Benutzermodellierung für benutzerspezifische Plangenerierung und -präsentation

Dissertation zum Erwerb des
akademischen Grades Dr. rer. nat.

vorgelegt beim
Fachbereich 6 (Mathematik und Informatik)
der Universität Essen

von

Detlef Küpper, geboren in Wuppertal

Datum der mündlichen Prüfung: 23. Oktober 2002

Gutachter:

Prof. Dr. Alfred Kobsa
(Universität Essen und University of California, Irvine)

Prof. Dr. Rainer Unland
(Universität Essen)

Diese Dissertation ist 2003 bei der Akademischen Verlagsgesellschaft Aka GmbH, Berlin in der Reihe “Dissertationen zur Künstlichen Intelligenz (DISKI)” als Band 269 erschienen.
ISBN: 3-89838-269-9
Detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de abrufbar.

Foreword

Plan generation by computers has gained increasing importance in the past few years. It has being used, e.g., in robotics applications, industrial process control, space mission planning, database systems, and CAD/CAM applications. To date, generated plans are thereby nearly always carried out by machines. Research on generating plans that are to be carried out by people is still in a relatively early stage. This book by Detlef Küpper lays important foundations for this new line of research.

Küpper specifically focuses on catering plans to the knowledge and abilities of each individual user. So far, plans have been generated for ”average” users only. The fact that people have different abilities that may or may not enable them to carry out such plans has been generally disregarded. Moreover, generated plans are currently communicated to users by simply listing their lowest-level steps. Hardly any attention has been paid to the fact that users may be unfamiliar with some plan steps, or may on the other hand be very acquainted with all low-level steps and therefore be better served with a higher-level description. Initial experiments by [Young, 1999] indeed demonstrated that people make fewer errors executing plans and achieve higher success rates if plan descriptions are tailored to their levels of expertise.

This dissertation constitutes the first scientific treatment of the first problem and, together with Young’s work, also of the second problem. The author takes an exceptionally broad approach, reviewing current research in planning, plan recognition, interactive planning, text and dialog planning, help systems, and planning for information gathering. He proposes models of each user’s knowledge and capabilities as central knowledge sources for generating personalized plans that users are able to carry out, and for communicating these plans to users in such a way that they will be easily understood. The former is achieved by a partial order planner that obtains its working set of plan operators from the user model (these operators correspond to users’ abilities and are obtained, e.g., through stereotypes). The latter is accomplished by a presentation algorithm that aims at presenting generated plans at a high level of abstraction, while ascertaining that users are still able to decompose them into plan steps that they are able to carry out.

The proposed framework for user-tailored plan generation and presentation was implemented as a research prototype, to verify its consistency and completeness. This framework is however also based on several assumptions about human comprehension and plan execution. While these assumptions are surely plausible, future research will have to verify them empirically to determine the ecological relevance of Küpper’s interesting work.

Alfred Kobsa
University of California, Irvine

Danksagung

Diese Arbeit hat einen recht langen Weg von ersten Ideen zu natürlichsprachlichen Dialogsystemen bis zu ihrer jetzigen Form hinter sich. Auf diesem Weg haben mich zahlreiche Menschen durch Diskussionen, Hinweise, technische Ratschläge oder auch ”nur” durch einige aufmunternde Worte unterstützt. Auch wenn ich hier bei weitem nicht alle nennen kann, gebührt Ihnen mein aufrichtiger Dank.

Die wichtigste Unterstützung kam zweifelsohne von meinem Doktorvater Alfred Kobsa. Nicht nur, daß seine Arbeiten zur Benutzermodellierung mein Interesse für dieses Thema geweckt haben. Er hat mich mit großer Geduld immer wieder ermuntert die Arbeit voranzutreiben – auch nach Phasen, in denen andere Dinge im Vordergrund standen und die Arbeit nur langsam voranging. Nicht zuletzt habe ich es ihm zu verdanken, dass ich die intensivste Phase der Arbeit als Doktorand der GMD St.Augustin verbringen konnte. In diesem Zusammenhang sei auch Erich Vorwerk genannt, dem ich für die Einbettung dieser Stelle und die notwendige technische Ausstattung zum Dank verpflichtet bin. Rainer Unland möchte ich für seine spontane Bereitschaft danken, das Zweitgutachten zu übernehmen.

Unverzichtbar für den Wert einer wissenschaftlichen Arbeit sind Diskussionen, die die verschiedenen Aspekte der Arbeit hinterfragen. Dafür möchte ich mich besonders bei Josef Fink, Jörg Höhle, Wolfgang Pohl und Jörg Schreck – den Mitarbeitern des BGP-MS Projekts an der Universität Konstanz und der GMD – bedanken. Besonderer Dank gilt Jörg Höhle, dessen tiefe Kenntnis des BGP-MS-Systems eine wertvolle Hilfe für mich war, und Wolfgang Pohl, dessen BGP-MS-Erweiterungen eine Basis für die Entwicklung des Prototyps dieser Arbeit darstellt.

Ich möchte nicht versäumen Wolfgang Hoeppner zu nennen, der meine ersten ”Gehversuche” auf dem Gebiet natürlichsprachlicher Dialogssysteme begleitete und mich unterstützt hat, meine vagen Ideen auf eine solide Grundlage zu stellen.

Üblicherweise enden Danksagungen mit dem Dank an die Familie des Autors, die während einer solchen Arbeit meist zu kurz kommt – Dank für das Verständnis, verbunden mit dem Versprechen sich in Zukunft mehr Zeit für sie zu nehmen. Auch ich möchte an dieser Stelle sowohl den Dank aussprechen als auch das Verprechen abgeben. Darüber hinaus will ich die aktive Unterstützung nicht unerwähnt lassen. So war bspw. die ganze Familie in Lektorat und Endkontrolle involviert, meine Frau Angela übernahm zudem Installation und Test neuer LaTeX-Module und meine ältere Tochter Janina schrieb eine Übersetzung eines mathematischen Zeichensatzes in LaTeX-Symbole. Als kleines Dankeschön widme ich diese Arbeit

Angela, *Janina* und *Nadja*

Zusammenfassung

Um Ziele mithilfe eines Computers zu erreichen, sind zumeist mehrere Einzelschritte erforderlich, wodurch ein geplantes Vorgehen erforderlich oder zumindest sinnvoll wird. Allerdings kann nicht erwartet werden, daß ein Computerbenutzer Pläne für alle derartigen Ziele kennt. Da auch die Erzeugung neuer Pläne keineswegs einfach ist, bietet es sich an, für den Benutzer eine Hilfestellung in Form geeigneter Pläne bereitzustellen. Viele Hilfesysteme enthalten dafür eine Bibliothek von Plänen zu fest vorgegebenen Zielen. Dadurch können aber nur die häufigsten Ziele *typischer* Benutzer abgedeckt werden, weil sich durch Kombination der Einzelfunktionalitäten des Computers sehr viele verschiedene Ziele erreichen lassen. Eine Adaption an den jeweiligen Benutzer erhöht die Vielfalt um eine weitere Dimension.

Eine solche Anpassung gewinnt aber zunehmend an Bedeutung, weil heute beinahe jedermann Umgang mit Computern hat[1], aber kaum erwartet werden kann, daß sich jeder Benutzer in allen Gebieten der Computernutzung gleich gut auskennt. In der vorliegenden Arbeit wird gezeigt, daß durch eine benutzerspezifische Generierung und Präsentation von Plänen eine weitgehende Anpassung einer Hilfestellung an den jeweiligen Benutzer möglich ist. Durch die Plangenerierung ist die Menge der Ziele, für die Unterstützung geboten werden kann, nicht eingeschränkt – solange die Ziele überhaupt erreichbar sind.

In dieser Arbeit orientiert sich die Benutzeradaption einer Hilfestellung erstmalig daran, *daß der Benutzer seine Ziele* mithilfe des Ratschlags auch tatsächlich *erreichen kann*. Dies setzt voraus, daß auch die spezifischen *Möglichkeiten eines Benutzers zu handeln* berücksichtigt werden, die – wie noch gezeigt wird – auf den Befugnissen und physischen Fähigkeiten des Benutzers beruhen. Vorausgesetzt die Annahmen über die Handlungsmöglichkeiten des Benutzers und das Planungsverfahren sind korrekt, kann der Benutzer mit einem so erzeugten benutzerspezifischen Plan *im Prinzip* seine Ziele erreichen (möglicherweise muß die Präsentation des Plans aber noch Erläuterungen zur Ausführung enthalten). Auch die Planpräsentation muß also an den Benutzer angepaßt sein, um sicherzustellen, daß er die zur Ausführung des Plans notwendigen Kenntnisse hat. In dieser Arbeit wird gezeigt, daß hierfür nicht nur ein Modell der Benutzerkenntnisse benötigt wird, sondern zusätzlich auch wieder ein Modell seiner Handlungsmöglichkeiten. Um die Adaption leisten zu können, habe ich ein Benutzermodell entwickelt, das zum ersten Mal neben dem Wissen des Benutzers auch seine Handlungsmöglichkeiten explizit repräsentiert und beide Aspekte klar von einander trennt.

Als Grundlage für die Realisierung dieses Modells werden Terminologie, Repräsentation und Verwendung der Begriffe *Handeln* und *Planen* mit ihrem Umfeld in den Forschungsbereichen *Handlungsplanung*, *Benutzermodellierung* und *planbasierte Hilfesysteme* analysiert und gegenübergestellt. Das Ergebnis stellt, in der hier erstmalig vorliegenden systematischen Form, auch über die Hauptzielrichtung der Arbeit hinaus einen eigenständigen Beitrag dar. Für die Arbeit selbst zeigt die Untersuchung die Notwendigkeit einer Repräsentation von Planungswissen, die auch für eine Plangenerierung ausdrucksstark genug ist. Gleichzeitig ist es wünschenswert, die im Bereich Benutzermodellierung häufig verwendete begriffliche Repräsentation von Wissen auch für Planungswissen einzusetzen, um die bereits entwickelten Verfahren zur Kommunikation mit dem Benutzer, zur Wissensverarbeitung und insbesondere zur Wissensakquisition übernehmen zu können. Es wurde deshalb eine neue beschreibungslogische Repräsentation von Planungswissen entwickelt, die diesen Anforderungen genügt und sich gleichermaßen für die Modellierung der Handlungsmöglichkeiten des Benutzers und seiner Annahmen über Veränderungsmöglichkeiten im Sachbereich eignet. Sie kann sowohl für die Generierung und Erklärung von Plänen aus Sicht des adaptiven Systems als auch – bspw. um Benutzerfehler erklären zu können – für die Generierung von Plänen aus Sicht des Benutzers eingesetzt

[1] lt. der ACTA-Studie des Instituts für Demoskopie Allensbach vom Sept. 2001 verfügen alleine 62,8% der unter-65-jährigen Bevölkerung in Deutschland über einen privaten Computer in ihrem Haushalt.

werden. Anhand der Stereotype-Technik wird gezeigt, daß sich bestehende Verfahren zur Wissensakquisition auch für das so repräsentierte Planungswissen eignen.

Zur benutzerspezifischen Präsentation eines Plans gibt es bereits verschiedene Arbeiten. Die Verfahren, die in der vorliegenden Arbeit entwickelt werden, dienen in erster Linie der Untersuchung, welche neuen Aspekte sich durch das hier vorgestellte Benutzermodell für die Präsentation von Plänen ergeben. Exemplarisch wird dies für die Präsentation der Planstruktur, der Ausführungsmöglichkeiten von Planschritten und für die Entscheidung, welche Plankomponenten explizit in einer Beschreibung enthalten sein müssen – jeweils bezogen auf den jeweiligen Benutzer – gezeigt.

Um einen Plan auszuführen muß bekannt sein, *wie* die einzelnen Schritte ausgeführt werden können. Für die Abschätzung, ob ein Benutzer dazu Erklärungen benötigt, wird in dieser Arbeit erstmals untersucht, das Benutzerwissen nicht nur hinsichtlich Korrektheit, sondern auch hinsichtlich *Nutzbarkeit* und *Verläßlichkeit* zu beurteilen. Dadurch soll verhindert werden, daß der Benutzer seine Ziele *nicht* erreicht, weil er eine falsche Ausführungsmöglichkeit wählt, oder weil die ihm bekannten Ausführungsmöglichkeiten jenseits seiner Handlungsmöglichkeiten liegen. Es wird gezeigt, daß erst das hier entwickelte neue Benutzermodell die notwendige Information für eine solche Beurteilung enthält. Für die Ermittlung der Information, die der jeweilige Benutzer insgesamt benötigt, um einen für ihn erzeugten Plan ausführen zu können, wurde ein Algorithmus entwickelt, der im Präsentationskapitel dieser Arbeit beschrieben ist.

Für einen Benutzer ist es oft wichtig, den Zweck einzelner Schritte in einem Plan zu kennen. Die im Benutzermodell repräsentierten Annahmen über das Planungswissen des Benutzers ermöglichen erstmals die Überprüfung, ob der Benutzer diesen Zweck erkennen kann und deshalb auf die explizite Präsentation dieser Strukturinformationen verzichtet werden kann. Dies gilt analog für einzelne Schritte des Plans; d.h. die explizite Präsentation eines Planschritts kann entfallen, falls der Benutzer diesen anhand seines Planungswissens leicht selbst ermitteln kann. Für die Entscheidung, welche Komponenten des Plans explizit genannt werden, wurde ein weiterer Algorithmus entwickelt. Die Entscheidung basiert erstmalig auch auf einer Abwägung der Aufwände des Benutzers für die Verarbeitung einer längeren Beschreibung und für die Rekonstruktion fehlender Beschreibungskomponenten.

Die Auswirkungen, die Annahmen über Kenntnisse und Handlungsmöglichkeiten eines Benutzers auf Pläne und Erklärungsbedarf für den jeweiligen Benutzer haben, werden anhand eines prototypischen Hilfesystems und verschiedener Beispielbenutzer gezeigt. Im abschließenden Kapitel werden weitere Anwendungsmöglichkeiten des neuen Benutzermodells angesprochen, die als Anknüpfungspunkt für weitere Forschungsarbeiten dienen können.

Abstract

Goal-oriented work with computers requires a number of steps usually. Thus computer users may benefit from plans in order to reach their goals. However, we should not expect users knowing plans for all their goals. Because generating plans is anything but easy, the computer itself should support the user by providing an adequate plan that may achieve his goals. Many help systems offer plans from a static plan library. The enormous number of possible goals and plans limit the coverage of plan libraries to *usual goals* of *typical users*. The great number of possible plans results from combining manifold functions of nowadays computers. Tailoring plans to the specific needs and properties of the current user adds another dimension of complexity. Nevertheless the importance of such an adaptation increases continuously. Because of the large circulation of computer usage[2] we must expect computer users with very different knowledge and capabilities. The work presented here deals with user adapted advice. It bases on user tailored plan generation and presentation, and thus is not limited by a static library.

Adaptation in this work not only means presenting the user a plan in a way that he may understand. Adaptation of plans must also consider the *specific capabilities of the user to act* in the real world. This is an indispensable precondition if we want to prevent user advice that the user cannot carry out. We will show that the *specific capabilities users to act* depend on their physical abilities and their authorization. Provided that the planning procedure and the system's assumptions of the current user's capabilities are correct, the system may generate a plan that enables this user to reach his goals *in principle*. The proviso *in principle* is added since – depending on his knowledge – the user may need additional explanations on how to perform the plan's steps. Thus, also the presentation of the plan should be adapted to the current user. We will show that this adaptation not only requires a model of the user's knowledge but again also a model of his capabilities. One contribution of this work is a new model of the user's properties that includes but strictly separates both topics.

A foundation of this model is laid by a study of terminology, representation, and usage of the terms *action* and *plan* within the related research areas *planning*, *user modeling*, and *plan based help systems*. This systematic study is another contribution of this work – independent of the main goal of the work. It shows the need for an expressive representation of planning knowledge that supports plan generation. Furthermore the use of a terminological representation seems to be beneficial. Many user modeling systems use such a representation. Thus we may exploit the rich research experience in this area (e.g., for explaining concepts to the users and gathering conceptual knowledge of the user) for planning knowledge too. Chapter 3 introduces a new representation of planning knowledge that shows the required properties and is employed by both parts of the user model. The model serves for the main goal of this work – user tailored planning and plan presentation – and for planning from the user's viewpoint. The latter may be useful for analyzing user misconceptions. We use stereotypes as an example how the approach may exploit well-known user modeling techniques for the acquisition of assumptions about the user's planning knowledge and capabilities.

Research on user tailored plan presentation is already available. However, it does not consider the user's capabilities. The methods developed in chapter 4 demonstrate the benefit of an exploitation of the new user model for plan presentation. They cover user tailored presentation of the plan's structure and execution alternatives as well as decisions on whether or not a component of the plan should be presented explicitly to the current user.

First of all, a user who wants to perform a plan needs knowledge of *how* to perform all the plan steps. In order to decide if and what additional knowledge the user needs, we not only judge if the (assumptions about the) user's knowledge is correct. Rather we introduce the terms *usable* and *re-*

[2] A survey of the *Institut für Demoskopie Allensbach* from Sept. 2001 shows that 62,8% of the less than 65 year old people in Germany use computers at home.

liable knowledge. The new user model enables this more specific rating of the user's knowledge. Furthermore, also the system's knowledge may or may not be usable for the current user. Considering usableness ensures that presentations only contain explanations that the user can perform. If the user has unreliable knowledge, he may choose a wrong execution alternative. Thus the system offers information that prevents him from doing so. Chapter 4 contains an algorithm that collects all information that the current user needs to perform a given plan.

Often users need or at least want to know about the reasons for the presence of steps in a plan. The planning knowledge contained in the new user model enables an estimation whether or not the user can recognize these reasons. Only in the latter case a presentation should contain this structural information about the plan. The same holds for plan steps, i.e., plan steps that the user can easily infer, might be regarded as redundant and thus should not be presented explicitly. The algorithm that decides whether or not a component of the plan should be mentioned explicitly in the presentation introduces a technique that is based on an estimation of the current user's effort for recognizing the presentation. This includes balancing between the user's effort to comprehend and memorize a description of this component and the effort for him to reestablish it when it is omitted.

A prototypical help system demonstrates the impact of assumptions about the knowledge and capabilities of different example users on plans and explanations that are generated for them. The last Chapter gives an outlook to possible applications of the new user model, which may be seen as a starting point for further research.

Inhaltsverzeichnis

1 Einleitung und Überblick **1**
1.1 Pläne und Computerbenutzer . . . 2
1.1.1 Handlungsplanung . . . 2
1.1.2 Präsentation eines Plans . . . 4
1.1.3 Benutzermodellierung . . . 5
1.2 Problemstellung und Lösungsansatz . . . 6
1.2.1 Motivation . . . 6
1.2.2 Ansatz der Arbeit . . . 8
1.2.2.1 Der Beispiel-Sachbereich . . . 8
1.2.2.2 Plangenerierung und Planpräsentation . . . 9
1.2.2.3 Inhalte des Benutzermodells und Wissensrepräsentation . . . 9
1.3 Aufbau der Arbeit . . . 11

2 Pläne in verwandten Forschungsgebieten **13**
2.1 Handlungsplanung . . . 13
2.1.1 Least Commitment Planning . . . 17
2.1.2 Abstraktion – Hierarchisches Planen . . . 19
2.1.2.1 Situationsabstraktion . . . 19
2.1.2.2 Operatorabstraktion . . . 21
2.1.3 Voraussetzungen beim Planen . . . 25
2.1.4 Abschließende Bemerkungen . . . 28
2.2 Planerkennung . . . 28
2.2.1 Wurzeln der Planerkennung . . . 28
2.2.2 Kautz' Theorie der Planerkennung . . . 31
2.2.3 Planerkennung mit Beschreibungslogik . . . 37
2.2.3.1 Modellierung von Aktionen und Plänen mit Hilfe von Zustand-Concepts . . . 37
2.2.3.2 Modellierung von Plänen durch Temporal Constraint Networks . . . 41
2.2.4 Weitere Ansätze der Planerkennung . . . 44
2.2.5 Zusammenfassung . . . 44
2.3 Gemeinsames Planen (mit einem Benutzer) . . . 46
2.3.1 Planungsunterstützung durch TRAINS/TRIPS . . . 47
2.3.2 Pläne als mentale Zustände . . . 52
2.3.3 Zusammenfassung . . . 58
2.4 Text- und Dialogplanen . . . 60
2.4.1 Textplanen als Planungsprozeß . . . 60
2.4.2 RST-basiertes Textplanen . . . 63

2.4.2.1 Hovys Textplaner . . . 64
2.4.2.2 Weitergehende Ansätze . . . 65
2.4.3 Decompositional Partial-Order Causal Link Planning . . . 67
2.4.4 Dialogplanen . . . 69
2.4.5 Zusammenfassung . . . 69
2.5 Hilfesysteme . . . 70
2.5.1 Erkennung von falschen oder suboptimalen Plänen . . . 71
2.5.2 Erkennung von Fehlerursachen . . . 74
2.5.3 Erzeugen von Plänen . . . 76
2.5.4 Benutzerspezifische Hilfe . . . 80
2.5.5 Zusammenfassung . . . 81
2.6 Planen zur Informationsbeschaffung . . . 82
2.7 Abschließende Bemerkungen und Zusammenfassung . . . 87

3 Modellierungs- und Plangenerierungsaspekte 91
3.1 Handlungspotential und Wissen von Benutzern . . . 92
3.1.1 Alternativen zum Verarbeitungsschema . . . 96
3.1.2 Veränderungsmöglichkeiten eines Handlungspotentials . . . 100
3.1.3 Berücksichtigung des Handlungspotentials in verwandten Forschungsarbeiten 102
3.1.4 Zusammenfassung . . . 103
3.2 Repräsentation von Planoperatoren . . . 104
3.2.1 Beschreibungslogiken . . . 105
3.2.2 Planoperatoren in Beschreibungslogik . . . 110
3.2.2.1 Spezifika der Repräsentation in SB-ONE . . . 115
3.3 Struktur eines Benutzermodells mit Handlungspotential und Wissen . . . 119
3.3.1 Akteurmodelle . . . 120
3.3.2 Repräsentation eines Benutzermodells mit Handlungspotential und Wissen im AsTRa-Formalismus . . . 124
3.3.3 Stereotypes . . . 128
3.3.4 Geschachtelte Annahmen über Handlungsmöglichkeiten . . . 133
3.3.4.1 Auswirkungen geschachtelter Annahmen auf die Partitionsstruktur 134
3.4 Planungsproblem und Planungssachbereich . . . 138
3.4.1 Das Planungsproblem . . . 139
3.4.2 Der Planungssachbereich . . . 140
3.4.2.1 Typen im Planungssachbereich . . . 142
3.4.2.2 Abhängigkeiten im Sachbereich . . . 143
3.4.2.3 Nichtatomare Formeln im Planungssachbereich . . . 145
3.4.2.4 Ableitung von Abhängigkeiten aus dem Benutzermodell . . . 146
3.4.2.5 Erzeugen von Objekten . . . 153
3.4.2.6 Vereinfachte beschreibungslogische Inferenzen . . . 154
3.4.2.7 Erzeugung des aktiven Teils eines Planungssachbereichs . . . 156
3.5 Das Umfeld des Benutzers: Werkzeuge und andere Akteure . . . 160
3.5.1 Der Einfluß von Werkzeugen auf Handlungsmöglichkeiten . . . 160
3.5.1.1 Die Repräsentation eines Verwendungspotentials . . . 162
3.5.1.2 Ermittlung des werkzeugspezifischen Handlungspotentials . . . 165
3.5.1.3 Realisierungsaspekte in SB-ONE . . . 168
3.5.2 Unterstützung durch andere Akteure . . . 168
3.5.2.1 Gemeinsames Handeln . . . 169

3.5.2.2 Verteiltes Handeln 169

4 Präsentation eines Plans **171**
4.1 Planstruktur 172
4.2 Wissen wie ein Plan-Concept ausgeführt wird 176
4.2.1 Dekomposition eines Plan-Concepts 176
4.2.2 Beurteilung von Wissen, wie ein Plan-Concept ausgeführt wird 180
4.2.2.1 Korrekte und falsche Annahmen über Ausführungsmöglichkeiten von Plan-Concepts 181
4.2.2.2 Nutzbares Planungswissen 183
4.2.2.3 Verläßliches Wissen über die Ausführungsmöglichkeiten von Plan-Concepts 184
4.2.3 Zusätzliche Akteure 185
4.3 Inhaltsbestimmung und Bewertung einer Planpräsentation 186
4.3.1 Der Algorithmus zur Planpräsentation 188
4.3.2 Gesamtbewertung einer Planpräsentation 191
4.3.3 Die Bedeutung falscher Annahmen des Benutzers für die Planpräsentation . . 192
4.3.4 Arbeiten zum Verstehen von Plänen und Anweisungen aus der Psychologie . 193
4.3.5 Alternative Modelle zur Planbewertung 195
4.4 Berücksichtigung von Benutzer-Inferenzen 197
4.4.1 Verwandte Ansätze zur Berücksichtigung von Benutzer-Inferenzen 197
4.4.2 Relativer Aufwand und Benutzerwissen 199
4.4.2.1 Bedingungen für das Entfernen von Komponenten aus einer Planpräsentation 200
4.4.2.2 Aufwandsabschätzung für die Rekonstruktion von Komponenten . 202
4.4.3 Algorithmus für das Entfernen von Komponenten aus einer Planpräsentation 205
4.4.4 Diskussion 211
4.5 Optimierte Plangenerierung 213
4.5.1 Nicht nutzbare Plan-Concepts 213
4.5.2 Steuerung des Planungsprozesses 214
4.6 Die innere Struktur von Plan-Concepts 217
4.6.1 Teil-Ganzes-Relationen in Beschreibungslogik 218
4.6.2 Repräsentation von Dekompositionen in BGP-MS 220
4.6.3 Beurteilung von Ausführungsmöglichkeiten im Vergleich zur Planerkennung 223
4.7 Zusammenfassung 225

5 Der Demonstrations-Prototyp **227**
5.1 Architektur des Gesamtsystems 227
5.1.1 Verarbeitung in der Modellierungsphase 229
5.1.2 Verarbeitung bei der Erzeugung benutzerspezifischer Handlungsanweisungen 231
5.1.3 Plangenerierung mit UCPOP 232
5.1.4 Eigenschaften der Algorithmen 234
5.1.4.1 Initiale Bewertung und Beurteilung von Plan-Concepts 235
5.1.4.2 Erzeugung eines Planungssachbereichs 237
5.1.4.3 Planerzeugung 240
5.1.4.4 Planpräsentation 240
5.1.4.5 Zusammenfassung 244
5.2 Eine Beispiel-Benutzermodellierungswissensbasis 244

5.2.1 Vorgehensweise . . . 245
5.2.2 Der Beispielsachbereich . . . 248
5.2.2.1 Das Dateisystem . . . 253
5.2.2.2 Erzeugen von Objekten . . . 258
5.2.3 Initialer Aufbau einer Benutzermodellierungswissensbasis . . . 259
5.3 Anwendung des Prototyps . . . 265
5.3.1 Beispiel-Planungsprobleme . . . 266
5.3.2 Benutzerspezifische Pläne . . . 267
5.3.2.1 Wissensakquisitation für das Benutzermodell . . . 268
5.3.2.2 Pläne für Oscar . . . 270
5.3.2.3 Pläne für Trixie . . . 273
5.3.2.4 Pläne für Nick . . . 275
5.3.2.5 Pläne für Theo . . . 278
5.3.2.6 Pläne für Sue . . . 280

6 Zusammenfassung und Ausblick 283
6.1 Zusammenfassung . . . 283
6.2 Ausblick . . . 285
6.2.1 Weitergehende Bewertung von Plänen . . . 286
6.2.2 “Bessere” Plan-Beschreibungen . . . 286
6.2.2.1 Abstraktere Plan-Beschreibungen . . . 286
6.2.2.2 Annahmen eines Benutzers über seine eigenen Handlungsmöglichkeiten . . . 287
6.2.3 Verallgemeinerung benutzerspezifischer Plangenerierung und -präsentation . 289
6.2.3.1 Abstrakte Benutzerziele – unvollständiges Systemwissen . . . 289
6.2.3.2 Lernbereitschaft des Benutzers – Adaptierbarkeit des Systems . . . 290
6.2.4 Weitere Einsatz- und Anwendungsmöglichkeiten . . . 291
6.2.4.1 Weitere Anwendungsbereiche . . . 291
6.2.4.2 Einsatz in der Planerkennung . . . 291
6.2.4.3 Einsatz in Dialogsystemen . . . 293
6.2.4.4 Fehlerdiagnose . . . 293
6.2.5 Empirische Überprüfung . . . 295
6.3 Abschließende Bemerkungen . . . 297

A Syntaxdefinitionen 299
A.1 Extended Backus-Naur Form (EBNF) . . . 300
A.2 Zusammenfassung syntaktischer Definitionen in EBNF . . . 301
A.2.1 Beschreibung von Concepts . . . 301
A.2.2 Beschreibung von Plan-Concepts . . . 301
A.2.3 Beschreibung von Dekompositionen . . . 302
A.2.4 Beschreibung von Axiomen für den Planungssachbereich . . . 303
A.2.5 Beschreibung von Facts für den Planungssachbereich . . . 304
A.2.6 Beschreibung von impliziten Effekten . . . 304
A.2.7 Beschreibung von Vorrang-Mustern . . . 304
A.3 SB-ONE: Syntax der graphischen Notation . . . 306

B Sachbereichsmodell **307**
B.1 Domain-Concepts . . . 307
B.2 Plan-Concepts . . . 309
B.2.1 Plan-Concepts für die Plangenerierung . . . 310
B.2.1.1 open-file-tpx . . . 310
B.2.1.2 mount . . . 310
B.2.1.3 mount-win . . . 311
B.2.1.4 mount-ux . . . 311
B.2.1.5 app-print . . . 311
B.2.1.6 user-get . . . 311
B.2.1.7 launch . . . 311
B.2.1.8 launch-office-app . . . 312
B.2.1.9 com-copy . . . 312
B.2.1.10 copy . . . 312
B.2.1.11 ftp-download-file . . . 312
B.2.1.12 ftp-download-file-bin . . . 313
B.2.1.13 ftp-download-file-text . . . 313
B.2.1.14 remote-copy . . . 313
B.2.1.15 download-file . . . 313
B.2.1.16 ftp-open-connection . . . 313
B.2.1.17 ftp-close-connection . . . 314
B.2.1.18 ftp-chmode-binary . . . 314
B.2.1.19 ftp-chmode-text . . . 314
B.2.1.20 ftp-cd . . . 315
B.2.1.21 ftp-lcd . . . 315
B.2.1.22 shutdown-pc . . . 315
B.2.2 Unvollständige Plan-Concepts . . . 315
B.2.3 Dekompositionen . . . 316
B.2.4 Abkürzungen für Plan-Concepts . . . 317
B.3 Potentielle Axiome . . . 317
B.4 Vorrang-Muster . . . 319

C Systemumgebung und ausgewählte Protokolle **321**
C.1 Systemumgebung und -voraussetzungen . . . 321
C.2 UCPOP-Ausgabe des Planungsresultats . . . 322
C.2.1 Plan O-C-e für Oscar und Ausgangssituation 1 . . . 323
C.2.2 Plan O-C-ne für Oscar und Ausgangssituation 2 . . . 324

Literaturverzeichnis **327**

Index **343**

Abkürzungen **347**

Abbildungsverzeichnis

2.1 Abstrakte Planschritte die nicht widerspruchsfrei angeordnet werden können. 24
2.2 Lösungsmöglichkeit mit dekomponierten Planschritten 25
2.3 Graphische Darstellung von Kautz' Beispiel *Cooking World* 32
2.4 Event Hierarchy mit Vorbedingungen und Effekten 35
2.5 Definition eines Action-Concepts aus [Devanbu und Litman, 1996, S.7] 38
2.6 Definition eines Plan-Concepts aus [Devanbu und Litman, 1996, S.12] 38
2.7 EBNF-Beschreibung eines Plan-Ausdrucks nach Devanbu und Litman 38
2.8 Endlicher Automat für die Schnittmenge von zwei regulären Sprachen 40
2.9 Plan Subsumption aus [Weida und Litman, 1992, S.284] 42
2.10 Plan Taxonomie aus [Weida und Litman, 1992, S.286] 45
2.11 Verband von widerlegbaren Regeln nach Ferguson 49
2.12 Argumentation über Pläne aus [Ferguson und Allen, 1994] 51
2.13 TRIPS System Architektur aus [Ferguson und Allen, 1998] 53
2.14 Gemeinsamer Plan und zwei Einzelpläne aus [Grosz und Sidner, 1990] 55
2.15 Grosz und Kraus' *Axiom schema for intending-that* 59
2.16 Request-Operator aus [Cohen, 1978, S.132] 61
2.17 Plan-Netzwerk aus [Appelt, 1985, S.139] 63
2.18 RST Rel. *Circumstance* aus [Mann und Thompson, 1987, App. I.1] 64
2.19 RST Planoperator *Circumstance* aus [Hovy, 1991, S.90]. 65
2.20 Decomposition Link in DPOCL 68
2.21 SINIX-Consultant Plan-Schema für das Löschen eines Verzeichnisses 72
2.22 Alternative ComfoHelp-Pläne aus [Mittermaier, 1995] 73
2.23 IUST+ Planbibliothek aus [Thies, 1994, S.81] 73
2.24 Beschreibung eines Fluges aus [Bauer und Dengler, 1999] 85
2.25 Inhaltsschema einer Flugbuchungsseite Fluges aus [Bauer und Dengler, 1999] 85
2.26 Darstellung des Operators *join* aus [Knoblock, 1996] 87
2.27 Informationsbeschaffungs-Plan aus [Knoblock, 1996] 88

3.1 Handlungspotential und Modell der Kentnisse 104
3.2 DL-Beschreibung: *Wichtige Nachricht* 109
3.3 EBNF-Syntax für Planoperatoren 111
3.4 Planoperator *open-file-tpx*. 111
3.5 Relation zwischen Rollenfüllern von Vorbedingungsattributen 114
3.6 EBNF-Syntax für die Definition von Plan-Concepts 114
3.7 Plan-Concept *open-file-tpx* 115
3.8 Plan-Concept *open-file-tpx* als Spezialisierung von *open_file* 116
3.9 Graphische SB-ONE Darstellung des Plan-Concepts *open-file-tpx* 119
3.10 Cohens geschachtelte Kontexte 121

3.11 Shared-Beliefs in BGP-MS 123
3.12 Eigene Partition für objektbezogenes Wissen 125
3.13 Partitionenhierarchie, bei der SB Sub-Partition aller SBA_iC Partitionen ist 127
3.14 Wünschenswerter Inhalt von SB 128
3.15 Geerbter Inhalt von SB bei Hierarchie nach Abb. 3.13 128
3.16 Partitionsstruktur mit Stereotype-Partitionen für einfache Annahmen 130
3.17 Partitionsstruktur der Systemannahmen 131
3.18 Partitionsstruktur mit Stereotype-Partitionen für Handlungspotentiale 132
3.19 Stereotypes aus Abb. 3.16 und 3.18 in AsTRa-Notation 133
3.20 Partitionsstruktur aus Abb. 3.16 mit separaten Partitionen für Objekte 135
3.21 Stereotypes mit Benutzer-Annahmen über das System 136
3.22 Partitionsstruktur aus Abb. 3.20 mit Benutzerannahmen über das System 137
3.23 EBNF-Syntax für die Definition von impliziten Effekten 151
3.24 EBNF-Syntax für die Definition eines Vorrang-Musters 158
3.25 KRSS-Darstellung des Plan-Concepts *app-print* 159
3.26 SB-ONE-Darstellung des Plan-Concepts *app-print* 159
3.27 Erzeugung des Planoperators *app-print* 159

4.1 Vereinfachter Planausschnitt für das Beispielziel 174
4.2 Algorithmus zur Erzeugung und Bewertung einer Planpräsentation 189
4.3 Algorithmus für das Entfernen von Komponenten aus einer Planpräsentation 206
4.4 Funktion *check-del* 207
4.5 Prozedur *del-set* 208
4.6 Funktion *adv-set* 209
4.7 Beispiel einer Komponente, die von mehreren anderen abhängig ist. 210
4.8 Algorithmus zur Berechnung des Erklärungsaufwandes für ein Plan-Concept 216
4.9 EBNF-Syntax für die Definition von Dekompositionen 222
4.10 Beschreibung einer Dekomposition des Plan-Concepts *ftp-download-file-bin*. 222

5.1 Architektur des Demonstrations-Prototyps 228
5.2 Definition und Verwendung von Konstanten für Concept-Beschreibungen 246
5.3 Definition von Konstanten für Plan-Concept-Beschreibungen 247
5.4 Die wichtigsten Plan-Concepts für die Beispiele 249
5.5 Dekompositionen von *shutdown-pc* 250
5.6 Dekompositionen von *ftp-download-file-bin* 250
5.7 Grafische SB-ONE-Repräsentation der wichtigsten Domain-Concepts der Beispiele . 251
5.8 Die potentiellen Axiome des Beispielsachbereichs 252
5.9 Ausschnitt der Dateisysteme der Computer *PC* und *Konstanz* 257
5.10 Ausschnitt der Dateisysteme von *PC* und *Konstanz* nach *mount* 257
5.11 Pläne für das Beispielziel in zwei Ausgangssituationen 272

A.1 EBNF-Syntax für die Definition von SB-ONE-Concepts 302
A.2 EBNF-Syntax für die Definition von Plan-Concepts. 302
A.3 EBNF-Syntax für die Definition von Dekompositionen 303
A.4 EBNF-Syntax für die Definition von potentiellen Axiomen. 303
A.5 EBNF-Syntax für die Definition von impliziten Effekten. 304
A.6 Graphische Darstellung von SB-ONE-Netzen 306

Tabellenverzeichnis

3.1 Erforderliche Eigenschaften für die Ausführung von Aktionen 95
3.2 Syntax und Semantik von Concept-Ausdrücken 107
3.3 KRSS-Ausdrücke von Vorbedingungskomponenten 113
3.4 Zusammenfassung Abhängigkeiten 153

4.1 *Teil-Ganzes* Beziehungen 219

C.1 Systemvariablen und Suchpfade für den Betrieb des Prototyps 322

Kapitel 1

Einleitung und Überblick

Der Einsatz von Plänen ist immer dann sinnvoll, wenn Ziele angestrebt werden, die nicht unmittelbar in einem Schritt erreicht werden können. Dies trifft insbesondere auch für Ziele zu, die mithilfe eines Computers erreicht werden sollen. Ein Plan besteht aus einer Folge von einzelnen Schritten bzw. Aktionen, durch deren Ausführung (im Idealfall) das Gesamtziel erreicht wird. Das erfolgreiche Ausführen eines Planschritts ist normalerweise an bestimmte Voraussetzungen – der Vorbedingung des Planschritts – geknüpft. Solche Vorbedingungen müssen durch das Ausführen anderer Schritte befriedigt werden, wenn sie nicht bereits vor Beginn des Plans erfüllt waren.

Die Zahl der möglichen Pläne, mit denen Ziele unter Zuhilfenahme eines Computers erreicht werden können, ergibt sich aus der Kombination der einzelnen Funktionen der Programme des Computers. Bereits diese Zahl der Einzelfunktionen ist bei handelsüblichen Standard-Computern so groß, daß nicht erwartet werden kann, daß jeder Benutzer alle kennt. Selbst für Computer ist es nicht praktikabel, alle daraus resultierenden Pläne zu speichern und dem Benutzer bei Bedarf als Hilfestellung anzubieten. Dementsprechend sind konventionelle Hilfesysteme auf Pläne für die häufigsten Ziele typischer Benutzer beschränkt.

Alternativ können Pläne bei Bedarf anhand konkret vorliegender Ziele erzeugt werden. Allerdings ist eine solche Planerzeugung – wie noch gezeigt wird – keineswegs einfach. Der naheliegende Wunsch, eine solch aufwendige Aufgabe einem Computer zu übertragen, ist Thema des Forschungsbereichs Handlungsplanung (vgl. bspw. [Yang, 1997], [Hertzberg, 1989] oder [Russell und Norvig, 1995, Kap.11-13]), wobei dort allerdings Pläne für Roboter im Vordergrund stehen. Die Grundidee besteht darin, Befehle bzw. Funktionen die ein Roboter ausführen kann als Basisbausteine für die Konstruktion eines Plans einzusetzen. Ein solcher Plan kann zur Ausführung gebracht werden, indem dem Roboter die entsprechende Folge von Befehlen gegeben wird. Das notwendige Wissen für die Erzeugung des Plans muß dabei im planenden Computerprogramm repräsentiert sein. Der Roboter muß lediglich (durch seine Konstruktion) sicherstellen, daß die Befehle genau so, wie in der Schnittstellenbeschreibung des Roboters angegeben, ausgeführt werden.

Die Grundidee läßt sich auch auf Pläne übertragen, die von einem Computerbenutzer ausgeführt werden sollen. Um sicherzustellen, daß der Benutzer (zumindest im Idealfall) mit einem solchen Plan seine Ziele erreicht, muß sich die Erzeugung des Plans – analog zu denen für einen Roboter – daran orientieren, welche Handlungen der Benutzer *ausführen kann*. Natürlich gibt es für einen Computerbenutzer keine festgelegte Menge von Befehlen, die er ausführen kann. Maßgeblich für die Planerzeugung sind die *Möglichkeiten zu handeln*, die der jeweilige Benutzer hat. Im Verlauf dieser Arbeit wird gezeigt, daß diese *prinzipiellen Handlungsmöglichkeiten eines Benutzers* durch seine physischen Fähigkeiten und seine Rechte zur Ausführung von Handlungen bestimmt werden[1]. Wird einem

[1] vgl. insb. Kap. 3.1 *Handlungspotential und Wissen von Benutzern*, das auch Definitionen der Begrifflichkeit enthält.

Benutzer das Ergebnis eines darauf basierenden Planungsprozesses präsentiert, müssen ihm dann allerdings noch Erklärungen über die Planschritte gegeben werden, die er nicht gut genug *kennt*, um sie korrekt ausführen zu können. Da nicht erwartet werden kann, daß alle Benutzer gleiche Kenntnisse haben, müssen die Erklärungen an den Wissenstand des jeweiligen Benutzers adaptiert sein.

Für den Gesamtprozeß aus Erzeugung und Präsentation eines benutzerspezifischen Plans müssen demnach als Eigenschaften des Benutzers seine *Möglichkeiten zu handeln* und sein Wissen – insbesondere sein *Planungswissen* – berücksichtigt werden.

Die Modellierung von Benutzereigenschaften, die Anwendungsprogrammen ermöglicht sich an den jeweiligen Benutzer anzupassen, ist Thema der Benutzermodellierung (vgl. [Kobsa und Wahlster, 1989], [McTear, 1993] und [UMUAI, 2001]). Die bisherigen Arbeiten in diesem Bereich befassen sich allerdings hauptsächlich mit den Präferenzen und dem Wissen, nicht aber mit den Handlungsmöglichkeiten eines Benutzers. Speziell das Wissen über Pläne wird im Teilgebiet Planerkennung verwendet. Allerdings geht es dort darum einen Plan zu erkennen, den der Benutzer bereits ausführt (und demnach auch kennt) und nicht darum einen Plan zu erzeugen. Deshalb wird das Planungswissen meist ohne Vorbedingungen und Effekte modelliert und ist so für die Planerzeugung ungeeignet.

Diese Arbeit greift die (aus Sicht einer benutzerspezifischen Planerzeugung und -präsentation) bestehenden Defizite auf und zeigt, welche Informationen über den Benutzer für die Erzeugung und Präsentation benutzerspezifischer Pläne notwendig sind, wie sie repräsentiert werden können und in welchem Zusammenhang sie stehen. Der wichtigste innovative Beitrag dieser Forschungsarbeit besteht in der Konzeption und Realisierung der Integration von *Planungswissen* und *Möglichkeiten zu handeln* in den Rahmen logikbasierter Benutzermodellierung[2]. Das daraus resultierende neue Modell der (relevanten) Eigenschaften eines Benutzers ermöglicht einem adaptiven System erstmals eine Hilfestellung oder einen Ratschlag dahingehend an den Benutzer anzupassen, daß dieser damit auch tatsächlich seine Ziele erreichen kann. Grundlage für die Realisierung dieses Modells ist eine Analyse und Gegenüberstellung der Modellierung und Verwendung von Plänen und deren Komponenten in angrenzenden Forschungsgebieten, die über die Hauptzielrichtung der Arbeit hinaus einen eigenständigen Beitrag darstellt.

1.1 Pläne und Computerbenutzer

1.1.1 Handlungsplanung

Ausgangspunkt für die systematische Forschung im Bereich Handlungsplanung, auf dessen Entwicklung in Kap. 2.1 *Handlungsplanung* noch ausführlicher eingegangen wird, war die Frage, wie Menschen beim Problemlösen und Planen vorgehen [Newell und Simon, 1961]. Aus [McCarthy und Hayes, 1969], [Fikes und Nilsson, 1971] und [Lifschitz, 1986] entwickelte sich daraus eine Vorstellung, die unter der Bezeichnung *STRIPS-Formalismus* auch heute noch Ausgangspunkt für viele Planungssysteme ist.

Diese Vorstellung beruht darauf, daß die reale Welt zu einem bestimmten Zeitpunkt eine Situation darstellt, die durch eine prädikatenlogische Formel beschrieben werden kann. Veränderungen werden durch ausführen von Aktionen hervorgerufen. Die konkret ausgeführten Aktionen oder Handlungen sind dabei Exemplare von Handlungs-Mustern – den sogenannten *Operatoren* – die damit die Veränderungs*möglichkeiten* beschreiben. Die Beschreibung der Veränderung selbst wird als *Effekt* des Operators bzw. der Aktion bezeichnet. Das Ausführen einer Aktion ist aber nur möglich, wenn die *Vorbedingung* der Aktion bzw. des Operators erfüllt ist. Vorbedingung und Effekt lassen sich wieder durch eine prädikatenlogische Formel beschreiben. Alle diese Formeln müssen im reinen STRIPS-

[2] vgl. dazu bspw. [Pohl, 1997].

Formalismus Konjunktionen von Literalen sein. Da der Effekt die *Veränderung* durch Ausführung einer Aktion beschreibt, muß er im logischen Widerspruch zur Situation *vor* der Ausführung stehen. Die Ausführung einer Aktion führt deshalb zu einer neuen Situation, die bis auf den Effekt der Aktion identisch zur ursprünglichen ist. Die Beschreibung der Situation kann im STRIPS-Formalismus dadurch erreicht werden, daß der Effekt der Aktion mit der Beschreibung der ursprünglichen Situation konjugiert wird, wobei alle Literale entfernt werden, die im Widerspruch zum Effekt der Aktion stehen. Die originale STRIPS-Darstellung sah das explizite Hinzufügen und Löschen von Teilformeln vor.

Ein Plan ist eine Folge von Aktionen – den Schritten des Plans –, deren Ausführung eine (vollständig bekannte) Ausgangssituation sukzessive in eine Situation überführt, in der die Ziele erreicht sind. Die Ziele werden dabei wiederum durch eine prädikatenlogische Formel beschrieben. Diese Formel kann als Bedingung, die in der Zielsituation erfüllt werden muß, aufgefaßt werden. Die Aufgabe der Planerzeugung besteht nun darin, bei gegebener Operatormenge einen solchen Plan für eine ebenfalls gegebene Zielformel zu bestimmen.

Mit dieser Modellierung sind zwei Probleme verbunden, die im ungünstigsten Fall zum Scheitern eines Plans führen können. Das erste ist unter dem Namen *Qualifikationsproblem* bekannt (vgl. [McCarthy und Hayes, 1969]). Es besteht darin, daß es in der Realität prinzipiell nicht möglich ist vollständig anzugeben, unter welchen Umständen eine Aktion ausgeführt werden kann. Aber selbst wenn dies möglich wäre, müsste sich eine sinnvolle Modellierung auf die relevanten Bedingungen beschränken, um die Planerzeugung nicht beliebig komplex werden zu lassen. Irrelevant sind dabei Bedingungen, die *normalerweise* erfüllt sind und auch nicht verändert werden (können). In außergewöhnlichen Fällen kann dadurch allerdings die (erfolgreiche) Ausführbarkeit einer Aktion und damit eines Plans der die Aktion enthält scheitern. Das zweite Problem besteht darin, daß zufällige Ereignisse, bzw. allgemein Ereignisse, die nicht unter Kontrolle des Akteurs stehen, bereits erreichte Teilziele eines Plans zerstören können. Beide Probleme sind prinzipiell nicht lösbar. Sie haben ihre Ursache in der Modellbildung an sich, die nur ein unvollständiges Bild der Realität darstellen kann, und sind deshalb bei jeglicher Art des Planens präsent. Eine *Garantie*, daß ein Plan in der Realität auch tatsächlich seine Ziele erreicht, kann es also nicht geben. Natürlich macht es dennoch Sinn zu planen, wenn Störungen einer Planausführung nur selten zu erwarten sind. Zudem besteht in einem solchen Fall oft die Möglichkeit den Plan so zu modifizieren, daß dadurch die Ziele trotzdem erreicht werden.

Die Planerzeugung selbst ist ein aufwendiger Prozeß, dessen Berechnungskomplexität exponentiell von der Länge des Plans, d.h. der Anzahl der Planschritte, abhängt. Die Schwierigkeit besteht darin, geeignete Schritte auszuwählen und so anzuordnen, daß die zugehörigen Vorbedingungen jeweils unmittelbar vor Ausführung eines Schritts erfüllt sind. Letzteres ist eine notwendige Voraussetzung für eine erfolgreiche Planausführung. Dabei muß beachtet werden, daß die einzelnen Schritte interagieren können, indem ein Teilziel, das bereits durch den Effekt eines Schritts erreicht wurde, durch den Effekt eines nachfolgenden Schritts wieder zerstört werden kann. Wird ein solches Teilziel noch für andere Schritte im Plan benötigt, muß die Reihenfolge der Schritte geändert werden, wodurch sich neue Konflikte ergeben können.

Von den Bemühungen, die Komplexität zumindest für praktische Fälle zu reduzieren, haben die beiden folgenden auch Auswirkungen auf die Vorstellung über einen Plan. Ein Ansatz besteht darin, Entscheidungen während des Planungsprozesses so spät wie möglich zu treffen, weil sich jede Entscheidung später als Fehlentscheidung herausstellen kann. Entscheidungen über die Reihenfolge der Planschritte brauchen teilweise nicht einmal für den endgültigen Plan getroffen werden. Bspw. ist es für eine Aufgabe, die zwei Anwendungsprogramme benötigt, meist gleichgültig in welcher Reihenfolge sie gestartet werden, sofern dies geschehen ist bevor sie zur Anwendung kommen. In einem solchen Fall spricht man von einem partiell geordneten Plan – im Gegensatz zum strikt geordneten

Plan, in dem jeder Schritt seine feste Position hat. Die Vorstellung, daß die Schritte eines Plans nur partiell geordnet sein können, hat sich inzwischen durchgesetzt und liegt auch dieser Arbeit zugrunde.

Der Ansatz des hierarchischen Planens mit Operatorabstraktion[3] greift wieder auf die Vorstellung über das Planen durch Menschen zurück. Demnach wird bei komplexen Aufgaben nicht gleich jede einzelne Aktion geplant, sondern gröbere Einheiten, die in einem oder mehreren Schritten verfeinert werden. Klassisches Beispiel ist der Hausbau, für den zunächst mit Einheiten wie Fundamenterstellung oder Innenausbau geplant wird und Details, wie das Streichen eines Türrahmens, erst in späteren Schritten hinzukommen. Die groben Einheiten stellen für die Plangenerierung abstrakte Operatoren dar, die in Teilpläne aus weniger abstrakten Operatoren dekomponiert werden, bis der endgültige Plan nur noch aus primitiven Planschritten besteht, die nicht weiter dekomponiert werden können. Einem Operator werden durch den Entwickler des Sachbereichsmodells i.a. mehrere solcher Folgen zugeordnet, von denen der Planungsprozeß eine geeignete auswählen muß.

Im Bereich Handlungsplanung wird ein solcher Entwickler als Benutzer eines Planungssystems angesehen. Seine Aufgabe ist, außer der Definition der genannten Zuordnungen, vor allem die Festlegung der Operatoren mit ihren Vorbedingungen und Effekten. Die Pläne sollen aber normalerweise nicht von dem Benutzer ausgeführt werden. Meist sind sie für einen Roboter gedacht, dessen Befehlssatz bzw. Funktionsschnittstelle die Basis für die Definition der Operatoren bilden. Dennoch sehen Planungssysteme eine Ausgabe des Plans für den Benutzer vor, damit er das Planungsergebnis beurteilen, eventuelle Fehler in der Modellierung entdecken oder in den Planungsprozeß eingreifen kann. Eine Anpassung an die Eigenschaften des Benutzers findet dabei allerdings nicht statt.

1.1.2 Präsentation eines Plans

Im Gegensatz zur Zielsetzung der Plandarstellung im Bereich Handlungsplanung geht es hier darum, den Plan so zu präsentieren, daß der Benutzer – als Leser einer Planpräsentation – den Plan versteht und danach handeln kann. Dabei macht es einen Unterschied, ob der Plan quasi als vorgefertigtes Rezept bekannt ist oder erst für ein aktuelles Ziel von einem Planungssystem erzeugt wurde.

Im ersten Fall kann der Entwickler, der die Bibliothek vorgefertigter Pläne zusammenstellt, auch bereits Vorgaben für die Präsentation machen. Dies gilt sowohl für die Darstellung selbst als auch für die Festlegung welche Informationen sinnvollerweise zu dem jeweiligen Plan gegeben werden. Dabei kann vom Informationsbedarf und Wissen eines typischen Benutzers oder typischer Benutzergruppen ausgegangen werden. Der Aufgabenschwerpunkt der Planpräsentation liegt dann auf der Darstellung selbst. Beispiele dafür sind [McKeown *et al.*, 1990], [Rösner und Stede, 1992], [Paris und Linden, 1996] für natürlichsprachliche Anweisungstexte, [Miksch *et al.*, 1998] für graphische Darstellungen und [Wahlster *et al.*, 1993], [André, 1995] für animierte Multimedia-Darstellungen.

Wird ein Plan erst auf Anforderung für ein konkretes Ziel generiert, muß für eine Präsentation dieses Plans auch festgelegt werden, ob die Ergebnisdatenstruktur des Planungsprozesses als Information für den Benutzer ausreicht bzw. andererseits nicht zu detailliert ist. Ein erster Ansatz, aus der Ausgabe eines Planungssystems (NONLIN [Tate, 1976]) eine natürlichsprachliche Beschreibung zu erzeugen, findet sich in [Mellish und Evans, 1989]. Die Beschreibungen waren allerdings zu ausführlich und zu umständlich, um tatsächlich für Benutzer eingesetzt werden zu können. Ein Ansatz, Beschreibungen auf den wesentlichen Inhalt zu reduzieren, besteht darin, auf die explizite Nennung von Details die der Benutzer selbst leicht schließen kann zu verzichten. Young adaptiert diese Idee für die Beschreibung von Plänen, die vom Planungssystem DPOCL [Young *et al.*, 1994b] erzeugt wurden. Young zeigt in [Young, 1999], daß sich dadurch deutlich kürzere Planbeschreibungen erzeugen lassen, die zudem dazu führen, daß die Leser solcher Beschreibungen merklich weniger Fehler bei der Ausführung des zugehörigen Plans machen.

[3] vgl. Kap. 2.1.2.2.

Unterschiedliche Benutzereigenschaften berücksichtigt Young, indem er vorschlägt, unterschiedliche Schlußfolgerungsfähigkeiten der Benutzer durch benutzerspezifische Plangenerierungsalgorithmen zu berücksichtigen. Unterschiede beim Wissen von Benutzern, das ja die Basis für Schlußfolgerungen darstellt, wird von Young allerdings nicht berücksichtigt. Die Frage, ob der jeweilige Benutzer die Schritte des Plans ausführen kann – also ob der Plan für den Benutzer überhaupt brauchbar ist – wird von keinem der Ansätze behandelt.

1.1.3 Benutzermodellierung

Pläne spielen im Forschungsbereich Benutzermodellierung vor allem für die Planerkennung und für intelligente Hilfesystem eine wichtige Rolle. Ausgangspunkt für die Arbeiten zur Planerkennung war die Feststellung, daß eine angemessene Antwort auf eine Frage oft nur möglich ist, wenn die Intention des Fragestellers berücksichtigt wird. Dies beinhaltet sowohl seine Zielsetzung als auch seinen Plan, diese Zielsetzung zu erreichen. Ist die Intention einer Frage bekannt, können zusätzliche Informationen gegeben werden, die für das Erreichen des Ziels nützlich sind oder die auf Fehler im Plan hinweisen. Klassisches Beispiel aus [Allen und Perrault, 1980] ist die Frage nach der Abfahrtszeit eines Zuges. Ist bekannt, daß der Fragesteller eine Reise mit diesem Zug vorhat und deshalb rechtzeitig zum Bahnsteig gehen will, so sollte eine Antwort bspw. zusätzlich das Abfahrtsgleis angeben, weil diese Kenntnis für den Plan notwendig ist. Ist der Plan in der aktuellen Situation nicht durchführbar, weil bspw. der Zug ausgefallen ist, sollte auch darüber informiert werden. Da bei einer Frage meist weder Plan noch Ziel angegeben werden, müssen sie zunächst ermittelt werden. Für Auskunfts- oder Dialogsysteme erfolgt eine solche Planerkennung durch Vergleich der beobachteten Handlungen des Benutzers bzw. Fragestellers mit den Plänen einer Planbibliothek. Zu den beobachteten Handlungen des Benutzers gehört insbesondere auch seine Fragestellung. Die Planbibliothek ist meist eine fest vorgegebene Menge aller Pläne, die im Anwendungssachbereich des jeweiligen Systems relevant sind. Da eine solche Menge sehr groß werden kann, empfiehlt es sich die Pläne in einer kompakten Darstellung zu repräsentieren. Dafür haben sich hierarchische Repräsentationen durchgesetzt, die auf Kautz' Theorie der Planerkennung ([Kautz, 1987], [Kautz, 1991]) basieren und große Ähnlichkeiten mit der Dekompositionshierarchie abstrakter Planoperatoren in der Handlungsplanung haben. Ein beobachteter Plan(anfang) gilt dann als erkannt, wenn er eindeutig einer Folge primitiver Planschritte zugeordnet werden kann, die sich durch Dekomposition aus der Wurzel der Hierarchie ableiten läßt. Dabei wird normalerweise aber nicht der für die Plan*erzeugung* notwendige Aufwand getrieben, der sicherstellt, daß nur solche Folgen erzeugt werden, die einen *korrekten* Plan darstellen. Dies ist für die Plan*erkennung* nicht unbedingt erforderlich, weil der Benutzer seinen Plan bereits ausführt und die Verantwortung für dessen Korrektheit auf den Benutzer abgewälzt werden kann. Dementsprechend spielen Vorbedingungen und Effekte der Planoperatoren in der Planerkennung nur eine untergeordnete Rolle oder werden gar nicht erst repräsentiert, wie bspw. in [Weida und Litman, 1992] und [Weida und Litman, 1994]. Ist der Plan des Benutzers allerdings fehlerhaft, kann ein solches Verfahren der Planerkennung dazu führen, daß die Intention des Benutzer nicht richtig erkannt wird. Pollack kritisiert dies in [Pollack, 1986] und [Pollack, 1990] und schlägt auch die Erkennung und Korrektur von *falschen* Benutzerplänen vor.

Im Kontext von Hilfesystemen muß auf jeden Fall mit falschen oder unvollständigen Kenntnissen des Benutzers gerechnet werden, denn diese begründen gerade die Notwendigkeit einer Hilfestellung. Dementsprechend können die Planbibliotheken intelligenter Hilfesysteme[4] auch falsche Pläne enthalten, die typische Fehler von Benutzer(gruppen) in dem Anwendungssachbereich repräsentieren. Unter falschen Plänen seien hier Pläne verstanden, mit denen sich ihr Ziel nicht oder nur umständlich erreichen läßt. In [Mittermaier, 1995] findet sich eine Klassifizierung unterschiedlicher Fehlertypen.

[4] im Gegensatz zu Hilfesystemen, die nicht die möglichen Zielsetzungen des Benutzers berücksichtigen

Die Planbibliotheken dienen in intelligenten Hilfesystemen sowohl dazu die Ziele des Benutzers zu erkennen als auch um festzustellen, ob ein Benutzer einen falschen Plan ausführt. Ist letzteres der Fall, geben *aktive* Hilfesysteme von sich aus eine Warnung und einen Alternativvorschlag, wie das Ziel erreicht werden kann, ab. Die Ratschläge, wie ein Ziel erreicht werden kann, können wiederum der Planbibliothek entnommen werden. Normalerweise sind solche Planbibliotheken statisch und damit auf Ratschläge für Ziele beschränkt, die der Entwickler der Planbibliothek vorhergesehen und dafür eine Lösung bereitgestellt hat. Eine Ausnahme ist das PHI-System [Bauer *et al.*, 1993], das in der Lage ist, die Planbibliothek durch die Generierung neuer Pläne zu erweitern.

Als Eigenschaften des Benutzers werden für Planerkennung und Hilfesystem vor allem dessen Kenntnisse und Präferenzen berücksichtigt. Die beobachteten Handlungen eines Benutzers können oft zu mehreren Plänen gehören. Von den für die Planerkennung in Frage kommenden Plänen können diejenigen ausgeschlossen werden, von denen angenommen wird, daß der Benutzer sie nicht kennt. Die Berücksichtigung der Benutzerpräferenzen läßt frühzeitig eine Entscheidung darüber zu, welche der Erkennungsalternativen der Benutzer wahrscheinlich ausführt (bspw. [Bauer, 1996]). Das Modell der Benutzerkenntnisse wird in einigen Hilfesystemen (bspw. UNIX-Consultant [Wilensky *et al.*, 1988] und ComfoHelp [Krause *et al.*, 1993]) auch dazu verwendet, um Hinweise oder Erklärungen zu vermeiden, die der Benutzer bereits kennt. Im PHI-System wird das Modell der Benutzereigenschaften auch eingesetzt, um Hilfsangebote zu erzeugen, die sich an den Präferenzen des Benutzers orientieren und um Pläne zu vermeiden, die Schritte enthalten die der Benutzer nicht kennt. Die Frage, ob der jeweilige Benutzer die Schritte eines Plans *ausführen kann*, wird allerdings auch im Bereich Benutzermodellierung nicht berücksichtigt,

1.2 Problemstellung und Lösungsansatz

1.2.1 Motivation

Der Nutzen einer Erzeugung und Präsentation von Plänen, die an die Eigenschaften des jeweiligen Benutzers angepaßt sind, wird besonders deutlich, wenn Benutzer bei den *Gesamt*aufgaben unterstützt werden sollen, die diese zu bewältigen haben. Der Computer stellt dabei meist ein Werkzeug dar, mit dem viele, aber oft nicht alle Teilaufgaben durchgeführt werden. Die Computerbenutzung selbst ist oft nicht auf *ein* Anwendungsprogramm beschränkt, sondern erfordert den Einsatz mehrerer Programme oder Betriebssystemfunktionen. Als Beispiel dafür möge das Ziel:

Einen Ausdruck einer Datei bekommen, die sich auf einem fremden Computer befindet.

dienen, das eine alltägliche Aufgabe im Umfeld einer heutigen Büroumgebung darstellt. Eine (hier zunächst benutzer*un*spezifische) Lösungsmöglichkeit ist, die Datei vom fremden Computer mithilfe eines FTP[5]-Programms zu transferieren, mithilfe eines Textverarbeitungsprogramms auszudrucken und dann vom Drucker abzuholen. Die Lösung erfordert damit bereits zwei Anwendungsprogramme für Dateitransfer und Ausdruck und eine physische Handlung des Benutzers außerhalb des Computers.

Der konventionelle Weg einen Computerbenutzer zu unterstützen besteht darin, “Online-Handbücher” für die Anwendungsprogramme zur Verfügung zu stellen. Damit kann ein Benutzer allerdings lediglich die Bedienung eines einzelnen Programms nachschlagen. Der Benutzer muß also bereits das Programm und seine Funktionen kennen, um überhaupt entscheiden zu können, ob der Einsatz des Programms für seine Aufgabenstellung sinnvoll ist. Solche Kenntnisse können von “ungeübten Benutzern” nicht erwartet werden. Allerdings bedingt die Vielfalt an heutigen Anwendungsprogrammen,

[5] File-Transfer-Protocol

die zudem oft wiederum über eine sehr große Anzahl von Funktionen verfügen, daß selbst "Experten" kaum mehr alle Funktionen aller Programme kennen können. Diese Art der Unterstützung kann also nicht verhindern, daß (evtl.) sinnvolle Funktionen des Computers und seiner Programme ungenutzt bleiben, weil sie nicht bekannt sind.

Eine bessere Unterstützung versprechen Assistenten, die inzwischen für viele Programme mitgeliefert werden. Diese stellen vorgefertigte Lösungen für komplexe Aufgaben zur Verfügung, die üblicherweise häufig mit dem jeweiligen Programm durchgeführt werden. Ein Beispiel dafür ist das Erstellen von Serienbriefen mit einem Textverarbeitungsprogramm. Der Benutzer muß dabei nicht die einzelnen Funktionen des Programms kennen, die für die Aufgabe notwendig sind, weil der Assistent die notwendige Information bereitstellt.

Allerdings sind auch solche Assistenten auf ein Programm oder eine zusammenhängende Programmgruppe und zudem meist auf sehr wenige Aufgaben beschränkt. Die Abläufe sind zudem starr, d.h. sie können nicht vom Benutzer beeinflußt werden, auch wenn er für einzelne Teilaufgaben bessere oder kürzere Lösungen kennt.

Selbst wenn man sich auf *ein* Programm beschränkt, ergibt sich aus den einzelnen Funktionen eine sehr große Anzahl von Kombinationsmöglichkeiten. Es ist also von vorn herein nicht realistisch, jede sinnvolle Kombination als vorgefertigte Lösungsmöglichkeit durch einen Assistenten zur Verfügung zu stellen. Diese Tendenz wird noch einmal verstärkt, wenn die Funktionen mehrerer Programme kombiniert und/oder die speziellen Eigenschaften des jeweiligen Benutzers berücksichtigt werden. Durch die damit praktisch unbegrenzte Zahl von Einsatzmöglichkeiten und -varianten, die den Computer als universelles Werkzeug auszeichnen, können Ansätze, die auf vorgefertigten Lösungen bzw. statischen Planbibliotheken beruhen, nur einen sehr geringen Teil dieses Potentials abdecken.

Als Alternative bietet es sich an, Lösungen auf Anforderung für eine konkret vorliegende Aufgabe zu ermitteln, also einen Handlungsplan dafür zu erzeugen. Für die Unterstützung von Computerbenutzern gibt es allerdings nur wenige Systeme, die ein solches Vorgehen realisiert haben (vgl. dazu Kap. 2.5 *Hilfesysteme*). Ebenfalls wenig berücksichtigt werden die Eigenschaften des konkreten Benutzers. Wenn überhaupt, ist dies auf die Kenntnisse und Präferenzen des Benutzers beschränkt. Gerade die Berücksichtigung der Benutzereigenschaften gewinnt aber zunehmend an Bedeutung, weil das fortschreitende Vordringen des Computereinsatzes in viele Lebensbereiche fast jeden Menschen zum Computerbenutzer macht. Damit kann nicht mehr von *dem typischen Computerbenutzer* gesprochen werden, sondern es muß mit Benutzern mit sehr unterschiedlichen Eigenschaften gerechnet werden.

Völlig unbeachtet blieb bisher die Frage, ob ein Benutzer mit einem unterstützenden Rat seine Aufgabe bewältigen, d.h. seine Ziele erreichen kann. Dies betrifft weniger die (evtl. unvollständigen oder falschen) Kenntnisse des Benutzers, die ein Hilfesystem ausgleichen können sollte. Wichtiger ist die Frage, ob der Benutzer die Handlungen, die er ausführen soll, auch *ausführen kann*. Dies heißt bei der Bedienung eines Computers vor allem, ob er über die notwendigen Zugriffs- bzw. Ausführungsrechte verfügt; bei Handlungen außerhalb des Computers kommen die physischen Fähigkeiten des Benutzers hinzu. Ob der Plan des obigen Beispiels von einem Benutzer ausgeführt werden kann, und damit, ob der Plan für diesen Benutzer sinnvoll ist, hängt u.a. davon ab, ob er auf fremde Rechner zugreifen darf. Ob der letzte Schritt des Beispiels – das Abholen des Ausdrucks – tatsächlich so trivial ist wie er auf den ersten Blick erscheint, hängt ebenfalls von den spezifischen Handlungsmöglichkeiten des jeweiligen Benutzers ab. Ist der Benutzer beispielsweise Rollstuhlfahrer und der Drucker befindet sich nicht auf dem gleichen Stockwerk wie der Benutzer, kann dafür eine weitere Hilfestellung notwendig sein. Gleiches gilt bspw., wenn der Benutzer keinen Zugang zu dem Raum hat, in dem sich der Drucker befindet.

Als zentrale Fragen, die auch das Thema dieser Arbeit umschreiben, ergeben sich damit:

- wie kann ein Plan für den jeweiligen Benutzer erzeugt und präsentiert werden, sodaß dieser damit seine Ziele erreichen kann?

- welche Eigenschaften des Benutzers müssen dafür modelliert werden und in welcher Relation stehen sie?

Dazu kommt die eher technische Frage, inwiefern die Ansätze der entsprechenden Forschungsgebiete Handlungsplanung und Benutzermodellierung eingesetzt werden können, bzw. wie sie für diese Zielsetzung modifiziert oder ergänzt werden müssen.

1.2.2 Ansatz der Arbeit

Diese Arbeit orientiert sich an einem Hilfesystem (im weiteren kurz: *System*), das zu einer gegebenen Aufgabe eine Lösungsmöglichkeit, d.h. einen Plan für den jeweiligen Benutzer, erzeugt und ihm diesen in Form einer benutzerspezifischen Handlungsanweisung präsentiert. Die Zielsetzung dabei ist, daß der Benutzer unter Berücksichtigung seiner spezifischen Eigenschaften durch eine solche Handlungsanweisung in die Lage versetzt wird die Aufgabe zu erledigen; d.h. die damit verbundenen Ziele zu erreichen. Handlungsanweisungen sind nicht auf die Funktionalität eines Anwendungsprogramms beschränkt und können auch Handlungen umfassen, die nicht mit einem Computer durchgeführt werden.

1.2.2.1 Der Beispiel-Sachbereich

Als Beispiel-Anwendungssachbereich wurde eine heute übliche Büroumgebung in Unternehmen und Organisationen mit vernetzten Computerarbeitsplätzen ausgewählt. Viele Aufgaben haben die erwähnte Charakteristik, daß sie sowohl Handlungen mit als auch außerhalb eines Computers erfordern, wobei erstere nicht auf die Verwendung eines einzigen Anwendungsprogramms beschränkt sind. Die Mitarbeiter haben meist in verschiedenen Bereichen sehr unterschiedliche Kenntnisse und Handlungsmöglichkeiten. Die heutigen kurzen Innovationszyklen von Hard- und insbesondere von Software bedingen, daß Mitarbeiter – wenn überhaupt – nur in den für sie wichtigsten Anwendungen geschult werden. Sie können deshalb schwerlich in allen Gebieten kompetent sein. Vielmehr kennen sich Mitarbeiter meist in einem engen Bereich gut aus, sind aber auf Hilfe angewiesen, wenn eine Aufgabe aus dem Rahmen der täglichen Arbeit fällt. Auch die Handlungsmöglichkeiten sind üblicherweise verschieden. Die Zuständigkeiten der einzelnen Mitarbeiter oder Abteilungen, die auch mit Zugriffs-, Änderungs- und Ausführungsrechten verbunden sind, haben in einer Büroumgebung typischerweise den größten Einfluß darauf. Aber auch die körperlichen Eigenschaften können eine Rolle spielen – insbesondere, wenn auch behinderte Mitarbeiter berücksichtigt werden.

Bezogen auf die Arbeit mit dem Computer demonstriert der gewählte Beispielsachbereich damit die Notwendigkeit einer anwendungsübergreifenden Unterstützung, die die Kenntnisse und Handlungsmöglichkeiten des jeweiligen Benutzers berücksichtigt, wie sie durch bestehende Hilfesysteme *nicht* gewährleistet werden kann.

Typische Objekte des Beispielsachbereichs sind (vernetzte) Computer (ggf. mit verschiedenen Betriebssystemen) mit Anwendungsprogrammen und einem Dateisystem. Dazu kommen Dokumente, Computer-Peripheriegeräte wie Drucker, Datenträger, u.v.m. Der für das Beispiel modellierte Ausschnitt wird in Kap. 5.2.2 *Der Beispielsachbereich* beschrieben. Die wichtigsten Handlungsmöglichkeiten ergeben sich aus den Funktionalitäten der Anwendungsprogramme und Peripheriegeräte, sowie aus den Betriebssystem- und Netzwerkfunktionen. Ohne auch nur annähernd vollständig zu sein, können in vielen Unternehmen Benutzergruppen mit unterschiedlichen Zuständigkeiten und Rechten wie *"normale" Angestellte*, *Systemadministratoren*, *Gäste* und *Hilfskräfte* identifiziert werden.

1.2.2.2 Plangenerierung und Planpräsentation

Methodische Grundlage für die Erzeugung der Pläne ist die (klassische) KI-Handlungsplanung. Für den Demonstrationsprototyp wird das Planungssystem UCPOP [Penberthy und Weld, 1992] eingesetzt. In Kapitel 5.1.3 *Plangenerierung mit UCPOP* wird die Wahl des Planungssystems begründet und auch Alternativen diskutiert. Es wurde weitgehend auf die Ausnutzung spezifischer Eigenschaften des Planungssystems verzichtet, um auch andere Planungssysteme einsetzen zu können. Die Berücksichtigung der Benutzereigenschaften bei der Planerzeugung wird dadurch erreicht, daß der Planungssachbereich anhand der (Systemannahmen über die) prinzipiellen Handlungsmöglichkeiten des Benutzers erzeugt wird. Der Planungssachbereich beschreibt die Planoperatoren und Abhängigkeiten im Anwendungssachbereich, die das Planungssystem für die Lösung des Planungsproblems verwenden darf[6].

Mit einer solchen Plangenerierung kann aber zunächst einmal nur die Existenz des Plans nachgewiesen werden, mit der der jeweilige Benutzer *im Prinzip* seine Ziele erreichen kann. Möglicherweise fehlen ihm aber noch notwendige Kenntnisse, die zur Planausführung notwendig sind. Abgesehen von der Identifizierung der an einem Plan beteiligten Objekte betrifft dies vor allem das Wissen über die einzelnen Schritte des Plans und *wie* sie (wiederum durch den Benutzer) ausgeführt werden können. Diese Arbeit behandelt den zweiten Aspekt, da für die Referenzierung von Objekten bereits viele Forschungsarbeiten vorliegen (bspw. [Kobsa, 1985], [Schmauks und Reithinger, 1988], [Sarner und Carberry, 1992]). Die Präsentation eines Plans kann deshalb nicht nur die Informationen über den Plan selbst enthalten, sondern auch zusätzliche Erklärungen über die einzelnen Planschritte und wie diese ausgeführt werden können. Damit der Benutzer solche Erklärungen nicht nur verstehen, sondern auch danach handeln kann, müssen sie an die Kenntnisse und – wie der Plan selbst – auch an die spezifischen Handlungsmöglichkeiten des Benutzers angepaßt sein.

In dieser Arbeit werden die beiden Prozesse der Plangenerierung und Planpräsentation zunächst getrennt behandelt, um die jeweiligen Eigenschaften und Anforderungen an das Benutzermodell leichter herausarbeiten zu können. Da aber zu erwarten ist, daß die Benutzerakzeptanz einer Handlungsanweisung stark beeinträchtigt wird, wenn diese viele langwierige Erklärungen enthält, wird der Planungsprozeß im Verlauf der Arbeit dahingehend modifiziert, daß die Erzeugung von Plänen mit möglichst wenig Erklärungsbedarf bevorzugt wird. Dies heißt allerdings nicht, daß Pläne, für die der Benutzer Erklärungen benötigt, von vorn herein unberücksichtigt bleiben, denn dies würde den Nutzen eines Hilfesystems stark einschränken. Ein wichtiger Aspekt eines Hilfesystems besteht ja gerade darin, einem Benutzer Handlungsmöglichkeiten zu erklären, die dieser im Prinzip hat, die er aber wegen unvollständigen oder falschen Kenntnissen nicht nutzen kann.

Es sei noch angemerkt, daß sich die Präsentation eines Plans in dieser Arbeit auf die Festlegung der Informationen beschränkt, die der jeweilige Benutzer benötigt, um den Plan korrekt ausführen zu können. Die Problematik der Erzeugung einer angemessenen Darstellung gehört dagegen nicht zur Aufgabenstellung. Die Festlegung der Informationen setzt keinen bestimmten Ausgabemodus voraus. Für Beispiele von Handlungsanweisungen wird natürlichsprachlicher Text gewählt. Dies ist aber nur als *mögliche* Darstellung der Informationen zu verstehen. Auch die anderen im Abschnitt 1.1.2 *Präsentation eines Plans* genannten Darstellungsformen wären geeignet.

1.2.2.3 Inhalte des Benutzermodells und Wissensrepräsentation

Ohne hier bereits eine ausführliche Erörterung vorweg zu nehmen wird klar, daß es wichtig ist, die prinzipiellen Handlungsmöglichkeiten eines Benutzers von seinen Kenntnissen – insbesondere denen über Handlungsmöglichkeiten – zu unterscheiden. Kapitel 3.1 *Handlungspotential und Wissen*

[6] vgl. Kap. 3.4 *Planungsproblem und Planungssachbereich* für eine ausführliche Darstellung.

von Benutzern enthält dazu sowohl einige Beispiele von Handlungen, die ein Benutzer zwar ausführen könnte, aber nicht kennt als auch für den umgekehrten Fall. In dieser Arbeit werden die beiden Aspekte im Benutzermodell strikt auseinander gehalten. Für den Aufbau und die Verwaltung des Benutzermodells wird das Benutzermodellierungsshellsystem BGP-MS [Kobsa und Pohl, 1995] eingesetzt, das eine solche Trennung durch den Partitionsansatz [Cohen, 1978] unterstützt. Kapitel 3.3 *Struktur eines Benutzermodells mit Handlungspotential und Wissen* zeigt, welche Partitionen des Benutzermodells für welche Teilaufgaben erforderlich sind und welche Beziehungen die Partitionen untereinander haben. Es sei hier aber bereits erwähnt, daß hier und im weiteren vom Modell des Wissens oder der Kenntnisse des Benutzers die Rede sein wird, ohne daß damit der Anspruch verbunden ist, daß der Inhalt des Modells nur absolut wahre Sachverhalte repräsentiert (wer könnte eine solche absolute Wahrheit auch feststellen?). Vielmehr repräsentiert dieser Teil des Benutzermodells die Annahmen, die ein Beobachter – hier also das System – über die Annahmen des Benutzers hat. Diese können sehr wohl von den eigenen[7] Annahmen des Systems verschieden sein. Die Begriffe Wissen, Kenntnisse oder Annahmen werden dabei synonym verwendet. Die eigenen Annahmen des Systems werden ebenfalls explizit und separat in der Benutzermodellierungswissensbasis von BGP-MS repräsentiert. Läßt sich eine im Benutzermodell repräsentierte Annahme nicht aus den so repräsentierten Annahmen des Systems ableiten, wird sie (aus der Sicht des Systems) als *falsch* angesehen und kann Anlaß für eine korrigierende Erläuterung des Systems sein.

Die Begrifflichkeit des Sachbereichs stellt einen wichtigen Teil des Wissens dar, das in der Benutzermodellierungswissensbasis repräsentiert wird. Viele Wissensbasen, die für die Benutzermodellierung eingesetzt werden, repräsentieren diese Begriffe in einer Abstraktionshierarchie und setzen dafür terminologische Wissensrepräsentationssysteme ein (u.a. [Sleeman, 1985], [Sarner und Carberry, 1992]). Der Ansatz wird für diese Arbeit übernommen und darüber hinaus auf die Repräsentation von Handlungsmöglichkeiten erweitert. Kap. 3.2.2 *Planoperatoren in Beschreibungslogik* schlägt dazu eine terminologische Repräsentation für Handlungsmöglichkeiten vor, die ausdrucksstark genug ist, um als Grundlage für die Verarbeitung von Plänen und insbesondere auch für die Erzeugung benutzerspezifischer Pläne zu dienen. Sie verläßt aber andererseits nicht den Rahmen der Beschreibungslogik, wie sie bspw. in [Patel-Schneider und Swartout, 1993] definiert ist. Der Vorteil dieser begrifflichen Repräsentation ist zum einen, daß gängige Wissensrepräsentationssysteme – wie auch das zu BGP-MS gehörende SB-ONE [Kobsa, 1991] – eingesetzt werden können und – wichtiger noch – daß die Erfahrungen bei der Erklärung und der Akquisition solchen Wissens genutzt werden können.

Eine Wissensquelle für den Aufbau von Benutzermodellen sind *Stereotypes*[8] (vgl. [Rich, 1979], [Rich, 1983]), die *typische Annahmen* einer Benutzergruppe (ggf. bezüglich eines Themas) enthalten. Dabei werden die Gruppen so definiert, daß die Mitglieder weitgehend *gleiche* Annahmen haben. Wegen der gleichartigen Repräsentation von Begriffen und Handlungsmöglichkeiten eignen sich Stereotypes für die Repräsentation beider Arten von Wissen. Auch hierbei ist wieder eine Trennung zwischen Annahmen und Handlungsmöglichkeiten erforderlich. Darüber hinaus ergeben sich bei der Festlegung der Stereotypes unterschiedliche Gruppierungen, je nachdem, ob man Gruppen mit möglichst homogenen Handlungsmöglichkeiten oder homogenen Annahmen zu Grunde legt. Die dazu notwendige Verwaltung von Stereotypes verschiedener Typen wird durch eine Erweiterung von BGP-MS unterstützt, die in [Pohl, 1997] beschrieben wird.

[7] Unter *eigenen Annahmen* eines Akteurs werden abkürzend solche Annahmen verstanden, die dieser für wahr hält; im Gegensatz bspw. zu Annahmen anderer Akteure, die sich davon unterscheiden.

[8] *Stereotype* wird in dieser Arbeit als englischer Begriff verwendet, weil die wohl treffendste Übersetzung *Klischee* in diesem Kontext wenig geläufig ist.

1.3 Aufbau der Arbeit

Die Arbeit beginnt mit einer Untersuchung wie Handeln, Handlungsmöglichkeiten und Pläne in verschiedenen, für diese Arbeit relevanten Bereichen der Forschung zur künstlichen Intelligenz betrachtet, repräsentiert und verarbeitet werden.

Die Ergebnisse sind Grundlage für Kapitel 3. Dort wird der Frage nachgegangen, wie Annahmen und Handlungsmöglichkeiten eines Menschen geeignet in einem Benutzermodell repräsentiert werden können, das Basis für die Erzeugung und Präsentation benutzerspezifischer Handlungsanweisungen ist. Es wird ein Benutzermodell vorgeschlagen, das sowohl für Annahmen als auch für Handlungsmöglichkeiten eine terminologische Repräsentation vorsieht. Dabei werden auch die hierfür relevanten Eigenschaften terminologischer Repräsentationen bzw. Beschreibungslogiken im allgemeinen und des eingesetzten Repräsentationssystems SB-ONE im speziellen behandelt. Als weiterer Aspekt wird auf die Struktur des gesamten Modells eingegangen, das neben den Annahmen und Handlungsmöglichkeiten des Benutzers auch Annahmen des Systems und ggf. Eigenschaften anderer Akteure enthalten kann. Wichtig ist dabei die Trennung zwischen Annahmen und Handlungsmöglichkeiten und die wechselseitigen Beziehungen der einzelnen Teile des Modells. Es folgt die Behandlung des Zusammenhangs dieses Modells mit der Aufgaben- und der Sachbereichsbeschreibung für das Planungssystem, die beide aus dem Benutzermodell gewonnen werden. Im letzten Teil des Kapitels wird noch erörtert, welchen Einfluß Hilfsmittel und andere Akteure auf die Handlungsmöglichkeiten des Benutzers haben.

Kapitel 4 widmet sich der benutzerspezifischen Präsentation eines Plans. Da Beschreibungen von Plänen stets abstrakt sind, wird zunächst erörtert, wie sich Pläne identifizieren lassen. Im Kontrast zu Beschreibungen von Gegenständen sind die Eigenschaften hier weniger wichtig. Stattdessen sind für Planbeschreibungen die Planschritte und deren gegenseitige Abhängigkeiten wichtig. Damit ein Benutzer auch entsprechend der Beschreibung handeln kann, muß er zudem wissen *wie* die Planschritte ausgeführt werden können. Es wird zunächst eine Modellierung solchen Wissens vorgeschlagen, die – vergleichbar zum Planen mit Operatorabstraktion – auf der Dekomposition der Schritte in Teilschritte beruht. Um ermitteln zu können, ob und ggf. welche zusätzlichen Informationen über die Ausführungsmöglichkeiten der Planschritte dem Benutzer gegeben werden müssen, muß zunächst beurteilt werden, ob sein Wissen ausreicht, um den Plan ausführen zu können. Hierbei spielt auch eine Rolle, ob der Benutzer die Ausführungsmöglichkeiten, die er für einen Planschritt *kennt*, auch tatsächlich *ausführen kann*. Parallel zur Ermittlung der Gesamtinformation, die dem Benutzer gegeben werden muß, wird eine Bewertung der Präsentation – i.w. anhand ihrer Länge – berechnet, die als Gütekriterium dient. Darüber hinaus werden alternative Bewertungskriterien vorgeschlagen und zum Vergleich Arbeiten zum Textverstehen aus der Psychologie angesprochen.

Informationen, die ein Benutzer selbst mit wenig Aufwand schließen kann, sollten nicht explizit präsentiert werden. Für die Frage, auf welche Teile der Gesamtinformation dies zutrifft, wird ein Entscheidungsverfahren vorgeschlagen, das sich am Benutzerwissen und einer Aufwand/Nutzen Abschätzung orientiert. Der Aufwand besteht für den Benutzer im Schlußfolgern der nicht explizit genannten Komponenten; der Nutzen in einer kürzeren Präsentation. Danach wird darauf eingegangen, wie der Planungsprozeß beeinflußt werden kann, um von vorn herein Pläne zu erzeugen, für die der Benutzer möglichst wenig zusätzliche Informationen benötigt und demnach zu einer guten Bewertung der Präsentation führen. Der letzte Teil dieses Kapitels widmet sich der konkreten Repräsentation des Wissens über die Ausführungsmöglichkeiten der Planschritte in der Benutzermodellierungswissensbasis.

Ziel des fünften Kapitels ist die Demonstration der bis dahin vorgestellten Ideen an einem Prototyp eines adaptiven Hilfesystems.

Zunächst wird die Architektur vorgestellt und ein Überblick über die Arbeitsweise des Systems

gegeben. Dazu kommt eine Beurteilung des Laufzeitverhaltens der neu entwickelten Algorithmen und die Begründung der Wahl des verwendeten Planungssystems bzw. Planungsverfahrens. Im nächsten Schritt wird die Modellierung des Beispiel-Sachbereichs gezeigt und dadurch demonstriert, wie die Vorschläge aus dem dritten Kapitel praktisch umgesetzt werden können. Das resultierende Modell enthält auch Stereotypes mit typischen Annahmen und Handlungsmöglichkeiten von beispielhaften Benutzergruppen, die im letzten Teil des Kapitels für die Anwendungsbeispiele eingesetzt werden. Es zeigt sich dabei, daß keine der Lösungsmöglichkeiten für die Beispielprobleme für alle fünf (hypothetischen) Beispielbenutzer geeignet ist und unterstreicht so die Notwendigkeit benutzerspezifischer Hilfe.

Das letzte Kapitel enthält eine Zusammenfassung der Ergebnisse dieser Arbeit. Da die Verarbeitung und insbesondere die Erzeugung benutzerspezifischer Pläne bisher praktisch nicht bearbeitet wurde, sind während der Arbeit viele Ideen zur weiteren Verbesserung, Erweiterung und auch zu Einsatzmöglichkeiten entstanden, die ebenfalls im letzten Teil der Arbeit kurz beschrieben werden.

Es bleibt anzumerken, daß der Anhang eine Zusammenfassung aller Syntaxdefinitionen, eine Beschreibung der Beispielmodellierung und Hinweise zum Betrieb des Prototyps enthält. Protokolle der Beispiele sind aus Platzgründen nur teilweise enthalten. Die vollständigen Protokolle finden sich ebenso wie der Programmquelltext des Prototyps im elektronischen Anhang.

Kapitel 2

Pläne in verwandten Forschungsgebieten

Für die Erzeugung einer Handlungsanweisung, mit der ein Computer-Benutzer eine gegebene Aufgabe bewältigen kann, müssen sowohl seine Kenntnisse als auch seine Handlungsmöglichkeiten berücksichtigt werden. Im Kapitel 3 *Modellierungs- und Plangenerierungsaspekte* wird beschrieben, welches Wissen dazu über den jeweiligen Benutzer erforderlich ist, wie es repräsentiert und organisiert und wie daraus ein Plan erzeugt werden kann, der Basis für eine solche Handlungsanweisung ist. Als Grundlage dafür soll hier zunächst untersucht werden, wie die Aspekte des Handelns in anderen Forschungsgebieten der künstlichen Intelligenz gesehen, repräsentiert und verarbeitet werden. Die Untersuchung orientiert sich eng am Zweck dieser Arbeit und hat deshalb keinesfalls den Anspruch, die Themen *Handeln* oder *Planen* erschöpfend darzustellen.

Ausgangspunkt ist das Gebiet der Handlungsplanung, für das umfangreiche Forschungsergebnisse für viele der damit zusammenhängenden Fragen vorliegen. Die dort entwickelten Modelle des Handelns und Planens sind Grundlage für viele weitere Forschungsgebiete. Bereiche mit engem Bezug zu dieser Arbeit, die teilweise auch das Handeln von Benutzern berücksichtigen, sind *Planerkennung*, *Planen für einen oder gemeinsam mit einem Benutzer*, *Dialog- und Textplanung*, *Hilfesysteme* sowie das *Planen der Beschaffung von Informationen*.

2.1 Handlungsplanung

Das Gebiet Handlungsplanung hat rund 30 Jahre intensiver Forschung hinter sich; die Anfänge sind noch einmal ca. 10 Jahre älter. Entsprechend groß ist das Gebiet. Hier soll nicht der Versuch einer umfassenden Beschreibung gemacht werden. Vielmehr sollen die wichtigsten Ideen und insbesondere die Modelle zur Beschreibung des Handelns vorgestellt werden, weil sie Grundlage für viele Arbeiten sind – insbesondere auch derjenigen, die in den nächsten Abschnitten vorgestellt werden.

Übersichten über die Ansätze und Methoden finden sich bspw. in [Yang, 1997], [Hertzberg, 1989] und den Kapiteln 11 bis 13 von [Russell und Norvig, 1995]. Die Entwicklung des Gebiets wird in [McDermott und Hendler, 1995] und [Tate *et al.*, 1990] beschrieben. [Tate *et al.*, 1990] ist im Einführungsteil von [Allen *et al.*, 1990] enthalten. Dieser Sammelband enthält eine Auswahl wichtiger Artikel, die Meilensteine der Entwicklung des Gebiets darstellen. [McDermott und Hendler, 1995] ist die Einführung zu einem Sonderheft der Zeitschrift *Artificial Intelligence*. Die darin enthaltenen Artikel geben ein Bild vom Stand der Forschung um 1995. Neuere Entwicklungen werden in [Weld, 1999] beschrieben.

Das Forschungsgebiet Handlungsplanung ist aus der Frage hervorgegangen, wie Menschen Probleme lösen. Der Artikel [Newell und Simon, 1961], der als Beschreibung des ersten Planungssystems *GPS* (*General Problem Solver*) angesehen wird, ist der psychologischen Forschung zuzuordnen. Den-

noch hat sich die Handlungsplanung selbst fast ausschließlich auf die *Erzeugung von Plänen für Roboter* konzentriert. Die grundlegenden Ideen zur Planung einer Handlung haben bis heute Bestand – wenngleich etwas modifiziert. McCarthy und Hayes schlagen in [McCarthy und Hayes, 1969] eine Formalisierung in Prädikatenlogik erster Stufe vor, die *Situationskalkül* (*situation calculus*) genannt wird. Ausgangspunkt einer Planerzeugung, bzw. kurz Planung, ist eine Beschreibung des Zustands (bzw. der *Situation*) in der sich die Welt *vor* Ausführung des Plans befindet (die *Startsituation*) und ein oder mehrere Ziele, die durch Bedingungen an einen *Endzustand* bzw. eine *Zielsituation* beschrieben werden. Ein Akteur hat die Aufgabe solche Handlungen bzw. Aktionen durchzuführen, die die Startsituation so *verändern*, daß die Bedingungen der Zielsituation erfüllt werden. Die Lösung eines Planungsproblems, oder kurz ein Plan, ist gerade eine solche Folge von Aktionen, mit der eine Startsituation in eine Zielsituation überführt werden kann. Für die Erzeugung eines Plans muß bekannt sein, welche Aktionen der Akteur mit welchem Ergebnis ausführen kann; d.h. welche *Veränderungen* einer Situation möglich sind. Dabei sind an die Ausführung einer Aktion oft sogenannte *Vorbedingungen* geknüpft, die unmittelbar vor Ausführung der Aktion erfüllt sein müssen.

Für die Beschreibung einer einzelnen Situation in Prädikatenlogik wird jedem (relevanten) Objekt ein Konstantensymbol zugeordnet. Die (relevanten) Merkmale der Objekte werden durch Prädikate modelliert. Dabei beschreibt ein solches Prädikat gerade die Menge der Objekte, die das Merkmal haben. Eine *Situation* kann dann durch eine Konjunktion aller negierten und nichtnegierten atomaren Aussagen modelliert werden, die in dieser Situation wahr sind. Wird die *Closed-World* Annahme (vgl. [Reiter, 1980]) vereinbart, gilt für beliebige Prädikate P und Argumente $x_1, \ldots, x_n$

$\neg P(x_1, \ldots, x_n)$ falls sich $P(x_1, \ldots, x_n)$ *nicht* ableiten läßt.

Die Beschreibung einer Situation kann sich dadurch erheblich verkleinern, weil negierte Aussagen nicht explizit notiert werden müssen. Die *Closed-World* Annahme wird für die meisten Planungssysteme und auch für diese Arbeit vereinbart. Eine weitere Verkleinerung des Beschreibungsaufwands kann sich durch allgemeingültige Formeln oder Axiome ergeben, mit denen sich die Gültigkeit von Formeln aus anderen ableiten läßt. Ist bspw. ein zweistelliges Prädikat P symmetrisch, so kann der Aufwand für die Beschreibung der entsprechenden Prädikat-Argument-Kombinationen halbiert werden, wenn die Symmetrie durch das Axiom $\forall x, y\ P(x, y) \rightarrow P(y, x)$ explizit gemacht wird.

Veränderungen werden durch den *Unterschied* zweier Situationen modelliert. Um mehrere Situationen in einem prädikatenlogischen Modell verarbeiten zu können, wird die Gültigkeit von Prädikaten auf eine Situation eingeschränkt, indem jedes Prädikat um eine Argumentstelle für die Situation erweitert wird. Bspw. kann das Merkmal, daß ein Objekt *Obj* in Situation *Sit rot* ist, durch das Prädikat *rot*(*Obj*, *Sit*) ausgedrückt werden. Eine Veränderung führt grundsätzlich zu einer neuen Situation, bspw. Sit'. Da die Gültigkeit von Merkmalen auf eine Situation bezogen ist, können die Merkmale der neuen Situation durchaus im Widerspruch zu der Beschreibung der alten Situation stehen. Bspw. kann eine logisch konsistente Beschreibung sowohl *rot*(*Obj*, *Sit*) als auch *grün*(*Obj*, Sit') enthalten, auch wenn etwa durch Axiome wie $\forall o,s\ rot(o,s) \rightarrow \neg grün(o,s)$ gefordert wird, daß ein rotes Objekt niemals grün ist.

Da eine Aktion eine *Veränderung* bewirkt, erfordert deren Modellierung auch eine Funktion, die eine Situation in eine Folgesituation abbildet. Die Veränderung, die durch eine Aktion A hervorgerufen wird, kann durch ein Axiom nach folgendem Schema

$$\forall x_1, \ldots, x_n, s \quad V(x_1, \ldots, x_n, s) \rightarrow E(x_1, \ldots, x_n, A(x_1, \ldots, x_n, s))$$

modelliert werden. Formal steht $V(x_1, \ldots, x_n, s)$ für eine prädikatenlogische Formel mit $x_1, \ldots, x_n, s$ als freie Variablen und $E(x_1, \ldots, x_n, A(x_1, \ldots, x_n, s))$ für eine prädikatenlogische Formel, die n freie Variablen $x_1, \ldots, x_n$ und eine Funktion A mit den Argumenten $x_1, \ldots, x_n, s$ enthält. Formel V modelliert die

Vorbedingung die vor der Ausführung der Aktion erfüllt sein muß und Formel E den Effekt der in der Situation danach gilt. Diese Situation wird durch die Funktion A bestimmt. Da die Variablen $x_1, \ldots, x_n$ und s für *beliebige* Objekte bzw. Situationen stehen, wird durch ein Axiom nach diesem Schema keine konkrete Aktion, sondern eine Menge von Aktionen beschrieben. Solche Axiome werden deshalb *Aktions-Schemata* oder *Operatoren* genannt.

Ebenso, wie für die Beschreibung einer Situation nur die *relevanten* Objekte herangezogen werden und davon wiederum nur die *relevanten* Merkmale, ist auch die Beschreibung einer Vorbedingung eines Operators auf die *relevanten* Sachverhalte beschränkt. In allen Fällen entscheidet der Entwickler des Sachbereichsmodells darüber *was* relevant ist. Diese Entscheidung ist unumgänglich, weil – zumindest für reale Objekte – niemals alle Eigenschaften bestimmt werden können und i.a. auch die Zahl der Einzel-Bedingungen, aus der sich die Vorbedingung einer Aktion zusammensetzt, praktisch unbeschränkt ist. Die Problematik, daß nicht alle Einzel-Bedingungen an das Ausführen einer Aktion berücksichtigt werden können und daß die Aktion deshalb im ungünstigen Fall scheitern kann, obwohl alle modellierten Bedingungen erfüllt sind, wird das *Qualifikationsproblem* (*Qualification Problem*) genannt [McCarthy, 1977]. Da dieses Problem prinzipiell unlösbar ist (vgl. bspw. [Hertzberg, 1989, S.28]), muß sich die Entscheidung über die Genauigkeit der Modellierung daran orientieren, daß die Aktionen *normalerweise* ausführbar sind, wenn ihre modellierten Vorbedingungen erfüllt sind.

Um Aussagen über die Situation treffen zu können, die über den Effekt eines Operators hinausgehen, muß auch berücksichtigt werden, was ein Operator *nicht* verändert. Die Problematik der Berücksichtigung der unveränderlichen Gegebenheiten bzw. des Rahmens der Operatoren wird *Rahmen-Problem* (*Frame Problem*) genannt[1]. Dazu ist für jeden Operator und jedes Prädikat, das durch den Operator nicht berührt wird, ein sogenanntes *Rahmen-Axiom* erforderlich. Dieses besagt, daß ein solches Prädikat in der Situation nach Ausführung der entsprechenden Aktion für alle Objekte gilt, für die es in der Situation davor galt. Nun kann die Existenz eines Plans auf die Ableitbarkeit einer Situation in der die Ziel-Bedingungen gelten zurückgeführt werden. Das *Protokoll des Ableitungsweges* enthält eine Folge der verwendeten Operatoren und stellt damit eine Beschreibung des Plans dar. Durch die Rahmen-Axiome wird die an sich elegante Modellierung der Veränderbarkeit allerdings sehr aufwendig und unübersichtlich. Kowalski hat in [Kowalski, 1979] einen Ansatz vorgeschlagen, der mit deutlich weniger Rahmen-Axiomen auskommt. Dieser verringert aber lediglich den Aufwand für die *Modellierung*, nicht für die *Erzeugung* eines Plans. Ein Theorem-Beweiser, der diese Aufgabe durchführen soll, kann durch die Rahmen-Axiome sehr viele Schlüsse ziehen, die für die eigentliche Aufgabe irrelevant sind und dementsprechend sehr viel Rechenzeit unnötig verbrauchen.

Ein Durchbruch in der Handlungsplanung gelang Fikes und Nilsson, indem sie die Vorteile von GPS und dem Situationskalkül verbanden. Im Gegensatz zum Situationskalkül verzichtete ihr *Stanford Research Institute Problem Solver* (*STRIPS*) [Fikes und Nilsson, 1971] darauf die Gültigkeit der einzelnen Prädikate durch eine logische Schlußfolgerung abzuleiten. Stattdessen versucht STRIPS die Repräsentation der Startsituation so zu verändern – d.h. praktisch zu editieren –, daß die Bedingungen der Zielsituation erfüllt werden. Die Veränderungsmöglichkeiten werden wie im Situationskalkül durch die Operatoren beschrieben. Da zu einem Verarbeitungszeitpunkt immer nur eine Situation betrachtet werden muß, ist keine Funktion für die Abbildung der Situation in eine Nachfolgesituation erforderlich. Der entscheidende Vorteil liegt in der eleganten Lösung des Rahmen-Problems. Da nur solche Veränderungen an einer Situation vorgenommen werden, die durch einen Operator festgelegt sind, bleiben alle anderen Merkmale erhalten, ohne daß es dazu spezieller Maßnahmen bedarf. Diese Idee, der die sogenannte *STRIPS Annahme*[2] zugrunde liegt, nutzen bis heute beinahe alle Planungssysteme.

[1] Eine ausführliche Behandlung dieses und weiterer Probleme findet sich in [McCarthy und Hayes, 1969].

[2] vgl. Abschnitt 2.1.3.

Im STRIPS-System besteht eine Operatorbeschreibung aus:

- der *Vorbedingung* (*precondition*) – einer prädikatenlogischen Formel
- der *Hinzu-Liste* (*add list*) – einer Menge prädikatenlogischer Formeln
- der *Lösch-Liste* (*delete list*) – einer Menge prädikatenlogischer Formeln

Dazu kommen Parameter; d.h. Variablen, die in der Beschreibung frei vorkommen und im Planungsprozeß bei der Verwendung eines Operators gebunden werden.

Ein Operator ist in einer Situation anwendbar, in der seine Vorbedingung erfüllt werden kann. Der Effekt eines Operators wird durch die *Hinzu-Liste* und die *Lösch-Liste* beschrieben, die die Formeln enthalten, die durch den Operator wahr werden bzw. nicht mehr wahr sind und deshalb gelöscht werden müssen. Die Idee ist, daß die Wirkung eines Operators allein dadurch beschrieben werden kann, daß aus der Beschreibung einer Situation die Formeln der *Lösch-Liste* gelöscht werden und die der *Hinzu-Liste* hinzugefügt werden. Ohne Einschränkungen für die Beschreibung der Situationen und Operatoren ist es aber (u.a.) wegen der Semi-Entscheidbarkeit der Prädikatenlogik gar nicht immer möglich tatsächlich alle Formeln, die durch einen Operator nicht mehr wahr sind, in der Lösch-Liste aufzuführen. Lifschitz untersuchte die Problematik in [Lifschitz, 1986] und machte Vorschläge für die Definition *korrekter* STRIPS-Systeme. Dabei müssen die Beschreibungen von Startsituation und Operatoren bestimmte Bedingungen einhalten. Ohne auf Details einzugehen, soll hier direkt auf heute gängige Beschreibungen von Planungsproblemen eingegangen werden, die die Konstruktion korrekter und vollständiger Planungsverfahren erlauben.

Die Beschreibungen lehnen sich nach wie vor an den STRIPS-Formalismus an. Sie sind meist bzgl. der Startsituation und aller drei Komponenten eines Operators auf eine Konjunktion atomarer Formeln eingeschränkt, wobei außer den Operatorparametern keine Variablen vorkommen dürfen. Für die Zielsituation ist eine Konjunktion von Literalen zulässig, deren Variablen (meist implizit) existentiell quantifiziert sind (vgl. [Russell und Norvig, 1995], [Yang, 1997], [Weld, 1999]). Mit Ausnahme von Konstantensymbolen sind Funktionssymbole bei der Beschreibung von Planungsproblemen grundsätzlich unzulässig.

Als Referenz für heute gängige Beschreibungen sei die *Planning Domain Definition Language* (PDDL) [Ghallab *et al.*, 1998] angegeben, die für den Wettbewerb von Planungssystemen auf der AIPS'98 [Long, 2000], [McDermott, 2000] entworfen wurde. Die Sprache basiert auf der Prädikatenlogik erster Stufe; d.h. für die Beschreibungen der Operatoren sind auch Quantoren zulässig. Es sei angemerkt, daß dabei von einem *statischen Sachbereich* (*Static Universe*) ausgegangen wird; d.h. durch Angabe der (endlich vielen) Konstanten sind alle im Sachbereich vorkommenden Objekte bekannt. Damit kann auch die Wahrheit einer allquantifizierten Formel grundsätzlich durch Aufzählung festgestellt werden[3].

Ein wichtiger Unterschied zur STRIPS-Beschreibung liegt in der Trennung von veränderlichen und unveränderlichen Sachverhalten. Die Startsituation des Beispiels aus [Fikes und Nilsson, 1971] enthielt sowohl veränderliche Aussagen zu einzelnen Merkmalen, u.a. daß ein Objekt O an Ort L ist, als auch die allgemeingültige Formel, daß eine Tür die Raum A mit Raum B verbindet auch Raum B mit Raum A verbindet. PDDL läßt für die Beschreibung der Startsituation selbst nur die Aufzählung der Merkmale die in der Startsituation gelten als Menge von atomaren Formeln zu[4]. Die Formeln sind implizit konjunktiv verknüpft. Allgemeingültige Formeln, die in jeder Situation wahr sind, gehören zum *Planungssachbereich*. Sie werden *Axiome* genannt und müssen die Form

$$\forall x_1, \ldots, x_n \quad G \rightarrow D$$

[3] vgl. [Weld, 1994] für eine ausführliche Darstellung.

[4] Negierte atomare Formeln sind ebenfalls zulässig, aber ohne Belang, wenn die Closed-World Annahme vereinbart ist.

haben, wobei G eine prädikatenlogische Formel und D eine atomare Formel ist. Die Formeln dürfen keine freien Variablen enthalten. Das Prädikat einer Konklusion wird auch *abgeleitetes Prädikat* genannt und darf nicht in einem Effekt eines Operators auftreten[5].

Die Beschreibung der Operatoren gehört in PDDL auch zum Planungssachbereich. Eine PDDL-Operatorbeschreibung[6] enthält wie bei STRIPS Parameter sowie eine Vorbedingung, die durch eine prädikatenlogische Formel beschrieben wird. Hinzu- und Lösch-Liste der STRIPS-Operatoren sind zum Effekt zusammengefaßt, der durch eine Konjunktion von Literalen beschrieben wird. Dabei hat eine negierte Atomformel die gleiche Wirkung wie die zugehörige nicht-negierte Formel in der Lösch-Liste eines Operators nach der ursprünglichen STRIPS-Definition; d.h. die Ausführung des Operators bewirkt, daß diese Formel nicht mehr wahr ist. Gegenüber der ursprünglichen STRIPS-Definition ist die Ausdrucksstärke eines Effekts damit stark reduziert, um effiziente und korrekte Planungsalgorithmen zu ermöglichen. Es sei angemerkt, daß manche Autoren anstelle von *einem* Effekt von *mehreren* Effekten eines Operators ausgehen. Damit ist die Menge der Literale gemeint, deren Konjunktion den (Gesamt-)Effekt bilden. Entsprechendes gilt auch für die Vorbedingung, falls ein Planungsformalismus auch dafür lediglich eine Konjunktion von Literalen zuläßt. Die einzelnen Literale werden in dieser Arbeit *Effekt-* bzw. *Vorbedingungskomponenten* genannt. Für die Beschreibung der Zielsituation ist wie bei STRIPS eine prädikatenlogische Formel ohne Funktionssymbole zulässig.

Aus heutiger Sicht ergibt sich damit für das (klassische) Handlungsplanen folgendes Bild:

Ein Planungsproblem wird durch eine *Startsituation* und *Bedingungen an eine Zielsituation* beschrieben. Das Planungsproblem bezieht sich dabei immer auf einen *Planungssachbereich*. Der Planungssachbereich beschreibt die Abhängigkeiten und Veränderungsmöglichkeiten des modellierten Weltausschnitts in Form *situationsunabhängiger Axiome* bzw. durch *Planoperatoren*. Die *Lösung eines Planungsproblems* – der *Plan* – ist eine Folge von Exemplaren von Planoperatoren – den *Planschritten* –, die angewendet auf die Startsituation zu einer Situation führen, in der die Bedingungen an die Zielsituation erfüllt sind.

2.1.1 Least Commitment Planning

Oft ist es nicht notwendig eine strikte Folge von Planschritten festzulegen, weil die geforderten Ziele durch die Ausführung der Planschritte in mehreren alternativen Reihenfolgen erreicht werden kann. Lösung eines Planungsproblems kann dann auch ein *partiell geordneter Plan* sein, der aus einer lediglich partiell geordneten Menge von Planschritten besteht. Der Akteur, der den Plan ausführt, kann dann selbst bestimmen in welcher Reihenfolge er die Planschritte ausführt.

Die Idee, nur solche Entscheidungen zu treffen die nicht verschoben werden können, kann auf den gesamten Planungsprozeß übertragen werden. Der Vorteil dieses sogenannten *Least Commitment Planning* besteht darin, die Zahl der Entscheidungen, die sich im weiteren Verlauf des Planungsprozesses als falsch herausstellen können, zu verringern. Damit ergibt sich eine veränderte Sicht auf den Planungsprozeß. Die Betrachtung einzelner *Situationen* tritt völlig in den Hintergrund. Stattdessen wird von einem *unvollständigen* bzw. *partiellen Plan* ausgegangen (dem *Startplan* (s.u.)) und dieser solange ergänzt, bis er vollständig und konsistent ist und damit die geforderten Ziele erreicht werden können[7]. Es sei auf den Unterschied zwischen einem *partiell geordneten* (aber vollständigen) und einem *partiellen* Plan hingewiesen.

[5] In Kapitel 3.4.2.3 *Nichtatomare Formeln im Planungssachbereich* wird auf die Gründe eingegangen.

[6] hier: Beschreibungen *einfacher* PDDL-Operatoren. Erweiterungen werden noch in Kap. 2.1.2.2 *Operatorabstraktion* behandelt.

[7] Im Bereich Handlungsplanung spricht man von einem *Plan-Suchraum* (*plan space*) im Gegensatz zu einem *Situations-Suchraum* (*situation space*) bei STRIPS.

Für die Beschreibung eines dafür geeigneten Plans sei als Vereinfachung vereinbart, daß auf Axiome verzichtet wird und die Vorbedingungen von Operatoren – ebenso wie die Effekte – Konjunktionen von Literalen (den Vorbedingungs- bzw. Effektkomponenten) sind. Ein (evtl. unvollständiger bzw. partieller) Plan besteht dann aus folgenden Komponenten (vgl. auch [Russell und Norvig, 1995, S.347]):

- einer Menge von Planschritten, die Exemplare von Planoperatoren sind.
- einer Menge von Reihenfolge- oder Ordnungs-Bedingungen für die Planschritte.
- einer Menge von (positiven oder negativen) Bedingungen für die Bindung von Variablen
- einer Menge von sogenannten *Causal Links*.

Ein *Causal Link* verbindet eine Effektkomponente *EL* eines Planschritts *E* mit einer Vorbedingungskomponente *VL* eines Planschritts *V*. Dabei müssen die Literale *EL* und *VL* unter Berücksichtigung der Bindungsbedingungen logisch äquivalent sein. Planschritt *E* wird oft Erzeuger und *V* Verbraucher des Literals *VL* genannt. In der Literatur findet man auch die Sprechweise *der Causal Link verbindet E mit V bezüglich Literal VL*.

Ein solcher partieller Plan kann als Beschreibung der Menge aller Linearisierungen des Plans aufgefaßt werden, die sich gemäß der Reihenfolgebedingungen bilden lassen. Eine *Linearisierung* eines partiellen Plans *p* ist ein Plan, der sich von *p* höchstens durch seine Reihenfolgebedingungen unterscheidet, wobei diese gegenüber den Reihenfolgebedingungen von *p* so *erweitert* sind, daß für die Planschritte nur genau eine strikte Folge zulässig ist. Werden die Reihenfolgebedingungen als Relation über den Planschritten aufgefaßt, so ist der transitive Abschluß der Reihenfolgebedingungen einer Linearisierung eines partiellen Plans *p* eine totale Ordnung, die die Reihenfolgebedingung von *p* enthält.

Um einen unvollständigen Plan zu vervollständigen, kann das Planungssystem dem Plan neue Planschritte, Causal Links sowie Bindungs- oder Ordnungs-Bedingungen hinzufügen. Dabei darf die Konsistenz des Plans nicht verletzt werden. Vollständigkeit und Konsistenz eines Plans seien dabei folgendermaßen definiert:

- Ein Plan ist *vollständig*, wenn jede Vorbedingungskomponente jedes Planschritts durch einen Causal Link mit einer Effektkomponente eines anderen, davor liegenden Planschritts verbunden ist und der Causal Link nicht bedroht ist. Eine Bedrohung (*threat*) eines Causal Links liegt vor, wenn ein Planschritt zwischen die beiden Planschritte des Causal Links angeordnet werden darf, der eine Effektkomponente besitzt, die dem Literal des Causal Links widerspricht, bzw. bei einer zulässigen Variablenbindung widersprechen kann.
- Ein Plan ist *konsistent*, wenn die Bindungs- und Ordnungs-Bedingungen nicht widersprüchlich sind.

Um wie bisher Planungsprobleme zu lösen, die durch Startsituation und Zielbedingung beschrieben sind, wird in einem Initialschritt ein *Startplan* erzeugt, der zwei Planschritte, die ausnahmsweise nicht Exemplare eines Operators sind, und eine Ordnungsbedingung enthält:

- Ein Planschritt heißt *Start*. Er hat keine Vorbedingung. Der Effekt ist gleich der Beschreibung der Startsituation[8].

[8] Zur Berücksichtigung der Closed-World Annahme – insbesondere für Causal Links – gehören zum Effekt auch alle Literale, die sich gemäß Closed-World Annahme ergeben. Sie werden bei einem Planungsprozeß aber meist nicht alle explizit repräsentiert, sondern erst bei Bedarf; d.h. wenn ein solches Literal für die Vorbedingung eines Operators benötigt wird.

- Ein zweiter Planschritt heißt *Ziel*. Seine Vorbedingung ist gleich der Zielbedingung und er hat keinen Effekt.
- Die Ordnungsbedingung lautet: *Start* vor *Ziel*.

Ein Planungsprozeß nach diesem Schema muß nicht unbedingt mit einem Initialplan beginnen, der aus Startsituation und Zielbedingung abgeleitet wurde. Vielmehr kann er auf einem beliebigen partiellen Plan aufsetzen. Umgekehrt kann auch das Ergebnis des Planungsprozesses für eine erweiterte Planungsaufgabe genutzt werden.

Obwohl die Idee des Zurückstellens von Entscheidungen nicht nur die Reihenfolge der Planschritte, sondern zumindest auch die Bindungsbedingungen für Variablen betrifft, werden solche Planungssysteme bzw. -verfahren meist *Partial Order Planner* (POP) bzw. *Partial Order Planning* genannt. NOAH [Sacerdoti, 1977] und NONLIN [Tate, 1976], die ersten Planungssysteme die nach diesem Verfahren arbeiteten, wurden *nichtlineare* Planer genannt, weil sie Pläne verarbeiten konnten, die nicht strikt geordnet (linear) waren. Heutige *Partial Order* Planungssysteme basieren zumeist auf dem SNLP-Algorithmus [McAllester und Rosenblitt, 1991]. Dieser einfache, systematische Algorithmus eignet sich sehr gut für Erweiterungen, die auch komplexe Operatorbeschreibungen und Axiome verarbeiten können[9].

2.1.2 Abstraktion – Hierarchisches Planen

Auch die Ideen zum *abstrakten Planen* haben die Vorstellung über die Repräsentation von Plänen und Planoperatoren beeinflußt. Abstraktion, also das Absehen von unwichtigen Details, wurde und wird als Möglichkeit gesehen der Komplexität sachbereichsunabhängiger Plangenerierung zu begegnen. Dabei wird das Planungsproblem zunächst auf einer abstrakten Ebene gelöst und diese Lösung dann schrittweise verfeinert, bis eine Folge[10] von elementaren Planschritten vorliegt, die alle Details abdeckt. Bei der sogenannten *Situationsabstraktion* ignoriert der Planungsprozeß auf den abstrakten Ebenen unwichtige Merkmale der Situationen (vgl. [Hertzberg, 1989]). Eine zweite Variante von Abstraktion wird *Operatorabstraktion* genannt. Dabei wird von abstrakten Operatoren ausgegangen, die abstrakte Effekte haben und nicht direkt ausgeführt werden können. Ein damit erzeugter Plan muß konkretisiert werden, indem die abstrakten Planschritte schrittweise in ihre Bestandteile zerlegt werden, bis der Plan nur noch aus Planschritten besteht, die der Akteur direkt ausführen kann.

2.1.2.1 Situationsabstraktion

Das erste Planungssystem das *Situationsabstraktion* verwendete war das auf STRIPS basierende ABSTRIPS[11] [Sacerdoti, 1974]. Die Beschreibung eines ABSTRIPS-Planungssachbereichs unterteilt die Vorbedingungen eines Operators in verschieden wichtige Ebenen ein. Das Planungssystem berücksichtigt in der ersten Phase nur die Vorbedingungen der wichtigsten Ebene – abstrahiert also von unwichtigen Merkmalen einer Situation. Danach wird der Plan schrittweise ergänzt, indem auch die Vorbedingungen weniger wichtiger Ebenen berücksichtigt werden. Die Abstraktion bewirkt damit praktisch die Beseitigung von (Teilen von) Operator-Vorbedingungen und wird auch *Precondition Elimination* genannt. Tenenberg untersucht in [Tenenberg, 1991] verschiedene Varianten dieses Ansatzes. Für seine Untersuchungen definiert er abstrakte STRIPS-Systeme. Als Basis dienen STRIPS-Systeme, die gegenüber der ursprünglichen Beschreibung in [Fikes und Nilsson, 1971] eingeschränkt

[9] Eine ausführliche Beschreibung findet sich in [Weld, 1994].

[10] Prinzipiell sind auch – wie beim PO-Planen – partiell geordnete Mengen möglich. Der Begriff *Folge* soll deshalb im weiteren auch in diesem Sinne verstanden werden.

[11] Abstraction-Based STRIPS

sind. Die wichtigsten Unterschiede sind, daß Vorbedingung, Hinzu- und Lösch-Liste jeweils Mengen von (implizit konjunktiv verknüpften) atomaren Formeln sind und daß alle Axiome Horn-Formeln sein müssen. Der Planungsprozeß kann auf abstrakten Ebenen zu inkonsistenten Situationen führen, wenn sich die Abstraktion nur auf die Vorbedingung von Operatoren beschränkt. Tenenberg schlägt deshalb zwei Alternativen vor die inkonsistente Situationen verhindern und weist dafür Eigenschaften der Lösungssuche nach. Beide Alternativen sehen vor, daß sich die Abstraktion nicht nur auf Vorbedingungen bezieht, sondern generell die modellierten Merkmale des Sachbereichs betrifft. Die Abstraktion wird für die dazugehörigen Prädikate definiert und auf Vorbedingungen *und* Effekte von Operatoren sowie auf Axiome und Start- und Zielsituation übertragen.

In Tenenbergs erstem Vorschlag, den er *Inheritance Abstraction* nennt, besteht die Abstraktion in einer Abbildung von Prädikaten auf allgemeinere Prädikate. Dabei ist ein Prädikat A allgemeiner als ein Prädikat S, wenn

$$\forall x_1, \ldots, x_n \quad S(x_1, \ldots, x_n) \rightarrow A(x_1, \ldots, x_n).$$

Da die Objekte des modellierten Sachbereichs durch Konstantensymbole bezeichnet werden, repräsentieren einstellige Prädikate Objekttypen und mehrstellige die Beziehungstypen des Sachbereichs. Eine daraus resultierende Abstraktionshierarchie von Objekttypen entspricht damit einer Begriffstaxonomie, wie sie bspw. in Benutzermodellierungssystemen zur Organisation des Benutzerwissens eingesetzt werden ([Sleeman, 1985], [Sarner und Carberry, 1992], [Kobsa *et al.*, 1994]). Die Verwendung allgemeiner Prädikate in Operatoren spezifiziert die zu verwendenden Objekte weniger genau als speziellere, läßt also Freiheitsgrade offen. Tenenberg gibt als Beispiel an, daß bei der Zubereitung einer Mahlzeit nicht genau festgelegt wird mit welchem Werkzeug (bspw. Messer, Kartoffelstampfer) eine Zutat zerkleinert wird, wobei je nach speziellem Typ der Zutat (bspw. Fleisch, Kartoffeln) unterschiedliche Werkzeuge sinnvoll sein können. Zu einer Abstraktion von Objekttypen ist eine Abstraktion von Operatoren definiert, die jedes Prädikat, das in der Vorbedingung oder dem Effekt vorkommt, auf seine Abstraktion abbildet. Zusätzlich wird gefordert, daß es für *jede* Kombination von Spezialisierungen jeder Vorbedingungskomponente eines abstrakten Operators auch einen spezielleren Operator geben muß, dessen Vorbedingung sich aus der Konjunktion einer dieser Kombination ergibt. Diese Bedingung ist allerdings so restriktiv, daß Tenenberg selbst sagt, daß sie in der Praxis selten anwendbar sein wird. Im Beispiel der Zerkleinerung von Lebensmitteln müßte es damit einen (eher realitätsfernen) Operator zur Zerkleinerung von Fleisch mit einem Kartoffelstampfer geben, wenn der abstrakte Operator als Vorbedingung eine Zutat und ein Werkzeug (jeweils mit den o.g. Spezialisierungen) verlangt.

Für ein dementsprechend definiertes abstraktes STRIPS-System kann Tenenberg die *Downward Solution Property* nachweisen, die besagt, daß die Existenz eines abstrakten Plans – also einer abstrakten Lösung eines Planungsproblems – die Existenz mindestens einer Spezialisierung dieses Plans impliziert. Unter Spezialisierung des Plans wird hier verstanden, daß für jeden Planschritt, der Exemplar eines abstrakten Operators ist, ein Exemplar einer geeigneten Spezialisierung des Operators eingesetzt wird, sodaß der speziellere Plan eine Zielsituation erreicht, die eine Spezialisierung der abstrakten Zielsituation ist. Dies muß allerdings nicht die ursprüngliche Zielsituation sein und es ist auch möglich, daß diese mit keiner Spezialisierung der abstrakten Lösung erreichbar ist. Der Vorteil durch die Abstraktion besteht bei diesem Vorgehen nicht darin, daß die abstrakten Pläne kürzer als die speziellen sind, sondern lediglich darin, daß die abstrakten Pläne weniger genau festgelegt sind. Für die Erzeugung eines abstrakten Plans sind deshalb weniger Entscheidungen erforderlich und es kann auch erwartet werden, daß damit weniger Entscheidungen revidiert werden müssen.

Die ebenfalls wünschenswerte *Upward Solution Property*, die besagt, daß es zu jeder speziellen Lösung auch eine abstrakte gibt, gilt hier nicht. Scheitert also die Planerzeugung auf einer abstrakten

Ebene, so muß die Planerzeugung auf einer spezielleren Ebene wiederholt werden, wenn die Vollständigkeit der Lösungssuche nicht in Frage gestellt werden soll.

Tenenberg vergleicht diese Sicht der Abstraktion mit dem Ansatz von Kautz [Kautz, 1991] für die Planerkennung. Darauf wird noch im Kapitel 2.2 *Planerkennung* eingegangen.

Tenenbergs zweiter Vorschlag, den er mit *Abstraction Using Relaxed Models*[12] überschreibt, kommt der ursprünglichen ABSTRIPS-Idee näher, indem auf höheren Abstraktionsebenen die unwichtigen Prädikate weggelassen werden. Es wird nachgewiesen, daß entsprechende Planungssysteme zu widersprüchlichen Situationen führen können, wenn die Axiome des Planungssachbereichs Prädikate unterschiedlicher Abstraktionsebenen verbinden. Um dies zu verhindern definiert er eingeschränkte ABSTRIPS-Systeme, die entsprechende Bedingungen an die Zuordnung von Prädikaten zu Abstraktionsebenen stellen.

Im Gegensatz zu *Inheritance Abstraction* muß eine Planerzeugung nach diesem Modell in einer abstrakten Ebene weniger Bedingungen berücksichtigen und führt deshalb i.a. auch zu kürzeren Plänen. Allerdings gilt hier nicht die *Downward Solution Property*, sodaß die Existenz einer abstrakten Lösung keine konkrete Lösung garantiert. Stattdessen läßt sich hier die *Upward Solution Property* nachweisen; d.h. läßt sich keine abstrakte Lösung finden, kann die Lösungssuche abgebrochen werden, ohne deren Vollständigkeit zu beeinträchtigen.

Eine signifikante Effizienzsteigerung des Planungsprozesses ist allerdings nur dann zu erwarten, wenn es selten vorkommt, daß eine abstrakte Lösung wieder verworfen werden muß. Im Idealfall wird ein abstrakter Plan lediglich in jeder Ebene verfeinert; d.h. die Planerzeugung einer Abstraktionsebene übernimmt jeweils die Ergebnisse aus der vorangegangenen nächst abstrakteren Ebene und bearbeitet lediglich die zusätzlichen Bedingungen der aktuellen Ebene. Die Möglichkeit eines solchen Vorgehens wird *Downward Refinement Property*[13] genannt. [Knoblock *et al.*, 1991] und [Bacchus und Yang, 1994] untersuchen verschiedene Eigenschaften von Abstraktionshierarchien und die Auswirkungen auf die Effizienz von Planungsprozessen. [Knoblock, 1994] und [Bacchus und Yang, 1994] stellen Verfahren für die *Erzeugung von Abstraktionshierarchien* vor, die diese Eigenschaften (teilw. angenähert) garantieren. Allerdings ist es nicht immer möglich eine *gute* Abstraktionshierarchie zu erzeugen. [Bäckström und Jonsson, 1995] zeigen sogar, daß der Gesamtprozeß aus Ermittlung einer Abstraktionshierarchie und Generierung eines Plans im ungünstigen Fall zu einer drastischen Verschlechterung der Laufzeit gegenüber einer Plangenerierung ohne Abstraktion führen kann.

2.1.2.2 Operatorabstraktion

Operatorabstraktion beruht auf der Beobachtung, daß es oft einfacher ist, zunächst einen gröberen (abstrakten) Plan zu erzeugen bei dem (abstrakte) Planschritte große Teile des Ziels in einem Schritt erreichen, als von vorn herein jedes Detail des Ziels durch einzelne Planschritte zu berücksichtigen. Ein solcher abstrakter Plan ist zwangsläufig kürzer und erfordert damit weniger Berechnungsaufwand. Ist der Plan nicht für einen Roboter, sondern für einen Menschen vorgesehen kommt hinzu, daß eine abstraktere Darstellung meist leichter verstanden werden kann und auf jeden Fall für eine Übersicht hilfreich ist. Als Beispiel wird oft der Bau eines Hauses genannt, der aus einer sehr großen Menge von Einzelaktionen besteht, die kaum sinnvoll als *ein* Plan dargestellt werden kann. Stattdessen ist es üblich mit einem groben (Standard-) Plan zu beginnen; bspw. *Baugrund vorbereiten*, *Fundament erstellen*, *Stockwerke bauen*, *Dach aufsetzen*, *Innenausbau*. Die Planschritte eines abstrakten Plans können allerdings nicht direkt ausgeführt werden. Vielmehr können die Teilaufgaben wiederum als

[12] [Knoblock, 1994] und [Yang, 1997] bezeichnen Abstraktion, die sich nur auf die Vorbedingungen von Operatoren beschränkt als *Relaxed Model*. *Reduced Model Abstraction* ist die Bezeichnung, falls auch Effekte mit einbezogen werden.

[13] Die Eigenschaft ist eng verbunden mit der *Downward Solution Property*. Allerdings garantiert eine abstrakte Lösung die Existenz eines konkreten Plan für das *ursprüngliche* Ziel, wenn die *Downward Refinement Property* gilt.

Planungsprobleme aufgefaßt werden, für die nun jeweils ein konkreterer Plan erzeugt werden muß. Die Teilpläne müssen dann wieder zu einem Gesamtplan zusammengefügt werden, wobei eventuelle Abhängigkeiten oder Unverträglichkeiten zwischen den Bestandteilen berücksichtigt und ggf. aufgelöst werden müssen. Je nach Komplexität des Planungsproblems sind für den Übergang von einem abstrakten zu einem konkreten Plan mehrere Schritte erforderlich. Die Planschritte eines endgültigen, konkreten Plans sind direkt ausführbar und werden – ebenso wie die Operatoren aus denen sie abgeleitet wurden – *primitiv* genannt. Ein solches Verfahren ist vergleichbar zum *Divide-and-Conquer* Ansatz bei der Entwicklung größerer Softwaresysteme.

Eine Beschleunigung des Verfahrens läßt sich erreichen, wenn für die Konkretisierung eines abstrakten Planschritts kein neuer Planungsprozeß angestoßen werden muß, sondern aus einer Menge vordefinierter Lösungsalternativen ausgewählt werden kann. Praktisch alle realisierten Planungssysteme die Operatorabstraktion einsetzen, nutzen diese Möglichkeit (bspw. NOAH [Sacerdoti, 1977], NONLIN [Tate, 1976], SIPE [Wilkins, 1988], O-PLAN [Currie und Tate, 1991], [Drabble und Tate, 1995], UMCP [Erol *et al.*, 1994b]). Den nichtprimitiven Operatoren ist dazu eine Menge von Realisierungsmöglichkeiten zugeordnet. Die Begrifflichkeit ist auch in der neueren Literatur keineswegs einheitlich; bspw. heißen die Realisierungsmöglichkeiten in [Erol *et al.*, 1994a] *Task Networks*, weswegen hier auch vom *Hierarchical Task Network* (HTN) Planning gesprochen wird. In [Yang, 1997] wird die Bezeichnung *Reduction* und in [Russell und Norvig, 1995] *Plan* verwendet. Die Planbeschreibungssprache PDDL [Ghallab *et al.*, 1998] ermöglicht die Definition solcher Realisierungsmöglichkeiten als Erweiterung der Operatorbeschreibung durch sogenannte *Action Expansions*. Wichtig ist, daß eine Realisierungsmöglichkeit in allen Varianten die Struktur eines partiellen Plans[14], wie im Kap. 2.1.1 *Least Commitment Planning* beschrieben, haben kann; d.h. sie besteht aus einer Menge von Planschritten, Reihenfolge- und Bindungs-Bedingungen sowie Causal Links.

Die Relation zwischen Operatoren und Realisierungsmöglichkeiten wird meist *Dekompositions-Relation* (dekomp) genannt und ein $p \in$ dekomp (o) eine *Dekomposition des Operators* o[15]. Die Operatoren, aus denen die Planschritte einer Dekomposition abgeleitet wurden, können – sofern es nichtprimitive Operatoren sind – wiederum dekomponiert werden. Damit lassen sich die Operatoren des Planungssachbereichs in einer *Dekompositionshierarchie* anordnen. Den primitiven Operatoren ist keine Dekomposition zugeordnet; sie bilden deshalb die Blätter der Hierarchie. Welche Operatoren primitiv sind, ist eine Entscheidung des Entwicklers des Sachbereichsmodells über die Granularität des Modells, denn prinzipiell gibt es (zumindest im Makrokosmos) keine Grenze, ab der ein Operator nicht noch zerlegt werden könnte. Auch der Akteur der den Plan ausführen soll spielt bei der Entscheidung eine Rolle. Im Beispiel des Hausbaus sind für den Generalunternehmer sicherlich andere Operatoren unteilbar, als für den Handwerker. Wird für einen Roboter geplant, sind die primitiven Operatoren durch dessen Funktionsschnittstelle nach außen vorgegeben.

Die Dekompositionshierarchie darf nicht mit einer Abstraktionshierarchie verwechselt werden (vgl. dazu auch [Kautz, 1991, S.80]). Teilweise wird ein Operator o als abstrakter als ein Operator p angesehen, wenn p zu einer Dekomposition von o gehört. Dennoch ist o i.a. keine Abstraktion von p. Bspw. können Vorbedingungen (und entsprechend auch die Effekte) von o und p verschieden sein, wobei zwischen den Vorbedingungen von o und p weder $\rightarrow$ noch $\leftarrow$ gilt. Die Beziehung zwischen o und p ist also keine Spezialisierung. Sie hat vielmehr den Charakter einer *Ganzes-Teil* Relation (s. [Artale *et al.*, 1996b] u. Kap. 4.6.1 *Teil-Ganzes-Relationen in Beschreibungslogik* für eine Diskussion von Eigenheiten solcher Relationen).

Allerdings stellt eine gesamte Dekomposition eines Operators eine Spezialisierung des Operators dar, wenn Vorbedingungen und Effekte jeweils als charakteristische Eigenschaften zugrunde gelegt werden. Dies ergibt sich aus der üblicherweise an eine Dekomposition gestellten Forderung nach *Vor-*

[14] Einige Varianten lassen darüber hinaus auch komplexere Beschreibungen zu (bspw. die *Action Expansions* in PDDL).

[15] Im weiteren wird deshalb der Begriff *Dekomposition* anstelle des Begriffs *Realisierungsmöglichkeit* verwendet.

bedingungs- bzw. *Effekt-Vererbung*; d.h., daß die Vorbedingung bzw. der Effekt einer Dekomposition eines Operators die Vorbedingung bzw. den Effekt des Operators implizieren muß.

Die Vorbedingung einer Dekomposition bzw. allgemein eines partiellen Plans sei dabei als die Bedingung definiert, die für *jede* zulässige Ausführungsreihenfolge der Planschritte dieses Plans erforderlich ist. Die Vorbedingung eines Plans ist in der Regel schwächer als die Konjunktion der Vorbedingungen aller seiner Planschritte, weil Planschritte, die am Anfang des Plans stehen, durch ihre Effekte (Teil-) Bedingungen für spätere Planschritte erzeugen können. Entsprechend leitet sich der Effekt eines partiellen Plans aus den Effekten der Planschritte ab, wobei berücksichtigt werden muß, daß spätere Planschritte die Effekte von vorangegangenen Planschritten (teilweise) wieder zunichte machen können. Der Effekt eines Plans beschränkt sich auf solche Veränderungen der Situation, die bei *jeder* erlaubten Reihenfolge der Planschritte erreicht werden. Werden – wie bspw. in [Russell und Norvig, 1995] und [Yang, 1997] – Vorbedingungen und Effekte auf Konjunktionen atomarer Formeln bzw. Literale beschränkt und Axiome ausgeschlossen, können Vorbedingung und Effekt eines Plans p wie folgt beschrieben werden:

Seien $pre(x)$ bzw. $eff(x)$ die Mengen der Vorbedingungs- bzw. Effektkomponenten eines Planschritts x, *steps(p)* die Menge der Planschritte eines Plans p und *tot-ord(p)* die Menge aller strikten Ordnungsrelationen, die die Reihenfolgebedingungen[16] von p enthalten. Dann ist die Vorbedingung eines Plans p die Konjunktion aller Elemente von $pre(p)$, wobei

$$\begin{aligned} pre(p) \quad = \quad & \{\, f \mid \forall L\ (L \in \textit{tot-ord}(p)) \rightarrow \\ & \exists s\ (s \in steps(p)) \wedge (f \in pre(s)) \wedge (\forall u\ u \notin L(s) \cup s \rightarrow (f \notin eff(u)) \wedge \overline{f} \notin eff(u)))\} \end{aligned}$$

Entsprechend ist der Effekt eines Plans p die Konjunktion aller Elemente von $eff(p)$, und

$$\begin{aligned} eff(p) \quad = \quad & \{\, f \mid \forall L\ (L \in \textit{tot-ord}(p)) \rightarrow \\ & \exists s\ (s \in steps(p)) \wedge (f \in eff(s)) \wedge (\forall u\ u \in L(s) \rightarrow \overline{f} \notin eff(u))\} \end{aligned}$$

In beiden Fällen steht $\overline{f}$ für $\neg f$, falls f eine atomare Formel ist und für g, falls $f = \neg g$ und g eine atomare Formel ist.

Die Forderung nach Vorbedingungs- bzw. Effekt-Vererbung zwischen einem Operator o und seiner Dekomposition d kann nun wie folgt beschrieben werden:

$$\begin{aligned} pre(o) \quad & \subseteq \quad pre(d) \\ eff(o) \quad & \subseteq \quad eff(d) \end{aligned}$$

Durch die Definition von Vorbedingungen und Effekten für Pläne wird die enge Verwandtschaft zwischen Plänen und Planoperatoren deutlich. In Planungssystemen, die keine Operatorabstraktion verwenden, können Pläne somit wie Operatoren eingesetzt werden. Ansonsten kann zu jedem Plan p ein *zum Plan p gehörender Operator op(p)* mit der gleichen Vorbedingung und dem gleichen Effekt wie p definiert werden, dessen Dekomposition genau aus einem Element, nämlich p besteht. Es gilt also:

$$\begin{aligned} pre(op(p)) \quad & = \quad pre(p) \\ eff(op(p)) \quad & = \quad eff(p) \\ dekomp(op(p)) \quad & = \quad \{p\} \end{aligned}$$

Abgesehen von syntaktischen Unterschieden der Definition können Pläne also als Operatoren aufgefaßt werden. Umgekehrt kann ein Operator immer auch als Stellvertreter für einen (oder mehrere)

[16] Die Reihenfolgebedingungen seien dabei als Elemente einer zyklenfreien, asymmetrischen Relation aufgefaßt.

Pläne aufgefaßt werden, in die er dekomponiert werden kann bzw. könnte, wenn die Modellierung ausreichend fein wäre.

Abschließend soll nun noch kurz auf die Frage eingegangen werden, ob für Operatorabstraktion ähnliche Eigenschaften für die Planerzeugung wie bei der Situationsabstraktion nachgewiesen werden können.

Ein abstrakter Plan garantiert bei Operatorabstraktion nicht die Existenz einer konkreten Lösung, weil es vorkommen kann, daß alle erlaubten Dekompositionen zu inkonsistenten Plänen führen; d.h. keine der Dekompositionen läßt sich in den Gesamtplan einbringen, ohne daß es zu widersprüchlichen Reihenfolge- oder Bindungs-Bedingungen kommt. Umgekehrt kann auch nicht geschlossen werden, daß es *keine* konkrete Lösung gibt, wenn keine abstrakte existiert. [Yang, 1997] zeigt dies am Beispiel eines Plans, der zwei Aussagen u und w erzeugen soll und aus zwei abstrakten Planschritten a und b besteht, die als Vorbedingung p_a bzw. p_b benötigen. Beide Bedingungen sollen in der Startsituation gelten. Schritt a erzeugt u und vernichtet p_b, während b w erzeugt und p_a vernichtet. Beide Schritte zusammen erzeugen zwar das gewünschte Resultat, sie lassen sich aber nicht anordnen, weil der eine jeweils die Vorbedingung des anderen zerstört (s. Abb. 2.1). Lassen sich a in a_1 und a_2, und b in b_1 und b_2 dekomponieren, so ist bspw. $a_1 < b_1 < a_2 < b_2$ eine Lösung, wenn p_a von a_1 und p_b von b_1 benötigt, aber erst von b_2 bzw. a_2 zerstört werden (s. Abb. 2.2). "$x < y$" steht für die Reihenfolge-Bedingung *x muß vor y stehen*. Lediglich wenn es in jeder Dekomposition eines jeden nichtprimitiven Operators einen (Sub-)Operator gibt, der sämtliche Vorbedingungen des abstrakten Operators benötigt und auch alle Effekte erzeugt, gilt auch hier die *Upward Solution Property*. Ein solcher zentraler Suboperator wird *Unique Main Subaction* genannt (vgl. [Yang, 1990]).

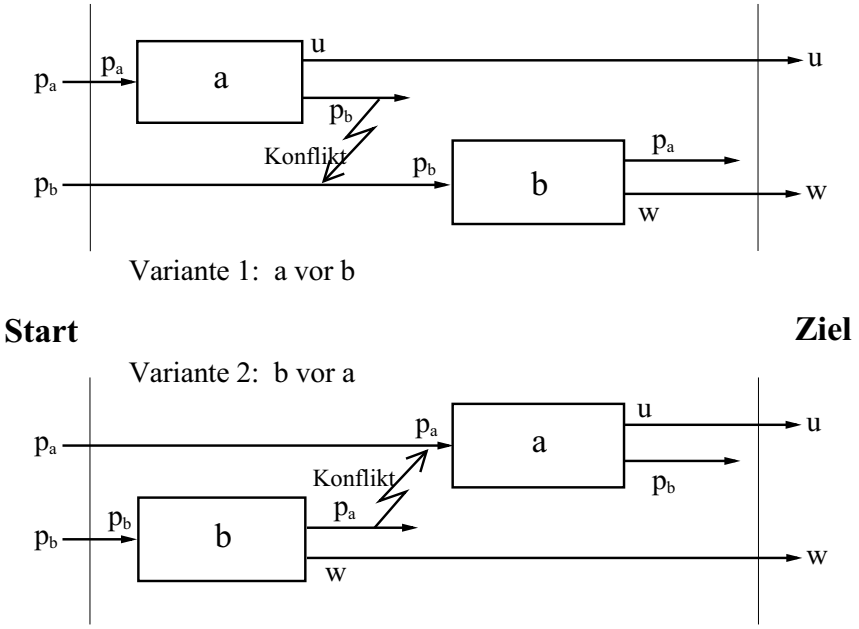

Abbildung 2.1: Abstrakte Planschritte die nicht widerspruchsfrei angeordnet werden können.

[Erol *et al.*, 1994c] und [Bäckström und Jonsson, 1995] zeigen allerdings, daß eine uneingeschränkte Verwendung von Abstraktion zu ähnlich unbefriedigenden Ergebnissen der Komplexitätsbetrachtungen führt wie beim Planen ohne Abstraktion (vgl. [Erol *et al.*, 1995], [Bylander, 1994]). Bspw. ist die Frage nach der Existenz eines Plans unentscheidbar. In beiden Arbeiten werden Einschränkungen genannt, die die Komplexität reduzieren. Allerdings bleibt das Problem der Planexi-

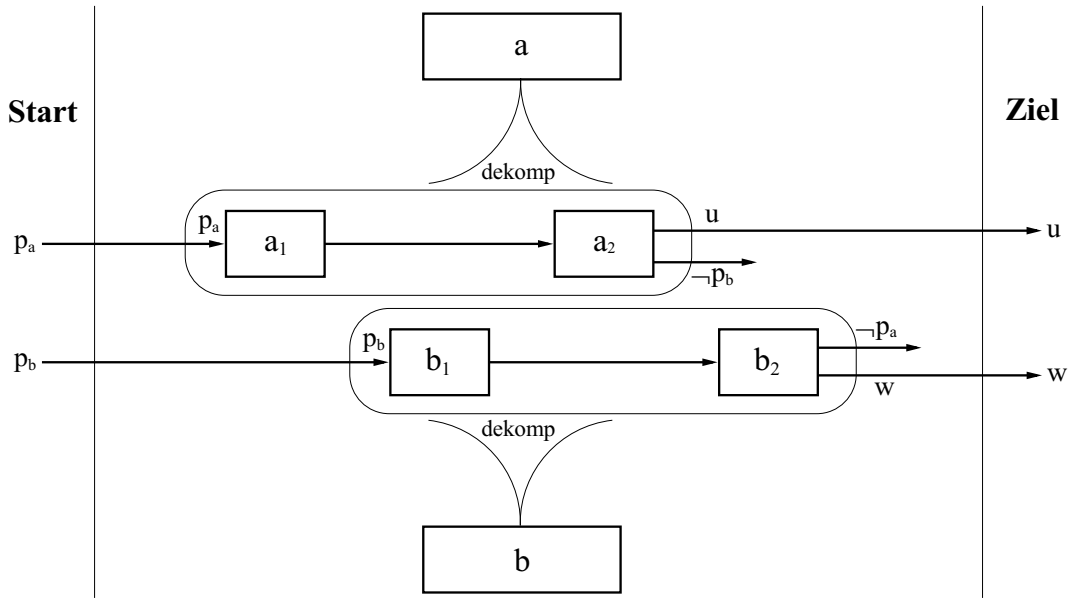

Abbildung 2.2: Lösungsmöglichkeit mit dekomponierten Planschritten

stenz bis auf wenige sehr spezielle Fälle immer noch unhandhabbar; d.h. es liegt in der Komplexitätsklasse NP oder ist noch schwieriger. Das heißt also, daß der Einsatz von Abstraktion an sich – zumindest im ungünstigsten Fall – *nicht* zu einer Beschleunigung des Planungsprozesses führt.

Diese unbefriedigenden theoretischen Ergebnisse stehen allerdings im Kontrast zum Einsatz von Operatorabstraktion in den meisten "praktischen" Planungssystemen, bspw. SIPE [Wilkins, 1988] und O-PLAN [Currie und Tate, 1991], [Drabble und Tate, 1995]; in [Nau *et al.*, 1998] finden sich weitere aktuellere Beispiele.

Die Vorteile eines drastisch verringerten Aufwands für die Lösungssuche überwiegt gegenüber dem Zusatzaufwand für die Erzeugung (evtl. unbrauchbarer) abstrakter Lösungen, wenn keine oder nur wenige störende Abhängigkeiten zwischen den (Teil-) Planschritten vorkommen. Dann kann es nur selten vorkommen, daß sich keine Dekomposition eines Operators in den Gesamtplan integrieren läßt, ohne die Inkonsistenz und damit die Unbrauchbarkeit des Gesamtplans zu bewirken (vgl. a. [Yang, 1997, K.6.2]). Beim vergleichbaren Ansatz in der Softwareentwicklung werden solche Abhängigkeiten durch Kapselung auf ein Minimum reduziert. Analog kann und sollte beim Entwurf des Planungssachbereichs vorgegangen werden; d.h. abstrakte Operatoren sollten möglichst so definiert werden, daß Abhängigkeiten zu anderen Operatoren weitgehend vermieden werden.

Der Entwickler des Sachbereichsmodells bringt damit mehr Wissen – insbesondere über die Struktur des Planungssachbereichs – ein, als dies bei der Modellierung elementarer Veränderungsmöglichkeiten durch primitive Operatoren der Fall ist. Im weiteren Verlauf dieser Arbeit wird noch gezeigt, wie dieses Wissen auch für die Präsentation eines Plans genutzt werden kann.

2.1.3 Voraussetzungen beim Planen

Die Generierung von Plänen hat natürlich nur dann Sinn, wenn der Akteur der den Plan ausführen soll durch die Ausführung des Plans die Ziele *in der realen Welt* erreicht. Dies ist aber – insbesondere beim bisher vorgestellten sogenannten *klassischen Planen* – nur unter bestimmten Voraussetzungen sichergestellt. Die wichtigsten davon sind:

1. Die kontinuierlichen Änderungen der realen Welt können adäquat durch die Beschreibung diskreter Situationen modelliert werden.
2. Alle Eigenschaften und Merkmale jeder Situation – insbesondere der Situation unmittelbar vor der Planausführung – sind vollständig bekannt.
3. Jeder Zustand der realen Welt kann durch eine endliche Menge von logischen Aussagen adäquat modelliert werden.
4. Die Welt ändert sich ausschließlich durch die Ausführung eines Planschritts, und dann nur genau so, wie durch den (deterministischen) Effekt des Planschritts angegeben. Die Änderung wird durch Aussagen die nach der Ausführung wahr bzw. nicht mehr wahr sind beschrieben. Diese Annahme wird *STRIPS Annahme* genannt. Wilkins [Wilkins, 1988] beschränkt die Änderungen darüber hinaus auf die *expliziten* Effekte eines Planschritts.
5. Die Ziele sind durch eine Folge von Planschritten erreichbar, wobei die Planschritte nacheinander und niemals gleichzeitig ausgeführt werden. Sussman [Sussman, 1975] nannte dies die *Linearitätsannahme*.
6. Der Akteur ist *zu dem Zeitpunkt* zu dem er einen Planschritt ausführen soll auch dazu in der Lage; d.h., er *weiß wie* der Planschritt auszuführen ist und hat die notwendigen *Fähigkeiten* dafür.

Die Annahmen sind allerdings keinesfalls immer erfüllt oder wenigstens erfüllbar. Vergleichbar zum Qualifikationsproblem ist es bspw. prinzipiell gar nicht möglich 2. zu erfüllen. 3. hängt vom Anwendungsbereich ab. (Quasi-) kontinuierliche Größen, wie bspw. der Füllstand eines Kraftstofftanks oder die Menge freien Arbeitsspeichers, lassen sich durch logische Aussagen nur inadäquat modellieren. 4. ließe sich bestenfalls erfüllen, wenn der Plan in einem abgeschlossenen System ausgeführt wird und auch nur dieses betrachtet wird. Eine Anforderung, die in der Praxis nur selten gewährleistet ist. 5. ist oft, aber keineswegs immer erfüllt, wie bspw. das Problem des Schusters Voigt in Zuckmayers *Der Hauptmann von Köpenick* zeigt. Sein Ziele *Arbeit* und *Wohnung zu bekommen*, waren nur gleichzeitig, aber nicht nacheinander zu erreichen. Annahme 6. beinhaltet zum einen die Forderung nach Anpassung der Plangenerierung an den vorgesehenen Akteur. Außerdem muß ein Plan so rechtzeitig fertiggestellt sein, daß die Voraussetzungen von der die Planerzeugung ausgegangen war bei Planausführung noch wahr sind.

Damit steht die Frage im Raum, ob die vorgestellten Vorgehensweisen der Planengenerierung gerechtfertigt sind. Sie kann auf die Frage nach dem Sinn von Planen überhaupt erweitert werden, da die Annahmen – zumindest teilweise und in abgewandelter Form – auch für das Planen von Menschen gelten. Auch dort können bspw. unerwartete Ereignisse oder unbekannte Voraussetzungen zum Scheitern eines Plans führen. Die Nützlichkeit von Planen für unser Handeln und auch für die Interaktion mit der Außenwelt steht aber außer Frage, wie bspw. Pollack dies für die Interaktion mit der Außenwelt in [Pollack, 1992, S.62] explizit macht:

> It turns out that, if you want to coordinate and communicate with other agents, it is extremely useful – and possibly even essential – for you and those other agents to be planners.

Die Nützlichkeit von Planen und der Planungsergebnisse kann aber nicht daran gemessen werden, ob ein Plan in der realen Welt seine Ziele *garantiert* erreicht, denn diese Garantie kann nicht gegeben werden. Selbst dann nicht, wenn die Plangenerierung *im Modell* nachweislich korrekt ist; denn im Modell werden bewußt nicht alle Eigenschaften und Zusammenhänge der realen Welt berücksichtigt. Es ist ja gerade das Ziel einer jeden Modellbildung, von der Komplexität der realen Welt zu abstrahieren und dann in einem vereinfachten Modell Schlüsse zu ziehen oder Prognosen zu stellen.

Kriterien für die Beurteilung des Nutzens sind bspw., wie häufig ein Akteur mit einem Plan in der realen Welt seine Ziele erreicht und wie groß der Aufwand für eine Korrektur ist, wenn die Ziele ohne Modifikation des Plans nicht erreicht werden können.

Um a priori abschätzen zu können, ob eine Plangenerierung in einem Anwendungsbereich sinnvoll ist, müssen die getroffenen Annahmen in Relation zu den Merkmalen und Eigenschaften dieses Anwendungsbereichs beurteilt werden. In [Hertzberg, 1993] findet sich eine breite Darstellung der Problematik.

Ebenso wichtig ist die Abschätzung, inwieweit Abweichungen von den Planungsergebnissen in der realen Welt *tolerierbar* oder *korrigierbar* sind. Letzteres kann bedeuten abzuschätzen, ob der Akteur – evtl. mit Hilfe des Planungssystems – in der Lage ist, ein unerwartetes Hindernis der Planausführung durch eine geringfügige Modifikation des Plans zu beseitigen. Im Beispielsachbereich dieser Arbeit kann bspw. ein Meldungsfenster, das an einen Termin erinnern soll, die Ausführung eines Plans unterbrechen. Nach einer Bestätigung der Meldung durch den Benutzer kann der Plan normalerweise aber wieder aufgenommen werden. Für die Modellierung dieses Sachbereichs ist es deshalb in vielen Fällen akzeptabel, das spontane Auftreten solcher Meldungsfenster nicht zu modellieren.

Die Anpassung eines Plans an eine (gegenüber den ursprünglichen Annahmen) veränderte Situation und die Reaktion auf unerwartete Ereignisse wird bspw. in [Georgeff und Lansky, 1987], [Wilkins, 1988] und [Wilkins *et al.*, 1995] behandelt.

Zum Abschluß seien noch einige Ansätze zum Planen erwähnt, die – im Vergleich zu den oben aufgeführten Annahmen des *klassischen Planens* – auf weniger oder weniger restriktiven Annahmen beruhen.

Zum *Planen unter Unsicherheit* zählen die Arbeiten zum *probabilistischen Planen* und zum *Anytime-Planen*. Zu ersterem gehört bspw. das Planungssystem BURIDAN [Kushmerick *et al.*, 1995], das einen Plan erzeugt, der das Erreichen der Ziele (auch im Modell) nicht garantiert, sondern lediglich Aussagen zur Wahrscheinlichkeit macht mit der die Ziele erreicht werden können. Auch die Merkmale und Eigenschaften der Startsituation werden nicht durch Wahrheitswerte, sondern durch Wahrscheinlichkeiten beschrieben. Beim Anytime-Planen [Dean und Boddy, 1988] geht es darum den Planungsprozeß so zu steuern, daß zu jedem Zeitpunkt ein Plan als Zwischenergebnis abgerufen werden kann. Dabei muß die Qualität des Plans – also die Chance, mit dem Plan die Ziele zu erreichen – mit fortscheitender Planungszeit besser werden. Ein solcher Ansatz ist bspw. in zeitkritischen Anwendungen sinnvoll, wenn die zur Verfügung stehende Zeit für die Planerstellung nicht im Voraus bekannt ist.

Die Voraussetzungen des klassischen Planens (insb. 1, 4, 5) ermöglichen einen Verzicht auf die explizite Modellierung der Zeit. Es sind dadurch lediglich die einzelnen Situationen vor und nach der Ausführung eines Planschritts, aber nicht deren *Ausführungsdauer* relevant. Auch von der, in der Realität vorkommenden, begrenzten Gültigkeitsdauer von Aussagen oder deren kontiunierliche Veränderung wird abgesehen. In Sachbereichen, in denen gerade die zeitliche Anordnung und Koordinierung die hauptsächliche Schwierigkeit von Aufgaben darstellt, reicht das einfache Zeitmodell, das nur ein *vor* und *nach* kennt, nicht mehr aus. Beispielsachbereiche dazu sind das Transportwesen oder chemische Prozesse. Arbeiten zu einer komplexeren Modellierung der Zeit bei der Plangenerierung sind bspw. [Vere, 1983] und [Pelavin, 1991]. Veres Planungssystem DEVISER berücksichtigt Zeitangaben und die Dauer von Aktionen. Pelavin behandelt die Konsequenzen des Wegfalls der Linearitätsannahme; d.h. wenn Planschritte auch nebenläufig ausgeführt werden dürfen.

Eine weitere Gruppe von Arbeiten befaßt sich mit der Berücksichtigung von begrenzten Ressourcen, die sich nicht adäquat durch logische Aussagen modellieren lassen. Dazu gehören bspw. [Wilkins, 1988], [Currie und Tate, 1991] und [Koehler, 1998].

2.1.4 Abschließende Bemerkungen

In diesem Abschnitt wurden Ansätze zur Plangenerierung vorgestellt, die prägenden Einfluß auf die Vorstellung über die Modellierung von Plänen und Veränderungsmöglichkeiten hatten. Die Eigenschaften der Verfahren selbst wurden nur am Rande behandelt – hierfür sei auf die eingangs erwähnte Literatur verwiesen. Nicht behandelt wurden Ansätze, deren Wert in der Erprobung von Planungstechniken oder der Verringerung des Laufzeitbedarfs der Plangenerierung liegen. Die wichtigsten Beispiele dafür sind deduktive Planungsverfahren [Bibel, 1986], [Biundo, 1993], Ansätze, die nach dem Graphplan-Verfahren [Blum und Furst, 1997] arbeiten, sowie Ansätze, die ein Planungsproblem auf Basis einer Übersetzung in eine aussagenlogische Formel lösen [Kautz und Selman, 1992], [Kautz und Selman, 1996]. Auf ein deduktives Planungsverfahren wird in Abschnitt 2.5.3 *Erzeugen von Plänen* noch kurz eingegangen. Für die anderen beiden Ansätze sei auf [Weld, 1999] verwiesen, der die grundsätzlichen Eigenschaften der Verfahren beschreibt und umfassende Hinweise zu weiterführender Literatur gibt.

2.2 Planerkennung

Der Forschungsbereich Planerkennung ist aus den Arbeiten zu natürlichsprachlichen Schnittstellen zu Datenbanken bzw. allgemeiner zu Informationssystemen entstanden[17]. Die ersten Prototypen solcher Systeme machten deutlich, daß eine wörtliche und isolierte Übersetzung einer einzelnen natürlichsprachlichen Frage in eine formale Sprache oft nicht das gewünschte und erwartete Ergebnis erreichte. Die Intention des Fragestellers – die ein wichtiges Kriterium für die Beurteilung der Antwort ist – wurde dabei nicht berücksichtigt. Nachfolgende Arbeiten versuchten deshalb die Intention des Fragestellers zu ermitteln. Es wurde (und wird) davon ausgegangen, daß ein Fragesteller Ziele hat und diese nun durch einen Plan, den er dafür erstellt hat, erreichen will. Eine Frage an das System – oder allgemeiner eine Äußerung des Benutzers – wird dabei als eine Aktion bzw. ein Schritt des Plans angesehen. Die Aufgabe der Planerkennung besteht nun darin, aus den beobachteten Äußerungen oder auch sonstigen Aktionen eines Benutzers zu schließen, welchen Plan er ausführt und welche Ziele er damit anstrebt. Dadurch wird es dann möglich eine zielgerichtete Antwort zu geben, die auch über die Frage hinausgehende Ergänzungen enthalten kann.

Im folgenden soll die Entwicklung des Gebiets kurz dargestellt werden und dann auf wichtige Arbeiten eingegangen werden, die die Vorstellung und Repräsentation von Plänen in der Planerkennung bestimmt haben. Für eine ausführliche Darstellung des Gebiets sei auf das Buch von Carberry [Carberry, 1990a] verwiesen, das auch auf die speziellen Probleme bei der Behandlung natürlicher Sprachen eingeht.

2.2.1 Wurzeln der Planerkennung

Die ersten Prototypen natürlichsprachlicher Schnittstellen machten deutlich, daß eine wörtliche und isolierte Übersetzung einer einzelnen natürlichsprachlichen Frage in eine formale Sprache oft nicht das gewünschte und erwartete Ergebnis erreichte. Obwohl empirische Untersuchungen wie bspw. [Zoeppritz, 1984] gezeigt haben, daß Benutzer einen Computer auch dann nicht als vollwertigen Dialogpartner ansehen, wenn sie Fragen in natürlicher Sprache stellen können, wird dennoch eine Antwort erwartet, die über eine wörtliche Interpretation der Frage hinausgeht und den normalen *Gebrauch* der Sprache berücksichtigt (vgl. [Cohen *et al.*, 1982, S.245]). Als Maßstab dafür wird eine

[17] vgl. [Küpper *et al.*, 1992] für einen Überblick über natürlichsprachliche Schnittstellen mit Deutsch als Interaktionssprache.

kooperative[18] Konversation zwischen zwei Menschen angesehen, wie sie durch das Grice'sche Kooperativitäts-Prinzip ([Grice, 1975]) beschrieben wird. Danach wird von einem kooperativen Dialogpartner erwartet:

> *Make your conversational contribution such as is required, at the stage at which it occurs, by the accepted purpose or direction of the talk exchange in which your are engaged.*

Wichtig dabei ist, daß der *Grund* für eine Frage bzw. den gesamten Dialog eine entscheidende Rolle spielt und sowohl bei der Interpretation einer Äußerung als auch bei der Erzeugung einer Antwort berücksichtigt werden muß. Dieser Grund ist aber nicht immer von vorn herein bekannt. Außerdem ist der vordergründige Zweck einer Frage, die Antwort wissen zu wollen, oft durch übergeordnete Ziele begründet, wie das Beispiel der Frage aus [Allen und Perrault, 1980] im Umfeld einer Bahnhofshalle zeigt:

> *Wann fährt der Zug nach Montreal ab?*

Das Wissen um die Uhrzeit eines abfahrenden Zuges ist in erster Linie interessant, wenn man das (übergeordnete) Ziel hat, mit diesem Zug verreisen zu wollen. In dem Beispiel berücksichtigt der Schalterbeamte als kooperativer Zuhörer dieses (wahrscheinliche) Ziel mit. Er antwortet mit der Angabe von Abfahrzeit *und* Bahnsteig, weil auch die Kenntnis des Bahnsteigs für dieses Ziel erforderlich ist. Der Schalterbeamte hat also aus der Aktion des Fragenden – die Frage nach der Abfahrtszeit zu stellen – geschlossen, daß dieser einen Plan verfolgt, der das Ziel hat mit dem nachgefragten Zug zu reisen. Zur Durchführung des Plans muß er den Zug erreichen und dazu wiederum muß er wissen auf welchem Bahnsteig der Zug abfährt. Diese Vorgehensweise des Schalterbeamten ist ein Beispiel für Planerkennung.

Planerkennung hat die Aufgabe aus den beobachteten Aktionen eines Akteurs (im Bsp. die Frage) das intendierte Vorgehen und letztlich die Ziele dieses Akteurs abzuleiten, um auf dieser Basis eine Reaktion erzeugen zu können, die den Akteur bei seinem Vorgehen und/oder beim Erreichen seiner Ziele unterstützt. Dabei muß auch berücksichtigt werden, daß ein Akteur falsche Vorstellungen haben kann und sein intendiertes Vorgehen deshalb nicht notwendigerweise dazu führt, daß er (alle) seine Ziele erreicht. Es sei angemerkt, daß das Grice'sche Kooperativitäts-Prinzip dabei verallgemeinert angewendet wird. Es geht hier nicht ausschließlich um Redebeiträge, die zwei Dialogpartner in einem Gespräch austauschen, sondern es geht um das Handeln der beteiligten Akteure. Handeln wird dabei – in Einklang mit der Sprechakttheorie aus der Sprachphilosophie (vgl. [Searle, 1969]) – als Oberbegriff auch für Redebeiträge angesehen. Der einzelne Akteur interpretiert dabei die beobachteten Aktionen (einschließlich eines evtl. Redebeitrags) seines Partners und entscheidet sich für eine Handlung, die dem Zweck der Kooperation dient. Das kann – wie im Bsp. oben – ein Redebeitrag sein, es kann aber auch eine physische Handlung sein. Bspw. wäre ein kooperativer Beitrag auf die Frage *Könnten Sie das Fenster schließen?* die Antwort *Ja!* und *das Schließen des offenen Fensters*. In der Literatur zur Planerkennung finden sich viele Beispiele für Handlungen, die der beobachtete Akteur außerhalb des Computers durchführt, bspw. Kautz' Cooking World (s.u.). Der Wert solcher Beispiele für derzeitige Mensch-Computerschnittstellen ist allerdings fragwürdig, weil heutige (Standard-) Computer keine Handlungen beobachten[19] können, die außerhalb des eigenen Systems durchgeführt werden.

Das Vorgehen bei der Planerkennung und der Erzeugung einer kooperativen Antwort ist für alle an einem Dialog beteiligten Akteure angebracht. Wie die Studie von Schmidt, Sridharan und Goodson [Schmidt *et al.*, 1978] zeigt, versuchen Menschen in einem Dialog auch tatsächlich die Pläne und

[18] Es sind durchaus auch unkooperative Dialogsysteme denkbar (vgl. [Jameson *et al.*, 1994], [Pollack, 1986]). Normalerweise wünscht sich ein Benutzer eines Computersystems allerdings, daß *seine* Ziele unterstützt werden und nicht die des Computersystems oder eines Dritten.

[19] Mit *beobachten einer Handlung* ist gemeint, daß die Handlung (ggf. mit ihren Parametern) durch den Computer identifiziert wird. Ein heute bereits anzutreffendes Aufnehmen eines Bildes oder einer Bildfolge reicht dazu nicht aus.

Ziele des Dialogpartners zu erkennen. Da für diese Arbeit – und die meisten Systeme, die Planerkennung betreiben – die Mensch-Computerschnittstelle im Zentrum des Interesses steht, sei aber im weiteren der beobachtete Akteur als der *Benutzer* und der Akteur, der beobachtet, daraus Schlüsse zieht und eine Reaktion erzeugt, als das *System* bezeichnet.

Das intendierte Vorgehen eines Benutzers wird bei der Planerkennung als Plan aufgefaßt, den dieser erstellt hat und zum Zeitpunkt der Beobachtung durchführt. Für die Planerkennung übernimmt Allen [Allen, 1979] die Vorstellung eines Plans von der Handlungsplanung – konkret von den Modellen der Systeme STRIPS [Fikes und Nilsson, 1971] und NOAH [Sacerdoti, 1977]. Wie im Abschnitt über das Handlungsplanen bereits beschrieben, besteht ein Plan danach aus einer (partiell) geordneten Menge von Planschritten, die Exemplare von Planoperatoren sind. Die Planoperatoren haben eine Vorbedingung und einen Effekt und ihnen kann – entsprechend der bei NOAH eingesetzten Operatorabstraktion – eine Dekomposition zugeordnet werden. Allen nutzt dieses Wissen über die zu den beobachteten Aktionen gehörenden Planoperatoren, um a) den Plan, den der Benutzer durchführt und b) die Hindernisse, die diesem Plan im Wege liegen, zu erkennen. Letzteres ist dann der Ausgangspunkt für die sogenannte *Überbeantwortung* von Anfragen des Benutzers; d.h. die ungefragte Angabe zusätzlicher, nützlicher Information (im Bahnhofsbeispiel der Bahnsteig). Allen nennt sein Vorgehen *plan-deduction and obstacle-detection cycle*. Für die Erkennung des Benutzerplans gibt Allen eine Reihe von Heuristiken an, nach denen sich aus einzelnen beobachteten Aktionen Hypothesen über den Benutzerplan aufstellen lassen. Die Heuristiken werden dabei als Umkehrung von Heuristiken[20] zur Planerzeugung angesehen. Bspw. wird aus der Erzeugungsheuristik

> *setze einen Planoperator o ein, wenn eine Bedingung p erfüllt werden soll und p Teil des Effekts von o ist*

die Erkennungsheuristik

> *wenn eine Aktion o beobachtet wird und p Teil des Effekts von o ist, will der Benutzer vermutlich p erfüllt haben*

Dabei kann *p* das Ziel des Benutzers, oder lediglich Voraussetzung für andere Aktionen sein; d.h. es kann weiter abgeleitet werden:

> *wenn der Benutzer p erfüllt haben will und p Teil der Vorbedingung des Planoperators o ist, will der Benutzer vermutlich o ausführen*

Die beobachteten Aktionen können Teil der Dekomposition eines abstrakteren Operators sein, was Anlaß für folgende Heuristik ist:

> *wenn eine Aktion o beobachtet wird und o in einer Dekomposition eines abstrakteren Operators a vorkommt, will der Benutzer vermutlich a ausführen*

Alle Ableitungen führen hier lediglich zu Hypothesen über das Vorgehen bzw. die Ziele des Benutzers. Außerdem sind die Schlüsse keineswegs eindeutig. Bspw. kann ein Operator in den Dekompositionen mehrerer abstrakter Operatoren vorkommen und so den Schluß auf mehrere abstrakte Operatoren rechtfertigen. Allen gibt dazu weitere Heuristiken zur Bewertung der konkurrierenden Hypothesen an, auf die hier aber nicht eingegangen werden soll. Es sei angemerkt, daß durch die Plan*erzeugungs*heuristiken lediglich ein sehr einfacher Planungsprozeß simuliert werden kann. Insbesondere werden keine konkurrierenden Abhängigkeiten zwischen Planoperatoren berücksichtigt, die ja die Komplexität der Handlungsplanung ausmachen.

[20] Der Begriff *Heuristik* ist zur Zeit von Allens Arbeit noch gerechtfertigt, weil die Verfahren zur Plangenerierung damals noch nicht systematisch vorgingen.

In Allens sehr einfachem und begrenztem Sachbereich kommen solche Abhängigkeiten nicht vor. Für komplexere Sachbereiche hat sich ein anderes Verfahren etabliert. Anstelle der Erzeugung von Planhypothesen während der Interaktion mit dem Benutzer, wird a priori eine Planbibliothek aufgebaut und dann abgeschätzt, inwieweit das Handeln des Benutzers zu den Plänen der Bibliothek paßt. Im Vergleich zu Allens Vorgehen lassen sich damit komplexere Pläne darstellen, und es ist auch möglich *falsche Pläne* zu repräsentieren. Als *falsche Pläne* seien Pläne bezeichnet, die nach Ansicht des Systems nicht zum gegebenen Ziel führen. In Abschnitt 2.5.1 *Erkennung von falschen oder suboptimalen Plänen* wird noch näher auf diesen Aspekt eingegangen. Die Erkennung auch solcher Pläne kann genutzt werden, um einen Benutzer vor unerwünschten Ergebnissen zu warnen.

2.2.2 Kautz' Theorie der Planerkennung

Eine formale Grundlage für die Planerkennung wurde von Kautz erarbeitet [Kautz und Allen, 1986], [Kautz, 1987], [Kautz, 1991]). Insbesondere seine Repräsentation von Plänen in einer Abstraktionshierarchie mit einer formalen Definition von Abstraktion und Dekomposition hat die Vorstellung von Plänen in der Planerkennung maßgeblich beeinflußt.

Kautz unterscheidet nicht zwischen Plänen und Planoperatoren und bezeichnet beides als *Events*. Das deklarative Wissen, das er für die Planerkennung benötigt, wird durch eine *Event Hierarchy* repräsentiert. Diese Hierarchie wird durch Mengen von einstelligen Prädikaten – den *Event Types* die für Mengen von *Events*[21] stehen – sowie durch Mengen von Formeln der Prädikatenlogik erster Stufe – den sogenannten *Abstraktions-*, *Dekompositions-* und *generellen Axiomen* – definiert.

Abstraktions-Axiome haben die Form:

$$\forall x \quad E_0(x) \rightarrow E_1(x)$$

und definieren die Relation E_1 *abstrahiert direkt* E_0 zwischen zwei Event Types E_1 und E_0. *Abstraktion* wird als transitiver Abschluß der direkten Abstraktion definiert. Der allgemeinste Event Type und damit Wurzel der Abstraktionshierarchie heißt *AnyEvent*. Außerdem gibt es noch den ausgezeichneten Event Type *End Event*, der alle Event Types umfaßt[22], die *nicht* Komponenten (s.u.) eines anderen Event Types sind. Ebenfalls ausgezeichnet ist die Menge der *Basic Event Types*, die keine Abstraktion eines anderen Event Types darstellen.

Dekompositions-Axiome haben die Form:

$$\forall x \quad E_0(x) \rightarrow E_1(f_1(x)) \wedge \ldots \wedge E_n(f_n(x)) \wedge \kappa$$

Dabei sind die E_i wieder Event Types. Die f_i sind Rollen-Funktionen, die die Relation *direkte Komponente* zwischen einem Event Type und einer seiner *direkten Komponenten* definieren. κ ist eine Formel, die weitere Bedingungen für E_0 beschreibt. Dazu gehören Zeitbedingungen zwischen seinen Komponenten, Gleichheits- bzw. Bindungsbedingungen und auch Vorbedingung und Effekt. Als Zeitbedingungen sind hier nicht nur – wie in den meisten Arbeiten zur Handlungsplanung – *Schritt A vor Schritt B* zulässig, sondern Allens Zeitrelationen [Allen, 1984], die auf Zeitintervallen definiert sind. Damit lassen sich bspw. auch Überlappung und Endsynchronisation von Planschritten beschreiben.

Die *generellen Axiome* dürfen keine Event Types enthalten und sind für die Definition sonstiger Fakten vorgesehen.

Eine solche Event Hierarchy definiert eine endliche Menge von Plänen und wird deshalb auch *Planbibliothek* genannt.

[21] Kautz nennt die einzelnen Individuen auch *Event Tokens*.

[22] und damit gleich oder abstrakter ist als diese Event Types

Kautz weist darauf hin, daß eine Event Hierarchy als Beschreibung eines semantischen Netzwerks ([Quillian, 1968], [Brachman, 1979]) aufgefaßt werden kann. Die Knoten des Netzwerks werden dabei durch die Event Types gebildet. Die Kanten repräsentieren die Relationen *abstrahiert direkt* (in Abb. 2.3 durch dicke graue Pfeile dargestellt) und *direkte Komponente* (in Abb. 2.3 durch dünne schwarze Pfeile dargestellt). Abbildung 2.3 zeigt die entsprechende graphische Darstellung von Kautz' Beispiel *Cooking World* aus [Kautz, 1991, S.81]. Die durch die Teilformel κ von Dekompositions-Axiomen beschriebenen Bedingungen und die generellen Axiome sind nicht dargestellt.

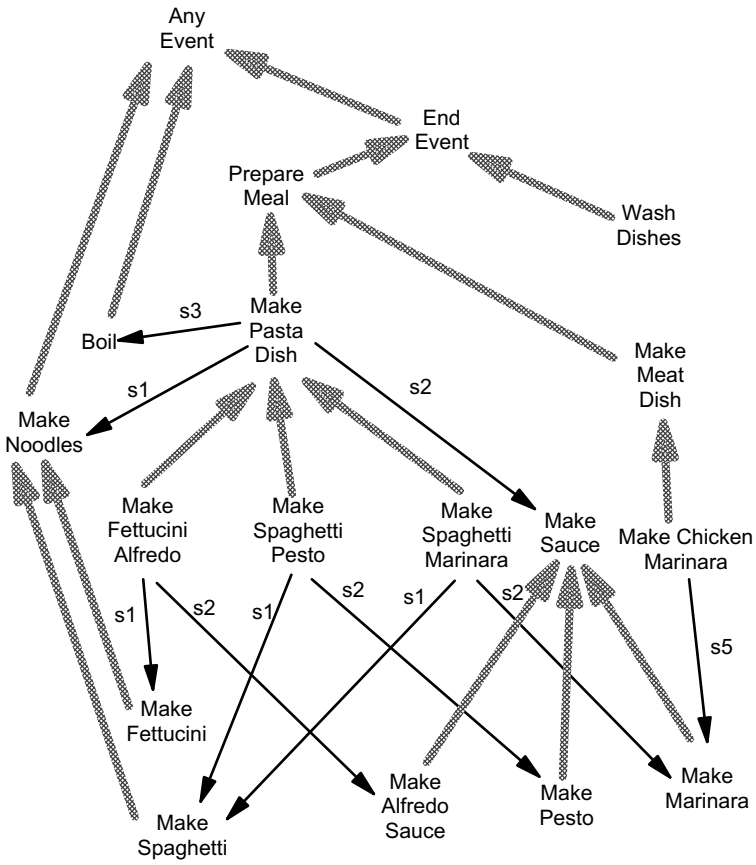

Abbildung 2.3: Graphische Darstellung von Kautz' Beispiel *Cooking World* aus [Kautz, 1991, S.81].

Es sei angemerkt, daß sich dieser Teil der Event Hierarchy – also die Abstraktions-Axiome und die Dekompositions-Axiome jeweils ohne die Teilformel κ – auch in heutigen Standard-Beschreibungslogiken, die ja ihre Wurzeln in der Theorie semantischer Netzwerke haben[23], darstellen läßt. In der Darstellungsweise der in [Patel-Schneider und Swartout, 1993] definierten Wissensrepräsentationssprache[24] läßt sich ein Abstraktionsaxiom $\forall x\, E_0(x) \rightarrow E_1(x)$ durch das Statement:

$$E_0 \sqsubseteq E_1$$

repräsentieren, womit E_0 als sogenanntes *primitives Sub-Concept* von E_1 definiert wird.

[23] vgl. bspw. [Baader, 1996] und Kap. 3.2.1 *Beschreibungslogiken*

[24] Auf Beschreibungslogiken und syntaktischen und semantischen Festlegungen der Sprache wird in Kap. 3.2.1 *Beschreibungslogiken* noch eingegangen.

Durch ein Dekompositions-Axiom $\forall x\ E_0(x) \rightarrow E_1(f_1(x)) \wedge \ldots \wedge E_n(f_n(x))$ wird definiert, daß jedes Element von E_0 in funktionalen Beziehungen $f_1 \ldots f_n$ zu Elementen aus $E_1 \ldots E_n$ steht. In der Ausdrucksweise von Beschreibungslogiken heißt dies, daß Concept E_0 Attribute mit entsprechenden Typbeschränkungen hat:

$$E_0 \sqsubseteq f_1 : E_1 \sqcap \ldots \sqcap f_n : E_n$$

Kautz' Event Types entsprechen dabei Concepts, die eine Abstraktionshierarchie bilden und deren kennzeichnende Attribute die (funktionalen) Beziehungen zu den direkten Komponenten sind.

Kautz weist darauf hin, daß die durch die Abstraktions-Axiome definierte Abstraktionshierarchie und die durch die Dekompositions-Axiome definierte Dekompositionshierarchie zwar zusammenhängen, aber wohl unterschieden werden müssen und das beide in der Handlungsplanung oft durcheinander gebracht werden ([Kautz, 1991, S.80]). Dies gibt Anlaß zu einem Vergleich mit den Ansätzen dort.

Kautz' Dekompositions-Axiome definieren die Vorbedingung und den Effekt eines Event Types, während sie in der Handlungsplanung zu einem Operator selbst gehören. Da Kautz in Dekompositions-Axiomen keine Disjunktion zuläßt, sind Vorbedingung und Effekt eines Events aber auch dort eindeutig definiert.

Ohne den Faktor κ für (u.a.) Vorbedingung und Effekt hat ein Dekompositions-Axiom Ähnlichkeit mit der Beschreibung einer Dekomposition eines Planoperators bei der Handlungsplanung mit Operatorabstraktion (vgl. Kap. 2.1.2.2 *Operatorabstraktion*). Anstelle der *Planschritte* werden bei Kautz die *direkten Komponenten* eines Event Types durch ein Dekompositions-Axiom zeitlich angeordnet und ggf. durch Bindungsbedingungen gekoppelt. Damit kann dieser Teil eines Dekompositions-Axioms als Beschreibung eines partiellen Plans aufgefaßt werden. Allerdings ergeben sich im Detail einige Unterschiede.

Es wurde bereits erwähnt, daß Dekompositions-Axiome komplexere Zeitbedingungen erlauben als bei der Handlungsplanung üblich sind.

Durch die Konstruktion der Dekompositions-Axiome läßt sich für einen Event Type maximal *ein* partieller Plan definieren. Alternative Dekompositionen eines Event Types E lassen sich ausdrücken, indem je Alternative eine Spezialisierung von E definiert wird, deren Dekompositions-Axiom die jeweilige Alternative beschreibt. Für E wird dann lediglich ein Dekompositions-Axiom für die Vorbedingung und den Effekt definiert.

Auffällig ist, daß die Struktur eines Dekompositions-Axioms und auch seine beschreibungslogische Entsprechung ausdrücken, daß ein Event Type eine Spezialisierung seiner Dekomposition ist. Bei der Behandlung von Operatorabstraktion wurde auf den Zusammenhang zwischen Dekomposition und Abstraktion bereits hingewiesen. Dort wurde von Vorbedingung und Effekt als kennzeichnende Attribute eines Planoperators und auch eines partiellen Plans ausgegangen. Es wurde gezeigt, daß ein Operator eine Abstraktion jeder seiner Dekompositionen ist, weil eine Dekomposition wegen der Vererbungsbedingung mindestens[25] die Vorbedingung und den Effekt des Operators hat bzw. erreicht. Auf das Modell von Kautz läßt sich diese Sicht nicht übertragen. Vorbedingung und Effekt einer Dekomposition sind nicht definiert und wären wegen der Möglichkeit der komplexen Anordnung der Komponenten auch nicht ohne weiteres berechenbar[26]. Der Unterschied läßt sich durch den

[25] in dem Sinne, daß die Vorbedingung bzw. der Effekt einer Dekomposition eines Operators die Vorbedingung bzw. den Effekt des Operators impliziert.

[26] Vilain und Kautz zeigten in [Vilain und Kautz, 1986], daß bereits die Frage: *gibt es ein Modell für eine Menge von Zeitintervallvariablen, die durch Allens Zeitrelationen angeordnet sind?* NP-vollständige Berechnungskomplexität hat. Die Größe des Problems ist dabei die Summe aus der Anzahl der Variablen und der Gesamtzahl der angegebenen Zeitbeziehungen zwischen den Variablen. Entsprechend schwierig ist die Frage, welche Vorbedingung bei *jeder* Linearisierung einer Dekomposition gelten muß bzw. welcher Effekt erreicht wird.

unterschiedlichen Anwendungszweck der beiden Ansätze erklären. Bei der Planerkennung kommt es darauf an einen Plan so zu *beschreiben*, daß er durch Beobachten einzelner Komponenten erkannt werden kann. Dabei sind Beschreibungen grundsätzlich allgemeiner[27] als das beschriebene Objekt. In der Handlungsplanung werden dagegen durch Dekomposition *mögliche* Realisierungen für einen komplexen Planoperator – und damit Spezialisierungen – angegeben, die aber nicht den Anspruch haben den Planoperator zu beschreiben.

Vorbedingungen und Effekte spielen in Kautz' Modell und auch in beinahe allen anderen Planerkennungssystemen nur eine untergeordnete Rolle. Ihre zentrale Stellung haben sie bei der *Erzeugung* eines Plans in der Handlungsplanung oder auch in Allens Modell der Planerkennung. Diese Funktion wird überflüssig, wenn Planbibliotheken eingesetzt werden, die alle Pläne die es im vorgesehenen Anwendungsbereich gibt *explizit* repräsentieren.

Eine Event Hierarchy selbst ist allerdings noch keine explizite Repräsentation aller möglichen Pläne des Sachbereichs, wenn solche Pläne als Folge beobachtbarer Aktionen aufgefaßt werden. Zwar sind die Pläne, die beobachtet werden können, durch die speziellsten Sub-Types von *End Event* definiert. Die Event Types repräsentieren allerdings Mengen von Plänen; d.h. es sind Ausführungsvarianten möglich. Diese ergeben sich einerseits daraus, daß die Schritte, bzw. die Komponenten eines Event Types nicht strikt geordnet sein müssen und deshalb in verschiedenen Reihenfolgen ausgeführt werden dürfen. Außerdem repräsentieren die Komponenten wiederum Mengen, aus denen für die Ausführung ein Element ausgewählt werden muß und das auch noch weiter dekomponiert werden darf. Eine explizite Aussage darüber, welche der Ausführungsvarianten tatsächlich korrekte Pläne darstellen, enthält die Event Hierarchy nicht. Dies kann nur durch einen Planprüfungsprozeß festgestellt werden, der auch Vorbedingungen und Effekte berücksichtigt. Wird darauf verzichtet, können auch Aktionsfolgen als Plan identifiziert werden, die nicht die Ziele des Plans erreichen. Wird vorausgesetzt, daß nur korrekte Pläne ausgeführt werden, kann dieser Fall zwar nicht auftreten, allerdings kann die Erkennung eines Plans unnötig verzögert werden. Die Event Hierarchy in Abb. 2.4 zeigt ein Beispiel, in dem E_0 auf zwei verschiedene Arten ausgeführt werden kann. Ohne Berücksichtigung von Vorbedingungen und Effekten lassen sich aber auch zwei weitere Aktionsfolgen bilden, die nicht ausführbar sind. In der Abbildung bezeichnen die Angaben über den Pfeilen vor bzw. nach dem Namen eines Event Types die Vorbedingung bzw. den Effekt des Event Types. Das Symbol "<" zwischen den Event Types soll die Zeitbedingung ausdrücken, daß E_1 vor E_2 ausgeführt werden muß. Wird dann bspw. beobachtet, daß ein Benutzer E_{1a} ausführt, läßt diese Event Hierarchy gemäß Kautz' Annahmen (s.u.) die Schlüsse zu, daß er auch E_1 und – weil E_1 ausschließlich Komponente von E_0 ist – E_0 ausführt. Um E_0 auszuführen ist zwingend notwendig, daß auch E_2 ausgeführt wird. Allerdings bleibt offen *wie*; d.h. ob durch Spezialisierung E_{2a} oder E_{2b}. Durch den Effekt von E_{1a} und den Vorbedingungen von E_{2a} und E_{2b} gibt es aber nur noch die Möglichkeit E_{2b} auszuführen, weil E_{1a} die Vorbedingung von E_{2a} zerstört hat. Diese Kenntnis wird in Kautz' Planerkennungsalgorithmus nicht berücksichtigt. Sie könnte die Planerkennung bereits zu diesem Zeitpunkt zum Ziel führen. D.h. ein System, das den Benutzer unterstützen soll, könnte dann schon Hinweise geben, die für die Ausführung von E_2 nützlich sind. Bspw. könnte das System – falls der Benutzer die Vorbedingung von E_{2a} nicht kennt – darauf hinweisen, daß nur noch E_{2b} ausgeführt werden kann.

Causal Links, die in der Handlungsplanung angeben, daß ein Planschritt eine Bedingung für einen anderen Planschritt erreicht[28], können in Kautz' Modell nur eingeschränkt repräsentiert werden. Kautz erwähnt bei der Erklärung seines *Cooking World* Beispiels, daß das *Ergebnis einer Komponente* eines Event Types für eine weitere genutzt werden kann. In seinem Beispiel setzt er dazu das Ergebnis der Funktion *result* (angewendet auf die eine Komponente) mit dem der Funktion *input* (angewendet auf die andere Komponente) gleich. Prädikatenlogische Funktionen bilden aber auf den Individuen-

[27] Im (theoretischen) Grenzfall kann eine Beschreibung auch exakt sein.

[28] vgl. Kap. 2.1.1 *Least Commitment Planning*.

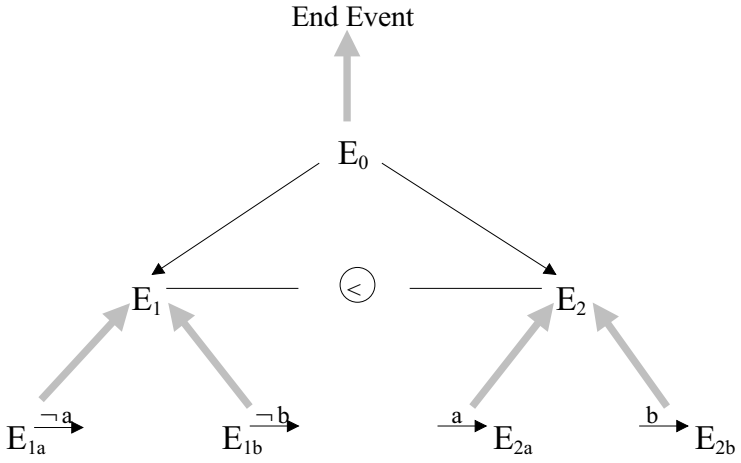

Abbildung 2.4: Event Hierarchy mit Vorbedingungen und Effekten

bereich ab und können damit keine Literale verbinden, wie dies für Causal Links erforderlich wäre. Außerdem sind Vorbedingungen und Effekte i.a. Konjunktionen von mehreren Literalen und können dementsprechend an verschiedenen Causal Links beteiligt sein, während die Funktionen *result* und *input* auf *eine* Komponente angewendet werden und damit das vollständige Ergebnis bzw. die vollständige Vorbedingung repräsentieren müssen.

Für das von Kautz vorgeschlagene Verfahren zur Planerkennung werden allerdings weder die Funktionen *input* und *result* noch Vorbedingungen oder Effekte berücksichtigt. Dies gilt auch für die Event Hierarchy insoweit, als Vorbedingungen oder Effekte zwar definiert werden *dürfen*, aber keinerlei Auswirkungen haben. Insbesondere ist nicht vorgesehen zu prüfen, ob die Komponenten eines Event Types zueinander passen; d.h. ob die Komponenten in ihrer zeitlichen Anordnung *korrekte* und damit auch *ausführbare* Pläne beschreiben. Daß die Beschreibungen in diesem Sinne korrekt sind und alle Komponenten auch einen Zweck für den Event Type erfüllen, setzt Kautz voraus. Er verläßt sich dabei auf das Planungswissen des Entwicklers der Event Hierarchy (vgl. a. Kap. 2.1.2.2 *Operatorabstraktion*). Für die *Erzeugung* eines Plans ist diese Sicht natürlich nicht akzeptabel, weil der Entwickler nicht alle Pläne kennen kann, die erzeugt werden können. Außerdem wird von einem Planungssystem erwartet, daß sein Formalismus ausschließlich korrekte Pläne erzeugt. (Vgl. dazu auch Tenenbergs Ausführungen in [Allen *et al.*, 1991, S.247], dem gleichen Buch in dem auch Kautz seine Arbeit beschreibt.)

Zusammenfassend liegen die wesentlichen Unterschiede der Sichtweisen von Handlungsplanung und Kautz' Ansatz in den verschiedenen Aufgaben und damit verbunden in dem unterschiedlichen Stellungswert von Vorbedingung und Effekt eines Plans oder Operators auf der einen und Komponenten eines Plans auf der anderen Seite. Vorbedingung und Effekt sind in der Handlungsplanung die entscheidenden Kriterien für die Korrektheit eines Plans und für den Aufbau von Abstraktionshierarchien. Für die Erkennung eines Plans sind dagegen die *beobachtbaren* Komponenten entscheidend. Sie werden (im Sinne von Beschreibungslogik) als kennzeichnende Attribute eines Event Types definiert. Für Definition von Event Types sind nur Implikationen – im Sinne von Beschreibungslogik also primitive Definitionen – vorgesehen. Damit ist zwar Vererbung möglich, nicht aber die Einordnung eines Plans bzw. Event Types in die Abstraktionshierarchie anhand seiner Attribute. Der Aufbau der Abstraktionshierarchie beruht damit ausschließlich auf dem vom Entwickler des Sachbereichmodells eingebrachten Wissen. Die für die Handlungsplanung wichtige Frage nach der Korrektheit eines Plans ist für die Planerkennung von untergeordneter Bedeutung. Durch die Beobachtung ist die Aus-

führbarkeit – zumindest bis zum letzten beobachteten Schritt – sichergestellt. Nicht allerdings die Korrektheit des gesamten Plans. Darauf wird noch im Zusammenhang mit Hilfesystemen (Abschnitt 2.5 *Hilfesysteme*) eingegangen.

Den beiden unterschiedlichen Ansätzen von Abstraktion in der Handlungsplanung fügt Kautz eine dritte hinzu. Bei Situationsabstraktion werden die Vorbedingungskomponenten von Planoperatoren oder die Prädikate eines Planungssachbereichs und damit auch Vorbedingungen und Effekte der Operatoren abstrahiert. Eine Dekomposition ist in diesem Ansatz nicht enthalten. Sie wird bei der Operatorabstraktion als mögliche Realisierung eines nichtprimitiven Operators eingesetzt. Eine Abstraktion wird dabei weder explizit repräsentiert, noch spielt sie für den Planungsprozeß eine Rolle. Allerdings kann ein nichtprimitiver Operator als Abstraktion jeder seiner Dekompositionen aufgefaßt werden. Die Abstraktion beruht dann auf Vorbedingungen und Effekten. Einem Operator sind dabei – im Gegensatz zu Kautz' Ansatz – keine Planschritte oder Komponenten *direkt* zugeordnet. Kautz verbindet Abstraktion und Dekomposition, indem er eine Abstraktions- bzw. Vererbungshierarchie definiert, die den Event Types direkte Komponenten als kennzeichnende Attribute zuordnet.

Es muß noch erwähnt werden, daß zu Kautz' Formalismus vier Annahmen gehören, ohne die die Erkennung eines Plans nicht gerechtfertigt wäre. Die Dekompositions-Axiome, die letztlich für die Erkennung entscheidend sind, geben ja nur notwendige, aber keine hinreichenden Kriterien für die Identifikation eines Event Type an. Durch die vier Annahmen wird aus der Event Hierarchy durch Hinzufügen weiterer Axiome ein *vervollständigtes Modell* abgeleitet. Die Erkennung eines Plans wird dann durch eine *deduktive* Schlußfolgerung von einer Beobachtung auf Event Types (Pläne) möglich, die die beobachtete Aktion als Komponente enthalten. Kautz zeigt, daß sich das *vervollständigte Modell* durch Anwendung von *Circumscription* [McCarthy, 1980] auf Teile der Event Hierarchy herleiten läßt. Die Annahmen im einzelnen:

1. *Exhaustiveness*: Ein Event Type zerfällt vollständig in seine direkten Spezialisierungen.
2. *Disjointness*: Verschiedene Event Types, die nicht in Abstraktions-Relation stehen, sind disjunkt.
3. *Component/Use*: Jedes Exemplar eines Event Types ist ein Exemplar von *End Event* oder eine direkte Komponente eines Exemplars eines anderen Event Types.
4. *Minimum Cardinality*: Können aus mehreren Beobachtungen mehrere Exemplare x und y von *End Event* abgeleitet werden, wird angenommen, daß $x = y$ sofern das Gesamtmodell dann noch konsistent ist.

Annahme 1. läßt den Schluß auf eine Spezialisierung eines Event Types zu, wenn alle anderen seiner Spezialisierungen ausgeschlossen werden konnten. Aus Annahme 2. läßt sich ableiten, daß jede Beobachtung einen eindeutigen Basistyp (d.h. Event Type ohne Spezialisierungen) hat. Die dritte Annahme besagt, daß jede Beobachtung von einem durch die Event Hierarchy definierten Plan herrühren muß. Das heißt auch, daß die Menge der durch die Event Hierarchy definierten Pläne bzw. die Planbibliothek ein vollständiges Modell *aller* Pläne (des Anwendungsbereichs) ist. Damit impliziert eine Beobachtung, daß einer der Pläne die durch die Event Hierarchy definiert sind ausgeführt wird. Die *Minimum Cardinality* Annahme besagt, daß Beobachtungen zu einem oder möglichst wenigen gemeinsamen Plänen gehören, sofern nichts dagegen spricht.

Eine *deduktive* Schlußfolgerung von einer Beobachtung auf Pläne scheint im Widerspruch zur naheliegenden Ansicht zu stehen, daß Planerkennung ein *abduktives* Problem ist (vgl. [Shanahan, 1989]), das den Schluß von einer Beobachtung auf seine Ursache erfordert. Die Rechtfertigung ergibt sich aus der Annahme, daß jede Beobachtung Teil einer der durch die Event Hierarchy bzw. die Planbibliothek definierten Pläne sein *muß*. D.h. die Anzahl der Möglichkeiten für die Ursache einer Beobachtung ist begrenzt.

2.2.3 Planerkennung mit Beschreibungslogik

Die Anordnung von Plänen in einer Abstraktionshierarchie und die Form der Abstraktions- und Dekompositions-Axiome in Kautz' Event Hierarchy legen es nahe, Pläne direkt in terminologischer Logik bzw. Beschreibungslogik zu repräsentieren. Diese Idee wurde von Devanbu, Weida und Litman aufgegriffen und untersucht. Es wurden zwei unterschiedliche Ansätze verfolgt, die beide die Inferenzmechanismen von terminologischen Wissensrepräsentationssystemen (TWRS) nutzen und diese um eine Repräsentation der zeitlichen Aspekte eines Plans ergänzen. Für die verwendete Begrifflichkeit sei wiederum auf das Kapitel 3.2.1 *Beschreibungslogiken* verwiesen.

2.2.3.1 Modellierung von Aktionen und Plänen mit Hilfe von Zustand-Concepts

Im ersten Ansatz [Devanbu und Litman, 1991], [Devanbu und Litman, 1996] schlagen Devanbu und Litman eine Repräsentation von Aktionen mit den Möglichkeiten des TWRS CLASSIC [Borgida *et al.*, 1989], [Brachman *et al.*, 1991] vor. Der Ansatz kann nicht direkt als Arbeit zur Planerkennung angesehen werden, bei dem aus einer beobachteten Folge von Aktionen ein Plan und daraus Ziele abgeleitet werden. Allerdings ist die Aufgabe von Devanbu und Litmans CLASP-System – die Klassifikation von Plänen und Szenarien – sehr vergleichbar. Szenarien werden dabei als Exemplare von Plänen angesehen.

Devanbu und Litman lehnen sich bei der Beschreibung von Aktionen an STRIPS an[29], wobei für Vorbedingung, Hinzu-Liste und Lösch-Liste jeweils Konjunktionen von Aussagen zulässig sind. Die Repräsentation erfolgt durch CLASSIC-Concepts mit der Rolle *Actor* und den Attributen[30] *Precondition*, *Add-List*, *Delete-List* und *Goal* für den Akteur, der die Aktion durchführt, sowie für Vorbedingung, Hinzu-Liste, Lösch-Liste und Ziel[31]. Abb. 2.5 zeigt die Definition der allgemeinsten Aktion ihres Modells. Der Wertebereich aller vier Attribute ist das Concept *State*, einer direkten Spezialisierung des allgemeinsten Concepts *Classic-Thing*. Zustände können als vollständige Situationsbeschreibungen aufgefaßt werden[32]. Dementsprechend repräsentieren State-Concepts Mengen von Situationsbeschreibungen. Sowohl Vorbedingungen als auch die Bedingungen, die durch Ausführen einer Aktion erreicht werden, werden damit durch Einschränkungen der Menge aller möglichen Situationsbeschreibungen modelliert. Devanbu und Litman fordern, daß State-Concepts ausschließlich primitiv, oder durch eine Konjunktion von State-Concepts definiert werden dürfen. Ein primitives State-Concept modelliert eine einzelne Aussage, indem es alle die Zustände beschreibt, in denen diese Aussage wahr ist. Entsprechend modelliert eine Konjunktion von State-Concepts eine Konjunktion von Aussagen. Dies entspricht der in der Handlungsplanung üblichen Beschreibung einer Situation durch Konjunktion von Aussagen, wenn keine Axiome zulässig sind. Allerdings ist zu beachten, daß CLASSIC keine Negation zuläßt und es keine Entsprechung der *Closed-World-Annahme* für Situationsbeschreibungen gibt (s.u.).

Die so definierten Action-Concepts repräsentieren zwar Mengen von Aktionen, sie müssen aber von den Planoperatoren bzw. Aktions-Schemata der Handlungsplanung unterschieden werden, weil keine Parameter für die Aktionen definiert werden können. Das heißt, daß für jede Kombination von

[29] vgl. Kap. 2.1 *Handlungsplanung*

[30] Rollen mit der Anzahlbeschränkung *genau 1* modellieren funktionale Beziehungen und werden in CLASSIC *Attribute* genannt.

[31] Devanbu und Litman machen keine Angaben über den Zusammenhang der Attribute. In den Beispielen ist die Wertebeschränkung von *Goal* eine Konjunktion, die aus den Faktoren der Wertebeschränkungen von *Precondition* und *Add-List*, vermindert um die von *Delete-List* besteht.

[32] Devanbu und Litman geben keine Definition eines Zustandes an. Seine Verwendung und die Referenz auf STRIPS legen diese Interpretation aber nahe.

```
(DEFINE-CONCEPT Action (PRIMITIVE
    (AND Classic-Thing
        (AT-LEAST 1 ACTOR) (ALL ACTOR Agent)
        (EXACTLY 1 PRECONDITION) (ALL PRECONDITION State)
        (EXACTLY 1 ADD-LIST) (ALL ADD-LIST State)
        (EXACTLY 1 DELETE-LIST) (ALL DELETE-LIST State)
        (EXACTLY 1 GOAL) (ALL GOAL State))))
```

Abbildung 2.5: Definition eines Action-Concepts aus [Devanbu und Litman, 1996, S.7]

Objekten, die an einer Aktion beteiligt sein können, ein eigenes Action-Concept definiert werden muß.

Auch Pläne werden in CLASP als Concepts repräsentiert. Attribute sind *Initial*, *Goal* und *Plan-Expression* (s.a. Abb. 2.6). Analog zu Action-Concepts beschreiben die ersten beiden Attribute Vorbedingung und Ziel des Plans durch State-Concepts. Der Plan-Ausdruck beschreibt die möglichen Folgen von Schritten des Plans. Während die Zustände direkt mit CLASSIC repräsentiert werden, ist dies für Folgen nicht möglich. CLASP sieht dafür eine eigene Syntax und einen eigenen Inferenzmechanismus vor. Ein Plan-Ausdruck besteht aus Action-Concepts und Sub-Plänen, die durch Kontrollstrukturen angeordnet werden können, die mit denjenigen prozeduraler Programmiersprachen vergleichbar sind (s.a. Abb. 2.7). Die Sub-Pläne stehen wiederum für einen Plan-Ausdruck. Eine rekursive Definition ist allerdings nicht zulässig.

```
(DEFINE-PLAN Plan (PRIMITIVE
    (AND Clasp-Thing
        (EXACTLY 1 INITIAL) (ALL INITIAL State)
        (EXACTLY 1 GOAL) (ALL GOAL State)
        (EXACTLY 1 PLAN-EXPRESSION)
            (ALL PLAN-EXPRESSION (LOOP Action)))))
```

Abbildung 2.6: Definition eines Plan-Concepts aus [Devanbu und Litman, 1996, S.12]

```
plan-concept-expression ::=
    action-concept |
    (SEQUENCE plan-concept-expression+) |
    (LOOP plan-concept-expression) |
    (REPEAT integer plan-concept-expression+) |
    (TEST (state-concept plan-concept-expression)+ ) |
    (OR plan-concept-expression+) |
    (SUBPLAN symbol) .
```

Abbildung 2.7: EBNF-Beschreibung eines Plan-Ausdrucks aus [Devanbu und Litman, 1996, S.11]

Die Menge der Folgen von Action-Concepts die sich ergibt, wenn alle Sub-Pläne durch ihre Bestandteile ersetzt werden, läßt sich als formale Sprache auffassen, deren Alphabet die Menge der Action-Concepts ist. Devanbu und Litman zeigen, daß sich durch Plan-Ausdrücke reguläre Sprachen über Action-Concepts definieren lassen. Für reguläre Sprachen läßt sich sowohl die Frage, ob eine Symbolfolge Wort der Sprache ist, als auch die Frage, ob eine Sprache in einer anderen enthalten ist,

effektiv berechnen[33]. Devanbu und Litman nutzen dies für die Hauptaufgabe des CLASP-Systems, der Klassifikation von Szenarien und Plänen – also deren korrekte Einordnung in eine Abstraktionshierachie.

CLASP Szenarien sind Exemplare von Plan-Concepts. Sie können durch CLASP erzeugt werden, indem dem System Füller für die drei Attribute eines Plan-Concepts – *Initial*, *Goal* und *Plan-Expression* – angegeben werden. Die Rollenfüller dürfen dabei unterspezifiziert sein. Für *Initial* und *Goal* heißt dies, daß nicht unbedingt alle Aussagen die im jeweiligen Zustand wahr sind angegeben werden müssen. Der Rollenfüller von *Plan-Expression* muß eine strikte Folge von Aktionen sein, wobei deren Attribute *Precondition*, *Add-List*, *Delete-List* und *Goal* normalerweise nicht spezifiziert werden. Solche Beschreibungen können bspw. durch Beobachtung gewonnen sein.

Die für CLASP (= *CLA*ssification of *S*cenarios and *P*lans) wichtige Frage, ob ein solches Szenarium von einem Plan(-Concept) subsumiert wird, teilen Devanbu und Litman in drei Teilfragen auf:

1. subsumiert die Vorbedingung des Plans die Vorbedingung des Szenariums?
2. subsumiert das Ziel des Plans das Ziel des Szenariums?
3. subsumiert der Plan-Ausdruck des Plans die Aktionsfolge des Szenariums?

Wenn alle drei Bedingungen erfüllt sind, subsumiert der Plan das Szenarium. Die ersten beiden Fragen werden direkt von CLASSIC bearbeitet. Für die dritte sieht CLASP einen Algorithmus vor, der auf erweiterten endlichen Automaten basiert. Die Erweiterung ist notwendig, weil für das Szenarium eine Folge von Aktionen definiert ist, während ein Plan-Ausdruck Folgen von Action-*Concepts* beschreibt. Für Frage 3 muß der Automat also nicht entscheiden, ob eine Folge von Symbolen (hier den Aktionen) Wort einer Sprache ist, sondern ob diese Folge *Exemplar* eines Wortes der Sprache ist, die durch den Plan-Ausdruck definiert ist. Die Entscheidung, ob eine einzelne Aktion des Szenariums vom zugehörigen Action-Concepts im Plan-Ausdruck subsumiert wird, wird wiederum dem Inferenzmechanismus von CLASSIC überlassen.

Die Subsumption zwischen zwei Plan-Concepts A und B wird, wie in Beschreibungslogiken üblich, über die Teilmengen-Relation zwischen den Extensionen der Concepts definiert. Die Berechnung wird – analog zur Frage, ob ein Szenarium von einem Plan-Concepts subsumiert wird – aufgeteilt in die Fragen:

1. subsumiert die Vorbedingung von Plan-Concept A die des Plan-Concepts B?
2. subsumiert das Ziel von Plan-Concept A das von Plan-Concept B?
3. subsumiert der Plan-Ausdruck von Plan-Concept A den des Plan-Concepts B?

Auch hier werden die ersten beiden Fragen durch CLASSIC entschieden, während die dritte mit Hilfe eines erweiterten endlichen Automaten entschieden wird. Für die Feststellung, ob durch den Plan-Ausdruck von Plan-Concept B ausschließlich solche Aktionsfolgen definiert sind, die auch durch den Plan-Ausdruck von Plan-Concept A beschrieben werden, konstruieren Devanbu und Litman einen Automaten, der die Schnittmenge der durch die beiden Aktionsfolgen definierten Sprachen akzeptiert. Die Zustandsmenge ist dabei das kartesische Produkt der Zustandsmengen der Automaten für die einzelnen Sprachen. Auch hier muß berücksichtigt werden, daß durch die Plan-Ausdrücke Sprachen über Action-*Concepts* definiert werden, während aber Folgen von Aktionen erkannt werden sollen.

[33] Vgl. bspw. [Hopcroft und Ullman, 1979]. Hopcroft und Ullman zeigen nicht direkt wie die Teilmengenbeziehung festgestellt werden kann, sondern daß reguläre Sprachen unter Mengendurchschnitt und Komplementbildung abgeschlossen sind und daß ihre Schnittmenge, Komplement und Leerheit effektiv berechenbar sind. Die Teilmengenbeziehung ergibt sich dann aus $L_1 \subseteq L_2$ gdw. $L_1 \cap \overline{L_2} = \varnothing$.

Dies wirkt sich insbesondere auf die Zustandsübergänge aus, die nicht für einzelne Action-Concepts, sondern für Konjunktionen von nicht-disjunkten Action-Concepts definiert sein müssen. Seien bspw. AC_1 und AC_2 zwei nicht-disjunkte Action-Concepts, PA_1 und PA_2 zwei Plan-Ausdrücke, die Aktionsfolgen beschreiben, die mit AC_1 bzw. AC_2 beginnen und FA_1 und FA_2 die zugehörigen endlichen Automaten. Für den kombinierten endlichen Automaten FA_{12} müssen dann vom kombinierten Zustand (S_q, S_s) drei Zustandsübergänge für die Konjunktionen $AC_1 \sqcap \neg AC_2$, $AC_1 \sqcap AC_2$ und $\neg AC_1 \sqcap AC_2$ definiert sein (vgl. Abb. 2.8). Davon kann lediglich der Übergang mit $AC_1 \sqcap AC_2$ zu einem Endzustand führen, während die anderen beiden – sofern es sich um erfüllbare Concepts handelt – anzeigen, daß es evtl. Aktionsfolgen geben kann, die durch Plan-Concept A, nicht aber durch B beschrieben werden, bzw. umgekehrt.

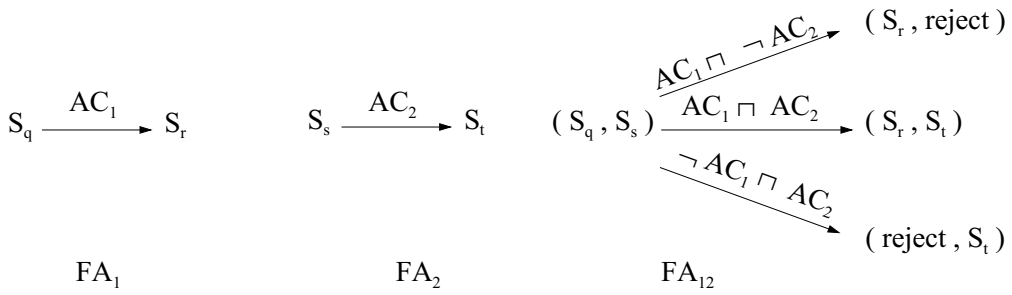

Abbildung 2.8: Zustandsübergänge bei der Kombination zweier erweiterter endlicher Automaten für die Schnittmenge von zwei regulären Sprachen

Es sei angemerkt, daß die Frage, ob ein Plan-Concept ein Szenarium bzw. ein anderes Plan-Concept subsumiert, nur dann positiv entschieden werden kann, wenn die Definitionen der Plan-Concepts *hinreichende* Beschreibungen – also als *Defined* Concepts – darstellen. Weitere Annahmen wie bspw. in Kautz' Ansatz sind hier aber nicht erforderlich.

Der Zusammenhang zwischen Vorbedingungen und Effekten von Action-Concepts eines Plan-Ausdrucks wird bei den Inferenzen weder in Bezug auf die Action-Concepts untereinander noch in Bezug zu dem Plan-Concept zu dem der Ausdruck gehört berücksichtigt. Lediglich bei der Definition von Szenarien werden einige Plausibilitätsprüfungen durchgeführt. Diese sollen sicherstellen, daß die Aktionsfolge des Szenariums auch das Ziel dieses Szenariums erreichen kann. Devanbu und Litman stellen dazu i.w. folgende Forderungen auf:

- Der Zustand unmittelbar nach einer Aktion A_i ist Rollenfüller des Attributs *Goal* des zugehörigen Action-Concepts und gleichzeitig Rollenfüller des Attributs *Precondition* des Action-Concepts der unmittelbar folgenden Aktion A_{i+1}. Dementsprechend muß der Zustand von den Werterestriktionen (VR) beider Attribute subsumiert werden. Zusätzlich wird gefordert, daß der Zustand von einem Concept S_{i+1} mit folgender Beschreibung subsumiert wird:

- $S_{i+1} := S_i - \mathrm{VR}(\textit{Delete-List}(A_i)) + \mathrm{VR}(\textit{Add-List}(A_i))$

Dabei steht S_i für das State-Concept, das die Situation vor Aktion A_i beschreibt. Die Operatoren + und – sind (vermutlich[34]) als Mengenoperatoren Differenz und Vereinigung auf den Konjunktionen der Concepts zu verstehen; d.h. die beschreibungslogische Entsprechung ist

$$(S_i \sqcap \neg\mathrm{VR}(\textit{Delete-List}(A_i))) \sqcup \mathrm{VR}(\textit{Add-List}(A_i))$$

[34]Devanbu und Litman geben keine Definition dafür an. Wiederum sei die Analogie zu STRIPS herangezogen.

Der Start dieser induktiven Definition wird nicht angegeben. Sinnvoll wäre die VR des Attributs *Precondition* der ersten Aktion. Devanbu und Litman gehen davon aus, daß die Rollenfüller der Attribute für die Vorbedingung und das Ziel der Aktionen nicht definiert sind und stattdessen durch CLASP gemäß der genannten Werterestriktionen gewählt werden.

Das Vorgehen modelliert den STRIPS-Ansatz allerdings nur unvollständig. Es fehlt insbesondere eine Entsprechung für die Closed-World-Annahme oder echte Negation. So wird bspw. die Undurchführbarkeit einer Aktionsfolge nicht erkannt, in der eine Aktion A_i eine Aussage p verneint[35] und die unmittelbar darauf folgende Aktion A_{i+1} p als Vorbedingung benötigt. Zwar wird das State-Concept *C(p)*, das die Aussage p modelliert, als Konjunktionsglied aus der Beschreibung des Concepts S_{i+1} für den Nachfolgezustand von A_i entfernt. Das bedeutet aber nur, daß S_{i+1} nicht von *C(p)* subsumiert wird; d.h. S_{i+1} beschreibt sowohl Zustände in denen p wahr ist als auch solche in denen p falsch ist. Da p für A_{i+1} wahr sein muß, wählt CLASP bei der Plausibilitätsprüfung einen Zustand in dem p wahr ist. Dies steht allerdings im Widerspruch zur STRIPS-Semantik der Lösch-Liste eines Operators, wonach p im Zustand direkt nach A_i *nicht* wahr ist.

2.2.3.2 Modellierung von Plänen durch Temporal Constraint Networks

Der Ansatz von Weida und Litman hat das Ziel, beobachtete Aktionen den Plänen einer Planbibliothek unter Zuhilfenahme eines Inferenzmechanismus für die Subsumption zuzuordnen [Weida und Litman, 1992], [Weida und Litman, 1994]. Wie Kautz gehen auch sie von der Vollständigkeitsannahme aus, daß alle beobachteten Aktionen Exemplare von definierten Concepts sind, die hier Action Types genannt werden. Die Action Types bilden eine Abstraktionshierarchie und werden im TWRS K-Rep [Mays *et al.*, 1991] repräsentiert. Für diese Concepts wird beispielhaft angegeben, daß für sie Rollen *agent* und *object* definiert sind. Ob und ggf. wie Vorbedingungen und Effekte von Action Types repräsentiert sind bleibt offen.

Für die Beschreibung von Plänen verzichten Weida und Litman auf Vorbedingungen und Effekte. Stattdessen wird ein Plan (-Concept[36]) in ihrem Ansatz lediglich durch seinen sogenannten *Plan-Rumpf* beschrieben. Ein Plan-Rumpf besteht aus einem Plan-Netzwerk, das aus einer Menge von Action Types besteht, die – wie in Kautz' Ansatz – durch Allens Zeitrelationen [Allen, 1984] in Beziehung gesetzt werden[37]. In einem solchen Netzwerk bilden die Action Types die Knoten und die zeitlichen Beziehungen die Kanten. Da die Zeitrelationen Bedingungen an die Anordnung der Aktionen stellen, nennen Weida und Litman ein solches Netzwerk *Temporal Constraint Network* (TCN). Anstelle von Action Types dürfen in einem Plan-Netzwerk auch Pläne vorkommen. Sie haben die Funktion von Makros; d.h. sie stehen lediglich für das Plan-Netzwerk ihres Rumpfes und haben keinerlei sonstige Bedeutung. Inferenzen werden grundsätzlich auf der Basis von expandierten Makros gezogen. Deshalb werden solche Makros nicht weiter beachtet, sondern es wird generell davon ausgegangen, daß Plan-Netzwerke lediglich Action Types enthalten.

Die Frage, ob ein Plan einen zweiten subsumiert, kann durch die Frage nach der Subsumption der zugehörigen Plan-Netzwerke ersetzt werden, weil die Pläne gerade durch die Netzwerke definiert werden. Die Subsumption basiert – wie in Beschreibungslogiken üblich – auf der Mengeninklusion der Extensionen; für Plan-Netzwerke also auf den Mengen zeitlich angeordneter Aktionen. Weida und Litman nutzen die in [Weida, 1996] bewiesene Aussage, daß ein Plan-Netzwerk *T1* genau dann ein Plan-Netzwerk *T2* subsumiert, wenn es eine sogenannte *Subsumptions-Abbildung* von *T1* nach

[35] indem das State-Concept das p modelliert die Werterestriktion des Attributs *Delete-List* der Aktion subsumiert.

[36] Pläne, wie sie Weida und Litman beschreiben, sind beschreibungslogische Concepts. In Anlehnung an die Terminologie der Autoren wird der Begriff *Plan* grundsätzlich für Plan-Concepts verwendet.

[37] Genaugenommen werden die Zeitintervalle, in denen die Aktionen der Concepts ausgeführt werden können, in Relation gesetzt. Im weiteren werden die Zeitintervalle immer durch die zugehörigen Aktionen oder Aktions-Concepts identifiziert.

T2 gibt. Eine Subsumptions-Abbildung von Plan-Netzwerk *T1* nach Plan-Netzwerk *T2* bildet jeden Knoten von *T1* auf einen Knoten von *T2* ab. Dabei müssen die Knoten von *T1* die zugehörigen von *T2* jeweils subsumieren, und sie müssen jeweils auf verschiedene Knoten von *T2* abgebildet werden. Außerdem müssen die Zeitbedingungen zwischen Knoten aus *T1* diejenigen zwischen den zugehörigen Knoten aus *T2* subsumieren. Weida und Litman machen dies am Beispiel der Plan-Netzwerke *Heat-Noodles* und *Assemble-Spaghetti-Marinada* deutlich (s.Abb. 2.9), wobei sie als Sachbereich Kautz' *Cooking-World* gewählt haben. Im Beispiel subsumieren die Action Types *C-Make-Noodles* bzw. *C-Heat* die Concepts *C-Make-Spaghetti* bzw. *C-Boil* (fette gestrichelte Pfeile). Da Action Types im TWRS K-Rep repräsentiert sind, kann diese Subsumption von K-Rep berechnet werden.

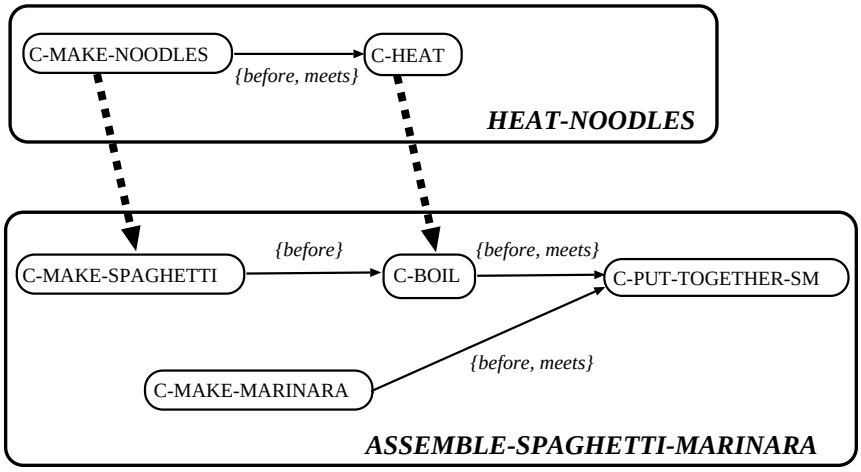

Abbildung 2.9: Plan Subsumption aus [Weida und Litman, 1992, S.284]

Weida und Litman definieren die Subsumption zwischen Zeitbedingungen ebenfalls durch Mengeninklusion: Seien *C1* und *C2* zwei Zeitbedingungen, die jeweils durch eine Menge von implizit disjunktiv verknüpften Allen'schen elementaren Zeitbeziehungen beschrieben sind, dann subsumiert *C1 C2* genau dann, wenn die zu *C1* gehörende Menge eine Obermenge der zu *C2* gehörenden Menge ist. Im Beispiel subsumiert die Zeitbedingung zwischen *C-Make-Noodles* und *C-Heat* diejenige zwischen *C-Make-Spaghetti* und *C-Boil*. Im allgemeinen Fall müssen für die Subsumption auch die impliziten Zeitbedingungen berücksichtigt werden. Inferenzen, die die zeitlichen Abhängigkeiten betreffen, werden mit dem Teilsystem MATS [Kautz und Ladkin, 1991] gezogen. Weida zeigt in [Weida, 1996], daß die Feststellung der Subsumption zwischen zwei Plan-Netzwerken ein NP-vollständiges Problem ist und argumentiert deshalb für einen unvollständigen Algorithmus.

Die so definierte Subsumption basiert auf der Menge der Action Types und der Zeitbedingungen, wobei ein Plan einen anderen subsumiert, wenn für ihn weniger und/oder allgemeinere Concepts bzw. Bedingungen definiert sind. Diese Eigenschaft ist der Schlüssel für die eigentliche Aufgabe dieses Ansatzes, der Planerkennung. Die Frage, ob die Beobachtung von Aktionen darauf schließen läßt, daß ein Plan ausgeführt wird, bilden Weida und Litman auf die Frage ab, ob der Plan bzw. das zugehörige Plan-Netzwerk die Beobachtung bzw. das zugehörige Beobachtungs-Netzwerk subsumiert oder zumindest *potentiell subsumiert*. Ein Beobachtungs-Netzwerk besteht dabei – analog zum Aufbau von Plan-Netzwerken – aus Aktionen mit zeitlichen Beziehungen. Die Definition für *Plan-Netzwerk P subsumiert Beobachtung-Netzwerk O potentiell* teilen Weida und Litman in drei *alternative* Bedingungen auf:

1. P subsumiert[38] O
2. es existiert eine *inverse Subsumptions-Abbildung* von O nach P
3. P subsumiert ein Plan-Netzwerk P', das O potentiell subsumiert.

Eine inverse Subsumptions-Abbildung von O nach P ordnet jeder Aktion A von O einen Action Type C von P zu, der A subsumiert. Die Zeitbeziehungen zwischen den Aktionen müssen von den Zeitbeziehungen der zugeordneten Action Types subsumiert werden. Jede beobachtete Aktion muß also ihre Entsprechung in dem Plan-Netzwerk haben. Es sei angemerkt, daß diese Bedingung strenger ist als bei der Subsumption. Aus P subsumiert O kann also nicht auf die Existenz einer inversen Subsumptions-Abbildung von O nach P geschlossen werden. Auch die Umkehrung gilt nicht. Allerdings kann ein Beobachtungsnetzwerk O, für das eine inverse Subsumptions-Abbildung nach P existiert, evtl. zu einem *späteren Zeitpunkt* von P subsumiert werden, wenn dem Netzwerk durch neue Beobachtungen weitere Knoten hinzugefügt wurden.

Weida zeigt in [Weida, 1996], daß eine Beobachtung genau dann Exemplar eines Plans ist oder zu einer ergänzt werden kann, wenn das zugehörige Beobachtungs-Netzwerk von dem zugehörigen Plan-Netzwerk potentiell subsumiert wird und folgende Voraussetzungen erfüllt sind:

- die *Planbibliothek ist vollständig*
- die *Beobachtung ist perfekt* (jede beobachtete Aktion läßt sich einem Blatt der Aktions-Hierarchie zuordnen, die beobachteten zeitlichen Beziehungen bestehen jeweils nur aus einer elementaren Zeitrelation und keine Beobachtung muß modifiziert oder zurückgenommen werden)
- alle Beobachtungen gehören zu *genau einem Plan*.

Eine Beobachtung einer Aktion oder einer Folge von Aktionen teilt die Menge der Pläne in folgende vier Partitionen ein:

1. Die *notwendigen Pläne*: die Menge der Pläne, die die Beobachtung subsumieren.
2. Die *direkt optionalen* Pläne: die Menge der Pläne, die die Beobachtung zwar nicht subsumieren, für die es aber eine inverse Subsumptions-Abbildung von der Beobachtung gibt.
3. Die *indirekt optionalen* Pläne: die Menge der Pläne, die weder notwendig, noch direkt optional sind, die aber einen direkt optionalen Plan subsumieren.
4. Die *unmöglichen* Pläne: alle verbleibenden Pläne.

Die Voraussetzungen der perfekten Beobachtung und daß alle beobachteten Aktionen zu einem Plan gehören, werden von Weida und Litman noch abgeschwächt. Darauf soll hier aber nicht weiter eingegangen werden, ebenso wie auf ihre Erweiterung der (qualitativen) Zeitrelationen um quantitative Angaben und Gleichheitsbedingungen, die dann zu sogenannten *Qualitative, Metric and Equality Networks* führen. Erwähnt werden soll aber dennoch, daß sie die Aktions-Hierarchie nutzen, um auch nicht perfekte Beobachtungen zuzulassen. Läßt sich eine “ungenaue” Beobachtung einer Aktion nicht eindeutig einem Blatt der Aktions-Hierarchie zuordnen, wird sie als Exemplar eines abstrakteren Action Types aufgefaßt.

[38] Die Subsumption zwischen Plan-Netzwerken wird dafür in naheliegender Weise um die Relation zwischen einem Plan-Netzwerk und einem Beobachtungs-Netzwerk erweitert; d.h. ein Plan-Netzwerk P subsumiert ein Beobachtungs-Netzwerk O genau dann, wenn jedem Knoten C aus P ein Knoten A aus O zugeordnet wird, der Exemplar von C ist. Die Zuordnung zwischen den Zeitrelationen wird unverändert übernommen.

2.2.4 Weitere Ansätze der Planerkennung

Um das Gebiet der Planerkennung abzurunden, sei hier noch kurz auf zwei Aspekte eingegangen, die aber keine grundlegenden Neuerungen in Bezug auf die Vorstellung und/oder Repräsentation von Plänen haben.

Die Verarbeitung einer natürlichsprachlichen Äußerung war ein Ausgangspunkt für die Forschung zur Planerkennung. Die erwähnten Modelle und Techniken wurden bewußt unabhängig von einem Einsatzgebiet entwickelt und berücksichtigen die Eigenschaften eines natürlichsprachlichen Dialogs höchstens marginal. Im Gegensatz zu den bisherigen Ansätzen, die die beobachteten Aktionen als lineare Folge ansehen mit der der Benutzer Ziele erreichen will, haben Dialoge – insbesondere Informationsbeschaffungs- oder Beratungs-Dialoge – eine komplexe Struktur (vgl. bspw. [Nothdurft, 1984]). Abweichungen vom zielgerichteten linearen Vorgehen ergeben sich bspw. durch Fokuswechsel [Grosz, 1981] und Klärungsdialoge [Litman und Allen, 1984]. Carberry, Kazi und Lambert und schlagen deshalb in [Lambert und Carberry, 1991], [Carberry *et al.*, 1992] ein Verarbeitungsmodell mit drei Ebenen vor:

- der *Diskurs-Ebene*, auf der die spezifischen Probleme der natürlichen Sprache behandelt werden, zu denen bspw. die Auflösung von Referenzen und die Wahl einer adäquaten Formulierung gehören.

- der *Problemlösungs-Ebene*, auf der die Strategie zur Problemlösung festgelegt wird. Dazu gehört bspw. auch die Frage, ob für ein Ziel ein (Sachbereichs-) Plan erstellt werden soll, ob ein fertiges Rezept genutzt werden kann oder ob ein Lösungsweg durch Exploration gefunden werden soll.

- der *Sachbereichs-Ebene*, zu der die Pläne gehören, mit denen die konkreten Ziele im jeweiligen Anwendungsgebiet erreicht werden können.

Die Autoren vergleichen ihr Modell mit den Ansätzen zum *Meta-Planen* sowohl bei der Verarbeitung natürlicher Sprache [Wilensky, 1981] als auch bei der Handlungsplanung [Stefik, 1981].

Ein generelles Problem aller bisher genannten Ansätze besteht darin, daß die Verwendung einer umfassenden Planbibliothek (um möglichst alle Beobachtungen erklären zu können) dazu führt, daß einzelne Beobachtungen zu sehr vielen Plänen passen und damit viele Beobachtungen notwendig sind um einen Plan und damit das angestrebte Ziel erkennen zu können. Dies steht im Widerspruch zum Zweck der Planerkennung, möglichst rasch Plan und Ziele des Benutzers zu erkennen, um möglichst früh eine adäquate Unterstützung oder Reaktion erzeugen zu können. In vielen Anwendungsgebieten sind die a priori Wahrscheinlichkeiten für die Ausführung von Plänen und Aktionen unterschiedlich. Dies läßt sich zur Bewertung von Plan-Kandidaten nutzen, die sich auf Grund der Beobachtung(en) ergeben. Bauer stellt in [Bauer, 1996] einen Ansatz vor, der auf der *Dempster-Shafer-Theorie* [Dempster, 1967], [Shafer, 1976] beruht und u.a. auch die Verarbeitung unsicherer Beobachtungen zuläßt. Die Arbeit enthält zudem einen ausführlichen Vergleich zu anderen Arbeiten zur Bewertung von Erkennungsalternativen.

2.2.5 Zusammenfassung

Die Vorstellungen über die Modellierung von Plänen beruhen im Bereich Planerkennung auf den Ansätzen in der Handlungsplanung. Da es sich bei den einzelnen Beobachtungen um die Komponenten eines Plans – Planschritte oder Aktionen – handelt, kommt der Beziehung zwischen einem Plan und seinen Dekompositionen besondere Bedeutung zu. Praktisch alle Ansätze gehen deshalb

von einem Modell von Plänen aus, das dem beim hierarchischen Planen mit Operatorabstraktion entspricht. Vorbedingungen und Effekte spielen hier, im Gegensatz zur Handlungsplanung, aber nur eine untergeordnete Rolle oder werden, wie bspw. im Ansatz von Weida und Litman, gar nicht modelliert.

Praktisch alle Ansätze zur Planerkennung gehen davon aus, daß die in dem Anwendungsbereich relevanten Pläne vollständig in einer Bibliothek repräsentiert sind. Sowohl für die Erkennungsaufgabe selbst als auch für Erklärungen und Hinweise an den Benutzer ist es wichtig, daß viele Pläne eines Sachbereichs in Relation zueinander stehen. Ein Plan kann bspw. eine Spezialisierung eines anderen sein, weil er spezifischere Objekte manipuliert oder weil er speziellere Anforderung an die Ausführung hat als der allgemeinere Plan. Außerdem kann ein Plan ein Teilbaustein bzw. eine Komponente eines übergeordneten Plans sein; d.h. der Plan entspricht damit einem Planschritt des übergeordneten Plans. Diese Vorstellung wurde von Kautz formalisiert und ist heute die Basis für die Repräsentation von Planbibliotheken in der Planerkennung. Wichtig ist dabei, daß Kautz die beiden wichtigsten Relationen, die *Abstraktion* und die *Ganzes-Teil-Beziehung* bzw. *Dekomposition*, explizit in der Planbibliothek repräsentiert hat. Seine Anordnung der Pläne in einer Abstraktionshierarchie kommt der häufig eingesetzten Repräsentation des begrifflichen Wissens eines Benutzers (oder generell eines Akteurs) durch Beschreibungslogik nahe.

(Standard-) Beschreibungslogiken sehen allerdings weder Konstrukte für die Modellierung von *Ganzes-Teil-* noch von zeitlichen Beziehungen vor[39]. Devanbu, Weida und Litman haben die Idee der Repräsentation von Plänen durch Beschreibungslogik auf zwei unterschiedliche Arten realisiert. Die beiden unterschiedlichen Realisierungen resultieren aus den unterschiedlichen Aufgaben der Ansätze. Die Unterschiede zeigen sich darin, welche Eigenschaften von Plänen *direkt in der Beschreibungslogik*, welche in einer *spezifischen Erweiterung der Logik* und welche *nicht* bzw. *nur eingeschränkt* repräsentiert wurden. Bemerkenswert ist dabei, daß der Ansatz von Weida und Litman zu einer Abstraktionshierarchie führt, die deutlich von der üblichen Vorstellung von Abstraktion abweicht. Die Ursache liegt darin, daß Weida und Litman ein Plan-Concept lediglich durch seine Komponenten und den zugehörigen zeitlichen Bedingungen beschreiben. Sie verzichten aber auf eine Modellierung von Vorbedingung und Effekt eines Plans. Im Beispiel ihrer Plan Taxonomie (s.Abb. 2.10) gibt es sowohl

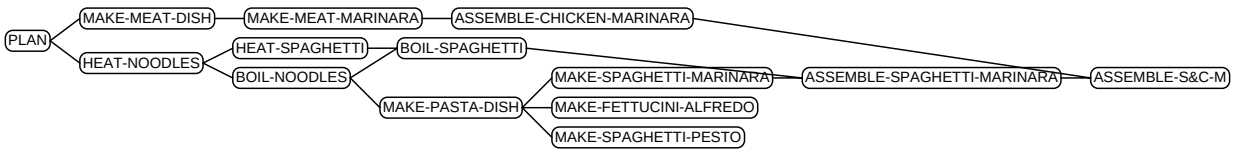

Abbildung 2.10: Plan Taxonomie aus [Weida und Litman, 1992, S.286]

eine Abstraktionsrelation zwischen Plan-Concepts, bei denen das speziellere Plan-Concept lediglich *speziellere Objekte* manipuliert (bspw. *Heat-Noodles* $\Leftarrow$ *Heat-Spaghetti*), als auch zwischen Plan-Concepts, bei denen das speziellere Plan-Concept *mehr Komponenten* hat (bspw. *Boil-Spaghetti* $\Leftarrow$ *Assemble-Spaghetti&Chicken-Marinara*). Die Namen der Concepts suggerieren, daß ein Concept für die Menge der Pläne steht, die sich mit diesem Namen bezeichnen lassen und damit auch die wesentlichen Eigenschaften haben, die mit der Bezeichnung verbunden sind. Dazu gehören bei einem Plan zweifelsohne auch die Vorbedingung und insbesondere der Effekt, der normalerweise der Grund für das Ausführen des Plans ist. Bei einer solchen Interpretation wäre es zumindest fragwürdig, wieso das Anrichten eines *Spaghetti-Hühnchen-Gerichts* eine spezielle Form des *Spaghettikochens* sein soll. Die Klassifikation in einem TWRS darf sich aber ausschließlich nach der formalen Beschreibung

[39] vgl. bspw. [Patel-Schneider und Swartout, 1993] u. Kap. 4.6.1 *Teil-Ganzes-Relationen in Beschreibungslogik*

richten, und die beschränkt sich im Ansatz von Weida und Litman auf die Komponenten eines Plans. Damit ist bspw. jeder Plan eine Spezialisierung des leeren Plans, der auch die völlige Untätigkeit beschreibt. Als Basis für eine Interaktion mit einem Benutzer, die ja auf dem Austausch von Begriffen beruht, ist diese Modellierung allerdings ungeeignet, weil ihr durch den Verzicht auf eine Repräsentation von Vorbedingungen und Effekten wesentliche Eigenschaften eines Plans fehlen. Beispielsweise kann ein Plan *der ein bestimmtes Ziel erreichen soll* mit diesem Ansatz nicht modelliert werden und steht damit auch nicht für die Interaktion mit einem Benutzer zur Verfügung.

2.3 Gemeinsames Planen (mit einem Benutzer)

Für die Erzeugung eines Plans durch ein Planungssystem ist in jedem Fall eine Interaktion mit dem Benutzer erforderlich: zumindest die Problemstellung muß durch den Benutzer in einer für das Planungssystem geeigneten Weise formuliert werden. Viele Planungssysteme sehen aber auch darüber hinaus einen Benutzereingriff vor, um den Planungsprozeß selbst zu steuern oder zu lenken (vgl. [Chien *et al.*, 1995]). Einige Systeme sehen einen Eingriff auf der Ebene einzelner Planungsentscheidungen vor (vgl. [Tate *et al.*, 1994], [Drabble und Tate, 1995], [Wilkins, 1993]). Ein fruchtbarer Eingriff auf dieser Ebene erfordert vom Benutzer allerdings sehr gute Kenntnisse der internen Struktur des Planungssystems und der Modellierung des Planungssachbereichs. Myers [Myers, 1996] stellt dem ein System gegenüber, das dem Benutzer Einfluß auf einer abstrakteren Ebene ermöglicht. Über eine quasi-natürlichsprachliche Schnittstelle kann der Benutzer Teilziele und Randbedingungen bestimmen und diese auch in Bezug zum Ausführungskontext setzen. Bspw. kann der Benutzer bestimmen, daß ein Ziel auf eine bestimmte Weise oder nur unter bestimmten Bedingungen erreicht werden soll. Diese Anweisungen werden dann in eine Menge von Randbedingungen für den Planungsprozeß übersetzt, der dann aber ohne Eingriff des Benutzers erfolgt.

Ein anderer Ansatz ergibt sich, wenn der Planungsprozeß selbst auf Benutzer und System oder, verallgemeinert, auf zwei oder mehrere Akteure verteilt ist. Ein solcher Ansatz ist vor allem dann vorteilhaft, wenn die verschiedenen Akteure unterschiedliche Fähigkeiten haben. Durch Zusammenarbeit können dann oft komplexere Probleme gelöst oder bessere Ergebnisse erzielt werden, als dies mit jedem einzelnen Akteur möglich wäre.

Im Beispiel der Zusammenarbeit von Mensch und Computer beim Planungsprozeß liegen wichtige spezifische Fähigkeiten des Menschen in der Berücksichtigung vager Kriterien und Präferenzen, die für eine Beurteilung der Tauglichkeit des Plans in der realen Welt oft eine wichtige Rolle spielen. Solche Kriterien lassen sich meist nicht oder nur mit großem Aufwand formalisieren und entziehen sich damit (praktisch) einer Bearbeitung durch einen Computer. Die spezifischen Stärken des Computers sind bei diesem Prozeß die Verwaltung und vollständige Berücksichtigung aller Abhängigkeiten und Randbedingungen. Eine naheliegende Konstellation für ein gemeinsames Planen von einem Computer mit seinem Benutzer wäre deshalb, daß der Benutzer einen unvollständigen Plan als Vorgabe entwickelt, dem Computer mitteilt und dieser den Plan dann prüft, vervollständigt oder auch Gegenvorschläge macht, wenn der Plan inkonsistent ist. Ein solcher *Planungsassistent* war Ziel des Forschungsprojekts TRAINS [Allen und Schubert, 1991], [Allen *et al.*, 1995], [Ferguson *et al.*, 1996b] und seines Nachfolgers TRIPS [Ferguson und Allen, 1998] an der Universität Rochester, auf die im nächsten Abschnitt noch näher eingegangen wird.

Beim Ansatz des *hybriden* Planens von Kambhampati, Tenenbaum und Lee [Kambhampati *et al.*, 1993] wird das gemeinsame Planen durch eine Zusammenarbeit von spezialisierten Problemlösungskomponenten realisiert. Die Komponenten arbeiten jeweils nach einem Algorithmus, der für eine bestimmte Klasse von Problemstellungen besonders gut geeignet ist. Die Kontrolle über den

gesamten Planungsprozeß erfolgt von zentraler Stelle; d.h. der Kontrollprozeß entscheidet, welcher spezielle Problemlösungsalgorithmus für welche Teilaufgabe eingesetzt wird.

Arbeiten mehrere Akteure ohne eine solche zentrale Kontrollinstanz zusammen, spricht man vom *Multiagent*- oder *verteilten* Planen. In [desJardins *et al.*, 1999] wird ein grober Überblick und eine Einteilung des Gebiets gegeben. Die Autoren sehen verteiltes Planen und auch die Aufteilung einer Gesamtaufgabe auf mehrere Akteure als Weiterentwicklung des klassischen Planens an, die ohne dessen vereinfachte Annahmen (vgl. Kap. 2.1.3 *Voraussetzungen beim Planen*) auskommt und so auch für komplexere Probleme geeignet ist. Für den Zweck dieser Arbeit sei im übernächsten Abschnitt ein Ansatz herausgegriffen, der Pläne als mentale Zustände ansieht und einen Aspekt behandelt, der sowohl für das verteilte Planen als auch generell für die Interaktion zwischen einem Benutzer und einem Computersystem wichtig ist: die Verhandlung darüber, welche Aufgaben im einzelnen von welchem Akteur zu erledigen sind. Dies beinhaltet auch die Angleichung ggf. unterschiedlicher Ansichten über Faktoren, die die Planung beeinflussen können.

2.3.1 Planungsunterstützung durch TRAINS/TRIPS

Ziel des Forschungsprojekts TRAINS [Allen und Schubert, 1991], [Allen *et al.*, 1995], [Ferguson *et al.*, 1996b] und seines Nachfolgers TRIPS [Ferguson und Allen, 1998] an der Universität Rochester war bzw. ist die Entwicklung eines Systems zur Planungsunterstützung eines Benutzers. Ein wichtiger Aspekt beider Projekte ist die *natürlichsprachliche* Interaktion zwischen Benutzer und Computer, die allerdings für diese Arbeit nicht im Zentrum des Interesses steht.

Bei der Verarbeitung von Plänen geht es im Gegensatz zum Namen (TRIPS = The Rochester Interactive *Planning System*) nicht primär um die Erzeugung eines Plans, sondern – wie für einen Planungsassistenten bereits gesagt – um die Prüfung, Vervollständigung und ggf. Modifikation eines Plans, der aber normalerweise vom Benutzer vorgegeben wird. Für den für die Projekte gewählten Beispielsachbereich *Transportaufgaben* ist dies angebracht, weil die Schwierigkeit von typischen Problemstellungen hier in erster Linie in einer geeigneten Parametrierung von Aktionen liegt und nicht in der Auswahl oder Anordnung der Aktionen. Der TRAINS-Prototyp von 1996 sieht für den Beispielsachbereich ohnehin nur die beiden Aktionen *go* und *stay* vor (vgl. [Ferguson *et al.*, 1996b, S.44]). Diese Aktionen können nicht direkt mit den Planoperatoren der Handlungsplanung verglichen werden. Vielmehr hat Ferguson in [Ferguson, 1994][40] ein Modell von *Handeln* und *Veränderung* entwickelt, daß sich nicht nur für die *Planerzeugung*, sondern auch für die *Vorhersage von Zuständen* und für die *Planerkennung* eignet. Viele Ansätze der *Planerkennung* haben die Modellierung von Plänen stark an die (heutige) Vorstellung in der Handlungsplanung angelehnt. Eine *Vorhersage von Zuständen*, wie sie im Situationskalkül[41] möglich war, wurde in der Handlungsplanung aber zugunsten einer effizienteren Lösung von Planungsproblemen durch den STRIPS-Ansatz eliminiert.

Ferguson und Allen trennen den Effekt von der Aktion und repräsentieren explizit die Zeitintervalle der Aktionsdurchführung und der Gültigkeit von Vorbedingungs- und Effekt-Komponenten. Eine (versuchte) Ausführung einer Aktion kann dabei Ursache für ein *Ereignis* (*Event*) sein, das zu der Aktion gehört und normalerweise im gleichen Zeitintervall stattfindet wie die Aktion. Das Ereignis muß aber nicht notwendigerweise stattfinden (s.u.). Der Effekt einer Aktion tritt nur dann ein, wenn das zugehörige Ereignis stattfindet. Er ist deshalb konsequenterweise mit dem Ereignis und nicht der Aktion verbunden.

Mit diesem Modell sind – wie im Situationskalkül – Schlüsse auf die Gültigkeit einzelner Aussagen in einzelnen Situationen bzw. hier Zeitintervallen möglich. Allerdings muß hier nicht der gesamte Effekt in der Situation unmittelbar nach Ausführung der Aktion erreicht werden, sondern für

[40] s.a. [Ferguson und Allen, 1994] und [Allen und Ferguson, 1994]

[41] vgl. [McCarthy und Hayes, 1969] u. Kap. 2.1 *Handlungsplanung*

die einzelnen Effekt-Komponenten können unterschiedliche Zeitintervalle der Gültigkeit angegeben werden, die durch Allens Zeitrelationen [Allen, 1984] angeordnet werden können. Alle diese Zeitintervalle dürfen grundsätzlich frühestens mit dem Zeitintervall des Ereignisses beginnen. Anstelle von Zeitintervallen können – je nach Zeitmodell – auch Zeitpunkte verwendet werden. Die in diesem Modell verwendete explizite Repräsentation von Ereignissen geht auf Kowalski und Sergots Ereigniskalkül [Kowalski und Sergot, 1986] zurück. Der Zusammenhang zwischen Aktionen und Ereignissen ist sehr vergleichbar zum Ansatz von McDermott [McDermott, 1985], dessen Zeitmodell allerdings auf Zeitpunkten und nicht auf Zeitintervallen beruht.

Eine weitere Problematik, die Ferguson und Allen mit ihrem Modell von Handeln und Veränderung angehen, ist das *Qualifikationsproblem* – das Problem, daß niemals alle Einzelbedingungen die für eine erfolgreiche Ausführung einer Aktion erforderlich sind modelliert werden können. Es wurde bereits zu Beginn des Abschnitts über die Handlungsplanung (s. Kap. 2.1) auf dieses Problem hingewiesen. Dort wurde der Entwickler des Sachbereichsmodells für die Entscheidung, welche Merkmale im Sachbereich relevant sind und deshalb für die Vorbedingungen von Operatoren berücksichtigt werden müssen, verantwortlich gemacht. Diese Entscheidung ist für ein Planungssystem bindend; d.h. ein Planungssystem wird nur solche Pläne erzeugen, bei dem (im Modell) sichergestellt ist, daß die definierten Vorbedingungen der Planoperatoren jeweils vor dem Ausführen der zughörigen Planschritte vollständig erfüllt sind. Merkmale, die als irrelevant eingestuft wurden, werden demgegenüber gar nicht berücksichtigt. Nichtsdestotrotz können sie in ungünstigen Fällen dazu führen, daß eine Aktion und damit ein Plan undurchführbar ist.

Ferguson und Allen greifen deshalb für ihr Modell die Idee von *Argumentations-Systemen* [Loui, 1987], [Pollock, 1987], [Pollock, 1992] auf. Die (versuchte) Ausführung einer Aktion wird zusammen mit der Vorbedingung der Aktion als *Argument für ein Ereignis*, das zu der Aktion gehört, modelliert. Findet dieses Ereignis tatsächlich statt, so gelten dann auch die Aussagen, die durch den Effekt der mit dem Ereignis verbunden ist beschrieben sind. Ist die Vorbedingung einer Aktion nicht erfüllt, ist dies ein *Argument gegen* das zu der Aktion gehörende Ereignis. Allerdings wollen sich Ferguson und Allen nicht von vorn herein auf eine feste Menge von Bedingungen für eine Aktion festlegen. Vielmehr gilt bereits die (versuchte) Ausführung einer Aktion als Argument *für* das Auftreten des zugehörigen Ereignisses. Allerdings ist dieses Argument sehr schwach, weil es maximal *unspezifisch* ist (s.u.). Außerdem gilt das gleiche Argument auch *gegen* das Auftreten des Ereignisses.

Formal beschreiben Ferguson und Allen den Zusammenhang zwischen Aktionen und Ereignissen durch widerlegbare Regeln (*Defeasible Rules*), die im Sinne von Argumentations-Systemen *Argumentschritte*[42] darstellen. Diese *Action Definition* Regeln enthalten – analog zu den Operatoren der Handlungsplanung – normalerweise freie Variablen. Die Regeln haben die Form:

$$p_1 \wedge \ldots \wedge p_n \wedge Try(\pi, t) \rightarrow \exists e\, Event(e) \wedge t \circ time(e) \wedge \phi$$

Dabei sind $p_1, \ldots, p_n$ die Prädikate, die den Vorbedingungskomponenten von Aktion π entsprechen. Die Argumente der Prädikate sind hier nicht dargestellt. Jedes Prädikat hat aber zumindest eine Argumentstelle für das Zeitintervall seiner Gültigkeit. Das Prädikat *Try* ist wahr, wenn eine Aktion π während des Zeitintervalls t durchgeführt wird. *Event* steht für das Prädikat, das den Ereignistyp angibt. ϕ ist eine Formel, die weitere Bedingungen an das Ereignis beschreibt – oft sind dies Bindungsbedingungen. "$\circ$" steht für eine von Allens Zeitrelationen (normalerweise *meets*[43] oder *equal*); $time(e)$ gibt das Zeitintervall an, in dem das Ereignis e stattfindet. Das Symbol "$\rightarrow$" steht hier *nicht* für eine logische Implikation, sondern für eine widerlegbare Verbindung zwischen der Aussage links von "$\rightarrow$" (der Menge der *Premises*) und der Aussage rechts von "$\rightarrow$" (der *Conclusion*). Sie besagt

[42] Argumente bestehen i.a. aus mehreren Argumentschritten (vgl. bspw. [Ferguson, 1994, Kap.3.2])

[43] *meets*(a, b), auch geschrieben a:b, besagt, daß das Ende des Zeitintervalls a mit dem Beginn des Zeitintervalls b zusammenfällt.

lediglich, daß die Wahrheit der Konjunktion der *Premises für* die Wahrheit der *Conclusion* spricht, sofern nicht eine spezifischere Regel *dagegen* spricht; d.h. die *Conclusion* negiert. Eine *Action Definition* besagt also, daß das Ausführen von Aktion π im Zeitintervall t dafür spricht, daß in einem Zeitintervall t oder unmittelbar danach ein Ereignis vom Typ *Event* stattfindet.

Eine widerlegbare Regel A ist *spezifischer* als B, falls die Konjunktion der *Premises* von A die von B impliziert, aber nicht umgekehrt. Abb. 2.11 zeigt den Verband, der sich aus der Relation *spezifischer* und den widerlegbaren Regeln die *für* ein Ereignis sprechen ergibt. Entsprechend gibt es einen Verband der widerlegbaren Regeln *gegen* das gleiche Ereignis. Die Zahl der Vorbedingungskomponenten, die repräsentiert werden können, ist zwar auch hier praktisch begrenzt. Allerdings erreicht der Ansatz zusätzliche Flexibilität, weil der Modellierer mehr Bedingungen berücksichtigen kann, ohne daß diese in jedem Fall *alle* bei der Verarbeitung der Pläne berücksichtigt werden müssen, wie das Beispiel unten zeigt.

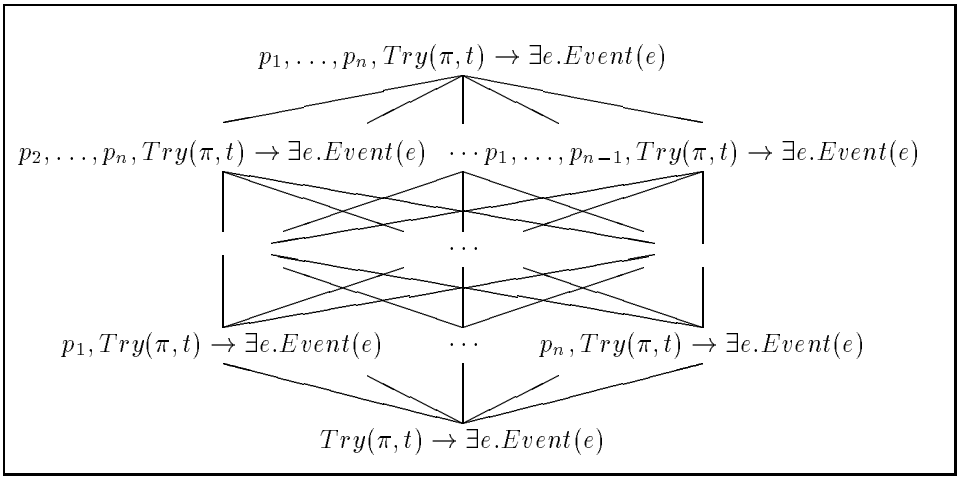

Abbildung 2.11: Verband von widerlegbaren Regeln *für* ein Ereignis *e* aus [Ferguson, 1994, S.80]

Den Zusammenhang zwischen Ereignissen und ihren Effekten definieren Ferguson und Allen durch sogenannte *Event Definition* Axiome, die folgende Form haben:

$$\forall e\ Event(e) \wedge \phi \supset \psi$$

Dabei steht *Event* wieder für ein Event-Prädikat. ϕ und ψ sind Formeln, in denen keine anderen Ereignisse vorkommen dürfen. Das Symbol "$\supset$" steht für die logische Implikation. ϕ dient zur Beschreibung von Bindungsbedingungen an *e*. Durch ψ werden die Literale angegeben die wahr werden, wenn ein Ereignis des Typs *Event* eintritt. Die entsprechenden Prädikate haben – wie bei den Prädikaten für die Vorbedingungskomponenten – jeweils eine Argumentstelle für das Zeitintervall ihrer Gültigkeit. Hinzu kommen Bindungsbedingungen, die insb. auch die zeitliche Beziehung zum Zeitintervall in dem das Ereignis *e* eintritt beschreiben. Aus Gründen der Schreibvereinfachung werden für Beispiele mehrstellige Event-Prädikate verwendet, durch die Bindungsbedingungen entfallen können. Vgl. dazu insb. das folgende Beispiel aus [Ferguson und Allen, 1994].

Hier verhandeln zwei Akteure darüber, wie ein Transportgut zu einem Hafen gebracht werden soll, damit es noch von einem Schiff das am selben Tag ausläuft weitertransportiert werden kann. Die Notation wurde dabei wie folgt geändert:

- Das Prädikat *Try* erhält ein zusätzliches Argument für das Ereignis das es auslösen soll.

- Das Zeitintervall in dem ein Ereignis eintritt, wird durch den Index des Ereignisses angegeben.
- Die einzelnen *Premises* der *Action Definitions* werden als Menge notiert.
- Auf die Angabe von Quantoren wurde verzichtet; es kann angenommen werden, daß alle Variablen implizit allquantifiziert sind.

Das Beispiel enthält folgende *Action Definitions*:

$$\begin{aligned} \{Try(sendByTrain(x,l),t,e_t)\} &\rightarrow Transport(e_t,x,l,5) \\ \{Try(sendByTruck(x,l),t,e_t)\} &\rightarrow Transport(e_t,x,l,3) \\ \{At(x,l,t),At(s,l,t),Try(load(x,s,l),t,e_t)\} &\rightarrow Load(e_t,x,s,l) \end{aligned}$$

Die beiden Ereignistypen *Transport* und *Load* sind wie folgt definiert:

$$\begin{aligned} Transport(e_t,x,l,n) &\supset At(x,l,t+n), \\ Load(e_t,x,s,l) &\supset In(x,s,t+1) \end{aligned}$$

Das Zeitmodell des Beispiels basiert auf Zeitpunkten, die volle Stunden in der realen Welt modellieren. Beim Ereignistyp *Transport* wird ausgenutzt, daß der Effekt eines Ereignisses in einem beliebigen späteren Zeitintervall (bzw. im Bsp. Zeit*punkt*) eintreten kann. Dazu gibt die letzte Argumentstelle des Prädikats *Transport* an, um wie viele Zeitschritte nach dem Ereignis das Transportgut x am Zielort l eintrifft. Beim Transport mit dem Zug also fünf Zeitschritte (bzw. hier Stunden) und beim Lkw-Transport drei. Außerdem ist bekannt, daß der Transport um 0 Uhr beginnen soll und die Auslaufzeit T_d des Schiffs zwischen 4 und 6 Uhr liegt. Das Prädikat *At*(*S, P, t*) ist wahr, wenn das Schiff S zum Zeitpunkt t im Hafen P ist; d.h. für Zeitpunkte t vor T_d ist es wahr, danach falsch. Abb. 2.12 zeigt, wie eine Verhandlung über den besten Plan für die Versendung des Transportguts modelliert werden kann. Im Schritt a) argumentiert Akteur A, daß die Ladung via Zug zum Hafen transportiert werden soll, zum Zeitpunkt 5 dort ist, verladen wird und damit zum Zeitpunkt 6 im Schiff ist. Von den Vorbedingungen für *Load* wird dabei nicht berücksichtigt, daß das Schiff zum Zeitpunkt des Ladens am gleichen Ort sein muß wie das Transportgut. Das alleine reicht aber noch nicht aus, um die Richtigkeit des Plans zu widerlegen. In Schritt b) geht Akteur B von der Annahme aus, daß das Schiff zum Zeitpunkt 4 den Hafen verläßt und damit *At*(*S, P, 5*) falsch ist. Da die *Premises gegen Load* in b) spezifischer sind als die *für Load* in a), ist der ursprüngliche Plan damit widerlegt. Schritt c) modelliert die Situation, die durch die Annahme von Akteur A, daß das Schiff doch erst zum Zeitpunkt 6 den Hafen verläßt, entsteht. Die *Premises für* bzw. *gegen Load* in b) und c) sind gleich spezifisch und sind damit weder bestätigt noch widerlegt. Ein Ausweg kann durch einen Transport mit dem Lkw erreicht werden, der in Schritt d) dargestellt ist. Da der Transport schneller ist, ist das Transportgut bereits zum Zeitpunkt 3 – also sicher vor dem Auslaufen des Schiffes – im Hafen. Die Strukturen der Abhängigkeiten in Schritt a) und d) sind gleich. Der Unterschied ergibt sich daraus, daß es kein Gegenargument gegen das *Load* aus Schritt d) gibt.

Die Graphen der einzelnen Schritte des Beispiels stellen für Ferguson und Allen jeweils einen Plan dar, ohne daß der Begriff formal definiert wird. Ihnen ist wichtig, daß ein Plan nicht nur eine Aneinanderreihung von Planschritten ist, sondern daß zusätzlich für jeden Planschritt die Gründe warum er im Plan ist mit repräsentiert werden. Diese Auffassung ist mit derjenigen der Handlungsplanung vergleichbar, die *Causal Links* als Bestandteil eines Plans ansehen (vgl. Kap. 2.1.1 *Least Commitment Planning*).

Die Ausdrucksstärke des zugrundeliegenden Formalismus für die Repräsentation von Aktionen und Ereignissen ist wesentlich höher als die der Modelle der Handlungsplanung. Außer der bereits genannten Möglichkeit, Vorhersagen für einzelne Situationen treffen zu können, entfällt durch die

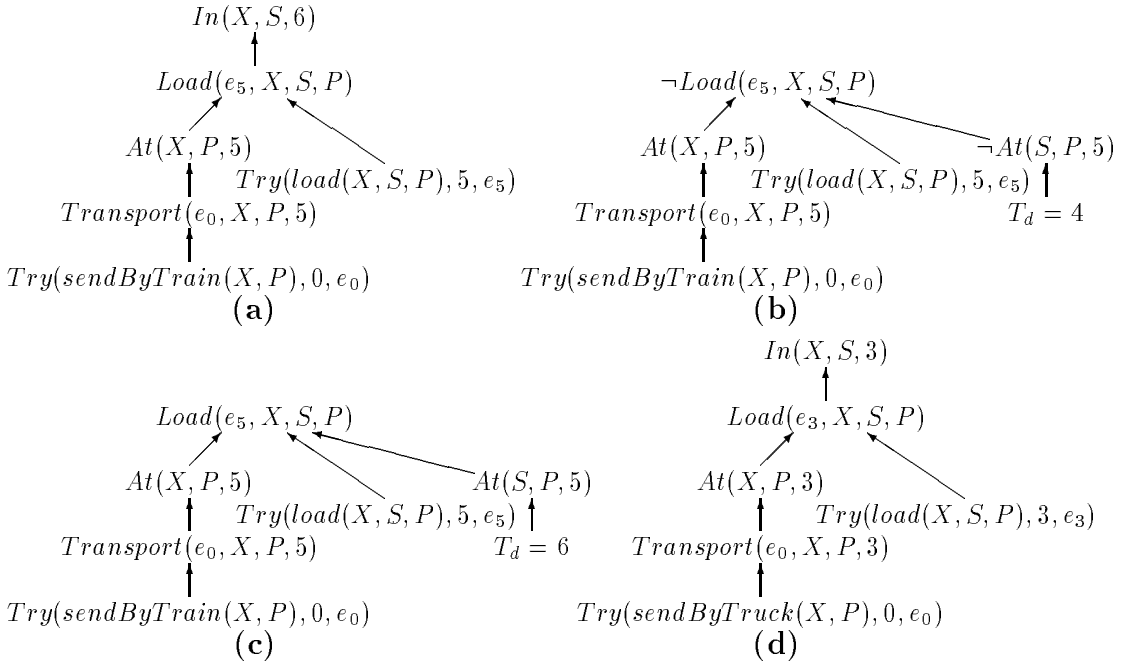

Abbildung 2.12: Argumentation über Pläne aus [Ferguson und Allen, 1994]

explizite Repräsentation der Zeitintervalle einer Aktionsausführung die unrealistische Annahme der Handlungsplanung, daß das Ausführen von Aktionen keine Zeit benötigt[44]. Durch die Möglichkeiten von Allens Zeitrelationen [Allen, 1984] können auch simultane oder teilweise überlappende Aktionen modelliert werden. Auch die STRIPS-Annahme, daß Veränderungen einer Situation ausschließlich auf Aktionen des handelnden Akteurs zurückgehen, entfällt hier. Durch die Trennung von Aktionen und Ereignissen lassen sich auch spontane, zufällige Ereignisse als Ereignisse ohne auslösende Aktionen modellieren. Auf der anderen Seite macht eine uneingeschränkte Verwendung solcher Ereignisse sowohl eine Vorhersage als auch eine Planung unmöglich. Ferguson und Allen verwenden deshalb die *Explanation Closure* Technik ([Haas, 1987], [Schubert, 1990]), mit der explizit und vollständig festgelegt wird, durch welche Ereignisse Veränderungen der Situation erklärt werden können und durch welche Aktionen Ereignisse hervorgerufen werden können. Vergleichbar mit den Rahmen-Axiomen des Situationskalküls wird

- für jedes Prädikat P, das eine Eigenschaft im Anwendungssachbereichs modelliert, ein Axiom der Form
  $$\forall t,t'\ P(t) \wedge \neg P(t') \wedge meets(t,t') \supset \exists e\ (E_1(e) \vee \ldots \vee E_n(e)) \wedge meets(time(e),t')$$
  wobei die E_i den Wahrheitswert von P verändern können, und

- für jeden Ereignistyp E ein Axiom der Form
  $$\forall e\ E(e) \supset G_1 \vee \ldots \vee G_n$$

definiert. Durch Axiome des ersten Punkts wird sichergestellt, daß sich ein Wahrheitswert nur ändern kann, wenn unmittelbar davor ein Ereignis aufgetreten ist, das diese Änderung bewirken kann. Die Axiome des zweiten Punkts besagen, daß das Auftreten eines Ereignisses von einer Ursache G_i her-

[44] als Folgerung aus der Annahme, daß Änderungen durch diskrete Situationen modelliert werden können (vgl. Kap. 2.1.3 *Voraussetzungen beim Planen*).

rührt. Dies kann eine Aktion oder ein anderes Ereignis sein. Die G_i können damit folgende Formen haben:

$$\exists t\ Try(\pi, t) \wedge t \circ time(e)$$
$$\exists e'\ E'(e') \wedge time(e) \circ time(e')$$

wobei "$\circ$" eine Allen'sche Zeitrelation (normalerweise "=") ist.

Die Axiome können zumindest teilweise automatisch aus den Definitionen der Aktionen und Ereignissen erzeugt werden (vgl. [Reiter, 1992] und [Schubert, 1990] für eine weitergehende Diskussion). Für die Beurteilung der praktischen Einsetzbarkeit des Formalismus ist die Frage wichtig, ob die gewünschten Inferenzen ausreichend effizient gezogen werden können, sodaß sie in einer Mensch-Computer-Schnittstelle mit akzeptablen Antwortzeiten genutzt werden können. Zumindest für die Plangenerierung sind hier Zweifel angebracht, denn zum einen wird Allens Zeitlogik eingesetzt, auf deren hohe Komplexität bereits im Kapitel 2.2 *Planerkennung* (Fußnote 26 S. 33) hingewiesen wurde. Außerdem sind – vergleichbar zu den Rahmen-Axiomen des Situationskalküls – viele Axiome allein dafür erforderlich, um Schlüsse auf Eigenschaften die sich *nicht* ändern zu ermöglichen. Die Frage, wie ein Inferenzmechanismus trotz solcher Axiome zielgerichtet und effizient ein Planungsproblem lösen kann, bleibt – ebenso wie für den Situationskalkül – unbeantwortet. Indirekt wird diese pessimistische Einschätzung durch den geringen Stellenwert der Plangenierung im TRAINS- und im TRIPS-System bestätigt. In beiden Systemen bedeutet Plangenerierung *Routenplanung* und unterscheidet sich deshalb deutlich von der allgemeinen Problemstellung der Handlungsplanung. Sowohl das Ziel (Transportgut von einem Ort zu einem anderen bringen) als auch der Typ der Planschritte (Transport auf einer Teilstrecke) sind festgelegt. Die Aufgabe liegt hier in einer geeigneten Parametrisierung (welche Teilstrecke, Zeitbedarf, Kosten). Dementsprechend wird in [Ferguson *et al.*, 1996a] klargestellt, daß TRAINS keine Komponente für (klassische) Planungsprobleme enthält und dies auch nicht für sinnvoll gehalten wird. Ihr Ansatz lautet vielmehr, so bald wie möglich einen Vorschlag (*Straw Plan*) in die Diskussion zu werfen, um so die Präferenzen und (Sub-)Ziele des Benutzers zu ermitteln. Die Architektur beider Systeme spiegelt die hauptsächliche Zielsetzung einer um Zeigegesten ergänzten *natürlichsprachlichen Verhandlung über Pläne* wieder (vgl. [Ferguson *et al.*, 1996b, S.4], [Ferguson und Allen, 1998] bzw. Abb. 2.13).

2.3.2 Pläne als mentale Zustände

Bisher wurde immer von *dem einen* Plan gesprochen – *der eine Plan*, der erzeugt werden soll oder *der eine Plan*, den der Benutzer ausführt und der erkannt werden soll. Pollack argumentiert in [Pollack, 1986] und [Pollack, 1990], daß traditionelle Modelle, die nur von *einem zentralen* Plan ausgehen[45], Schwächen bei der Erklärung von Phänomenen eines natürlichsprachlichen Dialogs haben und dementsprechend auch unzureichend für die Erzeugung einer Systemantwort in einem solchen Dialog sind. Die Beschränkung auf einen zentralen Plan impliziert die Annahme, daß sich ein Akteur, für den geplant wird oder der beobachtet wird, genau gemäß des Modells des Planers oder Beobachters verhält. Diese Annahme mag (ausreichend) korrekt sein, wenn das gesamte Verhalten des Akteurs deterministisch vorhergesagt werden kann – also beispielsweise bei einem Roboter, der meist der Adressat eines Planungsprozesses ist[46]. Hat der Akteur dagegen eigene Ansichten über seine Zielsetzung, über den Zustand der Welt und über Möglichkeiten diese zu verändern, so kann dieses Modell vom Modell des Planers oder Beobachters abweichen. Speziell in einer Situation eines Benutzers, der

[45] Pollack bezieht sich auf die Planungssysteme STRIPS [Fikes und Nilsson, 1971] und NOAH [Sacerdoti, 1977] und Ansätze zur Planerkennung aus [Allen, 1979], [Carberry, 1985], [Litman, 1985], [Sidner, 1985]. Vgl. auch die Kapitel 2.1 *Handlungsplanung* und 2.2 *Planerkennung* dieser Arbeit.

[46] bspw. bezeichnet Nilsson Planerzeugung als *Robot Problem Solving* (vgl. [Nilsson, 1980, S.275]).

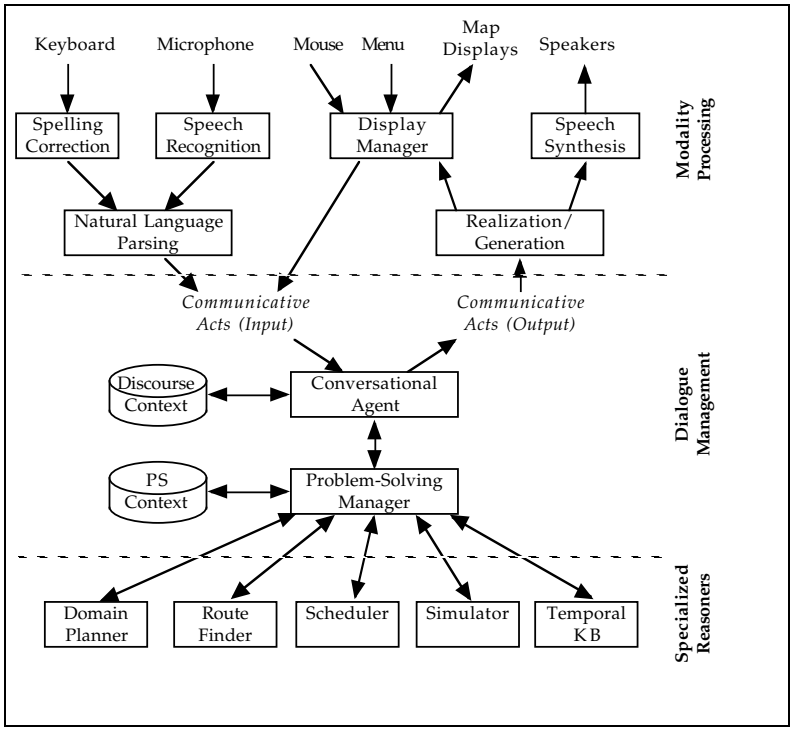

Abbildung 2.13: TRIPS System Architektur aus [Ferguson und Allen, 1998]

von einem Computer Unterstützung bei einem Problem sucht, sind unterschiedliche Ansichten der Normalfall. Einige Hilfesysteme[47] repräsentieren bspw. explizit sogenannte *typische falsche Pläne*; d.h. Pläne, die viele Benutzer für richtig halten, die aber (nach Ansicht des Systems) nicht zum Ziel führen. Solche *typischen falschen Pläne* können dazu dienen einen Benutzer vor möglichen Fehlern zu warnen oder einen Fehler zu erklären. Allerdings können niemals alle Fehlermöglichkeiten explizit repräsentiert werden, zumal die Beurteilung, ob ein Plan im o.g. Sinne falsch ist, auch vom Kontext abhängt in dem der Plan ausgeführt werden soll.

Pollack schlägt in Weiterentwicklung der Ideen aus [Bratman, 1987] ein alternatives Modell vor, das die verschiedenen Gesichtspunkte der Relation zwischen einem Akteur und einem Plan berücksichtigt. Insbesondere unterscheidet das Modell zwischen einem Plan den ein Akteur *kennt*, um ein bestimmtes Ziel zu erreichen oder eine Aktion durchzuführen, und der *konkreten Absicht*, einen Plan durchzuführen. Ersteres bezeichnet Pollack als *receipe-for-action* oder einfach nur als Rezept. Sie sieht die (nicht-primitiven) Operatoren der Handlungsplanung mit Operatorabstraktion (vgl. Kap. 2.1.2.2 *Operatorabstraktion*) bzw. speziell die Operatoren des NOAH-Systems [Sacerdoti, 1977] als einfache Rezepte an. Komplexere Rezepte werden als gerichteter azyklischer Graph aufgebaut, dessen Knoten aus Aussagen und Operatoren bestehen, die durch die Relationen *causes*, *is-a-precondition-of* oder *is-a-way-to* verbunden werden. Der wesentliche Aspekt in Pollacks Modell betrifft allerdings nicht die Rezepte oder deren Konstruktion, sondern die Frage, wie die konkrete Absicht eines Akteurs einen Plan auszuführen charakterisiert werden kann. Diese konkrete Absicht eines Akteurs *A* zusammen mit Annahmen über seine eigenen Fähigkeiten und den Zusammenhang zwischen Planschritten und Plan bezeichnet Pollack mit *Akteur A hat den Plan P*. Bevor Pollack einem Akteur *A* einen Plan

[47] vgl. Kap. 2.5 *Hilfesysteme*

P, der aus der Menge *S* von Planschritten besteht, in diesem Sinne zuschreibt, müssen die folgenden Bedingungen erfüllt sein:

1. *A* glaubt, daß er jeden Planschritt aus *S* ausführen kann.
2. *A* glaubt, daß die Ausführung der Menge der Planschritte *S* die Ausführung von *P* bewirkt.
3. *A* glaubt, daß jeder Planschritt aus *S* einen Beitrag für den Plan leistet.
4. *A* hat die konkrete Absicht jeden Planschritt aus *S* auszuführen.
5. *A* hat die konkrete Absicht die Menge der Planschritte *S* als Realisierungsmöglichkeit für *P* auszuführen
6. *A* hat die konkrete Absicht jeden Planschritt aus *S* auszuführen, um einen Beitrag für den Plan zu leisten.

Die Bedingungen 2 bzw. 3 lassen sich normalerweise[48] aus 5 bzw. 6 ableiten. Es werden dennoch alle Annahmen die einen Plan betreffen explizit aufgeführt, weil mögliche Unterschiede zwischen den *Annahmen* des Akteurs und eines Beobachters die Grundlage für eine Beurteilung eines Plans und damit auch für eine korrigierende Erklärung bieten.

Pollack gibt eine Formalisierung für eine vereinfachte Variante dieser Bedingungen an die beschreibt, daß ein Akteur *A* einen *einfachen* Plan hat. Darauf soll hier nicht weiter eingegangen werden, weil durch die Vereinfachung nur noch sehr eingeschränkte Pläne beschrieben werden können. Pläne bestehen dabei aus einer *kausal* geordneten Menge von Planschritten die alle *gleichzeitig* ausgeführt werden, indem die Ausführung eines Planschritts gleichzeitig die Ausführung des Nachfolgers bewirkt. Als Beispiel dafür wird das *Betätigen der DEL-Taste* angegeben, das in einem bestimmten Kontext das *Löschen einer Mail* bewirkt.

Pollack sieht ihren Ansatz als Beitrag zur Planerkennung indem sie Bedingungen angibt die erfüllt sein müssen, bevor von einem Akteur angenommen werden kann, *daß er einen Plan hat*. Sie gibt auch Klassifizierungskriterien und Korrekturvorschläge für verschiedene Arten *falscher* Pläne eines Benutzers an, auf die hier nicht eingegangen werden soll. Allerdings wird auf ihren Ansatz im Zusammenhang mit der Erkennung von Fehlerursachen in Abschnitt 2.5.2 *Erkennung von Fehlerursachen* noch einmal eingegangen.

Pollacks Ansatz wird von Grosz und Sidner in [Grosz und Sidner, 1990] aufgegriffen und für ein Szenario weiterentwickelt, bei dem mehrere Akteure an der Erstellung und Ausführung eines Plans mitwirken. Sie argumentieren, daß ein gemeinsamer Plan mehr ist als die Zusammenführung von mehreren Einzelplänen für jeweils eine Teilaufgabe und führen dafür den Begriff *SharedPlan* ein. Als Beispiel geben sie zwei Kinder an, die einen Turm aus Bauklötzen mit verschiedenen Mustern bauen wollen und wobei jedes Kind nur Bausteine mit einem Muster verwendet (s. Abb. 2.14). Als Konsequenz erweitern sie Pollacks Bedingungen zum einen in der Hinsicht, daß die dazu gehörenden Annahmen des einzelnen Akteurs nun *gemeinsame Annahmen* (*Mutual Beliefs*)[49] aller beteiligten Akteure sein müssen. D.h. jeder Akteur muß nicht nur der Überzeugung sein, daß er seine eigenen Aktionen ausführen kann und daß sie zur Gesamtaufgabe beitragen, sondern auch, daß die anderen Akteure ihre Aktionen ausführen können und auch zur Gesamtaufgabe beitragen. Diese Überzeugung muß wiederum allen Akteuren bekannt sein. Die einzelnen Akteure brauchen hier zwar nur die

[48] Pollack weist auf eine offene philosophische Diskussion über die Frage, ob diese Ableitung tatsächlich immer zulässig ist, hin und verweist dazu auf [Grice, 1971], [Davidson, 1980] und [Bratman, 1983].

[49] *gemeinsame Annahme* sei hier und im weiteren immer im Sinne von *Mutual Belief* verwendet. Solche *unendlich reflexiven Annahmen* (alle Akteure nehmen *p* an *und* alle Akteure nehmen an, daß alle Akteure *p* annehmen *und* alle Akteure nehmen an, daß alle Akteure annehmen, daß alle Akteure *p* annehmen *und* ...) werden ausführlich in [Kobsa, 1989] behandelt.

Absicht haben, ihre eigenen Aktionen auszuführen, es muß allerdings die gemeinsame Annahme bestehen, daß jeder einzelne Akteur die Absicht hat seine Aktionen (mit dem Ziel einen Beitrag für die Gesamtaufgabe zu leisten) auszuführen.

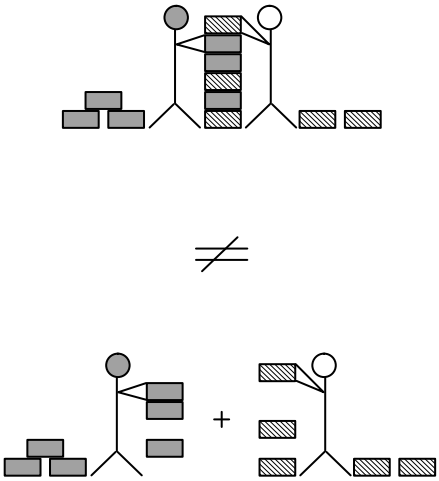

Abbildung 2.14: Gemeinsamer Plan und zwei Einzelpläne aus [Grosz und Sidner, 1990]

Der Ansatz wird in [Grosz und Kraus, 1996] weiterentwickelt, um auch komplexere Pläne erklären zu können. Die Arbeit basiert zwar auf einer Analyse der Zusammenarbeit von Menschen [Balkanski, 1990], sie ist aber nicht darauf beschränkt. Bspw. wird sie von Pasula [Pasula, 1996] für die Simulation der Zusammenarbeit künstlicher Akteure in einem Transport-Sachbereich genutzt.

Die Aufteilung einer komplexen Aufgabe in weniger komplexere Teilaufgaben orientiert sich bei Grosz und Kraus wie bei Pollack am Ansatz der Operatorabstraktion der Handlungsplanung. Vergleichbar zur Dekompositions-Relation dort, ist hier einer nicht-primitiven Aktion eine Menge von Rezepten zugeordnet. Ein Rezept R_a für eine Aktion a ist dabei eine Menge von (Sub-) Aktionen s_i und Bedingungen p_j.

$$R_a = \{s_1, p_1, \ldots, s_n, p_m\}$$

Die Bedingungen können sowohl die Vorbedingungen als auch die zeitliche Anordnung der Sub-Aktionen beschreiben. Aktionen, denen kein Rezept zugeordnet ist, werden als *basic-level action* (*primitive Aktionen*) bezeichnet.

Für einen komplexen Plan ist es unrealistisch, daß jeder einzelne Akteur über jedes Detail des Gesamtplans Bescheid weiß oder es sogar aktiv unterstützt. Ein gemeinsamer Plan setzt sich dementsprechend in Grosz und Kraus' Ansatz aus koordinierten Teilplänen zusammen, für die einzelne Gruppenmitglieder oder eine Teilgruppe verantwortlich ist. Es muß allerdings Konsens darüber bestehen, wer welche Teilaufgabe ausführt, daß er dies auch kann und daß damit letztlich die Gesamtaufgabe erfüllt wird. Im Gegensatz zu Searle [Searle, 1990] verzichten sie aber auf *gemeinsame Absichten* (*we-intentions*) und beziehen konkrete Absichten grundsätzlich auf *einen* Akteur. *Gemeinsam* können Akteure lediglich Annahmen und Pläne haben.

Ein solcher *gemeinsamer Plan* (*SharedPlan*) eine Aktion a auszuführen wird einer Gruppe zugeschrieben, wenn sie entweder einen *vollständigen gemeinsamen Plan* für die Aktion a hat oder lediglich einen *partiellen gemeinsamen Plan für a* und zusätzlich einen *vollständigen gemeinsamen Plan*, den *partiellen Plan für a zu vervollständigen*. Ein vollständiger gemeinsamer Plan setzt die

Existenz eines Rezepts voraus, mit dem die Aktion erreicht werden kann. Weitere Anforderungen an einen vollständigen gemeinsamen Plan sind:

1. Die Gruppe hat die gemeinsame Annahme, daß jedes einzelne Gruppenmitglied *will, daß* (*Intend-That* (s.u.)) die Gruppe a ausführt.
2. Die Gruppe hat die gemeinsame Annahme, daß $R_a = \{s_1, p_1, \ldots, s_n, p_m\}$ ein Rezept ist, um a zu erreichen.

   Für jede Subaktion $s_i \in R_a$, die durch ein einzelnes Gruppenmitglied G_k ausgeführt werden soll, gilt:
3. G_k *will* (*Intend-To* (s.u.)) s_i *ausführen*.
4. Die Gruppe hat die gemeinsame Annahme, daß G_k s_i ausführen will.
5. a) Falls s_i eine primitive Aktion ist,
   hat die Gruppe die gemeinsame Annahme, daß G_k s_i (unter den Rand-Bedingungen des Plans und des Rezepts) ausführen kann[50].

   b) Falls s_i eine nicht-primitive Aktion ist,

   i) gibt es ein Rezept R_{s_i} für s_i und G_k hat den vollständigen *individuellen* Plan (s.u.) s_i mit R_{s_i} auszuführen,

   ii) hat die Gruppe die gemeinsame Annahme, daß

   a) es ein Rezept R_{s_i} für die Ausführung von s_i gibt,
   b) G_k s_i mit diesem Rezept ausführen kann,
   c) G_k einen vollständigen *individuellen* Plan hat, s_i mit diesem Rezept auszuführen
6. Die Gruppe hat die gemeinsame Annahme, daß jedes einzelne Gruppenmitglied *will, daß* (*Intend-That*) es ein Rezept gibt mit dem G_k s_i ausführen *kann*.

Grosz und Kraus unterscheiden zwischen dem (aktiven) Willen eine Aktion auszuführen (*Intend-To*) und dem (aktiven) Willen, daß eine Eigenschaft oder Sachverhalt wahr sein soll (*Intend-That*). Intend-That bestimmt nicht unmittelbar das Handeln des Akteurs, kann aber zum Initiieren eines Planungsprozesses führen, dessen Ziel die gewünschte Eigenschaft ist. Es kann auch dazu führen, daß der Akteur die gewünschte Eigenschaft die bereits erreicht ist überwacht und – falls sie nicht mehr wahr ist – etwas zur Wiederherstellung tut. Im Falle von Bedingung 6) ist der Wille, daß der ausführende Akteur seine Aufgabe erledigen *kann*, die Basis für eine evtl. notwendige oder sinnvolle Unterstützung oder Hilfeleistung.

Varianten der o.g. Bedingungen ergeben sich für Subaktionen von R_a, die a) nicht von einem einzelnen Gruppenmitglied, sondern von mehreren Gruppenmitgliedern ausgeführt werden sollen oder b) an einen einzelnen oder mehrere *fremde* Akteure delegiert werden. Für den ersten Fall ist G_k als Teilgruppe anzusehen, die dementsprechend in 5b einen vollständigen *gemeinsamen* Plan hat s_i auszuführen. Eine Gruppe kann nur nicht-primitive Aktionen ausführen, deshalb entfällt Bedingung 5a). Bedingungen 3) und 4) entfallen, weil immer nur ein einzelner Akteur die Absicht haben kann eine Aktion auszuführen. Implizit ist die Absicht aber im gemeinsamen Plan (5b) enthalten, da ein solcher Plan beinhaltet, daß die einzelnen Planschritte von einem Gruppenmitglied oder einer Teilgruppe ausgeführt werden (rekursive Anwendung der Bedingungen für einen vollständigen *gemeinsamen* Plan).

[50] *Ausführen können* beinhaltet sowohl die Fähigkeit, die Aktion unter den gegebenen Bedingungen ausführen zu können, als auch – bei nichtprimitiven Aktionen – die Kenntnis eines Rezepts, um die Aktion auszuführen. Jede der einzelnen Subaktionen des Rezepts muß der Akteur ebenfalls ausführen können, oder einem anderen Akteur übertragen, der sie ausführen kann.

Für den Fall, daß eine Teilaktion an einen oder mehrere *fremde* Akteure delegiert wird, müssen die Bedingungen für die Ausführung eines bzw. mehrerer Akteure an diesen Akteur bzw. diese Akteure angepaßt werden. Hinzu kommt, daß einer der Gruppenmitglieder den oder die Akteure zur Durchführung der Aktion bewegen muß und daß in der Gruppe auch darüber eine gemeinsame Annahme besteht.

Wichtig ist, daß auch im Falle eines vollständigen gemeinsamen Plans nicht allen Gruppenmitgliedern jedes Detail der Planausführung bekannt sein muß. Für jede (nicht-primitive) Subaktion, die einem einzelnen Gruppenmitglied G_k übertragen wird, gibt es lediglich die gemeinsame Ansicht der Gruppe, *daß* es ein Rezept für die Ausführung der Subaktion gibt und *daß* G_k die Aktion mit dem Rezept ausführen kann und auch einen vollständigen *individuellen* Plan dafür hat.

Verantwortlich für Entscheidungen und Ausführung ist bei einem *individuellen* Plan ausschließlich der einzelne Akteur. Dementsprechend sind die Bedingungen für einen vollständigen *individuellen* Plan gegenüber dem gemeinsamen Plan auf die Annahmen und Absichten des Akteurs selbst beschränkt. Er entscheidet auch darüber, eine Subaktion von einem anderen Akteur (aber nicht von mehreren Akteure) ausführen zu lassen.

Die Eigenverantwortlichkeit bei der Planung und Ausführung von Subaktionen birgt die Gefahr, daß sich die verschiedenen Akteure für Aktionen entscheiden, die sich gegenseitig stören. Solange die Details bekannt sind wird dies durch Bedingung 6) verhindert, weil sie besagt, daß jedes Gruppenmitglied will, daß die Akteure, denen eine Aufgabe übertragen wurde, diese auch durchführen *können*. Ein einzelnes Gruppenmitglied darf also nichts tun, das ein anderes daran hindert seine Aufgabe durchzuführen. Konflikte, die durch Details entstehen die der Gruppe gar nicht bekannt sind, lassen sich so aber nicht verhindern. Um alle Möglichkeiten von Konflikten zu verhindern, müssten der gesamten Gruppe doch wieder alle Details bekannt gemacht werden, was zu einem erhöhten Kommunikationsaufwand führen würde. Grosz und Kraus lassen diese Abwägungsentscheidung ausdrücklich offen und weisen darauf hin, daß ihre Theorie lediglich Bedingungen für die Zusammenarbeit von Akteuren aufstellt.

Der Prozeß der Konstruktion eines vollständigen gemeinsamen Plans führt über *partielle* Pläne. Grosz und Kraus behandeln in [Grosz und Kraus, 1999] speziell die *Entwicklung* von gemeinsamen Plänen. Sie weisen auch darauf hin, daß es in einer dynamischen Umgebung sinnvoll sein kann, bereits vor einer Vervollständigung eines Plans mit der Ausführung zu beginnen. Dies ist immer dann vorteilhaft, wenn Sachverhalte, die für Planungsentscheidungen relevant sind, vor Ausführung des Plans noch nicht bekannt sind. Dieser Umstand kann allerdings dazu führen, daß sich der begonnene Plan als Sackgasse herausstellt und vollständig oder teilweise revidiert werden muß – sofern dies möglich ist. Auch hier ist wieder eine Abwägung zwischen dem Risiko der Ausführung eines falschen Plans und dem Abwarten bis gesicherte Erkenntnisse vorliegen notwendig. Partiellen Plänen kommt dabei eine besondere Bedeutung zu. Grosz und Kraus definieren sowohl für *vollständige gemeinsame* als auch für *vollständige individuelle* Pläne *partielle* Pendants. Die wesentlichen Unterschiede zu den *vollständigen* Plänen sind:

1. Das Rezept für die Ausführung der Aktion muß noch nicht festgelegt sein bzw. es muß nicht vollständig sein; d.h. es dürfen noch Aktionen und Bedingungen fehlen. In diesem Fall muß die Absicht bestehen, ein Rezept auszuwählen bzw. zu vervollständigen, wobei die (ggf. gemeinsame) Annahme bestehen muß, daß die Teilaktionen des ausgewählten Rezepts von dem planenden Akteur bzw. einem Mitglied der Gruppe ausgeführt oder an einen fremden Akteur delegiert werden kann.

2. Ist für eine der bereits feststehenden Aktionen ein Akteur oder eine Teilgruppe ausgewählt, so muß dieser bzw. diese lediglich einen partiellen Plan für die Ausführung haben.

3. Für eine bereits feststehende Aktion braucht nicht bereits ein Akteur oder eine Teilgruppe verantwortlich zu sein. In diesem Fall muß aber Konsens darüber herrschen, *daß* die Aktion von einem Akteur oder einer Teilgruppe ausgeführt oder an einen fremden Akteur delegiert werden kann.

Die Vervollständigung eines Plans wird hier – vergleichbar zur Auswahl einer Dekomposition bei der Handlungsplanung mit Operatorabstraktion – durch die Auswahl eines geeigneten Rezepts für eine nicht-primitive Aktion und deren Einbettung in den Plan vorangetrieben. Haben die Aktionen des Rezepts Vorbedingungen, die durch den Plan nicht erfüllt werden, oder zerstören Effekte der Aktionen Sachverhalte, die für den Gesamtplan notwendig sind, führt dies zu neuen Zielen, die fehlenden Vorbedingungen zu erfüllen bzw. die zerstörten Sachverhalte wiederherzustellen. Prinzipiell stehen dafür die gleichen Möglichkeiten zur Verfügung wie bei der Handlungsplanung; bspw.

- eine geeignete zeitliche Anordnung der Aktionen des Plans (etwa: eine Aktion, die einen Sachverhalt zerstört, wird nach der letzten Aktion die ihn als Vorbedingung benötigt angeordnet)
- die Einführung neuer Aktionen in den Plan (solche, die eine benötigte Vorbedingung als Effekt erzeugen).

Grosz und Kraus sehen lediglich die zweite dieser Möglichkeiten durch Axiom 2 in [Grosz und Kraus, 1999, Fig.7] vor (s. Abb. 2.15). Dieses Axiom besagt, daß ein Akteur eine Aktion *a* ausführen will (*Intend-To*) oder es zumindest erwägen muß (*Potential Intend-To*), von der er annimmt, daß sie einen Sachverhalt *prop* erzeugt und die nicht im Konflikt mit anderen Aktionen steht, wenn er will, daß (*Intend-That*) *prop* in einem Zeitintervall T_{prop} wahr ist und nicht glaubt, daß *prop* wahr ist. Dieses Axiom ist allerdings für eine Planerzeugung nicht ausreichend. Zwar betrifft die Absicht, daß *prop* wahr sein möge, ein bestimmtes Zeitintervall T_{prop}, aber die Annahme des Akteurs (zum Zeitintervall der Planung) über den Wahrheitswert von *prop* selbst ist zeit*un*abhängig. Das gilt auch für das Meta-Prädikat LEAD das ausdrückt, daß der Akteur annimmt, daß eine Aktion einen Sachverhalt erzeugt. Dabei wird lediglich das Zeitintervall der Ausführung der Aktion und wiederum das Zeitintervall der Planung berücksichtigt; der Sachverhalt selbst gilt dann aber zeitunabhängig. Dies würde bedeuten, daß er auch *vor* der Aktion gelten müsste und auch niemals wieder durch eine andere Aktion zurückgenommen werden könnte.

Die zweite Unzulänglichkeit besteht darin, daß die Ausführung einer Aktion *a* für *prop* ohne Alternative vermieden wird, wenn sie im Konflikt (Meta-Prädikat CONF) zu einer anderen Aktion steht, die dieser Akteur ebenfalls ausführen will. Anstatt die Ausführung von *a* von vorn herein zu vermeiden, wären die bekannten Konfliktlösungsstrategien aus der Handlungsplanung angebracht (bspw. eine geeignete zeitliche Anordnung der unverträglichen Aktionen). Dies setzt allerdings – wie generell bei der Plangenerierung – ein durchgängiges Zeitmodell voraus, das hier fehlt.

Allerdings ist der SharedPlan-Ansatz auch nicht als vollständiges Verfahren der Plangenerierung gedacht, sondern soll lediglich Erklärungen für das gemeinsame Planen mehrerer Akteure geben. Grosz und Kraus bezeichnen Axiom 2 dementsprechend auch als *Axiom schema encoding minimal conditions for intending-that*.

2.3.3 Zusammenfassung

Der zuerst vorgestellte Ansatz von Ferguson und Allen versteht sich als Ansatz zur Planungsunterstützung eines Benutzers, der zweite Ansatz wurde aus der Planerkennung heraus zu einem Ansatz des gemeinsamen Planens entwickelt. Beiden Ansätzen gemeinsam ist die Einschätzung, daß Planen in der realen Welt grundsätzlich mit Unsicherheit behaftet und unvollständig ist.

Axiom 2 :

If an agent intends-that $prop$ and does not believe $prop$

$(\forall G, T_i)$

(1) $[\text{Int.Th}(G, prop, T_i, T_{prop}, \text{IC}_{prop}) \ \wedge \ \neg\text{Bel}(G, prop, T_i)] \Rightarrow \ [$

and there is some action the agent can take that would lead to $prop$ holding then

(2) $[(\exists\alpha, T_\alpha)\text{LEAD}(G, \alpha, T_\alpha, prop, \text{con}(\alpha), T_i)] \Rightarrow \ [$

either there is some such action that the agent (potentially) intends to do

(3a) $[(\exists\alpha, T_\alpha)\text{LEAD}(G, \alpha, T_\alpha, prop, \text{con}(\alpha), T_i)\wedge$

(3b) $[\text{Int.To}(G, \alpha, T_\alpha, T_i, \text{con}(\alpha), prop) \vee \text{Pot.Int.To}(G, \alpha, T_\alpha, T_i, \text{con}(\alpha), prop)]] \ \bigvee$

or for every such action there is some conflict with the agent's other intentions

(4) $[(\forall\alpha, T_\alpha)\text{LEAD}(G, \alpha, T_\alpha, prop, \text{con}(\alpha), T_i) \quad \Rightarrow$

$[[(\exists\beta, T_\beta, \text{IC}_\beta)\text{Int.To}(G, \beta, T_i, T_\beta, \text{con}(\beta), \text{IC}_\beta)\wedge$

$\text{Bel}(G, \text{CONF}(\alpha, \beta, T_\alpha, T_\beta, \text{con}(\alpha), \text{con}(\beta)), T_i)]\vee$

$[(\exists prop, T_{prop}, \text{IC}_{prop})\text{Int.Th}(G, prop, T_i, T_{prop}, \text{IC}_{prop})\wedge$

$\text{Bel}(G, \text{CONF}(\beta, prop, T_\beta, T_{prop}, \text{con}(\beta)), T_i)]]]]]$

$\text{LEAD}(G, \alpha, T_\alpha, prop, \Theta, T_i)$

(1) $\text{Bel}(G, (\exists R_\alpha, T_\alpha)\ \text{CBA}(G, \alpha, R_\alpha, T_\alpha, \Theta), T_i) \bigwedge$

(2) $[[\text{Bel}(G, \text{Do}(G, \alpha, T_\alpha, \Theta) \Rightarrow prop, T_i)] \bigvee$

(3) $[\text{Bel}(G_1, \text{Do}(G_1, \alpha, T_\alpha, \Theta) \Rightarrow$

$[[(\exists\beta, G_2, R_\beta, T_\beta, T_i')$

$[\ T_\alpha \leq T_\beta \wedge \text{single.agent}(\beta) \ \wedge \ \text{CBA}(G_2, \beta, R_\beta, T_\beta, \text{con}(\beta)) \ \wedge$

$\text{Pot.Int.To}(G_2, \beta, T_\beta, T_i', \text{con}(\beta), \text{IC}_\beta) \wedge T_i' < T_\beta \wedge$

$[\text{Do}(G_2, \beta, T_\beta, \text{con}(\beta)) \Rightarrow prop]]] \quad \vee$

$[(\exists\beta, \text{GR}_2, R_\beta, T_\beta, T_i')$

$[T_\alpha \leq T_\beta \ \wedge \ \text{multi.agent}(\beta) \wedge \ \text{CBAG}(\text{GR}_2, \beta, R_\beta, T_\beta, \Theta) \wedge$

$(\forall G_j \in \text{GR}_2)\text{Pot.Int.Th}(G_j, \text{Do}(\text{GR}_2, \beta, T_\beta, \text{con}(\beta)), T_i', T_\beta, \text{IC}_\beta) \wedge T_i' < T_\beta \quad \wedge$

$[\text{Do}(\text{GR}_2, \beta, T_\beta, \text{con}(\beta))) \Rightarrow prop]]], T_i)]]$

Abbildung 2.15: Axiom 2 aus [Grosz und Kraus, 1999, Fig.7]: *Axiom schema encoding minimal conditions for intending-that*

Im Ansatz von Ferguson und Allen übernimmt der Benutzer normalerweise die initiale Planung, deren Ergebnis als Argument für das Erreichen der Ziele aufgefaßt wird. Das System kann diese Planung korrigieren, ergänzen oder Alternativvorschläge machen, wenn ihm speziellere (und damit bessere) Argumente für oder gegen die Ziele bekannt sind. Dies kann der Benutzer wiederum aufgrund seiner Kenntnisse oder Präferenzen modifizieren, sodaß sich zuletzt ein Plan ergibt, der nach Kenntnis und Präferenzen *beider* Dialogpartner am besten geeignet ist die Ziele zu erreichen. Die Kenntnisse beider Dialogpartner können zwar unvollständig sein, durch die Kombination wird aber die Erfolgswahrscheinlichkeit des Plans erhöht.

Auch im zweiten Ansatz wird davon ausgegangen, daß die Kenntnisse und zusätzlich hier auch die Fähigkeiten der Gruppenmitglieder unterschiedlich sind. Allerdings muß Konsens darüber herrschen, welche Aufgaben und Entscheidungen von welchen Gruppenmitgliedern ausgeführt bzw. getroffen werden. Damit verbunden ist die gemeinsame Annahme, daß die Kenntnisse und Fähigkeiten der Gruppenmitglieder für die jeweiligen Aufgaben korrekt bzw. ausreichend sind. Durch die Delegation von Aufgaben besteht die Gefahr, daß sich nicht abgestimmte Teilschritte des Plans gegenseitig behindern. Die Entscheidung, wie relevant diese Gefahr ist bzw. wieviel Abstimmungsaufwand getrieben werden soll, ist abhängig vom Sachbereich und wird dem Entwickler eines konkreten Systems

überlassen. Gleiches gilt für die Entscheidung, ob die Ausführung eines noch unvollständigen Plans bereits begonnen werden soll, wenn noch nicht alle relevanten Informationen vorliegen.

Auffällig ist, daß beide Ansätze keine Plangenerierung in der Allgemeinheit vorsehen, wie sie in der Handlungsplanung verstanden wird. Ferguson und Allens sehr ausdrucksstarke Modellierung von Aktionen und Ereignissen steht einem effizienten allgemeinen Planungsverfahren entgegen. Dem Ansatz von Grosz und Kraus fehlt ein durchgängiges Zeitmodell.

Der Wert der beiden Ansätze für diese Arbeit liegt vor allem in der Behandlung der unterschiedlichen Ansichten verschiedener Akteure *über* Pläne. Wichtig ist dabei auch die Erörterung des zweiten Ansatzes, welche Annahmen ein Akteur über die Annahmen, Ziele und auch Fähigkeiten anderer Akteure haben muß, damit er davon ausgehen kann, daß ein gemeinsamer Plan besteht.

2.4 Text- und Dialogplanen

Unter Textplanung im weiteren Sinne wird der Teil-Prozeß bei der Generierung eines natürlichsprachlichen Textes verstanden, der anhand eines oder mehrerer kommunikativer Ziele den Inhalt und die Struktur des Textes bestimmt. Dabei subsumiert *Text* sowohl längere Abhandlungen wie bspw. Benutzungsanleitungen oder Handbücher als auch einzelne Äußerungen in einem Mensch-Computer-Dialog. Für diese Aufgabe haben sich zwei Ansätze herauskristallisiert: der schemabasierte und der planbasierte Ansatz. Beispiele für ersteren sind u.a. [McKeown, 1985], [McCoy, 1989], [Paris, 1993]. Für diese Arbeit soll wiederum die Frage nach Vorstellung und Repräsentation von Plänen in diesem Forschungsbereich gestellt werden. Der für diese Frage relevante zweite Ansatz ist Thema dieses Abschnitts.

2.4.1 Textplanen als Planungsprozeß

Der Ansatz hat seinen Ursprung in Cohens Idee, nach der die Entscheidungen über die Äußerungen einer Person in einem Gespräch durch einen Planungsprozeß modelliert werden können [Cohen, 1978]. Cohen operationalisiert Austins Auffassung sprachliche Äußerungen als Handlungen zu betrachten [Austin, 1962], indem er Sprechakte durch Planoperatoren modelliert. Er setzt die Bedingungen, die Searle in [Searle, 1969] für eine erfolgreiche Ausführung eines Sprechaktes aufstellt, in Vorbedingungen der Operatoren um. Beispielsweise modelliert er eine *Bitte* durch den in Abb. 2.16 dargestellten Operator.

Cohen unterscheidet zwischen sogenannten *Cando*- und *Want*-Vorbedingungen. Der erste Typ beschreibt die Bedingungen die erfüllt sein müssen, damit der Akteur die zum Operator gehörende Aktion ausführen *kann*. Im Beispiel also, daß der Akteur sowohl glaubt, daß der Hörer die gewünschte Aktion durchführen kann als auch, daß der Hörer auch selbst glaubt, daß er die Aktion durchführen kann. Cando-Vorbedingungen sind also mit den Vorbedingungen der Handlungsplanung vergleichbar. Der zweite Typ beschreibt die Bedingungen an den *Willen* bzw. die *Intention* des Akteurs. Im Beispiel muß der Akteur also glauben, daß er die Aktion (hier: die Bitte) auch tatsächlich ausführen will.

Der Effekt besteht in der Annahme des Hörers, daß der Akteur glaubt, die Aktion zu wollen. Mit der Bitte eine Aktion auszuführen ist also nicht automatisch verbunden, daß der Hörer dies auch tut oder auch nur tun will. Er weiß lediglich, daß der Akteur dies wünscht; er kann diesen Wunsch aber auch ablehnen.

Der Effekt modelliert keine physikalischen Änderungen in der realen Welt, sondern verändert die die Annahmen des Hörers. Generell sind mit den Sprechakt-Operatoren keine physikalischen Änderungen, sondern Änderungen der mentalen Zustände der Dialogpartner verbunden. Darauf beziehen sich auch die Vorbedingungen. Cohen modelliert die mentalen Zustände der Akteure jeweils durch

Def-Operator: request (agent, recipient, act)[a]

```
CANDO.PRECOND: agent believe
                     recipient cando act
               and
               agent believe
                     recipient believe
                               recipient cando act
WANT.PRECOND:  agent believe
                     agent want request-instance
EFFECTS:       recipient believe
                          agent believe
                               agent want act
```

[a] Cohen modelliert Operatoren als mehrstellige Relationsschemata. Er verwendet für die Definition eine etwas abweichende Notation.

Abbildung 2.16: Request-Operator aus [Cohen, 1978, S.132]

getrennte Wissens-Partitionen für Annahmen (*Belief*) und Intentionen (*Want*). Die Annahmen-Partitionen eines Akteurs können wiederum Annahmen- oder Intentions-Partitionen enthalten, die auch den mentalen Zustand des Dialogpartners oder anderer Akteure betreffen. So läßt sich bspw. der o.g. Effekt des Beispiel-Operators modellieren. Auf diesen Partitionsansatz wird in Kap. 3.3 *Struktur eines Benutzermodells mit Handlungspotential und Wissen* noch näher eingegangen.

Der Wille des Akteurs ist nicht in allen Fällen für die Ausführung eines Operators erforderlich. Um bspw. zu modellieren, daß ein Akteur den Wunsch eines anderen automatisch übernimmt, definiert Cohen einen *cause_to_want* Operator, der eine leere *Want*-Vorbedingung hat. Es reicht dann aus, daß die *Cando*-Vorbedingung erfüllt ist, um den Effekt des Operators hervorzurufen. Wird dieser Operator eingesetzt, reicht die Annahme eines Akteurs, daß der Dialogpartner glaubt eine Aktion zu wollen aus, um sich diesen Wunsch zu eigen zu machen; d.h. der Akteur glaubt nun, daß auch er diese Aktion will.

Die Cando-Vorbedingung des *cause_to_want*-Operators läßt sich mit dem Effekt des *request*-Operators erreichen. Dies nutzt Cohen für ein einfaches Beispiel einer Planung ([Cohen, 1978, S.135]): um das Ziel zu erreichen, daß der Hörer glaubt eine Aktion zu wollen, wird der *cause_to_want*-Operator eingesetzt. Um dessen Vorbedingung zu erfüllen, der *request*-Operator. D.h. das Ziel kann erreicht werden, wenn der Hörer gebeten wird die Aktion auszuführen. Die Vorbedingung des *request*-Operators (Annahmen über Wollen und Können) wurden bereits genannt.

Während sich Cohen auf die Erzeugung von Sprechakten beschränkt, erweitert Appelt [Appelt, 1985] den Ansatz auf den gesamten Prozeß der Erzeugung einer natürlichsprachlichen Äußerung. Dabei wird der Prozeß in mehrere Abstraktionsstufen unterteilt, von denen die wichtigsten die Entscheidung über die illokutiven Sprechakte – bspw. einer Bitte, einer Aussage – und die Erzeugung der konkreten natürlichsprachlichen Äußerungen sind. Für diesen Prozeß setzt Appelt ein hierarchisches Planungsverfahren mit Operatorabstraktion (vgl. Kap. 2.1.2.2 *Operatorabstraktion*) ein, das sich am Planungssystem NOAH [Sacerdoti, 1977] orientiert. Die Planoperatoren werden durch ein Axiomensystem beschrieben, daß auf Moores Ansatz zur Beschreibung von Aktionen [Moore, 1980] mithilfe der *Mögliche-Welten-Semantik* [Kripke, 1963a], [Kripke, 1963b], [Hintikka, 1962], [Hintikka, 1969] zurückgeht. Die Operatoren bzw. Aktionen verbinden in Moores Ansatz zwei zeitlich nachfolgende Welten. Dazu definiert Moore das Meta-Prädikat *R(op, w1, w2)*, das genau dann wahr ist, wenn die Ausführung der Aktion *op* in Welt *w1* zu einer Situation führt, die durch *w2* beschrieben wird. Die Welten sind mit den Situationen des Situationskalküls der Handlungsplanung (vgl. [McCarthy

und Hayes, 1969] u. Kap. 2.1 *Handlungsplanung*) vergleichbar. Wie dort müssen auch hier Rahmen-Axiome definiert werden die beschreiben, was sich gegenüber der Vorgänger-Welt *nicht* geändert hat. Moore setzt Kowalskis Idee zur Reduzierung der Anzahl der Rahmen-Axiome [Kowalski, 1979] um, indem er die Objekte seiner Modellierung (Objekt-Ebene) in eine Meta-Ebene übersetzt.

Appelts Plangenerierung basiert aber nicht auf dieser Repräsentation, sondern auf sogenannten *Action-Summaries*, die Vorbedingung und Effekt eines Operators beschreiben. Analog zum Vorgehen im NOAH-System wird ausgehend von einem Ziel ein *Plan-Netzwerk* (*Procedural Network*) aufgebaut, dessen Knoten aus Zielen und Planschritten besteht. In sogenannten *Expansion-Critics* Zyklen wird ein Knoten zuerst expandiert und dann wird der resultierende Plan kritisiert. Unter Expansion versteht Appelt sowohl die Dekomposition eines Operators als auch die Einsetzung eines neuen Operators, dessen Effekt ein offenes Ziel des Plans erreicht. Im Beispiel unten wird der Operator *Pumpe-Entfernen* dekomponiert. Im Fall des Einsetzens eines neuen Operators wird jeweils ein Knoten für den neuen Operator selbst und für jede seiner Vorbedingungskomponenten[51] in das Netzwerk eingefügt[52]. In der grafischen Darstellung (s. Abb. 2.17) repräsentierten die horizontalen Kanten den Verlauf der Zeit von links nach rechts. Ein *Operator-Knoten* ist deshalb immer rechts von den *Ziel-Knoten* seiner Vorbedingung angeordnet. Da diese jeweils nur Teile der Vorbedingung repräsentieren, werden sie durch einen J*oin-Knoten* verbunden, der zwischen Ziel- und Operator-Knoten angeordnet ist. Entsprechend zum Join-Knoten gibt es links von den Ziel-Knoten einen S*plit-Knoten* der ausdrücken soll, daß die nachfolgenden Knoten nicht zeitlich geordnet sind; d.h. in einer beliebigen Anordnung erfüllt werden können. Um die Abhängigkeit zwischen einem Ziel und dem Operator der dieses Ziel erreicht zu repräsentieren, führt Appelt eine gestrichelte Kante von dem Ziel-Knoten zu dem Vorbedingungs-Knoten des Operators ein bzw. – falls es dafür mehrere Vorbedingungs-Knoten gibt – zum zugehörigen Split-Knoten.

Abb. 2.17 zeigt Appelts Beispiel, bei dem das Systems (*Rob*) das Ziel hat, eine Pumpe (*PU*) von ihrer Plattform (*PL*) zu entfernen (¬*Attached (PU, PL)*). Dieses Ziel wird als Knoten *P1* repräsentiert. Ein Operator der das Ziel erreicht ist *Remove*. Da das System selbst keine Aktionen in der realen Welt ausführen kann, muß ein anderer Akteur (*John*) die Aktion durchführen. Im Plan-Netzwerk wird deshalb *P1* expandiert; d.h. es wird der Knoten *P4* und für dessen Vorbedingung *P2* und *P3* als neue Ziel-Knoten eingeführt. Ziel *P3* ist bereits initial erfüllt und wird deshalb *Phantom-Knoten* genannt und gestrichelt dargestellt.

Die nachfolgenden Kritiken sollen feststellen, ob durch die Expansion bereits erreichte Ziele wieder zerstört werden oder ob ein Operator-Knoten überflüssig geworden ist, weil ein Ziel durch mehrere Operatoren erreicht wird. Falls erforderlich versucht Appelts System dann den Plan zu "verbessern", indem bspw. Reihenfolge-Bedingungen an die Operatoren gestellt werden, Operatoren entfernt oder Ziele neu hinzugenommen werden. Weder die Strategien zur Plan-Verbesserung noch die Kritiken selbst werden von Appelt formal definiert. Der Expansion-Critics Zyklus fährt nun mit der Bearbeitung von Knoten fort, bis kein Knoten der aktuellen Abstraktionsebene weiter expandiert werden kann. Das Plan-Netzwerk in Abb. 2.17 ist für die höchste Abstraktionsebene voll expandiert. Nach Abschluß des Expansion-Critics Zyklus für eine Abstraktionsebene wird – nun wieder anhand des Axiomensystems – versucht zu beweisen, daß die resultierende Folge von Planschritten ihr Ziel erreicht. Ist das nicht der Fall, müssen vorangegangene Entscheidungen zurückgenommen und ersetzt werden – bspw. solche über die Auswahl von Operatoren, die ein Ziel erreichen sollen.

Andernfalls wird die jeweils nächst niedrigere Abstraktionsebene in der gleichen Weise bear-

[51] Wie in der Handlungsplanung üblich (vgl. Kap. 2.1 *Handlungsplanung*), kann die Vorbedingung eines Operators aus einer Konjunktion von Literalen bestehen.

[52] Die entsprechenden Knoten in NOAHs *Procedural Nets* repräsentieren *sowohl* ein Ziel das sie erreichen sollen *als auch* eine Menge von Realisierungsmöglichkeiten, die der Dekomposition beim Planen mit Operatorabstraktion entspricht.

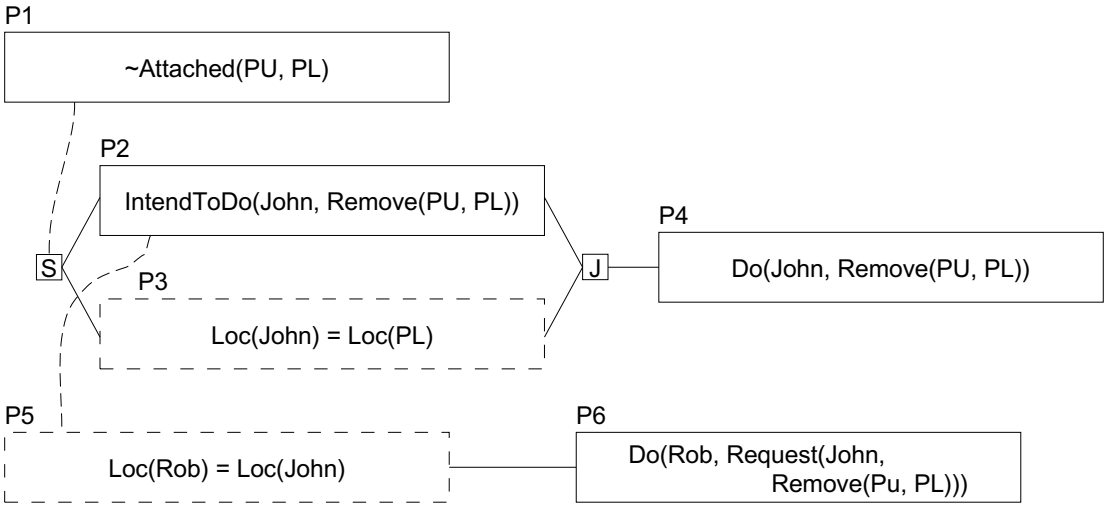

Abbildung 2.17: Plan-Netzwerk aus [Appelt, 1985, S.139]

beitet, bis schließlich auf der untersten Ebene die grammatikalische Struktur für die Ausgabe einer natürlichsprachlichen Äußerung vorliegt.

Im Beispiel wird auf der nächsten Ebene mit der Expansion von Knoten *P6* und *P4* fortgefahren ([Appelt, 1985, Fig. 7.7 bzw. 7.8]). Bemerkenswert ist dabei, daß nicht nur der Text, sondern auch die Aktionen des Dialogpartners (*John*) in der realen Welt geplant werden. Auch das Ziel selbst kann (wie im Beispiel) ein Ziel eines Akteurs in der realen Welt sein – im Beispiel ist es *Rob*. Dabei zeigt sich, daß die weitere Planung auf niedrigeren Ebenen Rückwirkungen auf höhere haben kann. Knoten *P4* (*Pumpe-Entfernen*) wird in eine Aktion *Lösen* (einer Schraube) dekomponiert. Voraussetzung für diese Aktion ist u.a., daß *John* das notwendige Werkzeug dafür kennt. Dieses Ziel kann dadurch erreicht werden, daß *John* über Merkmale des Werkzeugs informiert wird. *Informieren* ist aber wie *Bitten* ein illokutiver Sprechakt, der auf der höchsten Abstraktionsebene bearbeitet wird.

Angemerkt sei, daß Appelt nicht beschreibt, ob und wie die Zuordnung von Planoperatoren zu Abstraktionsebenen repräsentiert wird. Gleiches gilt für die Dekompositions-Relation zwischen Planoperatoren.

Appelts Ansatz eignet sich für die Erzeugung kurzer Äußerungen des Systems. Für längere Texte erscheint er vor allem aus zwei Gründen ungeeignet. Zum einen ist mit dem geforderten Korrektheitsbeweis für die Pläne bzw. Plankandidaten auf jeder Abstraktions-Ebene ein großer Berechnungsaufwand verbunden. Ohne Einschränkungen[53] geht Appelts Formalismus über die Prädikatlogik erster Stufe hinaus, womit die Berechnung eines solchen Beweises nicht garantiert werden kann (vgl. dazu auch [Moore, 1995, S.84f]). Eine weitere Beschränkung besteht darin, daß für die Erzeugung des Textes lediglich berücksichtigt wird, welche Informationen für das Ziel erforderlich sind. Information über die Struktur natürlichsprachlicher Texte wird dagegen nicht berücksichtigt.

2.4.2 RST-basiertes Textplanen

In der Linguistik befaßt sich eine Vielzahl von Arbeiten mit der Frage, welche Anforderungen an einen kohärenten, also inhaltlich zusammenhängenden, natürlichsprachlichen Text zu stellen sind (vgl. [Hovy, 1993] für einen Überblick). Hobbs [Hobbs, 1982] charakterisiert die Kohärenz von Texten durch Relationen die zwischen Teilen des Textes gelten. Mann und Thompson [Mann und

[53] [Appelt, 1985] gibt keinen Hinweis auf Einschränkungen, die die Berechnungskomplexität verringern würden.

Thompson, 1987] entwickeln diese Idee weiter und formalisieren den Ansatz. Ihre *Rhetorical Structure Theory* (RST) war und ist für die Erzeugung von Textplänen besonders einflußreich. Sie besagt, daß sich die Struktur eines kohärenten Textes durch rund 25 Relationen beschreiben läßt, die zwischen Teilen des Textes bestehen. Dabei weisen die Komponenten der Relationen wiederum eine Struktur auf, die sich auf die gleiche Weise beschreiben läßt, an die aber bestimmte Bedingungen geknüpft sind. Die Strukturierung kann so von längeren Textteilen bis hin zu einzelnen Satzbestandteilen verfeinert werden.

Die meisten Relationen sind zweistellig. Die erste Komponente wird *Nucleus*, die zweite *Satellite* genannt. Eine Relation wird durch einen Effekt auf den Leser/Hörer und die Bedingungen an Nucleus, Satellite und an beide gemeinsam beschrieben. Abb. 2.18 zeigt die Beschreibung für die Relation *Circumstance*.

relation name:	CIRCUMSTANCE
constraints on N:	none
constraints on S:	S presents a situation (not unrealized)
constraints on the N+S combination:	
	S sets a framework in the subject matter within which R (*Hörer*) is intended to interpret that the situation presented in N
the effect:	R recognizes that the situation presented in S provides the framework for interpreting N

Abbildung 2.18: RST Rel. *Circumstance* aus [Mann und Thompson, 1987, App. I.1]

Der Textteil der *Satellite* entspricht beschreibt die näheren Umstände des zu *Nucleus* gehörenden Textteils. Im Beispiel *Circumstance* ist der Effekt, daß der Hörer diese Zuordnung erkennt. Satellite muß einen bereits feststehenden Umstand beschreiben. Dadurch unterscheidet sich *Circumstance* bspw. von der Relation *Condition*, die hier eine hypothetische oder zukünftige Situation erfordert.

2.4.2.1 Hovys Textplaner

Hovy setzt die RST-Relationen in [Hovy, 1988] und [Hovy, 1991] für die Generierung von Texten ein, indem er die Relationen als Planoperatoren auffaßt. Er formalisiert dazu die Bedingungen – die nun Vorbedingungen bzw. *Subgoals* der Operatoren darstellen – und den Effekt mithilfe der Modal-Operatoren, die in [Cohen und Levesque, 1985] für die Ableitung von Sprechakten definiert wurden. Abb. 2.19 zeigt diese Formalisierung für die Relation *Circumstance*.

Der verwendete Modal-Operator *BMB* drückt die gemeinsame Annahme[54] zweier Akteure aus. Die Formel (*BMB a1 a2 p*) beschreibt die Annahme von Akteur *a1*, daß die beiden Akteure *a1* und *a2* gemeinsam *p* annehmen. Die Beschreibung der *Nucleus+Satellite requirements* sind von Hovy sehr speziell auf seinen Beispiel-Anwendungsbereich *Schiffahrt* abgestimmt. Für diesen Anwendungsbereich können *Kurs* (*Heading*) und *Ankunftszeit* (*Time*) als nähere Umstände angesehen werden.

Gegenüber der Beschreibung der Relation selbst, enthalten Hovys Operatoren drei zusätzliche Angaben. Die *Relation-phrases* können Wörter enthalten, durch die die beiden Komponenten der Relation verbunden werden. In Hovys Beispiel des *Sequence*-Operators sind dies “then” und “next”. Außerdem kann durch *Order* die Reihenfolge angegeben werden, in der die beiden Komponenten im Text angeordnet werden müssen. Die *Growth Points* sind zusätzliche optionale Bedingungen. Sie können, bspw. in Abhängigkeit von der zur Verfügung stehenden Information, vom Benutzerwissen

[54] s.a. Fußnote 49 zu gemeinsamen Annahmen.

```
Name: CIRCUMSTANCE
Results:
  ((BMB SPEAKER HEARER (CIRCUMSTANCE-OF ?X ?C)))
Nucleus+Satellite requirements/subgoals:
  ((OR (BMB SPEAKER HEARER (HEADING.R ?X ?C))
       (BMB SPEAKER HEARER (TIME.R ?X ?C))))
Nucleus requirements/subgoals:
  ((BMB SPEAKER HEARER (TOPIC ?X)))
Nucleus growth points:
  ((BMB SPEAKER HEARER (ATTRIBUTE-OF ?X ?ATT)))
Satellite requirements/subgoals:
  ((BMB SPEAKER HEARER (TOPIC ?C)))
Satellite growth points:
  ((BMB SPEAKER HEARER (ATTRIBUTE-OF ?C ?VAL)))
Order: (Nucleus Satellite)
Relation-phrases: ("")
```

Abbildung 2.19: RST Planoperator *Circumstance* aus [Hovy, 1991, S.90].

oder von der Länge des generierten Textes als zusätzliche Ziele bearbeitet oder auch weggelassen werden.

Die Textplanung wird auch bei diesem Ansatz als hierarchischer Prozeß aufgefaßt. Ausgehend von einem Diskursziel – bspw. daß der Hörer einen bestimmten Sachverhalt wissen soll – wird ein Planoperator ausgewählt, der dieses Ziel erreichen kann. Im nächsten Schritt wird versucht, die einzelnen und gemeinsamen Vorbedingungen für Nucleus und Satellite sowie ggf. die *Growth Points* zu erreichen. Werden dafür wiederum Operatoren eingesetzt, wiederholt sich der Vorgang bis die Ziele durch Oberflächensprechakte erreicht werden können. Das hierarchische Vorgehen ergibt sich auch bei diesem Ansatz nicht durch die explizite Definition von Dekompositions-Relationen. Vielmehr sind die Ziele der Operatoren jeweils nur durch weniger abstrakte Operatoren erreichbar. Außerdem werden für den endgültigen Textplan nur die Blätter einer solchen Herleitung berücksichtigt. Dies ist für den Plan gleichbedeutend mit der *Ersetzung* der abstrakten Operatoren durch die Operatoren, die für die Erfüllung der Vorbedingungen erforderlich sind und entspricht damit dem Vorgehen bei der Dekomposition eines Operators. Allerdings werden die abstrakten Operatoren nicht einfach aus der erzeugten Struktur entfernt, da sie ja die Struktur-Relationen des erzeugten Textes repräsentieren. Sie enthalten damit die Information, welche Rolle die einzelnen Teile des Textes spielen und werden deshalb für die endgültige Festlegung des Textes benötigt.

Hovy verzichtet – wie die meisten Nachfolger-Systeme – auf die Kritisierung des Planungsprozesses (vgl. [Hovy, 1993, Kap. 5.1.1]). Die Korrektheit des erzeugten Plans kann also nur sichergestellt werden, wenn Konflikte von vorn herein ausgeschlossen sind – etwa indem Ziele von keinem Operator zerstört werden können. In Hovys Beispielen ist dies der Fall. Hovys primäres Ziel war der Nachweis, daß die RST-Relationen als Planoperatoren in einem Textplaner eingesetzt werden können. Damit schuf er die Basis für die nachfolgend beschriebenen Arbeiten, die den Ansatz weiterentwickelten.

2.4.2.2 Weitergehende Ansätze

Moore und Paris [Moore und Paris, 1989], [Moore und Paris, 1993], [Moore, 1995] kritisieren, daß intentionale und rhetorische Ziele in Hovys Ansatz zu eng verknüpft sind. Mann und Thompson

hatten die rhetorischen Relationen in *Präsentations*- und *Thema*-Relationen eingeteilt. Beispiele für erstere sind *Motivation* und *Enablement*. Thema-Relationen sind bspw. *Purpose, Condition, Contrast* und auch *Circumstance* aus dem Beispiel oben. Paris und Moore zeigen, daß zwar die Präsentations-Relationen eindeutig mit einer Intention verknüpft sind – bspw. die Relation *Motivation*, mit dem Überreden des Hörers eine Handlung auszuführen –, nicht aber die Thema-Relationen. Die Relation *Circumstance* kann mit der Intention verbunden sein, daß der Sprecher möchte, daß der Hörer ein Objekt identifizieren kann oder daß dieser Eigenschaften eines Objekts kennenlernen soll. Diese Ziele können andererseits aber auch durch andere rhetorische Relationen erreicht werden – bspw. *Contrast*. Paris und Moore schlagen deshalb eine Entkopplung von Intention und rhetorischen Relationen vor, indem sie Planoperatoren definieren, deren Effekte entweder kommunikative Ziele sind, die die Intention des Sprechers repräsentieren, oder linguistische Ziele wie rhetorische Relationen oder Sprechakte. Die strikte Trennung ist insbesondere dann wichtig, wenn mit Reaktionen des Hörers gerechnet werden muß, also in einer Dialogsituation. Moore gibt in [Moore, 1995, S.105] als Beispiel eine Systemäußerung: "*Entfernen Sie den Deckel. Sie brauchen einen Phillips Schraubendreher. Er ist im oberen Fach des Werkzeugkasten. Haben Sie ihn?*", die der Hörer mit *Nein* beantwortet an. Geht man davon aus, daß der Hörer den Schraubendreher nicht identifizieren konnte, so ist es naheliegend zu versuchen, dieses gescheiterte kommunikative Ziel durch andere linguistische Mittel zu erreichen. Anstelle der näheren Umstände des Entfernens (*Circumstance-Relation* zwischen der *Information, daß der Deckel mit dem Schraubendreher entfernt werden soll* und der *Information über den Ort des Schraubendrehers*) können dem Hörer bspw. die Attribute des Schraubendrehers genannt werden. Dies entspricht aber einer anderen rhetorischen Relation, nämlich *Elaboration-Attribute*. Ohne die explizite Repräsentation des kommunikativen Ziels ist es aber für das System nicht möglich das Scheitern des Ziels überhaupt festzustellen, geschweige dazu zielgerichtet eine Alternative zu generieren.

Auch die Arbeiten von Horacek [Horacek, 1994], [Horacek, 1997] können als Erweiterung des auf RST-Relationen basierenden, hierarchischen Textplanungsansatzes angesehen werden. Horacek argumentiert, daß ein solcher Ansatz oft unnötig lange Texte erzeugt, wenn nicht berücksichtigt wird welche Informationen der Hörer bereits kennt oder selbst leicht aus dem Zusammenhang schließen kann. Auf diesen Aspekt, der bspw. auch in [Zukerman und McConachy, 1993b] und [Young, 1999] behandelt wird, wird noch in Kap. 4.4 *Berücksichtigung von Benutzer-Inferenzen* im Zusammenhang mit der Präsentation eines Plans eingegangen. Horacek setzt ein explizites Benutzermodell ein, das neben Kenntnissen auch die Inferenzfähigkeit eines Benutzers modelliert, um einen erzeugten Text auf die für den Benutzer wesentlichen Inhalte zu reduzieren.

Ein weiteres Beispiel das auf RST-basiertem Textplanen beruht ist das WIP-Projekt [Wahlster *et al.*, 1993]. André [André, 1995] konnte zeigen, daß sich der Ansatz für die Generierung von multimedialen Präsentationen erweitern läßt. Sie schlägt zunächst einen einfachen hierarchischen Planungsalgorithmus vor, der das Vorgehen beim reinen Textplanen um die Medienauswahl und Realisierung der Inhalte durch die verschiedenen Medien (Text und Grafik) erweitert. Die meisten Textplaner teilen den Planungsprozeß in mehrere unabhängige Phasen auf (vgl. [Reiter, 1994], [Fiedler, 1999] – Appelts Ansatz ist ein Gegenbeispiel). Diese Phaseneinteilung wird zwar auch von André übernommen, allerdings sieht sie für die letzte Phase medienspezifische Generatoren vor. Zwischen deren Ausgaben ergeben sich für eine zusammenhängende Gesamtdarstellung zwangsläufig Abhängigkeiten, die in dem einfachen Planungsalgorithmus nicht berücksichtigt werden. Sie schlägt deshalb vor, zunächst die zu präsentierenden Inhalte festzulegen und diese den medienspezifischen Generatoren zu übergeben. Werden bei der Generierung Abhängigkeiten festgestellt die sich nicht auflösen lassen, können die Generatoren eine Revision des ursprünglichen Präsentationsplans fordern. Dieses Vorgehen ist mit dem reaktiven Verfahren des Planungssystems SIPE [Wilkins, 1988] vergleichbar, das einen Plan mo-

difiziert, wenn er sich bei der Ausführung – bspw. durch unerwartete Ereignisse – als undurchführbar herausstellt.

2.4.3 Decompositional Partial-Order Causal Link Planning

Alle bisher vorgestellten Ansätze basieren auf hierarchischen Planungsverfahren mit Operatorabstraktion und lehnen sich am Planungssystem NOAH an. Dabei wird allerdings häufig auf die Kritisierung eines Plans, der aus einem Expansionsschritt resultiert, verzichtet. Die Planschritte, die aus einer Expansion hervorgegangen sind, können sich aber gegenseitig stören, wie in Kap. 2.2.2 *Kautz' Theorie der Planerkennung* Abb. 2.4 für die Dekomposition gezeigt wurde. Ohne Kritisierung und ggf. Beseitigung der Störung kann die Korrektheit des Plans nicht sichergestellt werden. Ein weiterer Kritikpunkt ist, daß Abhängigkeiten zwischen den Planschritten lediglich im Rahmen des Expansionsbaums – bspw. durch die RST-Relationen – berücksichtigt werden. Die kausalen Abhängigkeiten zwischen zwei Planschritten tieferer Abstraktionsebenen bleiben unberücksichtigt; bspw. wenn ein Planschritt eine Bedingung für den anderen erreicht.

Young, Moore und Pollack nahmen dies zum Anlaß, ein alternatives Planungsverfahren – *Decompositional Partial-Order Causal Link Planning* (*DPOCL*) – vorzuschlagen [Young und Moore, 1994], [Young *et al.*, 1994a], [Young *et al.*, 1994b]. Dieses erweitert den *SNLP*-Algorithmus [McAllester und Rosenblitt, 1991] für *Least Commitment Planen* um ein hierarchisches Verfahren mit Operatorabstraktion (vgl. Kap. 2.1.2.2 *Operatorabstraktion* und 2.1.1 *Least Commitment Planning*). Ein Operator (*Action*) wird durch ein *Action Schema* und eine optionale Menge von *Decomposition Schemata* definiert. Operatoren ohne Decomposition Schema sind *primitiv*, die anderen sind *nichtprimitiv*. Ein *Action Schema* entspricht einem (heute üblichen) nichthierarchischen Operator wie er im Kapitel über Handlungsplanung beschrieben wurde und wird durch eine Menge von Parametern, Vorbedingungen, Effekten und Bindungsbedingungen beschrieben. Die Elemente der Vorbedingungs- und Effekt-Mengen sind Literale, aus denen sich durch Konjunktion die Vorbedingung bzw. der Effekt des Operators ergibt. Ein *Decomposition Schema* eines Operators entspricht einer Dekomposition eines Operators wie im Abschnitt über Operatorabstraktion beschrieben; d.h. es besteht aus einer Menge von Planschritten, Reihenfolge- und Bindungs-Bedingungen sowie Causal Links. Außerdem müssen die Bedingungen für Vorbedingungs- und Effekt-Vererbung vom Operator auf seine Dekompositionen eingehalten werden. Zur Realisierung dieser Vererbungen fordern Young und Kollegen, daß jedes *Decomposition Schema* einen *Initial*- und einen *Final-Schritt* enthält, dessen Effekt- bzw. Vorbedingungs-Menge die Elemente der Vorbedingungs- bzw. Effekt-Menge des Operators enthalten. Für sich genommen entsprechen diese Schritte den Start- und Zielschritten beim Least Commitment Planen.

Der Planungsalgorithmus unterscheidet sich von Verfahren, wie sie in Lehrbüchern wie bspw. [Russell und Norvig, 1995] beschrieben werden i.w. durch zwei Punkte. Zum einen werden die Teilaufgaben des Algorithmus, *Auswählen des nächsten Ziels und eines passenden Operators*, *Dekomponieren eines nichtprimitiven Planschritts* und *Auflösung von Bedrohungen von Causal Links* nichtdeterministisch, insbesondere also ohne Vorgabe einer Reihenfolge, ausgewählt. Da die nichtdeterministische Auswahl auch zu der Entscheidung führen kann, daß *nur* die Schritte für das Least Commitment Planen ausgeführt werden, erbt DPOCL die Vollständigkeit des SNLP-Algorithmus. Gleiches gilt für die Korrektheit des Verfahrens, weil die Terminierungsbedingung nur korrekte Pläne zuläßt.

Die zweite Besonderheit besteht darin, daß DPOCL in einem Dekompositionsschritt *nicht* einen nichtprimitiven Planschritt durch eine seiner Dekompositionen *ersetzt*, sondern – wie bei den *Text*-Planern üblich – die Dekomposition dem Plan hinzufügt und durch einen *Decomposition Link* mit dem zugehörigen nichtprimitiven Planschritt verbindet. Der Link wird durch eine dreistellige Relation zwischen dem Planschritt sowie Initial- und Final-Schritt seiner Dekomposition realisiert (vgl. Abb.

2.20). Ein Plan, der durch einen solchen Planungsprozeß erzeugt wird, enthält also nicht nur die (ggf. partielle) Folge von primitiven Planschritten, die durch Causal Links verbunden sind, sondern auch alle nichtprimitiven Planschritte mit ihren Decomposition Links.

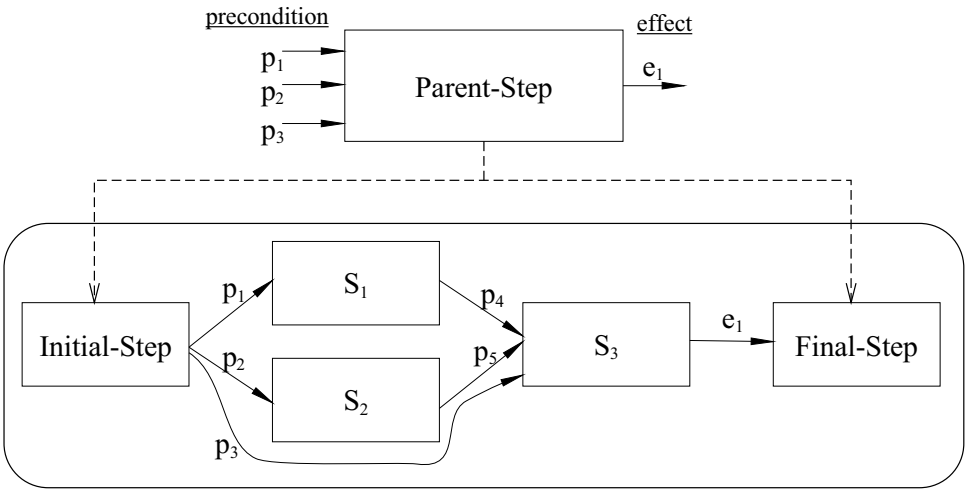

Abbildung 2.20: Decomposition Link (gestrichelt) zwischen Parent-Step sowie Initial- und Final-Step seiner Dekomposition

Bei geeigneter Definition der Operatoren können Causal Links und Decomposition Links die intentionalen und rhetorischen Relationen modellieren; also die beiden Relationstypen für die Generierung von natürlichsprachlichen Texten. Die Ziele betreffen dabei den mentalen Zustand des Hörers, der ja durch den Text verändert werden soll. Dementsprechend modellieren die Causal Links die intentionalen Relationen. Die rhetorische Struktur des zu erzeugenden Texts wird durch die Dekompositionsstruktur des erzeugten Plans repräsentiert, wenn – wie in Hovys Ansatz – die Planoperatoren zu den rhetorischen Relationen definiert werden und deren Dekompositionen dann (abgesehen von Initial- und Final-Schritt) durch die Komponenten der Relationen, also Nucleus und Satellite, gebildet werden.

Die Planstruktur ermöglicht auch die Unterscheidung zwischen *erwünschten Effekten* und *Nebeneffekten*. Young und Kollegen definieren, daß ein Effekt erwünscht ist, wenn er direkt oder indirekt zum Ziel beiträgt; d.h. wenn es einen Causal Link zum Ziel oder zu einem Planschritt, der einen erwünschten Effekt hat, gibt. Die Unterscheidung kann bspw. dann von Nutzen sein, wenn die Ausführung eines Plans nicht (alle) seine Ziele erreicht und der Plan deshalb modifiziert werden soll. Tritt durch die Modifikation der Fall ein, daß ein Planschritt keinen erwünschten Effekt mehr hat, so muß er aus dem Plan entfernt werden, weil er keinen Zweck mehr erfüllt und ein daraus generierter Text als inkohärent empfunden werden muß.

Abschließend sei bemerkt, daß die Kritik an der unzureichenden Definition der Semantik von Textplanern und damit verbunden die fehlende Beurteilungsmöglichkeit der Korrektheit eines Textplans sicherlich gerechtfertigt ist. Bisher fehlt aber auch der Nachweis, ob der Ansatz von Young und Kollegen für die Textplanung tatsächlich geeignet ist. Abgesehen davon, daß der in [Young *et al.*, 1994b] angekündigte Nachweis der formalen Eigenschaften des Algorithmus noch aussteht[55], entsprechen die Beispiele in [Young und Moore, 1994] und [Young *et al.*, 1994a] nicht der Definition für Operatoren. Bspw. wird im Beispiel aus [Young *et al.*, 1994a] als Effekt eine modallogische Formel

[55] pers. Kommunikation mit M.Young: ist in Arbeit; wird demnächst veröffentlicht.

(Bel(causes(I,U), L8)) verwendet, die sogar einen Wert (L8) für die Stärke der Annahme enthält, obwohl für Vorbedingung und Effekt lediglich eine Konjunktion von Literalen zulässig ist. In Fig. 3 in [Young und Moore, 1994] wird die Formel *causes(?prop2, ?prop1)* in der Dekomposition von *Support* sowohl als Bedingung als auch als Parameter des Planschritts *Cause-to-Believe-2* verwendet, obwohl Parameter von Operatoren prädikatenlogische Variablen sein sollen.

In den Arbeiten wird nicht gezeigt, daß die Auflösung von Bedrohungen von Causal Links tatsächlich für die Erzeugung eines Textplanes adäquat ist. In den Beispielen kommen solche Bedrohungen nicht vor. In der Handlungsplanung kann die Bedrohung eines Causal Links verhindert werden, wenn der Planschritt der die Bedrohung auslöst *zeitlich* vor oder nach den Schritten, die zu dem Causal Link gehören, angeordnet wird. Ob es in einem Text tatsächlich ausreicht bspw. eine Aussage, die für eine Argumentationskette benötigt wird, später einfach zu widerrufen ist zumindest fraglich. Ohnehin hat die *zeitliche* Reihenfolge von Planschritten für einen Text bzw. eine Argumentation nicht den gleichen entscheidenden Stellenwert wie in der Handlungsplanung.

Auch ein Nachweis, daß mit dem vorgeschlagenen Verfahren tatsächlich ein Text erzeugt wird, der durch die RST-Relationen strukturiert ist, wird nicht geführt. (Der DPOCL-Algorithmus ist in seiner Entscheidung frei und kann auch einen Plan ohne Dekomposition erzeugen.)

2.4.4 Dialogplanen

Obwohl es – wie für die Struktur von längeren Texten – auch spezielle Untersuchungen zur Struktur von Dialogen gibt (bspw. zu Beratungsdialogen [Nothdurft, 1984]), werden die Arbeiten zum Planen eines Dialogbeitrages nicht als eigenständiger Bereich angesehen, sondern finden sich in verschiedenen Forschungsbereichen der KI. An dieser Stelle sei deshalb lediglich auf entsprechende Bereiche verwiesen.

Die im Abschnitt 2.4.2.2 *Weitergehende Ansätze* genannten Ansätze zur Erweiterung des RST-basierten *Text*-Planens sind in erster Linie für die Planung eines *Dialog*-Beitrages wichtig. Dabei stellen Moore und Paris die Wichtigkeit der Behandlung von Benutzer-Reaktionen heraus. Horacek und André passen eine Systemäußerung gemäß einem Benutzermodell an den jeweiligen Benutzer an.

Der Forschungsbereich Planerkennung ist aus der Frage entstanden, wie ein KI-System einen *Dialog*beitrag erzeugen kann, der den Zielen und Plänen des Benutzers dienlich ist. Viele Arbeiten zur Planerkennung beschränken sich deshalb nicht nur auf die reine Erkennung eines Benutzer-Plans oder -Ziels, sondern erzeugen auch einen (natürlichsprachlichen) Dialogbeitrag.

Auch der Abschnitt 2.3 über das gemeinsame Planen mit einem Benutzer hat gezeigt, daß die Planerzeugung letztlich Basis für die Generierung einer angemessenen Systemantwort ist. In den Projekten TRAINS und TRIPS war die natürlichsprachliche Interaktion das zentrale Forschungsthema.

2.4.5 Zusammenfassung

Die planbasierten Ansätze zur Textplanung orientieren sich am Vorgehen hierarchischer Planungssysteme mit Operatorabstraktion. Allerdings ist – im Gegensatz zur Handlungsplanung – nicht die Einschränkung des Suchraums und die damit verbundene Effizienzsteigerung ausschlaggebend für die Verwendung dieses Ansatzes. Vielmehr spielt die am Ende von Kapitel 2.1.2.2 über Operatorabstraktion angesprochene Möglichkeit sachbereichsspezifisches Wissen in den Planungsprozeß einzubringen eine entscheidende Rolle. Dies betrifft vor allem das linguistische Wissen über die Struktur von Texten. Besonders der Ansatz, einen Text mittels rhetorischer Relationen hierarchisch zu strukturieren, erwies sich als fruchtbar und auch kompatibel zum hierarchischen Planungsansatz.

Gegenüber der Handlungsplanung, für die als Ergebnis letztlich nur die Folge primitiver, ausführbarer Planschritte interessant ist, ist für die Textplanung auch die hierarchische Struktur des Planungsprozesses wichtig, weil er die Struktur des zu erzeugenden Textes widerspiegelt. Dementsprechend werden nichtprimitive Planschritte nicht während des Planungsprozesses durch eine der Dekompositionen *ersetzt*, sondern bleiben in der Plan-Struktur und werden mit den Planschritten, die zu der Dekomposition gehören, verbunden. Die vollständige Plan-Struktur enthält dann sowohl Informationen über die kausalen Abhängigkeiten der Planschritte als auch die vollständige Information über den hierarchischen Erzeugungsprozeß. Bei geeigneter Modellierung repräsentieren diese die intentionale bzw. die rhetorische Struktur des zu erzeugenden Textes. Eine ausreichende formale Bearbeitung dieses Modells steht aber noch aus.

2.5 Hilfesysteme

Hilfesysteme sollen einen Benutzer bei der Arbeit mit einem Computersystem unterstützen. Die Bezeichnung subsumiert ein umfangreiches Spektrum von einfachen Systemen, die lediglich die Funktion eines Glossars haben, bis hin zu selbständig agierenden Systemen, die verschiedene Merkmale des Kontextes und des Benutzers berücksichtigen. Bauer stellt in [Bauer, 1988] Eigenschaften für die Charakterisierung von Hilfesystemen vor, die Thies in [Thies, 1994] unter besonderer Berücksichtigung *planbasierter* Hilfesysteme erweitert.

Entsprechend der Zielsetzung dieses Kapitels beschränkt sich dieser Abschnitt auf die Analyse planbasierter Systeme. Dies schließt Systeme, die bspw. auf Bayes'schen Netzen beruhen – wie [Gertner *et al.*, 1998] – oder die wie [Virvou und du Boulay, 1999] bewußt eine Alternative zum planbasierten Ansatz darstellen, auch dann aus, wenn sie als Hilfesystem interessante Eigenschaften haben. Gleiches gilt für tutorielle Systeme[56], die oft als Variante von Hilfesystemen angesehen werden, weil auch sie dem Benutzer (dem Lernenden, bzw. Studenten) Wissen vermitteln und ggf. dessen Anwendung trainieren sollen. Allerdings ist das zu vermittelnde Wissen meist allgemeiner, als dies für eine konkrete Hilfestellung sinnvoll wäre und die Menge des zu vermittelnden Wissens kann unabhängig von einer konkreten Aufgabenstellung im Voraus spezifiziert werden. Dabei rückt die Vermittlung des Wissens in den Vordergrund, für die psychologische und didaktische Kriterien relevant sind und deren Behandlung weit über die für diese Arbeit relevante Fragestellung nach der (formalisierbaren) Sicht von Plänen hinausgeht. Generell stehen in diesem Abschnitt nicht die Eigenschaften der Systeme für ihre *Beurteilung als Hilfesysteme* im Vordergrund, sondern sie werden nur im Zusammenhang mit der *Darstellung und Verwendung von Plänen* oder zur allgemeinen Charakterisierung behandelt.

Planbasierte Hilfesysteme zeichnen sich dadurch aus, daß sie bei der Festlegung von Inhalt und Form, den *Kontext* für die Hilfestellung berücksichtigen. Als Kontext wird dabei sowohl der Zustand des Systems mit dem der Benutzer arbeitet als auch der bisherige Verlauf der Interaktion und die Ziele des Benutzers verstanden. Auch die Eigenschaften und Fähigkeiten des Benutzers gehören zum Kontext einer Hilfestellung. Allerdings wird deren Berücksichtigung meist als eigenständiger Aspekt behandelt. Die Pionierarbeit für die Berücksichtigung des Kontextes in Hilfesystemen – das System WIZARD [Shrager und Finin, 1982], [Finin, 1983] – nutzte die Folge von Betriebssystem-Kommandos eines Benutzers, um *freiwillig*, d.h. ohne Anforderung, Hilfe anzubieten. Für eine solche *aktive* Hilfe ist die Berücksichtigung des Kontextes notwendige Voraussetzung für die Entscheidung *ob* und *wann* dem Benutzer *welche* Hilfe angeboten werden soll. Allerdings ist der Nutzen der Kontextberücksichtigung keineswegs darauf beschränkt. Für *passive* Hilfe, die nur auf Anforderung des Benutzers gegeben wird, entfällt lediglich die Frage, *ob* und *wann* Hilfe gegeben wird. Für die Fest-

[56] vgl. [McCalla, 2000] für eine aktuelle Übersicht und Perspektive des Forschungsbereichs

legung von Form und Inhalt einer Hilfestellung ist der Kontext sowohl für aktive als auch für passive Hilfe gleichermaßen von Bedeutung (vgl. [Breuker, 1990, S.26]).

Ein besonders wichtiger Aspekt des Kontextes sind die Ziele des Benutzers. Werden sie – wie bei aktiver Hilfe – nicht vom Benutzer angegeben, muß das Hilfesystem sie zunächst bestimmen. Dabei wird davon ausgegangen, daß die (bisherigen) Aktionen des Benutzers zielgerichtet sind; d.h. daß sie Teil eines Plans sind, mit dem er seine Ziele verfolgt. Die Feststellung der Ziele aus den Aktionen eines Benutzers ist die klassische Aufgabe der Planerkennung. Aktive Hilfesysteme verfügen deshalb immer über eine Planerkennungskomponente. Aber auch passive Hilfesysteme können Planerkennung nutzen, um Benutzerziele zu bestimmen, die über die explizit genannten hinausgehen. Bspw. verfügt das Hilfesystem UNIX-Consultant [Wilensky *et al.*, 1988] als ausschließlich passives Hilfesystem über eine Planerkennungskomponente.

Bei der Planerkennung für Hilfesysteme spielen die möglichen Fehlannahmen der Benutzer eine besonders wichtige Rolle. Zwar weisen bspw. auch bereits Pollack[57] [Pollack, 1986], [Pollack, 1990] und Carberry [Carberry, 1986] darauf hin, daß bei der Erkennung eines Benutzerplans die Annahmen des Benutzers berücksichtigt werden müssen und daß sich diese durchaus von den Annahmen eines beratenden Systems unterscheiden können. Bei Hilfesystemen sind solche Diskrepanzen aber essentiell, weil sie ja gerade die Hilfesituation charakterisieren. Wenn der Benutzer vollständige und ausschließlich korrekte Annahmen über das zu bedienende System hätte, würde er weder Hilfe benötigen noch wäre Hilfe möglich, da das Hilfesystem (im Idealfall) auch nur die gleichen vollständigen und korrekten Annahmen haben kann.

Für die Behandlung der Diskrepanz zwischen den "Vorstellungen" des Hilfesystems und den (angenommenen) des Benutzers werden unterschiedliche Ansätze verwendet, deren Auswirkungen auf die Darstellung von Plänen in den folgenden Abschnitten gezeigt wird. Dazu kommt die Frage nach der Herkunft der Pläne – sowohl der falschen als auch korrekten, die einem Benutzer als Rat gegeben werden.

2.5.1 Erkennung von falschen oder suboptimalen Plänen

Im Kapitel 2.2 *Planerkennung* wurde bereits festgestellt, daß die Verfahren zur Planerkennung zumeist von einer festen Menge von *korrekten* Plänen – der Planbibliothek – ausgehen, die (im Idealfall) alle Pläne beinhaltet mit denen sich ein beliebiges Ziel im Anwendungsbereich erreichen läßt. Wenngleich dies nicht immer explizit erwähnt wird, spiegelt diese Planbibliothek die Sicht des beobachtenden Systems wider. Alle Folgen von Aktionen, die keinem der Pläne entsprechen, müssen konsequenterweise als *falsch* klassifiziert werden. Allerdings ergibt sich damit kein Hinweis auf die *Intention* des Benutzers.

Als Alternative sind in WIZARD und vielen späteren Systemen auch speziell markierte Pläne repräsentiert, mit denen ein Ziel zwar angestrebt, aber nicht oder nur umständlich erreicht wird. Solche sogenannten *Bug-Lists* repräsentieren typische Fehler von Benutzern und wurden bereits im Bereich intelligenter Tutor-Systeme eingesetzt [Brown und Burton, 1978], [Anderson *et al.*, 1985]. Während WIZARD auf eine Fehlerkategorie beschränkt war, sehen nachfolgende Systeme meist mehrere Kategorien vor und unterscheiden dabei zumindest zwischen:

- *suboptimalen* Plänen – mit denen der Benutzer zwar sein Ziel erreicht, aber unnötig viel Aufwand betreibt oder Systemressourcen verbraucht und

- *falschen* Plänen – mit denen das Ziel nicht erreicht werden kann.

[57] s.a. Abschnitt 2.3 *Gemeinsames Planen (mit einem Benutzer)*

Mittermaier untersucht in [Mittermaier, 1995] verschiedene Möglichkeiten von Fehlern im Rahmen des Hilfesystems ComfoHelp [Krause *et al.*, 1993] zum Textverarbeitungssystem ComfoTex und die möglichen Reaktionen des Hilfesystems. Sie leitet daraus weitere Fehlerkategorien ab; bspw.

- *eventuell falsch* – für Pläne, die zwar ein angenommenes Ziel nicht erreichen, aber in einem andern Kontext Sinn machen und
- *leicht suboptimal* – für Aktionsfolgen, die nur unwesentlich von einem optimalen Plan abweichen und deren Auftreten die Unterbrechung des Benutzers durch ein Hilfeangebot nicht rechtfertigen würde.

Systemen die mit *Bug-Lists* arbeiten ist gemein, daß die repräsentierten Pläne ausschließlich den Anwendungsbereich betreffen – bspw. ein Betriebssystem (VMS in WIZARD, SINIX in SINIX-Consultant [Wahlster *et al.*, 1988]), ein Textverarbeitungssystem (ComfoTex in ComfoHelp) oder generell eine Fensteroberfläche (in IUST+ [Thies, 1994]). Jeder Plan (auch ein suboptimaler oder gar falscher) wird als Mittel zum Erreichen (genau) eines Ziels angesehen, dessen Planschritte Kommandos oder sonstige Funktionen des Systems sind. Das Ziel ist dabei eine Veränderung des Systemzustandes, der aber nicht explizit repräsentiert wird. Die Pläne werden nicht erzeugt, sondern vom Entwickler des Hilfesystems vorgegeben, der auch für die ausreichende Abdeckung und korrekte Klassifikation in eine Fehlerklasse verantwortlich ist. Abhängigkeiten der Planschritte[58], die sich bei einer Planerzeugung ergeben würden, werden nicht repräsentiert. Ein Plan besteht also letztlich aus einem Ziel das der Plan erreichen soll und einer linearen Folge von Systemfunktionen. Meist läßt die Plan-Repräsentation Parameter für die Objekte zu, auf die die Funktionen angewendet werden können. Abb. 2.21 zeigt die Darstellung eines solchen Plan-Schemas im SINIX-Consultant für das Löschen eines Verzeichnisses. Nach dem Plan-Namen (*DeleteDir*) ist das Ziel als String angegeben, dann folgen die zugehörigen Kommandos mit ihren Parametern. Parameter die mit ";" beginnen sind *Schemaparameter*; d.h. Variablen die bei der Erkennung gebunden werden – im Beispiel an den Namen des zu löschenden Verzeichnisses.

```
(DeleteDir (delete a directory and its content)
           (cd ;dir)
           (rm *)
           (cd ..)
           (rmdir ;dir))
```

Abbildung 2.21: SINIX-Consultant Plan-Schema für das Löschen eines Verzeichnisses nach [Wahlster *et al.*, 1988]

Abb. 2.22 zeigt die Darstellung von alternativen Plänen in ComfoHelp. Für das Ziel *Cursor an das Dokumentenende bewegen* ist der mit "*o*" markiert Plan optimal (die Tastenkombination Strg+Ende). Mit "a" werden (fast) gleichwertige Alternativen markiert. Der Plan, das *Dokumentenende mit den Cursor-Tasten zu erreichen*, ist leicht suboptimal (*ls*), wenn er mit wenigen (< x) Tastendrücken zu erreichen ist, ansonsten ist er suboptimal (*s*); d.h. ein Kandidat für die unaufgeforderte Hilfe.

Eine Sonderstellung nimmt IUST+ ein. Thies sieht eine hierarchische Planbibliothek (s. Abb. 2.23) vor, die durch die Plansprache GPL+ beschrieben wird. In der Abbildung stehen die Zielfahnen für (Sub-)Ziele, ein einzelnes Rechteck für eine Aktion und ein Stapel Rechtecke für einen Plan. Die Schritte eines Plans sind Aktionen oder Subziele. Für Schritte ist keine strikte Ausführungsreihenfolge vorgegeben. Stattdessen können Bedingungen für die zeitliche Anordnung der Planschritte

[58] vgl. *Causal Links* im Kap. 2.1.1 *Least Commitment Planning*

C_DokEnde

o	<Strg+Ende>
a	BEARBEITEN/Gehe_zu;Ende;OK
ls	Scrollen;C [Zielpos=DokEnde]
ls	n<CTasten> [Zielpos=DokEnde;n<x]
s	n<CTasten> [Zielpos=DokEnde;n>x]

Abbildung 2.22: Alternative ComfoHelp-Pläne für das Ziel *Cursor an das Dokumentenende bewegen* aus [Mittermaier, 1995, S.204]

definiert werden, womit sich eine oder mehrere zulässige Ausführungsreihenfolgen ergeben. Durch Bindungsbedingungen wird der Zusammenhang zwischen den Parametern des Plans und der Subziele bestimmt. Den Subzielen sind wiederum – i.a. mehrere alternative – Pläne zugeordnet, mit denen die Ziele erreicht werden können. Aktionen sind dagegen nicht weiter teilbar. Ihnen können Systemfunktionen zugeordnet werden. Diese Sichtweise von Plänen ist zur Handlungsplanung mit Operatorabstraktion vergleichbar (vgl. Kap. 2.1.2.2 *Operatorabstraktion*). Dabei entspricht die Beziehung zwischen Subzielen und den zugehörigen alternativen Plänen der Dekomposition; die Aktionen entsprechen den primitiven Operatoren. Allerdings sieht Thies keine Planerzeugung vor und verzichtet auf Definition und Berücksichtigung von Vorbedingungen und Effekten[59]. Kausale Abhängigkeiten zwischen den Planschritten werden hier ebenfalls nicht berücksichtigt.

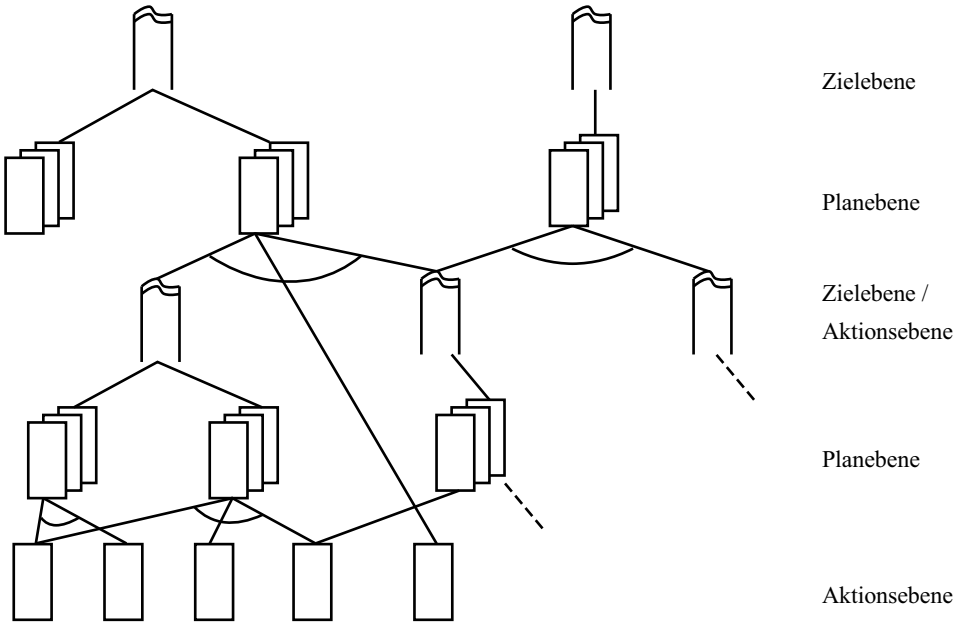

Abbildung 2.23: IUST+ Planbibliothek aus [Thies, 1994, S.81]

Thies unterscheidet allerdings zwei Typen von Aktionen: *navigatorische* und *nicht-navigatorische*. Unter navigatorischen Aktionen werden solche verstanden, die direkt mit der Manipulation der Fensteroberfläche zu tun haben, wie etwa das *Rollen eines Fensters* oder das *Selektieren ei-*

[59] Zwar kann für einen Plan eine *Precondition* definiert werden, die anzeigt wann der Plan eingesetzt werden darf. Die Bedingung wird allerdings nur für die Erzeugung eines Hilfetextes verwendet. Die *Pre-* und *Postdescriptions* von Aktionen und Zielen stellen parametrisierbare Texte dar und haben lediglich erklärenden Charakter.

nes Menüeintrags. Da solche Aktionen in sehr vielen Plänen an fast allen Stellen auftreten können, schließt Thies sie von der Planerkennung aus und erwartet stattdessen, daß das Anwendungssystem dem Planerkenner die *nicht-navigatorischen* Aktionen des Benutzers meldet, die der Funktionalität des Anwendungssystems entsprechen; bspw. das *Erzeugen eines Fensters* oder das *Einfügen eines Textes*. Die beschriebene Planbibliothek bezieht sich dementsprechend nur auf nicht-navigatorische Aktionen. Die Erzeugung eines Hilfetextes *basiert* zwar auf den Plänen der Bibliothek, allerdings müssen dafür auch die navigatorischen Aktionen berücksichtigt werden. Jede (nicht-nagivatorische) Aktion eines Plans muß durch eine Folge von navigatorischen Aktionen realisiert werden. Diese Folgen werden durch einen einfachen Planungsprozeß erzeugt. Für die navigatorischen Aktionen werden deshalb – analog zu den Operatoren der Handlungsplanung – Vorbedingungen und Effekte definiert.

Durch die Repräsentation falscher bzw. suboptimaler Pläne läßt sich bei allen Systemen das (wahrscheinlich) intendierte Ziel eines Benutzers auch dann erkennen, wenn die Aktionen des Benutzers nicht bzw. nur über Umwege zum Ziel führen würden. Als Hilfestellung oder Verbesserungsvorschlag kann ihm dann der für das erkannte Ziel *optimale Plan* der Planbibliothek präsentiert werden; bzw. – wie bei IUST+ – ein darauf basierender verfeinerter Plan. Für eine Erkennung und Erklärung der Fehlerursache reicht dies aber nicht aus.

2.5.2 Erkennung von Fehlerursachen

Beschränkt sich eine Hilfestellung ohne Rücksicht auf mögliche Fehleinschätzungen des Benutzers ausschließlich darauf einen korrekten Plan für das angenommene Ziel zu präsentieren, kann dies beim Benutzer zu neuen Fehlannahmen und/oder zu einer Rückweisung des Plans führen, weil er den Plan für falsch hält. Anhand der (indirekten) Frage:

I want to talk to Kathy, so I need to find out the phone number of St.Eligius.

demonstriert Pollack in [Pollack, 1990] wie eine Fehlannahme eines Ratsuchenden zu einem falschen Plan (der sein Ziel nicht erreicht) führt.

Im Beispiel würde die Beantwortung der Frage durch die Nennung der Telefonnummer von St.Eligius für das Ziel *Kathy anzurufen* nicht weiterhelfen, weil sich Kathy nicht in St.Eligius, sondern zu Hause aufhält. Die Nennung der Telefonnummer von zu Hause wäre zwar für einen Plan der das Ziel erreicht hilfreich, kann aber (u.a.) zu der Fehlannahme führen, daß die genannte Telefonnummer diejenige von St.Eligius ist. Kennt der Ratsuchende Kathys Telefonnummer zu Hause wird er annehmen, daß ihm die falsche Nummer genannt wurde und die Antwort zurückweisen. Eine kooperative Antwort darf aber weder falsche Angaben enthalten noch dürfen falsche Annahmen daraus ableitbar sein[60]. Angemessen wäre also eine Antwort, die zumindest die (implizite) Fehlannahme über den Aufenthaltsort von Kathy korrigiert.

Pollacks Sicht über die Annahmen – und damit auch mögliche Fehlannahmen –, die mit einem Plan eines Benutzers verbunden sind, wurden bereits in Kap. 2.3.2 *Pläne als mentale Zustände* behandelt. Sie betrachtet Pläne zwar nur eingeschränkt, ihre Analyse über die Intentionen und Annahmen eines Akteurs der einen Plan ausführen will kann aber auf Pläne im allgemeinen übertragen werden. Wichtig sind besonders die Annahmen, daß jeder einzelne Planschritt ausführbar ist und einen Beitrag für den Plan leistet. Damit verbunden sind Annahmen über die Vorbedingungen und – um zu beurteilen *welchen* Beitrag ein Planschritt leistet – über die Effekte eines Planschritts.

Auf dieser Basis können die mit einem Plan verbundenen Annahmen abgeleitet werden, wenn festgestellt wird, daß der Benutzer die Absicht hat diesen Plan auszuführen. Sind falsche Annahmen darunter (Annahmen, die den Annahmen des Systems widersprechen), muß eine kooperative Antwort bzw. Hilfestellung eine entsprechende Korrektur beinhalten.

[60] vgl. Grices Kooperativitätsbedingungen [Grice, 1975].

Quilici widmet sich in [Quilici, 1989] der Frage, wie ein Benutzer über falsche Annahmen, die mit seinem Plan verbunden sind, aufgeklärt und wie ihm eine Hilfestellung gegeben werden kann. Die Grundlage für beides sind die Annahmen des Systems über Pläne, von denen das System annimmt, daß der Benutzer sie *nicht* hat. Quilici schlägt dazu eine Taxonomie von möglichen Fehlannahmen und potentiellen Erklärungen vor. Annahmen über Pläne werden durch die Relationen *causes* und *applies* zwischen einem Plan und einem Effekt sowie *enables* zwischen Vorbedingung, Plan und Effekt repräsentiert. Um auch falsche Annahmen des Benutzers repräsentieren zu können, werden sie von den Annahmen des Systems getrennt. Sowohl *causes* als auch *applies* drücken aus, daß der Plan den Effekt erreicht. *Applies* gilt allerdings nur dann, wenn (der Akteur annimmt daß) der Plan *normalerweise* für den Effekt verwendet wird. Quilici gibt keine präzise Definition, sondern erläutert den Unterschied am Beispiel des UNIX-Kommandos "*rm **". Dieser "Plan" bewirkt (*causes*) zwar das Löschen einer Datei, wird aber normalerweise nicht für das Löschen *einer* Datei verwendet, weil damit *alle* Dateien gelöscht werden.

Die Relationen *causes* und *enables* sind nach Quilicis Ansicht semantisch identisch zu Pollacks Relationen *gen* bzw. *cgen* [Pollack, 1986], [Pollack, 1990]. Allerdings repräsentiert Pollack damit die kausalen Abhängigkeiten zwischen Aktionen und nicht zwischen Aktionen und Zuständen wie Quilici dies beschreibt. Gleichwohl deuten die Beispiele auf eine Verwendung im Sinne von Pollack hin. Bspw. hat das System die Annahme, daß der Plan "*rm*" im Zustand *writeable directory* den Zustand *the file's removal* bewirkt (*enables*). Letzteres ist aber eher als Aktion denn als Zustand zu verstehen und ist damit sehr vergleichbar zu Pollacks Beispiel, bei dem das *Drücken der DEL-Taste* (unter bestimmten Voraussetzung) die Aktion *aktuelle-mail-löschen* bewirkt.

Außer den Relationen selbst sieht Quilici auch die Repräsentation ihrer Verneinungen vor. Damit kann explizit ausgedrückt werden, daß das System glaubt, daß ein Plan einen bestimmten Effekt *nicht* hat. Diese Kenntnis kann bei der Ermittlung einer Fehlererklärung genutzt werden und ist vergleichbar zu den *Bug-Lists* im vorangegangenen Abschnitt.

Die Pläne selbst sind primitive Operatoren oder eine Folge davon. Quilici verwendet die Pläne aber ausschließlich atomar. In den Beispielen für seinen Anwendungsbereich *UNIX-Ratgeber* sind es einzelne UNIX-Kommandos.

Calistri-Yeh schlägt in [Calistri-Yeh, 1991] ebenfalls eine Klassifikation von möglichen Fehlannahmen bzgl. Plänen vor. Seine Einteilung basiert auf einer detaillierteren Repräsentation eines Plans. Demnach besteht ein Plan aus den folgenden vier Komponenten:

- *Parameter*
- *Vorbedingungen* – im Sinne von Anwendungsvorbedingungen, die vor der Ausführung des Plans erfüllt sein müssen, die der Akteur aber nicht beeinflussen kann.
- *Planschritte* – wobei die folgenden Typen unterschieden werden:
  - die Schritte, die für die Ausführung des Plans notwendig sind.
  - Alternativen für die Planausführung.
  - Hinzu- und Lösch-Listen von Aussagen zur Modellierung atomarer Aktionen[61].
- (Zeitliche) *Bedingungen* für
  - die zeitliche Anordnung der Planschritte.
  - die Bildung von gegenseitig disjunkten Gruppen alternativer Planschritte.

[61] Calistri-Yeh gibt keine genauere Beschreibung. Es liegt nahe, sie wie die entsprechenden Listen von STRIPS-Operatoren zu verstehen.

Die Sichtweise eines Plans hat wieder Ähnlichkeit zu derjenigen der Handlungsplanung mit Operatorabstraktion. Allerdings werden hier Planschritte, wie sie in der Handlungsplanung betrachtet werden, mit der Dekomposition und dem Effekt des Plans auf eine Ebene gestellt. Dies macht den Kunstgriff der Bedingungen für die Gruppierung alternativer Planschritte erforderlich.

Calistri-Yeh bezieht die potentiellen Fehlannahmen auf die Komponenten eines Plans und kommt dabei zu folgenden Möglichkeiten:

- *Verletzte Parameterbindung* – der Plan kann nicht zum Ziel führen, weil die falschen Objekte verwendet werden.
- *Unerfüllte, verwechselte* oder *fehlende Vorbedingungen*[62] – der Plan kann nicht zum Ziel führen, weil die notwendigen Vorbedingungen nicht erfüllt sind.
- *Verwechselte* oder *fehlende Planschritte* – der Plan kann nicht zum Ziel führen, weil er nicht richtig ausgeführt wird.
- *Falsche Reihenfolge* – der Plan kann nicht zum Ziel führen, weil die Planschritte in einer falschen Reihenfolge ausgeführt werden.

Weitere Fehlerkategorien beeinträchtigen nicht das Erreichen des Ziels, es wird aber unnötiger, zusätzlicher Aufwand getrieben oder das Ergebnis des Plans ist nicht optimal. In der Sprechweise des vorangegangenen Abschnitts handelt es sich um suboptimale Pläne. Diese liegen vor, wenn der Benutzer von *zusätzlichen Vorbedingungen* ausgeht, der Plan (evtl. deshalb) *zusätzliche Schritte* enthält oder wenn der Benutzer eine *ungünstige Wahl* trifft, wenn es mehrere Alternativen gibt.

Bemerkenswert ist, daß Calistri-Yeh – bspw. im Gegensatz zu Quilici – den Effekten von Plänen nur eine sehr untergeordnete Rolle beimißt und sie lediglich für primitive Pläne vorsieht. Diese Sicht entspricht vielen Ansätzen der Planerkennung (vgl. Kap. 2.2 *Planerkennung*), sie eignet sich allerdings nicht für die Analyse von Situationen, in denen ein Benutzer zwar einen Plan korrekt ausgeführt, dieser aber ein anderes, als das vom Benutzer erwartete Ergebnis erzielt.

Als Besonderheit von Calistri-Yehs Ansatz sei hervorgehoben, daß die Erkennung falscher Pläne ohne eine explizite Repräsentation solcher Pläne auskommt. Der Erkennungsalgorithmus seines Systems PATHFINDER versucht stattdessen Pläne als Modifikation der korrekten Pläne, die in der Planbibliothek repräsentiert sind, zu erkennen. Das Einschätzen der Benutzerannahmen als (leichte) Modifikation der Systemannahmen ist vergleichbar zum Ansatz der *Perturbation* in der Benutzermodellierung [Ballim und Wilks, 1991]. Die möglichen Modifikationen eines Plans leitet Calistri-Yeh aus den genannten Fehlermöglichkeiten ab, wobei er davon ausgeht, daß diese unterschiedlich wahrscheinlich auftreten.

2.5.3 Erzeugen von Plänen

Sowohl für die Erkennung von Zielen und Plänen eines Benutzer als auch für die Erzeugung einer Hilfestellung setzen viele Systeme eine statische Planbibliothek ein, die (meist) den Anspruch hat, alle bekannten Pläne für den Anwendungsbereich zu enthalten. Dazu gehören bei Hilfesystemen auch Pläne, die ihre Ziele nicht oder nur unzureichend bzw. umständlich erreichen. Um eine größere Flexibilität hinsichtlich der konkreten Situation und des jeweiligen Benutzers zu erreichen, passen einige Systeme die Pläne der Bibliothek an die jeweilige Situation oder den Benutzer an. Bspw. berücksichtigt IUST+ den jeweiligen Zustand des Anwendungssystems bei der Generierung der Folge von navigatorischen Aktionen für eine Hilfestellung. PATHFINDER erkennt auch Pläne, die nicht in

[62] der Unterschied dieser Varianten ist – wie Calistri-Yeh selbst sagt – etwas spitzfindig und für die Frage, was unter einem Plan verstanden wird, nicht relevant.

der Bibliothek enthalten sind, sondern benutzerabhängig von den darin enthaltenen korrekten Plänen abweichen. Die Idee eine Hilfestellung auf Basis einer Plangenerierung zu erzeugen, findet sich in EUROHELP [Breuker, 1990] – allerdings benutzer*un*spezifisch. Der Planer wurde zudem nicht in das Hilfesystem integriert (vgl. [Breuker, 1990, S.347]).

Einen integrierten Ansatz für Planerkennung und Planerzeugung verfolgt das Hilfesystem PHI [Bauer *et al.*, 1993]. Die Zusammenarbeit von Planerkenner und Plangenerator erfolgt dabei in zwei sogenannten Verzahnungs-Modi (*cross-talk modes*). Im ersten stellt der Plangenerator dem Erkenner abstrakte Pläne zur Verfügung, die der für die jeweilige Situation konkretisiert; d.h. bspw. Variablen an konkrete Objekte bindet oder aus alternativen Aktionen eine geeignete auswählt. Der Plangenerator erzeugt die Pläne anhand von Zielen, die für den Anwendungsbereich oder den jeweiligen Benutzer typisch sind. Dazu *kann* er auch auf bereits vorliegende Pläne zurückgreifen und sie ggf. modifizieren. Stellt der Planerkenner fest, daß der Benutzer einen suboptimalen Plan verfolgt, führt dies zum zweiten Verzahnungs-Modus, in dem der Plangenerator aufgefordert wird, zu einem erkannten suboptimalen Plan einen optimalen zu erzeugen. Auch in diesem Fall kann der Plangenerator auf bereits vorliegende Pläne zurückgreifen und sie ggf. modifizieren.

Die verschiedenen Einsatzbereiche der Pläne erfordern, daß die Plangenerierungskomponente an ihre jeweilige Aufgabe anpaßbar ist. Pläne für den Erkennungsprozeß abstrahieren sinnvollerweise von den konkreten Objekten und – wo möglich – auch teilweise von einer linearen Anordnung aller Planschritte (vgl. Bsp. weiter unten). Ein solcher abstrakter Plan ist damit für viele Situationen verwendbar und bewahrt den Plangenerator davor sehr viele Pläne zu erzeugen, die sich lediglich geringfügig unterscheiden. Für Pläne, die als Basis für eine Hilfestellung dienen sollen, ist dagegen eine Anpassung an die konkrete Situation sinnvoll. Auch die Eigenschaften des Benutzers, wie bspw. Kenntnisse oder Präferenzen, sollten dann bei der Plangenerierung Berücksichtigung finden.

Die dafür notwendige Adaptierbarkeit des Plangenerierungsprozesses wird erreicht, indem ein deduktives Planungsverfahren eingesetzt wird, das durch spezifische *Taktiken* gesteuert wird. Als Formalismus für die Darstellung von Aktionen und Plänen kommt dabei LLP (*logical language for planning*) [Biundo *et al.*, 1992] zum Einsatz. Dies gilt nicht nur für die Plangenerierung, sondern – entsprechend der engen Kopplung beider Komponenten – auch für Planerkennung. LLP ist eine Intervall-basierte modale Temporallogik, die auf Prädikatenlogik erster Stufe basiert und *Choppy-Logik* [Rosner und Pnueli, 1986] mit einer *Zeitlogik für Programme* [Kröger, 1987] kombiniert. Die Zeit wird durch diskrete Zustände modelliert. Ein Zustand in LLP besteht neben einer Beschreibung der veränderlichen Gegebenheiten des Anwendungsbereichs auch aus einer *Kontrollkomponente*, die die Aktion die in diesem Zustand ausgeführt wird beschreibt.

Ein (Zeit-)Intervall ist eine zusammenhängende Folge von Zuständen. Für LLP-Formeln sind zusätzlich zu den normalen prädikatenlogischen Operatoren als Modaloperatoren "$\bigcirc$" (*next*) und "$\Box$" (*always*) sowie "$\Diamond$" (*sometimes*) als Abkürzung für $\neg\Box\neg$ und der zweistellige Operator ";" (*chop*) für die zeitliche Sequenz vorgesehen. Die unären Operatoren sind dabei auf ein Referenzintervall bezogen; d.h. bspw., daß eine Formel $\Box\phi$ genau dann *in einem Intervall i gilt*, wenn ϕ in i und jedem Intervall gilt das nach i beginnt. Die Formel $\phi\ ;\ \psi$ gilt genau dann (im Referenzintervall i), wenn sich i in zwei direkt anschließende Intervalle i_1 und i_2 aufteilen läßt und ϕ in i_1 und ψ in i_2 gilt[63]. Dazu kommen – vergleichbar zu Programmiersprachen – die Kontrollstrukturen Selektion (*if then else*) und Iteration (*while do od*), die eine kompakte Darstellung komplexer Pläne erlauben.

Pläne werden in LLP durch Formeln mit folgender Struktur dargestellt:

- $EX(c)$ – Das Spezialprädikat ist in einem Intervall wahr, wenn c im ersten Zustand des Intervalls ausgeführt wird. c ist dabei ein Term, der eine atomare Aktion bezeichnet (im PHI-Anwendungsbereich ein UNIX-Mail-Kommando).

[63] wobei i_1 endlich sein muß.

- a ; b – die Sequenz der beiden Plan-Formeln a und b.
- **if** e **then** a **else** b $:\Leftrightarrow$ $[e \rightarrow a] \vee [\neg e \rightarrow b]$
  wobei a und b Plan-Formeln sind und e eine Bedingung ist (d.h. eine Formel die keine Zeitoperatoren enthalten darf und keine Plan-Formel ist).
- **while** e **do** a **od** ; b $:\Leftrightarrow$ [**if** e **then** [a ; **while** e **do** a **od** ; b] **else** b]
  wobei a und b Plan-Formeln sind und e eine Bedingung ist.

Der Beispielplan aus [Bauer *et al.*, 1993] zu der Aufgabe eine

Mail x einer bereits geöffneten Mailbox M lesen

hat folgende LLP-Darstellung:

$$\textbf{if } df(x,M) = T \textbf{ then } EX(undelete(x,M)) \ ; \ EX(type(x,M))$$

Für das Lesen einer Mail selbst eignet sich die Aktion *type*. Diese hat die Vorbedingungen, daß die Mailbox geöffnet sein muß und nicht bereits mit *Löschen* markiert sein darf. Der Plan sieht deshalb die Aktion *undelete* für das Entfernen einer Lösch-Markierung vor. Diese Aktion wird allerdings nur dann ausgeführt, wenn die Mail eine Lösch-Markierung hat; d.h. wenn $df(x,M) = T$.

Der Plan abstrahiert sowohl von der konkreten Mail als auch vom Zustand der Lösch-Markierung. Die Konkretisierung des Plans, also die Variablenbindung und die Entscheidung, ob die Aktion *undelete* notwendig ist oder nicht, kann durch den Planerkennungsprozeß erfolgen. Eine explizite Repräsentation von Abstraktions- oder Dekompositionsbeziehungen zwischen Plänen ist in PHI ebensowenig vorgesehen wie die der kausalen Abhängigkeiten zwischen einzelnen Planschritten.

Die Erzeugung eines Plans basiert auf einer *Planspezifikation*, die in LLP die folgende Form hat:

$$[preconditions \wedge \mathsf{Plan}] \rightarrow goals$$

preconditions bzw. *goals* stehen für LLP-Formeln, mit denen die Startsituation bzw. das Ziel beschrieben wird. Dabei enthält die *preconditions*-Formel normalerweise keine Modaloperatoren, während die *goals*-Formel irgendwann in der Zukunft gelten soll; d.h. sie hat meist die Form "$\Diamond goals$". Der Planungsprozeß hat die Aufgabe eine logische Formel für die Metavariable Plan zu finden, sodaß deren Einsetzung in die Spezifikation zu einer wahren Aussage führt. Die Aufgabe wird von einem taktischen Theorem-Beweiser durchgeführt, der versucht die Planspezifikation unter Zuhilfenahme von Axiomen über die Veränderungsmöglichkeiten im jeweiligen Sachbereich zu beweisen. Die Axiome werden nicht im einzelnen modelliert, sondern während des Beweisprozesses nach Bedarf aus Axiomenschemata abgeleitet, die die möglichen Aktionen im Sachbereich modellieren. Da der Formalismus – wie im Situationskalkül[64] – Schlußfolgerungen über einzelne Situationen und die darauf bezogene Gültigkeit von Aussagen ermöglicht, sind für den Deduktions-Prozeß auch Rahmen-Axiome erforderlich, die angeben was durch eine Aktion *nicht* verändert wird. Die Rahmen-Axiome werden ebenfalls bei Bedarf aus den Axiomenschemata abgeleitet. Ein Axiomenschema hat folgenden Aufbau:

$$[\ precond \wedge \mathrm{P}_{effect} \wedge EX(action)\ ] \rightarrow \bigcirc \mathrm{P}$$

wobei freie Variablen implizit allquantifiziert sind.

precond ist eine Formel ohne Modaloperatoren, die die Vorbedingung der Aktion *action* beschreibt. Der Effekt wird durch eine als *effect* bezeichnete Substitution (bzw. Folge von Substitutionen) beschrieben, die auf die Metavariable P in der Prämisse des Axiomenschemas angewendet

[64] vgl. Kap. 2.1 *Handlungsplanung*

wird. Die Substitution gibt an welche Teilformeln durch das Ausführen der Aktion wahr bzw. falsch werden. Das Ausführen der Aktion wird durch das Spezialprädikat *EX* modelliert.

Um aus dem Schema die benötigten Axiome zu erzeugen, setzt der Beweiser während des Planungsprozesses geeignete Formeln für die Metavariable P ein und führt die Substitution durch. Alle Teilformeln der Prämisse beziehen sich auf ein Referenzintervall. Der Modaloperator $\bigcirc$ in der Konklusion zeigt an, daß die für P eingesetzte Formel in den Nachfolgeintervallen[65] gilt.

Das Axiomenschema für das Mail-Kommando *type* lautet bspw.

$$[of(mb) = \mathrm{T} \wedge df\ (x, mb) = \mathrm{F} \wedge \mathbf{P}_{[\boldsymbol{rf(x,\,mb)}\,/\,\mathbf{T}]} \wedge EX(type(x, mb))\ ] \rightarrow \bigcirc \mathbf{P}$$

Dabei stellt $[of(mb) = \mathrm{T} \wedge df\ (x, mb) = \mathrm{F}]$ die Vorbedingung der Aktion *type* dar; d.h. die Mailbox *mb* muß geöffnet sein und die Nachricht *x* darin darf nicht gelöscht sein. Der Effekt der Aktion besteht darin, daß das Read-Flag dieser Nachricht wahr wird. Dementsprechend lautet die *effect*-Substitution $[rf(x, mb)\ /\ \mathrm{T}]$.

Um bspw. ein Axiom zu erzeugen, das die durch Ausführen des Kommandos *type* hervorgerufene Veränderung beschreibt, wird für P die Formel $rf(x, mb) = \mathrm{T}$ eingesetzt. Durch die Anwendung der Substitution wird aus $\mathrm{P}_{[rf(x,mb)/T]}$ die Tautologie T = T und das resultierende Axiom lautet:

$$[of(mb) = \mathrm{T} \wedge df(x, mb) = \mathrm{F} \wedge \mathbf{T = T} \wedge EX(type(x, mb))\ ] \rightarrow \bigcirc \boldsymbol{rf(x, mb) = \mathbf{T}}$$

Wird für P eine Formel eingesetzt, die nicht den Term $rf(x, mb)$ enthält, bleibt P in der Prämisse unverändert. Wird bspw. für P $df(x, mb) = \mathrm{F}$ eingesetzt ergibt sich:

$$[of(mb) = \mathrm{T} \wedge df(x, mb) = \mathrm{F} \wedge \boldsymbol{df(x, mb) = \mathbf{F}} \wedge EX(type(x, mb))\ ] \rightarrow \bigcirc \boldsymbol{df(x, mb) = \mathbf{F}}$$

Damit ist eine solche Formel im Folgeintervall nach Ausführen der Aktion genau dann wahr, wenn sie auch davor wahr war. Das erzeugte Axiom ist demnach ein Rahmen-Axiom.

Um zu verhindern, daß diese Erzeugung von Axiomen bzw. der Beweisprozeß insgesamt ziellos erfolgt, werden Taktiken eingesetzt, die das Vorgehen des Beweisers steuern. Solche Taktiken müssen nicht fest vorgegeben sein, sondern können je nach Anwendungsfall variieren. Dengler zeigt in [Dengler, 1996] wie sie in Form von Ableitungsregeln dazu eingesetzt werden können, (einige der) Verarbeitungsstrategien klassischer Planungssysteme zu modellieren. Beispiele sind u.a. das *progressive* Vorgehen (Zielsuche ausgehend von einer Startsituation), das *regressive* Vorgehen (ausgehend vom Ziel feststellen welche Aktionen und damit welche Voraussetzungen dafür notwendig sind) und das *Aufteilen eines Ziels* in mehrere Teilziele. Für letzteres gibt Dengler die *Task Reduction Refinement* Regel an:

$$\frac{\mathrm{T} \rightarrow \Diamond[g_1 \wedge \Diamond g_2]}{\mathrm{T} \rightarrow \Diamond g}$$

Die Regel besagt, daß das Ziel g erreicht werden kann, indem die beiden Teilziele g_1 und später g_2 erreicht werden. Trotz Namensähnlichkeit darf die Regel allerdings nicht mit *Task Reduction* beim HTN-Planen[66] [Erol *et al.*, 1994c], [Erol *et al.*, 1994a] verwechselt werden. Dort geht es darum, einen nichtprimitiven *Operator* zu reduzieren; d.h. durch ein Task-Netzwerk bzw. eine Dekomposition zu ersetzen. Es geht aber nicht um die Aufteilung eines *Ziels*.

Die Beeinflussung des Verhaltens des Planers durch die Definition von Ableitungsregeln kann in PHI dazu genutzt werden, den Planungsprozeß an seine unterschiedlichen Aufgaben anzupassen. Darüber hinaus stellt Dengler fest, daß auch der Benutzer auf diese Weise volle Kontrolle über das

[65] Als Nachfolgeintervall seien alle Intervalle bezeichnet, die mit der zweiten Situation des Referenzintervalls beginnen.
[66] Planen mit Operatorabstraktion, vgl. auch Kap. 2.1.2.2 *Operatorabstraktion*

Vorgehen des Planers erhält[67]. In [Bauer *et al.*, 1996] wird vorgeschlagen dem Benutzer eine Schnittstelle zur Verfügung zu stellen, die vom konkreten Formalismus abstrahiert und die notwendigen Definitionen dadurch vereinfacht.

Dennoch bleibt es fraglich, ob einem Benutzer, der in einer normalen Anwendungssituation Unterstützung durch ein Hilfesystem benötigt, zugemutet werden kann, zunächst eine geeignete Strategie für die Erzeugung der Hilfe oder gar für die Erkennung seiner Ziele zu definieren. Ohnehin ist nicht bekannt in welchem Bezug Planungsstrategien zu den Eigenschaften oder Präferenzen eines Benutzers stehen. Planungsstrategien wurden bisher lediglich in Bezug auf ihre Effizienz – also wie schnell ein Plan gefunden werden kann – untersucht. Offen bleibt insbesondere die Frage, ob und vor allem *wie* eine Anpassung an den Benutzer automatisch erfolgen könnte, die auch Dengler bspw. in [Dengler, 1994] fordert.

Zweifel bleiben auch, ob der Ansatz des deduktiven Planens überhaupt für die Aufgabe in einem Hilfesystem geeignet ist (vgl. auch [André, 1995, S.123]). Prinzipiell ergeben sich dabei die gleichen Probleme wie beim Planen im Situationskalkül[68]. Zwar kann das Vorgehen des Beweisers durch Taktiken in Hinblick auf ein zielgerichtetes Vorgehen beeinflußt werden, dennoch sind deduktive Planer bislang auf sehr eingeschränkte Einsatzbereiche und/oder auf kurze Pläne beschränkt [Biundo, 1993].

2.5.4 Benutzerspezifische Hilfe

Bereits in der Einleitung zu dieser Arbeit wurde die Wichtigkeit *benutzerspezifischer* Hilfestellung herausgestellt. Eine Hilfestellung, die für *einen* Benutzer adäquat ist, kann für *einen anderen* in Abhängigkeit von dessen Kenntnissen oder Fähigkeiten als zu langatmig, zu kompliziert, unverständlich oder gar als falsch beurteilt werden. Einige Hilfesysteme versuchen deshalb die Eigenschaften des jeweiligen Benutzers zu ermitteln und berücksichtigen diese Annahmen bei der Erzeugung der Hilfestellung. Speziell für Hilfesysteme sind nicht nur die Kenntnisse und Annahmen des Benutzers über Fakten, sondern insbesondere sein Planungswissen wichtig; d.h. die Kenntnisse welche Pläne wie welche Ziele erreichen können[69].

Im UNIX-Consultant ist die Komponente KNOME [Chin, 1989] für den Aufbau eines Benutzermodells zuständig. Die Einträge in das Benutzermodell werden aus den Eingaben des Benutzers an der Konsole abgeleitet, wobei ein (erweiterter) Stereotype-Ansatz [Rich, 1989] zum Einsatz kommt. Das Modell des planungsspezifischen Benutzerwissens bezieht sich auf UNIX-Kommandos mit ihren Optionen. Mit der Kenntnis eines Kommandos ist auch die Kenntnis über dessen Vorbedingungen, Effekte und Hauptzweck (*planfor*) verbunden. Wie bei Quilici (s.o.) wird im UNIX-Consultant zwischen Effekten und dem Hauptzweck, für den das Kommando normalerweise eingesetzt wird, unterschieden. Die Annahmen über das Planungswissen des Benutzers werden sowohl für die Disambiguierung bei der Planerkennung als auch für die Unterdrückung bereits bekannter Fakten bei der Präsentation einer Hilfestellung genutzt. Letzteres wird im System ComfoHelp auf Sequenzen von Befehlen erweitert. Wird (aufgrund mehrmaliger Beobachtung) geschlossen, daß der Benutzer eine Sequenz von Textverarbeitungs-Befehlen kennt, so werden diese nicht mehr als Vorschlag für eine effizientere Planausführung erzeugt.

Quilicis Ansatz zur Erklärung von Fehlannahmen basiert wie beim UNIX-Consultant auf Annahmen über die Kenntnis des Benutzers über UNIX-Kommandos. Quilici modelliert dazu explizit die Annahmen des Benutzers über deren Vorbedingungen, Effekte und Hauptzweck. Dabei läßt sich

[67] [Dengler, 1996, S.613]: "The transparency of the refinements allows a human to have complete control over the application of refinements."

[68] vgl. Kap. 2.1 *Handlungsplanung*

[69] Das Wissen *wie ein Ziel erreicht werden kann* wird oft zu den *Fähigkeiten* des Benutzers gerechnet. Kap. 3.1 *Handlungspotential und Wissen von Benutzern* enthält eine ausführliche Diskussion dieses Begriffs.

auch eine Diskrepanz zwischen Hauptzweck und Effekt ausdrücken. Bspw. geht Quilicis System nach einem gescheiterten Kommando *Datei-Löschen* davon aus, daß der Benutzer zwar weiß, daß das Kommando *eigentlich* für das Löschen geeignet ist, daß es aber offenbar (im konkreten Fall) *nicht* den Effekt hat eine Datei zu löschen, sondern eine Meldung "*permission denied*" hervorruft.

Auch im PHI-System werden Annahmen über den Benutzer berücksichtigt; und zwar sowohl für die Planerkennung als auch für die Erzeugung einer Hilfestellung. Die Erkennung des Benutzerplans und der damit verbundenen Ziele wird von einem quantitativen Benutzermodell unterstützt [Bauer, 1996]. Dieses repräsentiert Informationen über Präferenzen bzgl. der Ausführung von Plänen und über das gemeinsame Auftreten von Aktionen und Plänen durch Basiswahrscheinlichkeiten der Dempster-Shafer Theorie [Dempster, 1967], [Shafer, 1976]. Wird das beobachtete Vorgehen des Benutzers als suboptimal eingestuft und deshalb ein alternativer Plan für ein Hilfeangebot generiert, können wiederum die Präferenzen des Benutzers berücksichtigt werden. Um zu verhindern, daß ein solcher Plan Aktionen enthält, die der Benutzer nicht kennt, können solche Aktionen von der Planerzeugung ausgeschlossen werden.

2.5.5 Zusammenfassung

Die vorgestellten Hilfesysteme unterstützen einen Benutzers bei der Bearbeitung einer Aufgabe mit einem Anwendungsprogramm. Das System muß dem Benutzer dazu eine Handlungsanweisung d.h. einen Plan geben können, mit dem dieser seine Ziele (effizient) erreicht. Fordert ein Benutzer Hilfe an, hat er oft bereits einen Plan begonnen und festgestellt, daß er nicht die erwarteten Ergebnisse erzielt. Eine umfassende Hilfe analysiert das Vorgehen des Benutzers und versucht eine Erklärung für das Scheitern des Plans und ggf. auch nicht explizit genannter Ziele des Benutzers zu finden. Dieses kann auch ohne Hilfeanforderung des Benutzers sinnvoll sein, um unaufgefordert Hilfe anbieten zu können wenn abzusehen ist, daß das Vorgehen des Benutzers wahrscheinlich nicht oder nur umständlich zum Ziel führt. Die Erkennung der Benutzerpläne ist deshalb ein zentraler Aspekt planbasierter Hilfesysteme. Im Gegensatz zur "klassischen" Planerkennung steht hier aber die Erkennung falscher oder zumindest suboptimaler Pläne im Vordergrund. Einige Hilfesysteme repräsentieren solche Pläne als Folge von Aktionen und versehen sie mit einer entsprechenden Markierung. Eine detailliertere Analyse ist möglich, wenn nicht nur komplette Pläne, sondern auch einzelne Eigenschaften von Plänen oder Aktionen falsch sein können bzw. sich vom Wissen des Systems unterscheiden dürfen. Außer der Dekomposition, die in vielen Planerkennungs-Systemen repräsentiert wird, sind dabei Effekte und Vorbedingungen der Pläne besonders wichtig. Wie Pollacks Beispiel zeigt, kann die Ursache für Pläne die ihr Ziel nicht erreichen auch in einer falschen Einschätzung der aktuellen, die Voraussetzungen des Plans betreffenden Gegebenheiten liegen. Eine Trennung von Annahmen des Systems und des Benutzers – wenngleich meist nicht über Pläne – findet sich auch in (natürlichsprachlichen) Dialogsystemen, wobei die Annahmen des Benutzers oft in einem expliziten Benutzermodell repräsentiert sind[70]. Einige der hier vorgestellten Hilfesysteme lassen sich auch in diesen Forschungsbereich eingruppieren.

Die Repräsentation von Vorbedingungen und Effekten von Aktionen und Plänen hat nur in sehr wenigen Fällen dazu geführt, daß Pläne für die Ziele des Benutzers *generiert* werden. Tatsächlich realisiert wurde dies lediglich im System PHI, während die anderen Systeme Pläne für den Benutzer aus einer statischen Bibliothek korrekter Pläne entnehmen.

Überraschenderweise nutzt keines der Systeme die Abhängigkeiten zwischen den Schritten eines Plans, die als *Causal Links* Teil der Definition von Plänen in der Handlungsplanung sind. Die Verletzung solcher Abhängigkeiten führt zum Scheitern eines Plans und könnte deshalb zur Fehlererklärung

[70] vgl. [Kobsa und Wahlster, 1989] für eine Übersicht.

genutzt werden. Teilweise (bspw. im SINIX-Consultant und in ComfoHelp) werden die Abhängigkeiten allerdings implizit genutzt, indem für die Pläne sogenannte *Ignore-Aktionen* definiert werden können. Solche Aktionen dürfen bei der Planerkennung auch zwischen den Schritten eines Plans auftreten ohne dessen Erkennung zu beeinflussen. Dabei wird davon ausgegangen, daß der Plan auch dann seine Ziele erreicht, wenn solche Aktionen eingeschoben werden. Die Aktionen dürfen also keine der Abhängigkeiten zerstören. Ein Beispiel ist das *Sichern eines Dokuments*, das unschädlich in jeden Plan im Anwendungsbereich *Textverarbeitung* eingeschoben werden darf.

2.6 Planen zur Informationsbeschaffung

Ein wichtiges Einsatzgebiet von Computern, das insbesondere durch das Wachstum des Internets und des World-Wide-Web noch zunehmend an Bedeutung gewinnt, ist die Informationsbeschaffung. Über das Internet kann heute auf eine sehr große Menge an Information einfach zugegriffen werden. Da diese Informationen weitgehend unkoordiniert abgelegt sind, ist es dennoch schwierig auf *bestimmte* Informationen zuzugreifen. Dies gilt besonders, wenn an die Information Qualitätskriterien wie bspw. Aktualität oder Zuverlässigkeit gestellt werden. Oft steht die gewünschte Information nicht explizit zur Verfügung, sondern läßt sich nur durch eine Kombination von verschiedenen Informationsquellen ermitteln. Dabei kann Information, die durch Abfrage einer Informationsquelle gewonnen wird, für eine nachfolgende Abfrage genutzt werden. Werden solche Abfragen als Aktionen aufgefaßt, wird die Analogie zur Ausführung eines Plans offensichtlich. Auch dabei dienen die einzelnen Planschritte dazu, einen Beitrag für nachfolgende Schritte und/oder das Ziel zu leisten. Das Informationsbeschaffungsproblem läßt sich also auf ein Planungsproblem übertragen. Dabei nimmt die bereitgestellte Information die Rolle der Startsituation ein. Möglichkeiten von Informationsabfragen oder zur Informationsverarbeitung werden zu Planoperatoren und die Spezifikation der gesuchten Information stellt die Zielspezifikation für den Planungsprozeß dar.

Dennoch unterscheiden sich beide Problemklassen in einigen Aspekten, die dann Auswirkungen auf das Planungsverfahren und auch auf die Modellierung des Planungssachbereichs haben. Beispielsweise ist das Problem der Informationsbeschaffung monoton; d.h. die Beschaffung neuer Information zerstört keine bereits vorliegende[71]. Zur Bedrohung von *Causal Links*[72], deren Auflösung eine wichtige Aufgabe von klassischen Planungssystemen ist, kann es also bei der Informationsbeschaffung nicht kommen. Andererseits muß berücksichtigt werden, daß Informationsquellen unvollständig sein können und manche – auch gleichartige – Informationen auf mehrere Informationsquellen verteilt sind. Außerdem gibt es meist sehr viele Möglichkeiten die gleiche Information zu beschaffen, die sich aber in Eigenschaften wie bspw. dem Ressourcenverbrauch unterscheiden. Damit gewinnt die Beurteilung der *Qualität* eines erzeugten Plans an Bedeutung, die ansonsten in der Handlungsplanung weitgehend unbeachtet blieb (vgl. bspw. [Knoblock, 1996], [Lin, 1998]).

Die im folgenden vorgestellten Ansätze behandeln speziell das Problem der Plangenerierung für die Informationsbeschaffung.

Kwok und Weld stellen in [Kwok und Weld, 1996] einen Ansatz vor, der auf speziellen Operatoren für die Informationsbeschaffung beruht. Ein solcher Operator besteht aus:

- *Kopf*: Operatorname und einer Liste von Variablen $X_1, \ldots X_n$, den *Argumenten*, die jeweils einem Typ zugeordnet sind.
- *Rumpf*: einer Konjunktion von atomaren Formeln (der Prädikatenlogik erster Stufe) mit den Prädikaten $r_1 \ldots r_m$, die – unabhängig vom jeweiligen Operator – die Relationen im Sachbereich

[71] Zumindest solange die Ressourcen für die Speicherung der Information nicht berücksichtig werden. Andernfalls wäre denkbar, daß alte Information gelöscht wird, um Speicherplatz für neue zu schaffen.

[72] vgl. Kap. 2.1.1 *Least Commitment Planning*

modellieren. Die Prädikate können, müssen aber nicht, die gleichen Variablen wie der Kopf enthalten.

Die Argumente werden in *freie* und *gebundene* Variablen unterteilt. Gebundene Variablen werden durch Voranstellen des Zeichens "$" markiert. Solche Argumente müssen an einen Wert gebunden werden, bevor ein Operator als Informationsabfrage ausgeführt werden kann. Das Resultat einer entsprechenden Abfrage sind Tupel von Werten für die Argumente des Operators, für die der Rumpf zu einer wahren Aussage wird (bzw. werden kann), wenn die Werte dort eingesetzt werden. Variablen des Rumpfes die keine Argumente sind werden *ungebunden* genannt und sind implizit existenzquantifiziert. Das heißt, daß es für jedes Ergebnistupel einer Anfrage eine Belegung für die ungebundenen Variable geben muß, sodaß der Rumpf zu einer wahren Aussage wird.

Die Schreibweise für Operatoren ist:

$$\langle \textit{Op-Name} \rangle\ (X_1, \ldots X_n) \Rightarrow r_1(\ldots) \wedge \ldots \wedge r_m(\ldots)$$

Durch "$\Rightarrow$" wird ausgedrückt, daß ein Operator nicht notwendigerweise alle Tupel liefert, für die der Rumpf wahr ist bzw. bei geeigneter Belegung der ungebundenen Variablen wahr wäre. Liefert ein Operator alle entsprechenden Tupel, wird "$\Rightarrow$" durch "$\Leftrightarrow$" ersetzt.

Solche Operatoren haben im klassischen Sinne weder Vorbedingungen noch Effekte. Allerdings können die gebundenen Argumente – vergleichbar zu [Moore, 1985] und [Etzioni *et al.*, 1992]) – als Wissens-Vorbedingungen und die freien als Beobachtungs-Effekte wie in [Etzioni *et al.*, 1992] aufgefaßt werden. Die Effekte bewirken keine Veränderung im modellierten Sachbereich – insbesondere wird dadurch kein Wahrheitswert eines Prädikats verändert. Dementsprechend entfällt hier auch die strikte Trennung von Vorbedingung und Effekt in getrennte Zeitbereiche (vor bzw. nach Ausführung) wie bei den klassischen Operatoren. Veränderungen durch Informationsbeschaffungs-Operatoren betreffen lediglich die *Verfügbarkeit von Information*, die aber nicht explizit modelliert wird.

Kwok und Weld geben als Beispiele in einer UNIX-basierten Büroumgebung die Operatoren

- *finger* (F, L, $E, O, Ph) $\Rightarrow$ email (F, L, E) $\wedge$ office (F, L, O) $\wedge$ phone (O, Ph)
- *userid-room* ($O, E) $\Rightarrow$ office (F, L, O) $\wedge$ email (F, L, E)

an. Dabei sind die Variablen F und L im Rumpf des zweiten Operators ungebunden. Den Variablen F, L, E, O, Ph sind die Typen *Vorname, Nachname, E-Mail-Adresse, Büro-Nr., Telefon-Nr.* zugeordnet. Diese Typzuordnung gilt in den folgenden Beispielen auch für indizierte Variablen, die mit den genannten Anfangsbuchstaben beginnen.

Die Aufgabe der Plangenerierung besteht darin, aus einer Zielbeschreibung, d.h. der Spezifikation einer Informationsabfrage, einen Plan zu erzeugen, mit dem die gewünschte Information ermittelt werden kann. Eine *Zielbeschreibung*, bzw. einfach eine *Frage*, hat die gleiche Struktur wie ein Operator, besteht also in gleicher Weise aus Kopf und Rumpf. Im Gegensatz zu einem Operator, der das tatsächliche – evtl. unvollständige – Verhalten einer Informationsquelle modelliert, stellt der Rumpf einer Frage eine Bedingung dar, die durch den Plan realisiert werden soll. D.h. *jedes* Tupel von Werten für die Argumente des Frage-Kopfs, durch die der Rumpf (ggf. bei geeigneter Wahl der ungebundenen Variablen) zu einer wahren Aussage wird, ist eine Antwort auf die Frage. Dementsprechend ist die Schreibweise für Fragen:

$$\langle \textit{Frage-Name} \rangle\ (X_1, \ldots X_n) \Leftarrow r_1(\ldots) \wedge \ldots \wedge r_m(\ldots)$$

Auch für eine Frage sind gebundene Argumente in gleicher Notation vorgesehen. Durch sie lassen sich einzelne Argumente auf einen vorgegebenen Wert einschränken. Ein Beispiel ist die Suche nach den Vornamen der Mitarbeiter eines bestimmten Büroraums:

query-for-first-names ($O, F) $\Leftarrow$ office (F, L, O)

Durch den Frage-Rumpf wird festgelegt, daß alle *Büro-Nr.–Vornamen* Kombinationen eine Antwort auf die Frage sind, die in der auf *Büro-Nr.* und *Vornamen* projizierten Relation *office* enthalten sind. Da die Büro-Nr. ein gebundenes Argument ist, sind dies die Vornamen der Mitarbeiter in dem angegebenen Büro.

Ein Plan ist wie ein Operator aufgebaut. Die Schritte des Plans sind wie in der Handlungsplanung Exemplare von Operatoren. Der Rumpf eines Plans ist die Konjunktion dieser Operatorexemplare:

$$p\ (X_1, \ldots X_n) \Rightarrow O_1 \wedge \ldots \wedge O_k$$

Die Ausführung der Schritte (hier also der Einzelabfragen) setzt die Verfügbarkeit der Information für die gebundenen Argumente voraus. Diese kann entweder durch vorangegangene Planschritte ermittelt oder durch die Fragestellung bereitgestellt worden sein. Die Ausführungsreihenfolge der Planschritte ist also keineswegs beliebig, wie durch die Schreibweise als logische Konjunktion suggeriert wird. Vielmehr muß der Plan genau in der Reihenfolge der Operator-Indizes erfolgen. Daran ist die Bedingung geknüpft, daß es zu jedem gebundenen Argument eines Planschritts O_b entweder einen Planschritt O_a mit $a < b$ geben muß, der das jeweilige Argument ungebunden enthält und der damit bei der Ausführung einen Wert dafür erzeugt, oder das Argument muß im Kopf der Frage bzw. des Plans gebunden sein. Damit ein Plan $p\ (X_1, \ldots X_n) \Rightarrow O_1 \wedge \ldots \wedge O_k$ eine *Lösung* einer Frage $q\ (X_1, \ldots X_n) \Leftarrow r_1(\ldots) \wedge \ldots \wedge r_m(\ldots)$ ist, muß außerdem sichergestellt sein, daß jedes vom Plan gelieferte Tupel auch ein zulässiges Tupel für die Fragebedingung ist, wobei vorausgesetzt wird, daß Frage- und Plankopf gleich sind. Es muß also für jede Belegung der Argumente $X_1, \ldots X_n$ gelten:

$$p\ (X_1, \ldots X_n) \Rightarrow O_1 \wedge \ldots \wedge O_k \Rightarrow r_1(\ldots) \wedge \ldots \wedge r_m(\ldots) \Rightarrow q\ (X_1, \ldots X_n)$$

Da die Ergebnisse der Operatoren durch deren Rümpfe bestimmt wird, wird eine Überprüfung durch Einsetzen der Operatorrümpfe möglich. Für die Frage nach den Vornamen der Mitarbeiter eines Büroraums ist bspw. folgender Plan eine Lösung:

p ($O, F) $\Rightarrow$ *userid-room* ($O, E) $\wedge$ *finger* (F, L, $E, O, Ph)

Die notwendigen Informationen $O für *userid-room* und $E für *finger* werden von der Frage bzw. von *userid-room* zur Verfügung gestellt. Für eine beliebige Belegung c_1, c_2 für die Variablen O und F ergibt sich durch Einsetzen der Operatorrümpfe:

$$\begin{aligned} p\ (c_1, c_2) \Rightarrow\ & \textit{userid-room}\ (c_1, E) \wedge \textit{finger}\ (c_2, L, \$E, c_1, Ph) \Rightarrow \\ & \text{office}\ (F_0, L_0, c_1) \wedge \text{email}\ (F_0, L_0, E) \wedge \\ & \text{email}\ (c_2, L, E) \wedge \mathbf{office\ (c_2, L, c_1)} \wedge \text{phone}\ (c_1, Ph) \end{aligned}$$

Dies impliziert wiederum office (c_2, L, c_1) – die einzige Bedingung der Beispielfrage. Jedes Tupel (c_1, c_2) das der Plan liefert erfüllt also die Fragebedingung und ist damit ein gültiges Antworttupel.

Die Planerzeugung beginnt mit einem leeren Plan und verlängert diesen schrittweise um Exemplare von Operatoren die ausgeführt werden können; d.h. für die die benötigten Informationen durch die Frage oder durch vorangegangene Operatoren bereitgestellt werden. Kwok und Welds Algorithmus bearbeitet dabei alle alternativen Möglichkeiten, sofern die Berechnung eine zuvor gesetzte Ressourcenbegrenzung nicht überschreitet. Nach der Erzeugung eines Plans muß festgestellt werden, ob der Plan eine Lösung für die Frage darstellt. Dazu wird geprüft, ob es eine Abbildung τ der Frage-Argumente auf die Variablen des Plans und ggf. Bindungsbedingungen der Plan-Variablen untereinander gibt, sodaß der Plan-Rumpf den Frage-Rumpf impliziert, wenn jede Variable v_q des Frage-Rumpfes durch $\tau(v_q)$ ersetzt wird. Sofern der Algorithmus nicht durch die Ressourcenbegrenzung gehindert wird, erzeugt er alle Lösungen zu einer Frage.

Eine vergleichbare Modellierung von Operatoren zur Informationsbeschaffung findet sich in [Bauer und Dengler, 1999]. Bauer und Dengler wollen einen Benutzer bei der Informationsbeschaffung im Internet, speziell im World-Wide-Web, unterstützen. Sie modellieren dazu das Informationsangebot einer Informationsquelle durch Inhalts- und Frageschemata sowie einer Beschreibung des konkreten Zugriffs auf die Information. Für die Generierung eines Plans zur Informationsbeschaffung sind die ersten beiden relevant. Das Inhaltsschema definiert welche Informationen die Quelle bereitstellt, wobei auch hier jeder Information ein Typ zugeordnet ist. Im Gegensatz zum o.g. Ansatz werden diese Typen durch Concepts im Ontobroker-Ansatz [Decker *et al.*, 1999] definiert. Vergleichbar zu beschreibungslogischen Concepts[73] werden Concepts durch die Angabe von Attributen und der Einordnung in eine Abstraktionshierarchie definiert. Zusätzlich wird hier auch die Definition von Teil-Ganzes-Beziehungen unterstützt. Abb. 2.24 zeigt die Beschreibung des Beispiel-Concepts *Flight* im Sachbereich Flugbuchungen.

```
Flight[ carrier =>> Airline;
        departure =>> CityCode;
        destination =>> CityCode;
        travel_date =>> Date;
        departure_time =>> Time;
        arrival_time =>> Time;
        number =>> Flight_Number;
        ...].
```

Abbildung 2.24: Beschreibung eines Fluges aus [Bauer und Dengler, 1999]

Die Inhaltsschemata zur Beschreibung von Informationsquellen beziehen sich auf diese Concepts, indem sie angeben, für welche Attribute sie Werte bereitstellen oder benötigen. Das Inhaltsschema aus Abb. 2.25 beschreibt eine Flugbuchungsseite im WWW. Die Seite stellt demnach Werte für Attribute des Concepts *Flight* zur Verfügung. Die Typangaben sind teilweise spezieller als die des Concepts. Bspw. ist für *arrival_time* und *departure_time time_12_3* anstelle von *Time* angegeben. Damit wird ausgedrückt, daß die Information in einer speziellen Form (hier die Zeitangabe im 12 Std. Format) bereitgestellt wird.

```
itnflight( Flight.carrier: Airline,
           Flight.departure: CityCode,
           Flight.destination: CityCode,
           Flight.travel_date: date_us_2,
           Flight.arrival_time: time_12_3,
           Flight.departure_time: time_12_3 )
```

Abbildung 2.25: Inhaltsschema einer Flugbuchungsseite Fluges aus [Bauer und Dengler, 1999]

Zu einem Inhaltsschema können ein oder mehrere Frageschemata definiert werden, die angeben, welche der Informationen als Eingabe erwartet werden, welche als Ergebnis geliefert werden und welche von der Frage nicht betroffen sind. Bspw. besagt das Frageschema

$< +, +, +, u, -, - >$

[73] vgl. Kap. 3.2.1 *Beschreibungslogiken*

zum o.g. Inhaltsschema, daß die Informationsquelle Ankunft- und Abflugzeit liefert, wenn Fluggesellschaft, Abflugort und Ziel angegeben werden. Ein Frageschema bildet zusammen mit dem Inhaltsschema einen Operator, wobei – wie bei Kwok und Weld – die Eingabeargumente als (Wissens-) Vorbedingung und die Ausgaben als Effekt in einem Planungsprozeß verwendet werden.

Die Planung *effizienter* Abfragen von Informationsquellen ist das Ziel von Knoblocks Ansatz [Knoblock, 1996]. Die Effizienz, im Sinne möglichst geringer Wartezeit zwischen Abfrage und Bereitstellung des Ergebnisses, stellt ein Qualitätskriterium für einen erzeugten Plan dar. Im Gegensatz zur klassischen Abfrageoptimierung in Datenbanken (vgl. bspw. [Jarke und Koch, 1984]) geht Knoblock davon aus, daß Informationen verteilt sind, wobei auch die gleiche Information auf verschiedenen Servern bereitstehen kann. Er unterscheidet deshalb zwischen dem *Datenbestand* und einem *Server*, der Datenbestände zur Verfügung stellt. Wenngleich der Ansatz keine explizite Repräsentation von Ressourcen bzw. Ressourcenverbrauch vorsieht, wird die (mögliche) Verteiltheit der Informationen zur Effizienzsteigerung genutzt. Grundlage ist die Annahme, daß ein Server nicht mehrere Operationen gleichzeitig durchführen kann. Dadurch kann eine Abfrage beschleunigt werden, wenn Teiloperationen auf mehreren Servern parallel ausgeführt werden. Für die Planung der Teiloperationen muß also von der Annahme der klassischen Handlungsplanung abgegangen werden, daß zu einem Zeitpunkt nur genau ein Planschritt ausgeführt wird[74]. Partial-Order-Planer[75] stellen keine Bedingungen an die Reihenfolge der Ausführung von Planschritten, wenn dies nicht für die Korrektheit eine Plans notwendig ist. Planschritte, die zueinander in keiner zeitlichen Ordnungsrelation stehen, können auch parallel ausgeführt werden, sofern sich ihre Ausführungen nicht gegenseitig stören. Solche Störungen können auftreten, wenn man von der unrealistischen Annahme abgeht, daß die Ausführung eines Planschritts atomar ist; d.h. keine Zeit verbraucht und auch keine – evtl. nicht modellierte – Zwischenzustände bewirkt. Für das Anwendungsgebiet Informationsbeschaffung kann im wesentlichen die Belegung eines Servers als Störung auftreten, die eine parallele Ausführung von Planschritten verhindert. Zur Generierung guter Pläne (im o.g. Sinne) gehört also außer der Auswahl der geeigneten Datenbestände auch eine geeignete Wahl der Server, um den Aufwand für die Ausführung des Plans auf möglichst viele Server zu verteilen und dadurch die Gesamtbearbeitungszeit so gering wie möglich zu halten.

Knoblocks Lösung besteht darin, das klassische Partial-Order Planungssystem UCPOP[76] [Penberthy und Weld, 1992] so zu erweitern, daß auch solche Abhängigkeiten berücksichtigt werden, die aus der Möglichkeit Planschritte parallel auszuführen resultieren. Im Gegensatz zu den beiden anderen Ansätzen modellieren die Planoperatoren aber nicht die Verfügbarkeit von Informationen. Vielmehr sind allgemeine Operatoren zur Datenmanipulation definiert – bspw. *join* und *move* und zur Fragereformulierung bspw. *choose-source* und *substitute-definition*. Letztere dienen lediglich der Anpassung von Anfragen an die jeweiligen Datenquellen. Welche Informationen die Operatoren verarbeiten können wird i.w. durch die Vorbedingungen der Operatoren bestimmt. Diese greifen ihrerseits auf ein statisches Modell des Anwendungsbereichs zurück, das im Wissensrepräsentationssystem LOOM [MacGregor, 1991] definiert ist. Bspw. besagt die Definition des in Abb. 2.26 gezeigten Operators *join*, daß zwei Datenbestände die beide lokal verfügbar sind[77] unter Berücksichtigung einer join-Bedingung zu einem lokalen Datenbestand zusammengeführt werden können. Welche Datenbestände wie kombiniert werden, wird durch die Vorbedingung *join-partition* bestimmt. Dabei wird ausgenutzt, daß UCPOP prozedurale Vorbedingungen zuläßt; d.h. Lisp-Funktionen, die eine beliebige Verarbeitung ermöglichen. Durch solche Vorbedingungen werden auch Bewertungen alternativer (Teil-) Entscheidungen vorgenommen, um den Planungsprozeß zu steuern.

[74] vgl. Kap. 2.1.3 *Voraussetzungen beim Planen*: die Linearitätsannahme
[75] vgl. Kap. 2.1.1 *Least Commitment Planning*
[76] Das Planungssystem wird auch in dieser Arbeit eingesetzt. Näheres dazu s. Kap. 5.1.3 *Plangenerierung mit UCPOP*.
[77] Dies wird durch die beiden Argumente *local* und *sims* repräsentiert. SIMS steht für den lokalen Informations-Server.

```
(define (operator join)
  :parameters (?join-op ?data ?data-a ?data-b)
  :precondition
     (:and (join-partition ?data ?join-op ?data-a ?data-b)
           (available local sims ?data-a)
           (available local sims ?data-b))
  :effect (available local sims ?data))
```

Abbildung 2.26: Darstellung des Operators *join* aus [Knoblock, 1996] in UCPOP-Syntax

Knoblock zeigt für seinen Anwendungsbereich *Transportwesen* die Aufteilung einer Beispielabfrage nach Seehäfen, die (unter Berücksichtigung des Tiefgangs) für ein bestimmtes Schiff geeignet sind. Die Abfrage wird in zwei Teile gespalten, weil die Information über Schiffe im Datenbestand *assets* und die über Häfen im Datenbestand *seaports* enthalten sind. Da die Reihenfolge der beiden Teilabfragen irrelevant ist und die Datenbestände auf verschieden Servern verfügbar sind, können die Teilabfragen parallel ausgeführt werden. Abb. 2.27 zeigt den Gesamtplan für die Beispielabfrage. Die Planschritte sind durch Pfeile verbunden, die ihre Abhängigkeiten repräsentieren. Parallele Zweige können auch parallel ausgeführt werden, weil sie verschiedene Server betreffen.

Abgesehen von der Möglichkeit der parallelen Ausführung von Planschritten, entspricht dieser Plan einem partiell geordneten Plan der Handlungsplanung. Die parallele Ausführung von Planschritten impliziert die Annahme, daß mehrere Akteure mit der Ausführung des Plans beschäftigt sind – im Beispiel die insgesamt vier Server. Von der Eigenschaft, daß die Beschaffung von Information normalerweise keine Information zerstört, profitiert der Ansatz nur indirekt, indem sich der Aufwand für den Planungsprozeß reduziert. Es müssen bspw. keine Bedrohungen von Causal Links aufgelöst werden. Der eingesetzte Planungsalgorithmus ist aber unabhängig davon. Im Gegensatz dazu setzen Kwok und Weld in ihrem eingangs beschriebenen Ansatz einen spezialisierten und damit deutlich einfacheren Algorithmus ein, der lediglich total geordnete Pläne erzeugt. Die Planoperatoren beschränken sich zudem auf die *Abfrage* der Information und berücksichtigen nicht, woher die Information beschafft werden soll und wie sie kombiniert werden muß.

2.7 Abschließende Bemerkungen und Zusammenfassung

In diesem Kapitel wurden verschiedene Ansätze für die Modellierung des Handelns und von Plänen vorgestellt. Es wurden nur *logikbasierte* Modelle berücksichtigt, die ein *zielgerichtetes Handeln eines Akteurs* beschreiben können. Damit wurden Modelle oder Systeme ausgeschlossen, für deren Sachbereich *kontinuierliche Veränderungen* von Bedeutung sind. Ein Beispiel wäre eine Heizungsregelung, deren Komplexität darin liegt, daß die einzig mögliche Aktion (Heizleistung festlegen) eine *kontinuierliche* raum- und zeitabhängige Veränderung der (Raum-)Temperatur bewirkt. Ebenfalls unberücksichtigt blieben Modelle, die die mit Handeln und Veränderung verbundenen Phänomene beschreiben, ohne sie für ein zielorientiertes Handeln einzusetzen. Einen Überblick dazu gibt bspw. [Sandewall, 1994].

Die Basis für alle behandelten Ansätze ist die Annahme der Handlungsplanung, daß es für den gewählten Weltausschnitt adäquat ist, anstelle der kontinuierlichen Veränderungen der Realität nur für festgelegte Zeitpunkte Momentaufnahmen der aktuellen Sachverhalte zu modellieren – und diese dann jeweils durch eine (prädikaten-) logische Formel. Die Veränderungen selbst werden dabei ausschließlich durch Aktionen hervorgerufen und stellen den Effekt einer Aktion dar. Aktionen können i.a. nur ausgeführt werden, wenn unmittelbar vor der Ausführung eine (endliche) Menge von Bedingungen erfüllt ist. Da sich nach dem Ausführen einer Aktion bis zum Ausführen der nächsten

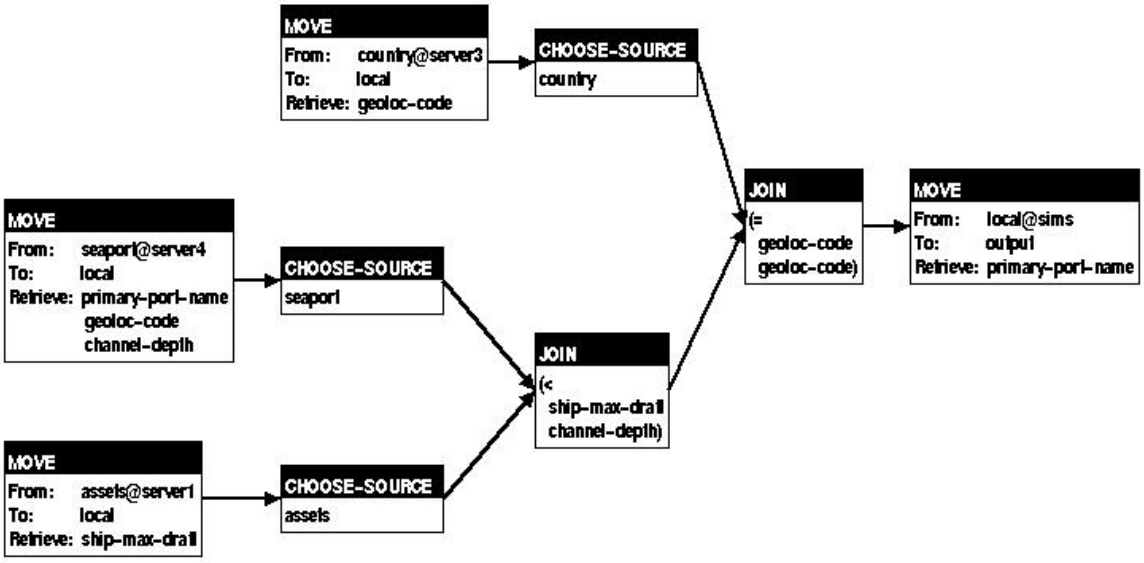

Abbildung 2.27: Informationsbeschaffungs-Plan für die Frage nach *See-Häfen, die für den Tiefgang eines gegebenen Schiffes ausreichen* aus [Knoblock, 1996].

nichts ändert, sind die "interessanten" Zeitpunkte für die Momentaufnahmen unmittelbar vor bzw. nach einer Aktion. Ein Plan, d.h. eine Lösung eines Planungsproblems, besteht dann in einer Folge von Aktionen, die eine (gegebene) Ausgangssituation schrittweise verändern, sodaß letztlich die Ziele erreicht werden. Da die Aktionen (die meist *Planschritte* genannt werden) für das Erreichen der Ziele oft in mehreren alternativen Reihenfolgen ausgeführt werden können, wird unter *Folge* eine partiell geordnete Menge subsumiert.

Wie jedes Modell stellt auch ein solcher Ansatz eine Vereinfachung der Realität dar. Deshalb kann auch für einen Plan, der mithilfe eines beweisbar korrekten Planungsverfahrens erzeugt wurde, nicht *garantiert* werden, daß seine Ausführung in der realen Welt auch tatsächlich die erwarteten Ziele erreicht. Der Ansatz läßt sich aber dennoch für die Erzeugung eines Plans einsetzen, wenn die Gegebenheiten der Realität ausreichend präzise modelliert sind, sodaß seine Ausführung *in den meisten Fällen* die Ziele erreicht oder die Ziele wenigstens durch Planmodifikation während der Ausführung erreicht werden können. Ist dies nicht der Fall, können ausdrucksstärkere Modelle oder Formalismen zum Einsatz kommen, die die Realität besser modellieren. Allerdings ist damit eine weitere Erhöhung der ohnehin schon hohen Berechnungskomplexität der Plangenerierung verbunden – eine Entscheidung über den Einsatz muß also sorgfältig abgewogen werden.

Auch das Abstraktionsniveau der Modellierung, und damit der Umfang an Details, die für die Plangenerierung berücksichtigt werden müssen, hat Einfluß auf die Berechnungskomplexität. Offensichtlich sinkt die Berechnungskomplexität mit steigender Abstraktion. Allerdings sinkt damit auch die Wahrscheinlichkeit, daß die Ausführung eines generierten Plans die erwarteten Ziele erreicht. Hierarchische Planungs-Ansätze nutzen dies, indem sie zunächst mit wenig Aufwand einen Plan auf einem hohen Abstraktionsniveau erzeugen und ihn dann schrittweise um Details ergänzen, sodaß letztlich ein zuverlässiger Plan entsteht. Die Bestimmung von Abstraktionsniveaus, die für die Plangenerierung möglichst gut geeignet sind, ist allerdings wiederum mit hohem Aufwand verbunden. Der Ansatz der Operatorabstraktion (Kap. 2.1.2.2 *Operatorabstraktion*) geht nicht von expliziten Abstraktionsniveaus aus, sondern von abstrakten Operatoren, die in Sub-Pläne mit weniger abstrakten Operatoren dekomponiert werden können. Das dabei notwendige Wissen, wie Teilaufgaben im Anwendungssachbereich normalerweise ausgeführt werden können, ist besonders dann nützlich, wenn

es im Sachbereich für viele Problemstellungen vorgefertigte Lösungen gibt oder wenn häufig eine Zerlegung in Teilaufgaben möglich ist, für die Lösungen bekannt sind, die sich nur selten gegenseitig stören. Interagieren die Teillösungen dagegen stark, ist eine Komposition zu einer Gesamtlösung aufwendig oder sogar unmöglich.

Das Wissen über mögliche Zerlegungen abstrakter Operatoren kann auch zur Strukturierung eines Plans oder einer Menge von Plänen genutzt werden. Beispiele für Letzteres finden sich insbesondere in der Planerkennung und bei Hilfesystemen. Eine Strukturierung eines einzelnen Plans ist für die Generierung eines natürlichsprachlichen Textes, sowie generell für die Kommunikation mit einem Benutzer nützlich.

Arbeiten mehrere Akteure zusammen, haben sie oft unterschiedliche Auffassungen über den oder die verfolgten Pläne. Dies wurde besonders bei Arbeiten zum gemeinsamen Planen und zu Hilfesystemen deutlich. Die unterschiedlichen Auffassungen können neben dem Plan selbst, alle Aspekte eines Plans betreffen – insbesondere auch seine Schritte, sowie deren Vorbedingungen, Effekte und mögliche Dekompositionen. Beim gemeinsamen Planen kam hinzu, daß unterschiedliche Auffassungen darüber bestehen können, ob ein Akteur Teile eines Plans ausführen kann oder will.

Abschließend sei angemerkt, daß Qualitätskriterien für Pläne kaum untersucht sind (vgl. [Lin, 1998], [Knoblock, 1996]). Von den in diesem Kapitel behandelten Arbeiten wird lediglich beim Planen zur Informationsbeschaffung erwähnt, daß die Qualität von Plänen für ihre praktische Einsetzbarkeit wichtig ist.

Zusammenfassend ergibt sich damit folgendes Bild für die Repräsentation und Verarbeitung von Wissen über Handeln und Pläne.

Für Pläne, die der Unterstützung eines Computerbenutzers dienen, lassen sich Veränderungen – auch wenn sie in der Realität kontinuierlich erfolgen – zumeist ausreichend genau durch Folgen von Momentaufnahmen beschreiben. Auch die Zeitdauer von Aktionen, sowie deren Unsicherheit und Ressourcenverbrauch spielen nicht so wichtige Rollen, daß ihre Berücksichtigung bei der Plangenerierung gerechtfertigt wäre. Im Verlauf der Arbeit wird noch auf Beispiele dieser Aspekte eingegangen. Die Modellierung kann sich also an den Vorstellungen der klassischen Handlungsplanung orientieren. Das heißt insbesondere, daß ein Plan aus einer Folge (im o.g. Sinne) von Planschritten besteht, die Exemplare von Planoperatoren sind. Die Planoperatoren modellieren mit ihren Vorbedingungen und Effekten die *Veränderungsmöglichkeiten* im Sachbereich. Ein Planoperator (bzw. ein Exemplar davon) kann nur dann ausgeführt werden, wenn seine Vorbedingung unmittelbar zuvor erfüllt ist. Der Effekt beschreibt die *Veränderungen*, die die Ausführung des Operators gegenüber der Situation unmittelbar davor bewirkt.

Bereits in der Einleitung wurde argumentiert, daß eine Unterstützung eines Benutzers auf der Basis *vorgefertigter* Pläne aufgrund der Vielzahl von Kombinationsmöglichkeiten unterschiedlicher Programme bzw. Funktionalitäten nicht realistisch ist. Da auch eine Interaktion denkbarer Teillösungen für einzelne Problemstellungen nicht ausgeschlossen werden kann, wird auf die Verwendung von Abstraktion bei der Plan*generierung* verzichtet. Es wird sich aber noch zeigen, daß eine eingeschränkte Variante von Operatorabstraktion für die *Präsentation* eines Plans sinnvoll eingesetzt werden kann.

Das konkrete Verfahren zur Plangenerierung ist für die Zielsetzung dieser Arbeit allerdings nebensächlich, solange es korrekt und (möglichst) vollständig ist. Wichtig ist vielmehr, daß bei der Plangenerierung berücksichtigt wird, daß der Benutzer den Plan auch ausführen kann. Das dafür notwendige Wissen über die Fähigkeiten – bzw. präziser über die *prinzipiellen Handlungsmöglichkeiten* – des Benutzers wird sich normalerweise vom Planungswissen des Benutzers und auch vom Planungswissen des Systems unterscheiden. Alle drei Komponenten müssen allerdings vergleichbar sein, um dem Benutzer – falls notwendig – Teile eines für ihn erzeugten Plans näher erklären oder ihn über Fehlannahmen aufklären zu können, die mglw. eine korrekte Ausführung des Plans behindern würden. Das nachfolgende Kapitel widmet sich der Frage einer geeigneten Repräsentation dieses

Wissens, die sich als Basis sowohl für die Kommunikation mit dem Benutzer eignet als auch für die Erzeugung einer Aufgabenspezifikation, anhand der ein Planungssystem einen benutzerspezifischen Plan erzeugen kann. Im vierten Kapitel wird das Zusammenspiel der Wissenskomponenten für die Erzeugung einer benutzerspezifischen Präsentation eines solchen Plans behandelt.

Kapitel 3

Modellierungs- und Plangenerierungsaspekte

Ziel dieser Arbeit sind Techniken zur Unterstützung eines Computerbenutzers, sodaß dieser seine Ziele erreichen kann. Für das Erreichen der Ziele sind Handlungen des Benutzers erforderlich, deren Abfolge von einem Hilfe- bzw. Unterstützungssystem geplant werden. Ein solcher Plan ist dann Basis für einen Rat bzw. eine Handlungsanweisung an den Benutzer. Im vorangegangenen Kapitel wurden verschiedene Ansätze zur Modellierung von geplantem Handeln vorgestellt und der Schluß gezogen, daß sich die Generierung eines solchen Plans an den Vorstellungen der klassischen Handlungsplanung orientieren kann. In diesem Kapitel wird darauf aufbauend ein Modell entwickelt, daß die Eigenschaften des *speziellen Benutzers* berücksichtigt, um diesem eine angemessene Unterstützung geben zu können.

Besonders die Arbeiten zum gemeinsamen Planen und zu Hilfesystemen berücksichtigen unterschiedliche *Ansichten* über die Gegebenheiten und Veränderungsmöglichkeiten im Anwendungssachbereich zwischen den Akteuren bzw. zwischen System und Benutzer. Im ersten Teil dieses Kapitels wird allerdings gezeigt, daß dies für einen benutzerspezifischen Rat nicht ausreichend ist. Das erfolgreiche Handeln eines Akteurs hängt nicht nur von seinen *Kenntnissen* ab, die durch eine Hilfestellung verändert werden können. Vielmehr sind auch seine *Fähigkeiten* bzw. generell seine *Möglichkeiten zu handeln* ein entscheidender Einflußfaktor. Die Begriffe werden noch präzisiert und es wird ein Benutzermodell als Basis für die Adaption an die Eigenschaften des Benutzers vorgestellt, das unterschiedliche Modalitäten handlungsspezifischen Wissens in getrennten Bereichen enthält. Im Anschluß wird auf die konkrete Repräsentation der einzelnen Teile des Benutzermodells in Beschreibungslogik allgemein und speziell im Formalismus von SB-ONE [Kobsa, 1991] eingegangen. Die Teile des Benutzermodells sind allerdings keineswegs unabhängig voneinander. Insbesondere stehen sie auch in Relation zu den Annahmen des Systems über die statischen und dynamischen Eigenschaften des Anwendungsbereichs. Dieser Zusammenhang ist ein weiteres Thema dieses Kapitels.

Der Abschnitt *Planungsproblem und Planungssachbereich* widmet sich der Generierung eines Plans, der als Basis für einen Rat an den Benutzer dienen soll. Wesentlich ist dabei die Einbeziehung des Benutzermodells in die Konstruktion einer Aufgabenspezifikation für das Planungssystem, damit dieses die Eigenschaften des Benutzers bei der Plangenerierung berücksichtigt.

Das Kapitel endet mit einer weiteren Differenzierung des Ansatzes. Manche Aufgaben lassen sich nur von mehreren Akteuren oder nur unter Zuhilfenahme von Werkzeugen lösen. Im Beispielsachbereich Büroumgebung können dies bspw. Anwendungsprogramme oder Bürogeräte sein. Da solche Werkzeuge die Handlungsmöglichkeiten eines Akteurs beeinflussen, bietet es sich an, sowohl die Ver-

fügbarkeit der Werkzeuge als auch die Beeinflussung der Handlungsmöglichkeiten – die je nach Art und Ausprägung des Werkzeugs unterschiedlich sein können – mit in das Modell einzubeziehen.

3.1 Handlungspotential und Wissen von Benutzern

Eingangs wurde gesagt, daß das Ziel der Arbeit darin liegt, Techniken zur Unterstützung eines Computerbenutzers zu entwickeln, sodaß dieser seine Ziele erreichen kann. Es wurde etwas unpräzise davon gesprochen, daß dazu die Eigenschaften des jeweiligen Benutzers, insbesondere Kenntnisse und Fähigkeiten, berücksichtigt werden müssen. Diese Arbeit folgt der Methodik einer expliziten Benutzermodellierung, die für die Anpassung eines technischen Systems an einen Benutzer ein explizites Modell der dafür relevanten Benutzereigenschaften vorsieht, das von anderen Informationen des Systems unterscheidbar ist (vgl. [Wahlster und Kobsa, 1989]). Bevor in den nächsten Abschnitten auf die konkrete Repräsentation des Benutzermodells eingegangen wird, sei hier zunächst präzisiert, was unter *Kenntnisse und Fähigkeiten eines Benutzers* verstanden wird.

Unter Kenntnissen eines Benutzers bzw. generell eines Akteurs werden in dieser Arbeit grundsätzlich *Annahmen* verstanden. Folgt man der grundlegenden Arbeit von Hintikka [Hintikka, 1962], muß zwischen *Wissen* (*knowledge*) und *Annahmen* (*belief*) eines Akteurs unterschieden werden. Während zu den Annahmen eines Akteurs beliebige Aussagen gehören können, kann das Wissen eines Akteurs nach Hintikka nur wahre Aussagen enthalten. Ein Akteur kann also bspw. die Annahme haben, daß sich ein DOS-PC durch das Betätigen der <*Ende*>-Taste herunterfahren – also beenden – läßt. Da die Annahme aber falsch ist, kann der Akteur dies nicht *wissen*. Für den Einsatz in der Benutzermodellierung ist eine explizite Unterscheidung allerdings nicht sinnvoll, da es normalerweise – und insbesondere auch in dieser Arbeit – kein Modell für die Bestimmung der absoluten Wahrheit einer Aussage gibt[1]. Stattdessen sei hier von einem Modell ausgegangen, das die Wahrheit einer Aussage grundsätzlich nur in Bezug zu den Annahmen eines Akteurs bestimmt. Dabei hält ein Akteur eine Aussage genau dann für wahr, wenn diese Aussage aus seinen Annahmen ableitbar ist – ansonsten hält er sie für falsch. Sofern nicht explizit ein beurteilender Akteur erwähnt ist, sei das (Hilfe- bzw. Unterstützungs-) System als beurteilende Instanz für die Annahmen des Benutzers vereinbart. Die Schreibweise *Annahme p des Benutzers ist falsch* ist damit als Abkürzung für: *das System hält Annahme p des Benutzers für falsch* oder *Annahme p des Benutzers läßt sich nicht aus den Annahmen des Systems ableiten* zu verstehen.

Annahmen eines Akteurs sind nicht auf direkte Aussagen beschränkt. Die Beurteilung der Annahmen des Benutzers durch das System sind bspw. nur möglich, wenn das System diese Annahmen kennt bzw. genauer, wenn es Annahmen über die Annahmen des Benutzers hat. Eine systematische Einteilung der Annahme-Typen, die in Benutzermodellen verschiedener Arbeiten vorkommen, findet sich in [Kobsa, 1989]. Für diese Arbeit kommen die Annahmen über (die noch zu präzisierenden) *Fähigkeiten* des Akteurs selbst und anderer Akteure hinzu.

Der Begriff *Fähigkeit* wird bspw. in Meyers Enzyklopädie [Mey, 1971] als "*Gesamtheit der zum Erbringen einer Leistung notwendigen Bedingungen*" definiert. Diese Definition ist allerdings sehr weit gefaßt. Sie umfaßt neben Aspekten, die in der Natur des Akteurs der die Leistung erbringt liegen, auch (veränderliche) Kenntnisse und sogar Aspekte die unabhängig vom Akteur sind. Beispiele für diese drei Aspekte sind:

- die physische Fähigkeit *Treppen steigen zu können* – bspw. für das Erreichen eines Raumes auf einem höheren Stockwerk
- die Kenntnis, *wie eine Türe geöffnet werden kann* – bspw. für das Eintreten in einen Raum

[1] vgl. auch [Finin, 1989, S.415].

- die akteurunabhängige Bedingung, *ob eine Türe verschlossen ist* – bspw. für das Eintreten in einen Raum

Für diese verschiedenen Typen von Bedingungen sind unterschiedliche Verfahren zur Verarbeitung sinnvoll. Darauf wird weiter unten eingegangen.

Ein Blick zu den Arbeiten die im letzten Kapitel vorgestellt wurden zeigt, daß der Begriff *Fähigkeit* dort nur sehr selten vorkommt. Horacek bspw. nutzt in seinen Arbeiten zur Dialogplanung die *Inferenz*fähigkeit eines Benutzers, um einen erzeugten Text auf die für den Benutzer wesentlichen Inhalte zu reduzieren (vgl. [Horacek, 1997] u. Kap. 2.4.2.2 *Weitergehende Ansätze*). Die Fähigkeit bezieht sich darauf Schlußfolgerungen zu ziehen. Im Zusammenhang mit Plänen werden solche Fähigkeiten in erster Linie für die *Erzeugung* des Plans benötigt. Liegt dagegen bereits ein Handlungsplan vor, kommt es eher auf die Fähigkeit *Handlungen auszuführen* an[2].

Solche Fähigkeiten bezieht Pollack in ihre Arbeiten [Pollack, 1986] und [Pollack, 1990] ein (vgl. auch 2.3.2 *Pläne als mentale Zustände*). Bspw. stellt sie an die Annahme eines Beobachters, daß ein Akteur *einen Plan hat*, u.a. die Bedingung, daß dieser Akteur glauben muß, daß er *jeden Planschritt des Plans ausführen kann*. Diese Annahme des Beobachters über die Annahme des Akteurs bedeutet nicht, daß der Akteur den Planschritt auch tatsächlich ausführen kann oder daß der Beobachter dies glauben muß. Wenngleich Pollack dieses *Ausführen-Können* nicht für weitergehende Schlußfolgerungen nutzt, bietet er einen Ansatz für eine Definition von Fähigkeiten, die für die Spezifikation eines Benutzermodells geeignet ist.

Im Bereich klassischer Handlungsplanung spielt die explizite Berücksichtigung von Fähigkeiten keine Rolle. Daß der Akteur der einen Plan ausführen soll die einzelnen Planschritte auch ausführen kann, ist allerdings eine (implizite) Voraussetzung für eine erfolgreiche Planausführung. Der Grund für die Vernachlässigung der expliziten Berücksichtigung von Fähigkeiten liegt darin, daß in erster Linie für Roboter geplant wird. Die Operatoren, die für die Erzeugung eines Plans verwendet werden, spiegeln dabei die Funktionalität des Roboters – also des Akteurs, der den Plan ausführen soll – wider. Sie können direkt aus der Schnittstellenspezifikation, die die Kommandos bzw. die Funktionen des Roboters beschreibt, abgeleitet werden. Wie bei Pollack können die Fähigkeiten eines Akteurs damit auf der Basis von Planschritten, oder allgemeiner von Operatoren, aus denen Planschritte abgeleitet werden können, definiert werden.

Allerdings wird hier nicht versucht den Begriff *Fähigkeit* auf dieser Basis präzise zu definieren, weil dadurch eine Diskrepanz zur umgangssprachlichen Verwendung des Begriffs entstehen würde. Stattdessen seien die Begriffe *Handlungsmöglichkeiten* und *Handlungspotential* eingeführt, um die Eigenschaften eines Benutzer zu bezeichnen, die für die Erzeugung eines benutzerspezifischen Plans relevant sind.

Definition 3.1 ***Handlungsmöglichkeiten***
Die *Handlungsmöglichkeiten eines Benutzers* umfassen die Menge der Aktionen, die dieser Benutzer *im Prinzip* ausführen *kann* und für die er die *Befugnis* hat.

Definition 3.2 ***Handlungspotential***
Das *Handlungspotential eines Benutzers* ist ein Modell seiner Handlungsmöglichkeiten.

Soll berücksichtigt werden, daß auch andere Akteure handeln und dadurch zum Erreichen eines Ziels beitragen, werden die Definitionen sinngemäß auf die *Handlungsmöglichkeiten* bzw. das *Handlungspotential eines* beliebigen *Akteurs* erweitert. Die Überlegungen sind nicht auf den Benutzer eines Computersystems eingeschränkt.

[2] Im Zusammenhang mit der Präsentation eines Plans wird in Kap. 4.4 *Berücksichtigung von Benutzer-Inferenzen* noch auf Schlußfolgerungsfähigkeiten eingegangen, die mit Horaceks Ideen verwandt sind.

Mögliche Aktionen und Mengen gleichartiger Aktionen werden wie in der Handlungsplanung durch Planoperatoren modelliert. Diese Modellierung sei für das Handlungspotential übernommen; d.h. mit dem Begriff *Handlungspotential* ist im folgenden immer auch die Repräsentation durch eine Menge von Planoperatoren verbunden. Für die konkrete Repräsentation der einzelnen Planoperatoren sei auf den nächsten Abschnitt 3.2 verwiesen.

Das Handlungspotential eines Benutzers kann direkt zur Generierung eines Plans herangezogen werden. Verwendet das Planungssystem dazu *ausschließlich* die Operatoren des Handlungspotentials, ist per Definition sichergestellt, daß der Benutzer jeden einzelnen Planschritt (der ja aus den Operatoren abgeleitet ist) zumindest prinzipiell ausführen kann und darf. Dies gilt auch für den gesamten Plan, sofern die Plangenerierung korrekt und die Modellierung der möglichen Aktionen ausreichend genau ist[3].

Der Zusatz *im Prinzip* in den Definitionen soll ausdrücken, daß möglicherweise noch Bedingungen erfüllt werden müssen, bevor ein Benutzer eine Aktion oder einen Planschritt tatsächlich ausführen kann. Eine Klasse von Bedingungen wird im Handlungspotential bereits durch die Vorbedingungen der Planoperatoren berücksichtigt. Die Erfüllung dieser Bedingungen ist ein wesentlicher Aspekt bei der Lösung eines Planungsproblems. Der erzeugte Plan garantiert, daß solche Bedingungen jeweils unmittelbar vor Ausführung eines Planschritts erfüllt sind.

Eine weitere Klasse von Bedingungen betrifft die Kenntnisse eines Benutzers. Mit der umgangssprachlichen Verwendung des Begriffs *Fähigkeit* oder mit *Ausführen können* sind meist auch Kenntnisse verbunden. Bspw. setzt die Befähigung zum Studium ein Abitur voraus, also umfassende schulische Kenntnisse. Entsprechendes gilt etwa für die Befähigung, einen Büro-Drucker reparieren zu können. Die Modellierung von Kenntnissen des Benutzers sind aber nicht in einem Handlungspotential enthalten. Im Gegenteil:

> Das Handlungspotential und das Modell der Kenntnisse eines Benutzers werden strikt getrennt.

Die Gründe dafür sind zum einen, daß die Möglichkeiten eines Benutzer zu handeln und seine Kenntnisse auch bezüglich ein und derselben Aktion verschieden sein können. Eine gezielte Hilfestellung muß die beiden Aspekte also differenzieren können. Außerdem wirken sich die beiden Aspekte unterschiedlich auf die Erzeugung eines benutzerspezifischen Ratschlages bzw. einer Unterstützung aus (s.u.).

Ein Beispiel für Unterschiede zwischen Handlungsmöglichkeiten und Kenntnissen bezogen auf eine Aktion ist der Neustart eines DOS-PCs. Um die Aktion durchzuführen, müssen die Tasten <*Strg*>, <*Alt*> und <*Entf*> gleichzeitig betätigt werden[4]. Es gibt sicherlich Benutzer, die das nicht wissen. Andere haben diese Kenntnis vielleicht, können die Aktion aber dennoch nicht ausführen, weil sie – bspw. wegen einer Körperbehinderung – nicht alle drei Tasten gleichzeitig erreichen können. Ein weiteres Beispiel ist das Kopieren einer Datei von einem fremden Computer mit FTP. Auch hier reicht die Kenntnis über das Ausführen der Aktion nicht aus, sondern es ist zusätzlich eine Zugangsberechtigung erforderlich. Dieses Beispiel zeigt, daß die Möglichkeiten eines Benutzers zu Handeln nicht nur von seinen eigenen (physischen) Eigenschaften abhängen, sondern auch von seinen Befugnissen. Für die Arbeit mit einem Computer sind Befugnisse sogar die dominierende Komponente für das Handlungspotential. Tabelle 3.1 zeigt weitere Beispiele. Für die Erzeugung eines Plans ist es allerdings irrelevant, *warum* ein Benutzer eine Aktion durchführen kann bzw. nicht kann. Entsprechend ist dieser Unterschied durch das Handlungspotential auch nicht modelliert. Dies ist konträr zur umgangsprachlichen Bedeutung von *Fähigkeit*, die normalerweise Befugnisse nicht subsumiert, wenngleich die o.g. Definition aus Meyers Lexikon ja sämtliche Bedingungen umfaßt. Im weiteren

[3] vgl. auch die Anmerkungen zur Korrektheit von Plänen in Kap. 2.1.3 *Voraussetzungen beim Planen*.

[4] Das ist eine der Möglichkeiten, die auf eine deutschsprachige Tastatur bezogen ist.

Operator	Erforderliche Eigenschaften des Akteurs		
	Kenntnisse	physisch	Befugnisse
Treppe steigen		Uneingeschränkte Gehfähigkeit (kein Rollstuhlfahrer)	
Klavier heben		physische Kraft	
Neustart DOS-PCs	DOS Kenntnisse	Beweglichkeit beider Arme	
Kopieren mit FTP	Bedienung eines FTP-Programms, Netzwerkkenntnisse		Zugriffsberechtigung, Berechtigung eine Firewall zu passieren
e-mail lesen	Bedienung eines Mail-Programms		Zugriffsberechtigung
Drucken mit einem besonderen (kostenintensiven) Drucker	Name des Druckers, Grundlegende Kenntnisse des Betriebssystems		Zugriffsberechtigung

Tabelle 3.1: Erforderliche Eigenschaften für die Ausführung von Aktionen

wird der Unterschied nicht mehr thematisiert. Handlungsmöglichkeiten und Handlungspotential eines Benutzer werden ausschließlich in diesem Sinne verwendet; d.h. sie umfassen bzw. modellieren die Aktionen, zu deren Ausführung der Benutzer physisch in der Lage ist und für die er auch die dazu evtl. notwendigen Befugnisse hat. Dies gilt auch dann, wenn aus Gründen einer einfacheren Sprechweise lediglich vom *Können* des Benutzers die Rede ist – bzw. davon, daß ein Benutzer eine Aktion oder Handlung *ausführen kann*.

Gehen wir davon aus, daß ein Ratschlag, der einen Benutzer beim Erreichen seiner Ziele unterstützen soll, eine Folge von Aktionen beschreibt die dieser Benutzer ausführen soll, und fassen wir diese Folge von Aktionen als Plan auf, so ergibt sich folgendes Bild von den Bedingungen, die bei der Erzeugung des Ratschlags berücksichtigt werden müssen.

1. Der Plan muß im Sinne der Handlungsplanung korrekt sein; d.h. unmittelbar vor jedem Planschritt muß die dazu gehörende Vorbedingung erfüllt sein.
2. Der Benutzer muß zumindest im Prinzip, d.h. wenn er über ausreichende Kenntnisse verfügt, in der Lage sein jeden Planschritt auszuführen.
3. Für jeden Planschritt muß der Benutzer über ausreichende Kenntnisse verfügen, um den Planschritt ausführen zu können.

Bedingung 1 kann durch ein klassisches Planungssystem sichergestellt werden. Dabei muß gewährleistet sein, daß die Annahmen, die der klassischen Handlungsplanung zugrunde liegen[5], in ausreichender Annäherung erfüllt sind. Die Annahme, daß der handelnde Akteur in der Lage sein muß jeden Planschritt auszuführen, ist – entsprechend der Zielsetzung dieser Arbeit – explizit als zweite Bedingung herausgestellt. Da die verschiedenen Benutzer unterschiedliche Möglichkeiten zu handeln haben, kann nicht ohne weiteres angenommen werden, daß ein Benutzer jeden beliebigen erzeugten

[5] vgl. Kap. 2.1.3 *Voraussetzungen beim Planen*.

Plan ausführen können wird. Vielmehr muß der Planungsprozeß so gesteuert werden, *daß* die Bedingung erfüllt wird. Wie bereits gesagt, kann dies sichergestellt werden, wenn ein Planungssystem ausschließlich Planoperatoren aus dem Handlungspotential des jeweiligen Benutzers für die Generierung verwendet. Die dritte Bedingung kann immer erfüllt werden, indem das Hilfesystem den Benutzer mit den erforderlichen Kenntnissen versorgt. Dies gilt zumindest, wenn – wie in diesem Kapitel vorausgesetzt – das Hilfesystem selbst über ausreichende Kenntnisse verfügt und diese dem Benutzer vermitteln kann.

Sind Handlungspotential und Kenntnisstand des Benutzers bekannt, ergibt sich daraus auch bereits ein erstes Vorgehensschema für die Erzeugung eines benutzerspezifischen Ratschlags zu einer gegebenen Problemstellung:

- Erzeugen eines Plans, der das gegebene Problem löst, wobei ausschließlich Operatoren des Handlungspotentials des Benutzer eingesetzt werden.
- Erzeugung einer Beschreibung dieses Plans, die dem Benutzer ggf. auch fehlende Kenntnisse vermittelt.

Da die beiden Aufgaben eine verschiedenartige Verarbeitung erfordern, werden sie in dieser Arbeit von getrennten Komponenten wahrgenommen: der Planungs- bzw. Plangenerierungskomponente und der Präsentationskomponente. Beide Komponenten arbeiten allerdings nicht völlig unabhängig. Der Grund liegt darin, daß für einen Plan, der ohne Berücksichtigung der Benutzerkenntnisse erzeugt wurde, zunächst nicht bekannt ist wieviel Erklärungen erforderlich sind, um evtl. fehlende Benutzerkenntnisse auszugleichen. Im ungünstigsten Fall führt die Menge der erforderlichen Erklärungen dazu, daß der Benutzer überfordert wird und das Ziel der Hilfestellung damit verfehlt wird. Als einfachste Lösung kann die Präsentationskomponente einen Plan, der eine zu aufwendige Beschreibung erfordern würde, zurückweisen und von der Planungskomponente einen alternativen Plan anfordern. Darauf wird in Kap. 4.3.2 *Gesamtbewertung einer Planpräsentation* noch eingegangen. Weitergehende Lösungen finden sich in Kap. 4.5 *Optimierte Plangenerierung*.

Aus dem Schema werden die unterschiedlichen Rollen deutlich, die die *Möglichkeiten eines Benutzers zu handeln* und seine *Kenntnisse* haben. Während erstere durch die Operatoren des Handlungspotentials den Rahmen der *möglichen* Pläne absteckt, bilden die Benutzerkenntnisse eine Basis für evtl. erforderliche Erklärungen. Dabei werden die Möglichkeiten eines Benutzers zu handeln als statisch angesehen (vgl. Abschnitt 3.1.2 *Veränderungsmöglichkeiten eines Handlungspotentials*). Die Kenntnisse eines Benutzers sind dagegen veränderlich, denn die Veränderung der Benutzerkenntnisse – die Ergänzung fehlender Kenntnisse und die Korrektur falscher Annahmen – ist ja gerade die zentrale Aufgabe eines Unterstützungs- oder Hilfesystems.

Anzumerken ist noch, daß es für das Erreichen der Benutzerziele natürlich nicht ausreicht, daß der Benutzer einen Plan kennt, den er auch ausführen kann. Er muß den Plan auch *ausführen*, was wiederum die *konkrete Absicht* des Benutzers erfordert (vgl. bspw. [Pollack, 1986] und Kap. 2.3.2 *Pläne als mentale Zustände*). Hier sei zunächst davon ausgegangen, daß der Benutzer – da er ja seine Ziele erreichen will – die Absicht hat einen Plan, den er für das Erreichen seiner Ziele genannt bekommt, auch auszuführen. Dennoch ist es möglich, daß ein Benutzer Präferenzen bzgl. bestimmter Handlungen bzw. Operatoren hat, sodaß er einen vorgeschlagenen Plan nicht akzeptiert und damit nicht ausführt. Die Berücksichtigung solcher Präferenzen bei der Plangenerierung wird hier allerdings zunächst zurückgestellt. Es sei dafür auf Kap. 4.5 *Optimierte Plangenerierung* verwiesen.

3.1.1 Alternativen zum Verarbeitungsschema

Weiter oben wurden Bedingungen angegeben, die bei der Erzeugung des Ratschlags berücksichtigt werden müssen. Für Bedingung 2 (der Benutzer muß in der Lage sein jeden Planschritt auszuführen)

und Bedingung 3 (der Benutzer muß über ausreichende Kenntnisse verfügen, um jeden Planschritt ausführen zu können) drängt sich die Frage auf, ob die Bedingungen nicht als Vorbedingungen der Planoperatoren modelliert werden können. Das Planungssystem würde dann von einer feststehenden Menge von Planoperatoren ausgehen, die *alle* Möglichkeiten im Anwendungsbereich zu handeln modellieren würde. Durch eine benutzerspezifische Komponente unter den Vorbedingungen der Operatoren könnte dann erreicht werden, daß das Planungssystem für die Plangenerierung ausschließlich solche Operatoren verwendet, die die geforderten Bedingungen erfüllen. Für den resultierenden Plan wären dann – Korrektheit des Planungsverfahrens vorausgesetzt – alle drei Bedingungen erfüllt. Dieser Abschnitt behandelt Möglichkeiten einer solchen Modellierung mit ihren Konsequenzen – getrennt nach den Handlungsmöglichkeiten des Benutzers und seinen Kenntnissen.

Für die Modellierung der Planoperatoren sei die übliche Vorstellung der Handlungsplanung zugrunde gelegt, daß die Vorbedingung eines Operators aus einer Konjunktion von Literalen besteht[6]. Damit ein Plan ausschließlich Planschritte enthält die der Benutzer ausführen kann, müssen die Vorbedingungen der zugrundeliegenden Operatoren jeweils eine benutzerspezifische Komponente enthalten, die genau dann den Wahrheitswert "wahr" ergibt, wenn der Benutzer den Operator ausführen kann. Als Beispiel gehen wir davon aus, daß Benutzer mit der Bezeichnung $B_1, \ldots, B_n$ fest vorgegeben sind und in Benutzergruppen $G_1, \ldots, G_m$ eingruppiert werden können. Soll modelliert werden, daß die Benutzer B_1 und B_2 sowie alle Benutzer der Gruppe G_3 einen Operator *Op* ausführen können, kann dies durch die Formel

$$\textit{curr-user}(B_1) \vee \textit{curr-user}(B_2) \vee (\textit{curr-user}(x) \wedge \textit{group-member}(x, G_3))$$

als ein Faktor der Vorbedingung des Operators *Op* realisiert werden. Dabei sind B_1, B_2 und G_3 Konstanten für die Benutzer bzw. die Gruppe. x ist eine implizit existenzquantifizierte Variable. $\textit{curr-user}(x)$ sei dabei genau dann wahr, falls x den aktuellen Benutzer bezeichnet. $\textit{group-member}(x, g)$ ist wahr, wenn x zur Gruppe g gehört. Eine solche Modellierung erfordert, daß die Definition der Vorbedingung von Operatoren auch Disjunktionen zuläßt, wie dies bspw. im Planungssystem UCPOP[7] [Penberthy und Weld, 1992], das in dieser Arbeit verwendet wird, der Fall ist.

Nachteilig ist die Vermischung der Aspekte der Benutzermodellierung und des Planungssachbereichs, zu dem auch die Definition der Operatoren gehört[8]. Bei der Modellierung des Planungssachbereichs müssen bereits alle zukünftigen Benutzer mit ihren Eigenschaften berücksichtigt werden. Dies führt nicht nur zu einem schlecht wartbaren Gesamtmodell, sondern verhindert auch die Berücksichtigung von möglichen Veränderungen der Ansichten über Benutzereigenschaften. Besser wäre eine klare Trennung beider Aspekte wie sie zu Beginn dieses Abschnitts und bspw. auch in [Wahlster und Kobsa, 1989] gefordert wird.

Eine bessere Trennung von Planungssachbereich und Benutzermodell ergibt sich, wenn für jeden Planoperator eine Konstante eingeführt wird und ein Prädikat – bspw. *user-can* – genau für die Konstanten wahr ist, die zu einem Planoperator gehören, den der Benutzer ausführen kann. Die Vorbedingung eines Planoperators erhält dann den Faktor

$$\textit{user-can}(\textit{Ref_Op}),$$

wenn *Ref_Op* die zu dem Operator gehörende Konstante ist. Die Definitionen der Operatoren erfordern damit keine Referenzen zu einzelnen Benutzern oder Benutzergruppen und sind damit unabhängig davon, welche Benutzer oder Gruppen für das System vorgesehen sind. Auch Änderungen der

[6] vgl. Kap. 2.1 *Handlungsplanung*.
[7] näheres zu diesem Planungssystem s. Kap. 5.1.3 *Plangenerierung mit UCPOP*
[8] vgl. auch Kap. 3.4 *Planungsproblem und Planungssachbereich*

Ansichten über die Handlungsmöglichkeiten des Benutzers sind unabhängig von den Operatordefinitionen möglich. Stattdessen kann eine Benutzermodellierungskomponente vor jedem Planungsprozeß bestimmen, welche Operatoren dem Benutzer zugetraut werden. Dazu fügt sie der Beschreibung der Startsituation für jeden solchen Operator den Faktor *user-can* mit der zum Operator gehörenden Konstante als Argument hinzu.

Beiden Realisierungsmöglichkeiten ist gemeinsam, daß die *möglichen* Eigenschaften aller denkbaren Benutzer im Voraus modelliert werden müssen, sofern sie die Handlungsmöglichkeiten beeinflussen. Besonders nachteilig ist dies, wenn die Möglichkeiten verschiedener Benutzer eine Handlung auszuführen auf verschiedene Weise eingeschränkt sind und deshalb unterschiedliche Varianten eines Planoperators modelliert werden müssen. Die Einschränkungen können sowohl von physischen Eigenschaften als auch von Befugnissen herrühren. Ein Beispiel für ersteres ist das Heben eines Gegenstandes bis zu einem vom Benutzer abhängenden Gewicht. Ein Beispiel für Einschränkungen durch unterschiedliche Befugnisse ist das Ausdrucken einer Datei – etwa indem bestimmte Benutzergruppen nur bestimmte Drucker verwenden dürfen. In beiden Beispielen müssen mehrere Operatoren mit dem gleichen Effekt definiert werden. Dies führt beim Planungsprozeß zu einer Vergrößerung des Verzweigungsfaktors, weil das Planungssystem aus vielen, beinahe gleichen Operatoren auswählen muß, um einen bestimmten Effekt zu erreichen. Da die einzelnen Faktoren einer Vorbedingung keine unterschiedliche Priorität haben, besteht die Möglichkeit, daß eine Planalternative sehr weit entwickelt wird, bevor sich herausstellt, daß ein Operator gar nicht von dem Benutzer ausgeführt werden kann. Bei der Plangenerierung ist die Größenordnung der Anzahl der Entscheidungen und damit des Zeitbedarfs $O(v^s)$. Dabei ist v der mittlere Verzweigungsfaktor und s gibt die Anzahl der Planschritte des Plans an. Bereits eine Verdoppelung des Verzweigungsfaktors führt bei einem Plan mit 10 Schritten zu einer rund tausendfachen Laufzeit.

Die Vielfalt an Operatoren für das gleiche Ziel, die sich durch die verschiedenen möglichen Benutzer ergibt, besteht für ein konkretes Planungsproblem allerdings nur scheinbar. Solange die Handlungsmöglichkeiten eines Benutzers nicht durch geplante Aktionen verändert werden können[9], dürfen nur die Operatoren ausgewählt werden, die durch die benutzerspezifischen Vorbedingungskomponenten anzeigen, daß der Benutzer die zugehörige Aktion ausführen kann. Um nicht Bearbeitungszeit mit der Untersuchung von Alternativen zu verschwenden, die letztlich nicht zum Ziel führen, kann entweder der Planungsalgorithmus erweitert werden, oder dem Planungssystem werden – wie in dieser Arbeit realisiert – nur die Operatoren aus dem Handlungspotential des jeweiligen Benutzers zur Verfügung gestellt. Eine sinnvolle Erweiterung des Planungsalgorithmus besteht darin, daß er zwischen "normalen" Vorbedingungskomponenten und *Anwendbarkeitsvorbedingungen* unterscheiden kann. Anwendbarkeitsvorbedingungen sind dabei Bedingungen, deren Wahrheitswerte ausschließlich durch die Beschreibung der Startsituation vorgegeben werden und nicht durch Operatoren verändert werden können – hier also die benutzerspezifischen Vorbedingungskomponenten. Im Planungsprozeß muß dann nur festgestellt werden, ob eine solche Bedingung erfüllt ist oder nicht. Im negativen Fall wird ein solcher Operator *nicht* für die weitere Konstruktion eines Plan verwendet. Der Aufwand reduziert sich damit auf die Feststellung, *daß* der Operator nicht geeignet ist. Eine letztlich sinnlose Weiterverarbeitung eines Plans *mit* einem solchen Operator wird so vermieden.

Auf die Frage inwiefern eine Veränderung des Handlungspotentials eines Benutzers sinnvoll sein kann, wird im nächsten Abschnitt eingegangen.

Während die Berücksichtigung der Handlungsmöglichkeiten eines speziellen Benutzers direkt durch den Planungsprozeß zwar zu einem erhöhten Berechnungsaufwand führt, aber wenigstens denkbar ist, lassen sich die Erörterungen nicht auf die Kenntnisse eines Benutzers übertragen.

[9] s. nachfolgenden Abschnitt.

Zum einen können die Kenntnisse eines Benutzers nicht als unveränderlich angesehen werden. Wie bereits erwähnt, gehört die Veränderung der Benutzerkenntnisse ja gerade zu den zentralen Aufgaben eines Unterstützungs- oder Hilfesystems. Stützt sich ein Ratschlag nur auf Handlungen, die der Benutzer bereits kennt, so ist sein Wert zumindest fraglich, denn der Benutzer könnte auch selbst darauf kommen, oder er kennt die vorgeschlagene Handlungsweise bereits.

Dennoch bleibt die Frage, ob es nicht sinnvoll ist dem Planungssystem die Aufgabe zu übertragen, die evtl. notwendigen Erklärungen zur Ergänzung oder Vervollständigung des Benutzerwissens zu planen. In diesem Fall müsste die Vorbedingung jedes Operators um folgende Bedingungen erweitert werden:

- der Benutzer muß wissen, *wie* er eine zum Operator gehörende Aktion ausführen kann.
- der Benutzer muß alle Parameter des Operators kennen; d.h. insbesondere, daß er alle an einer Aktion beteiligten Objekte identifizieren kann.

Solche Bedingungen werden in der Literatur als *Wissens-Vorbedingungen* bzw. *knowledge preconditions* bezeichnet (bspw. [Moore, 1985], [Morgenstern, 1987]). In dieser Arbeit wird davon ausgegangen, daß das Hilfesystem die Aufgabe und auch die Kenntnisse hat, um die notwendigen Informationen liefern zu können. Die Bedingungen lassen sich dann also grundsätzlich mit einer Aktion *erkläre (Benutzer, <Sachverhalt>)* erfüllen, die das System auszuführen hätte. Da solche Aktionen keine Interaktionen mit anderen Aktionen haben – insbesondere keine bereits erzielten Ergebnisse zerstören können – können sie auch nachträglich zu einem Plan hinzugefügt werden, der ohne Berücksichtigung der Wissens-Vorbedingungen erzeugt wurde. Da die Rechenzeit für die Plangenerierung exponentiell von der Anzahl der Planschritte abhängt, führt der Verzicht auf die Berücksichtigung von Wissens-Vorbedingungen und damit von *erkläre-Aktionen* bei der Plangenerierung zu einer deutlichen Verringerung des Planungsaufwandes. Unabhängig davon *kann* eine Erklärung für den Benutzer durchaus auch durch ein Planungssystem erzeugt werden. Wichtig ist, daß die Planungsprozesse getrennt sind und die Länge der einzelnen Pläne nicht zu groß wird.

Wissens-Vorbedingungen werden in der Literatur in verschiedener Weise verwendet, die hier noch kurz zum Vergleich angesprochen werden sollen. In Kap. 2.6 *Planen zur Informationsbeschaffung* wurden bereits zwei Ansätze behandelt. Eine erste Erwähnung von Wissens-Vorbedingungen findet sich bereits in [McCarthy und Hayes, 1969]. Die Autoren weisen darauf hin, daß ein Akteur, der eine Aktion durchführen soll bzw. will, wissen muß *wie* er sie durchführen kann. Moore ([Moore, 1980], [Moore, 1985]) schlägt eine Formalisierung von Wissens-Vorbedingungen auf Basis des *Mögliche-Welten Ansatzes* [Kripke, 1963b] vor. Sein Schwerpunkt sind Test-Aktionen, durch deren Effekte sich Rückschlüsse auf unbeobachtbare Eigenschaften ziehen lassen. Ein Beispiel ist der Test, ob eine Flüssigkeit sauer oder basisch ist, mithilfe von Lackmus-Papier. Ein Akteur, der (1) die Test-Aktion (und alle ihre Parameter) kennt, der (2) weiß, daß er die Test-Aktion mit diesem Lackmus-Papier und einer Flüssigkeit F durchgeführt hat und der (3) nach der Aktion eine rote Färbung eines Lackmus-Papiers erkennt, kann schließen, daß Flüssigkeit F sauer ist; bzw. exakter: zum Zeitpunkt der Test-Aktion sauer *war*. Moore fordert, daß es sich bei der Kenntnis der Aktion um eine *ausführbare* Beschreibung – vergleichbar einem Computerprogramm – handelt. Morgenstern erweitert diesen Ansatz um die Interaktion mehrerer Akteure [Morgenstern, 1987]. Damit läßt sich bspw. beschreiben, daß ein Akteur bereits mit der Ausführung eines (partiellen) Plans beginnt und erst durch die Interaktion mit einem anderen Akteur ausreichende Kenntnisse erlangt, um seine Ziele zu erreichen. Im Beispiel fragt ein Akteur einen anderen nach einer Test-Aktion um festzustellen, ob eine Flüssigkeit sauer oder basisch ist.

Die Zielsetzung beider Ansätze liegt in erster Linie im Aufbau einer Theorie, die ausdrucksstark genug ist, Schlußfolgerungen auf das Wissen eines bzw. mehrerer Akteure zu ziehen, wobei Aktionen

die das Wissen beeinflussen können berücksichtigt werden. Sie geben keine Hinweise darauf, ob oder wie die Theorie in einem Planungssystem Anwendung finden kann. Gleiches gilt auch für die Frage, welche Anforderungen an eine Beschreibung der Aktionsparameter zu stellen sind, damit der Akteur die an der Aktion beteiligten Objekte eindeutig identifizieren kann. Lochbaum schlägt bspw. in [Lochbaum, 1995] eine "*Orakel-Funktion*" vor, die eine solche Beschreibung unter Berücksichtigung des jeweiligen Kontextes erzeugt. Für die Realisierung der Funktion verweist sie auf Arbeiten aus der Textgenerierung (bspw. [Appelt und Kronfeld, 1987], [Hobbs, 1985]).

Golden und Weld behandeln in [Golden und Weld, 1996] Wissens-Vorbedingungen aus der Sicht der Handlungsplanung zur Informationsbeschaffung (vgl. auch Kap. 2.6 *Planen zur Informationsbeschaffung*). Ihr Ziel ist, einen Kompromiß zwischen der großen Ausdrucksstärke von Moores und Morgensterns Theorie und der Realisierbarkeit durch einen Planungsalgorithmus mit handhabbarem Laufzeitverhalten zu finden. Ein erzeugter Plan soll von einem Informationsbeschaffungs-Roboter – einem sogenannten *Softbot* [Etzioni und Weld, 1994] – ausgeführt werden. Die Notwendigkeit der Berücksichtigung von Kenntnissen *wie* ein Planschritt ausgeführt werden soll wird deshalb von vorn herein ausgeklammert. Weiterhin argumentieren sie, daß Schlußfolgerungen aufgrund von Wissens-Vorbedingungen, wie sie Moore für Test-Aktionen vorsieht, für die Plangenerierung ungeeignet sind. Der Grund liegt darin, daß ein Planungssystem für jeden Planschritt sicherstellen muß, *daß* die Vorbedingung vor Ausführung des Planschritts erfüllt ist. Für das obige Beispiel müßte das Planungssystem also sicherstellen, *daß* die Flüssigkeit sauer ist, bevor ein Operator mit der Vorbedingung *Flüssigkeit ist sauer* und dem Effekt *Lackmuspapier ist rot* eingesetzt werden kann. Dadurch wird der Operator als Test, *ob eine Flüssigkeit sauer ist* natürlich unbrauchbar. Die Argumentation gilt allerdings nicht mehr, wenn der Planungsalgorithmus in der Lage ist *bedingte Effekte* zu behandeln. In diesem Fall könnte die Test-Aktion ohne Vorbedingungen und mit folgendem Effekt modelliert werden:

falls *Flüssigkeit sauer ist* **dann** *ist das Lackmuspapier rot* und
falls *Flüssigkeit basisch ist* **dann** *ist das Lackmuspapier blau*

Bei solchen bedingten Effekten ist zu beachten, daß eine Konklusion *nach* dem Ausführen der Aktion gilt, wenn die Prämisse *vor* dem Ausführen der Aktion wahr *war*. Bedingte Effekte werden bspw. von UCPOP und auch von Etzioni und Welds Planungssystem unterstützt. Sie nutzen die bedingten Effekte aber nicht für Schlußfolgerungen auf den Wahrheitsgehalt der Prämisse. Vielmehr führt die Verwendung von Operatoren mit bedingten Effekten zu sogenannten *Subgoals*, d.h. zu untergeordneten Zielen, die notwendigen Informationen zu beschaffen. Dafür wiederum können Operatoren mit speziellen *Observe*-Effekten eingesetzt werden.

3.1.2 Veränderungsmöglichkeiten eines Handlungspotentials

Im letzten Abschnitt wurde als Modellierungsalternative für die Handlungsmöglichkeiten eines Benutzers diskutiert, die Planoperatoren, die die Veränderungsmöglichkeiten im Anwendungssachbereich repräsentieren, um eine benutzerspezifische Vorbedingungskomponente zu ergänzen. Sofern solche Vorbedingungskomponenten nicht als spezielle Anwendungsvorbedingungen repräsentiert sind, unterscheiden sie sich technisch nicht von benutzer*un*spezifischen Vorbedingungskomponenten. Damit können die Möglichkeiten eines Benutzers zu handeln – wie andere Sachverhalte des Anwendungsbereichs – durch Aktionen des Benutzers verändert werden. Gehen wir von der o.g. Repräsentation durch eine Vorbedingungskomponente *user-can (Ref_Op)* eines Operators *Op* aus, so kann ein anderer Operator Op_2 eine solche Komponente als Effekt (bzw. Teil des Effekts) haben. Die Ausführung von Op_2 bewirkt damit, daß der Benutzer danach auch *Op* ausführen kann.

Für die klassische Handlungsplanung, die üblicherweise von einem Roboter als Akteur ausgeht, erscheint eine solche Option nicht sinnvoll, da die Möglichkeiten des Roboters zu handeln durch seine Schnittstellenspezifikation fest vorgegeben ist. Eine andere Situation ergibt sich allerdings, wenn die Funktionalität des Roboters durch Programmierung oder Lernverfahren veränderbar ist. Im Gegensatz zu einem Roboter ist die Veränderung der Handlungsmöglichkeiten eines Menschen durch Lernen oder Training grundsätzlich gegeben. Dies gilt – in Grenzen – sowohl für die physischen Eigenschaften als auch für Befugnisse. Für die Beispiele des letzten Abschnitts *Klavier heben* und *Drucken mit einem besonderen Drucker* könnte ein Akteur bspw. Krafttraining durchführen, bzw. ein Paßwort stehlen und seinen Benutzerstatus ändern, um seine Handlungsmöglichkeiten zu erweitern.

Dennoch ist es sicher nicht sinnvoll *alle denkbaren* Aktionen, die ein Benutzer unter bestimmten Voraussetzungen ausführen *könnte*, zu modellieren und bei der Erzeugung eines Plans zu berücksichtigen. Die negativen Auswirkungen von unnötig großer Entscheidungsfreiheit auf den Zeitbedarf der Plangenerierung wurden bereits im vorangegangenen Abschnitt behandelt. Aber auch in Hinblick auf eine zielgerichtete Unterstützung eines Benutzers ist eine Berücksichtigung aller denkbaren Aktionen eines Benutzers nicht sinnvoll. Bspw. trägt ein Rat "*Betreiben Sie drei Monate Krafttraining, danach können Sie das Klavier transportieren*" bestenfalls zur Erheiterung des Benutzers, aber weniger zu einer konkreten Problemlösung bei. Das Stehlen eines Paßworts würde gesellschaftlichen Konventionen widersprechen, die allerdings in dieser Arbeit nicht explizit repräsentiert werden. Andererseits sind Anwendungsbereiche denkbar, in denen ein Wechsel des Benutzerstatus legitim ist. Bspw. ist es für einen UNIX-Systemadministrator nicht sinnvoll immer den Status *Superuser* zu haben. Vielmehr wird er diesen Status aus Sicherheitsgründen nur bei Bedarf einnehmen und ansonsten im normalen Benutzerstatus arbeiten.

Im Rahmen eines Hilfesystems oder generell einer Benutzerunterstützung gibt es also sowohl Beispiele *für* eine sinnvolle Berücksichtigung von Veränderungen der Handlungsmöglichkeiten eines Benutzers bei der Plangenerierung als auch *dagegen*. Für die Entscheidung, welche Veränderungsmöglichkeiten sinnvoll sind, lassen sich keine absoluten Kriterien angeben. Vielmehr muß für den jeweiligen Anwendungs- und Einsatzbereich entschieden werden, ob und ggf. welche Handlungsmöglichkeiten der potentiellen Benutzer veränderbar modelliert werden sollen. Der Rahmen eines Hilfe- bzw. Unterstützungssystems bietet dazu als Anhaltspunkt den Zeitbedarf für die Ausführung eines Ratschlages: die Ziele eines Benutzers sollten möglichst kurzfristig – in Relation zu einer Interaktion zwischen System und Benutzer – erreichbar sein. Dieses Kriterium läßt das obige Beispiel des mehrmonatigen Krafttrainings ebenso ausscheiden wie Handlungen, die eine längere Lern- oder Einarbeitungsphase erfordern. Die o.g. Änderung des Benutzer-Status eines UNIX-Systemadministrator ist eines der wenigen Beispiele für eine kurzfristige Änderung von Handlungsmöglichkeiten. Die meisten Handlungsmöglichkeiten eines Benutzers können dagegen in diesem Rahmen als unveränderlich angesehen werden; d.h. sie können nicht durch Handlungen des Benutzers verändert werden. Sie lassen sich damit wie die Funktionen eines Roboters durch Planoperatoren modellieren.

Die Definition der Handlungsmöglichkeiten eines Benutzers (Def. 3.1 S. 93) fordert lediglich, daß der Benutzer die Aktionen *im Prinzip* ausführen kann und darf. Ob er eine konkrete Aktion (in der konkreten Situation) tatsächlich ausführen kann oder nicht, kann grundsätzlich noch von weiteren Bedingungen abhängen. Dem Entwicklers des Sachbereichmodells steht also offen, solche Bedingungen auch zur Modellierung veränderbarer Handlungsmöglichkeiten zu nutzen – wie im o.g. Beispiel die Möglichkeiten eines UNIX Superusers. Die Menge der Planoperatoren, die das Handlungspotential eines Benutzers bilden, kann aber damit *nicht* durch das Handeln des Benutzers[10] verändert werden. Der Prozeß der Plangenerierung auf Basis eines Handlungspotentials kann also – wie in der Handlungsplanung üblich – von einer festen Menge von Planoperatoren ausgehen.

[10] oder verallgemeinert: beliebiger Akteure

Es sei betont, daß die so festgelegte Konstanz des Handlungspotentials eines Benutzers nur in Bezug auf das *Handeln* des Benutzer oder eines anderen Akteurs gilt. *Annahmen* eines Akteurs – insbesondere des Systems – *über* die Handlungsmöglichkeiten eines Benutzers können sich – vergleichbar zu Annahmen über seine Kenntnisse – im Verlauf einer Interaktion zwischen Benutzer und System sehr wohl ändern. Auf die Einzelheiten wird in Kap. 3.3 *Struktur eines Benutzermodells mit Handlungspotential und Wissen* noch eingegangen.

3.1.3 Berücksichtigung des Handlungspotentials in verwandten Forschungsarbeiten

Zu Anfang dieses Kapitels wurde gezeigt, daß eine explizite Berücksichtigung der Fähigkeiten eines Benutzers in verwandten Forschungsarbeiten – wenn überhaupt – nur eine marginale Rolle spielt. Selbst die Arbeiten, die ein explizites Modell des Benutzers aufbauen, trennen dabei nicht die Kenntnisse eines Benutzers von seinen Fähigkeiten bzw. präziser von seinem Handlungspotential. Im Gegensatz dazu wird in dieser Arbeit argumentiert, daß diese Trennung wichtig ist. Das wirft die Frage auf, wie diese Diskrepanz zu erklären ist, denn die anderen Arbeiten kommen ja offenbar ohne die Trennung und teilweise ganz ohne explizite Berücksichtigung eines Handlungspotentials des Benutzers aus.

Zunächst betrifft die Frage nur solche Systeme, die das Handeln des Benutzers – oder allgemeiner eines Akteurs – im modellierten Anwendungsbereich berücksichtigen. Als wichtigste Vertreter seien hier Planungssysteme und Systeme zur Planerkennung genannt.

Wie bereits erwähnt[11] gehen Planungssysteme implizit davon aus, daß der Akteur für den geplant wird alle Operatoren, die für die Plangenerierung zur Verfügung stehen, ausführen kann. Damit stellen die Planoperatoren – auch wenn dies in der Handlungsplanung meist nicht explizit erwähnt wird – ein Modell der Handlungsmöglichkeiten des Akteurs dar. Diese Sicht entspricht der Modellierung von Handlungsmöglichkeiten eines Akteurs durch das Handlungspotential, wie es in dieser Arbeit eingesetzt wird. Um die Kenntnisse des handelnden Akteurs kümmert sich die Plangenerierung nicht. Implizit ist damit die Annahme verbunden, daß der Akteur die Planschritte kennt und weiß wie er sie ausführen kann. Üblicherweise planen Systeme der Handlungsplanung allerdings für Roboter; d.h. es ist zumindest fragwürdig, ob Begriffe wie *Wissen* oder *Kennen* darauf übertragbar sind. Die Schnittstellenspezifikation eines Roboters beschreibt dessen Funktionalität, also die Handlungen die der Roboter ausführen kann. Soll für einen solchen Roboter ein Plan erzeugt werden, so werden die Planoperatoren entsprechend dieser Funktionsbeschreibung definiert. Damit ist sichergestellt, daß ein Plan nur solche Schritte enthält, die der Schnittstellenspezifikation des Roboters entsprechen; d.h. wird ein Planschritt als Kommando an den Roboter gegeben, so *wird* dieser – Funktionstüchtigkeit vorausgesetzt – das Kommando bzw. den Planschritt ausführen. Für das Erreichen eines Ziels ist nur dies relevant, während die Frage nach dem *Wissen* oder *Kennen* des Roboters dafür keine Rolle spielt.

Bei der Planerkennung geht es darum, aus einer beobachteten Folge von Einzelaktionen auf den Plan zu schließen, den der Akteur ausführt. Voraussetzungen für das Ausführen einer Aktion wurden weiter oben schon besprochen. Neben der Erfüllung der akteurunabhängigen Vorbedingungen muß der Akteur die Aktion kennen, wissen wie sie ausgeführt wird und natürlich auch dazu in der Lage sein. Wenn eine Aktion beobachtet wird, sind diese Bedingungen offenbar alle erfüllt. Aktionen, die ein Benutzer zwar kennt, aber nicht ausführen kann bzw. darf, können also ebensowenig beobachtet werden wie solche, die ein Benutzer zwar im Prinzip ausführen kann, aber nicht kennt. Eine Differenzierung zwischen Handlungspotential und Wissen eines Benutzer macht also keinen Sinn. Wenn beide Aspekte dennoch getrennt modelliert würden, könnten nur solche Aktionen beobachtet werden, die der Benutzer kann *und* kennt, die also in der Schnittmenge von Wissen und Handlungspotential liegen.

[11] vgl. bspw. Kap. 2.1.3 *Voraussetzungen beim Planen*.

Eine völlig andere Sicht ergibt sich für ein Hilfe- oder Unterstützungssystem. Der Rahmen einer Hilfestellung ist durch die (prinzipiellen) Handlungsmöglichkeiten des Benutzers bzw. sein Handlungspotential gegeben. Das Potential einer Unterstützung liegt in der Differenz zwischen diesen prinzipiellen Möglichkeiten eines Benutzers und seinen tatsächlich einsetzbaren Handlungsmöglichkeiten, die zusätzlich durch seine Kenntnisse begrenzt sind. Normalerweise wird ein Benutzer nicht alle seine (prinzipiellen) Handlungsmöglichkeiten kennen und/oder nicht von allen Aktionen wissen, wie sie ausgeführt werden können. Dies gilt besonders, wenn ein Benutzer von sich aus Hilfe anfordert. Eine wichtige Aufgabe eines Hilfesystems besteht darin, diese "Kenntnis-Lücke" zu schließen und damit die tatsächlich einsetzbaren Handlungsmöglichkeiten des Benutzers zu erweitern. Im Gegensatz zu einem Lehr- oder Tutor-System begrenzt eine zielgerichtete Hilfestellung die dafür notwendigen Erklärungen auf solche Aktionen, die für das Erreichen der Benutzerziele erforderlich sind. Die Aktionen werden in dieser Arbeit durch die Plangenerierung bestimmt, die das System anhand der Ziele und des Handlungspotentials des jeweiligen Benutzers durchführt. Sie sind damit auf die (prinzipiellen) Handlungsmöglichkeiten des Benutzer beschränkt.

Würde der Plan auf Basis der Benutzer-Kenntnisse erzeugt, wären keinerlei Erklärungen notwendig. Der Nutzen des Hilfesystems würde dann nur darin bestehen dem Benutzer den Planungsprozeß abzunehmen, den dieser aber auch selbst durchführen könnte. Der entscheidende Nachteil eines solchen Ansatzes wäre, daß Problemlösungen von vorn herein ausgeschlossen sind, die Aktionen erfordern, die der Benutzer nicht bereits kennt. Hat der Benutzer falsche Annahmen über Aktionen oder kennt er Aktionen, die er aber gar nicht ausführen kann oder darf, könnte der Planungsprozeß dann sogar falsche Ergebnisse liefern; d.h. Pläne, mit denen der Benutzer seine Ziele nicht erreichen kann. Je geringer die Benutzerkenntnisse sind, desto unwahrscheinlicher wird es, daß überhaupt eine korrekte Problemlösung gefunden wird. Dies betrifft sowohl Benutzer, die allgemein geringe Kenntnisse haben, als auch solche, die lediglich in einem Spezialbereich Kenntnis-Lücken haben und dafür spezifische Hilfe benötigen. In beiden Fällen könnte das System die Benutzer nicht mit den erforderlichen Informationen versorgen.

Abb. 3.1 faßt die Verwendung von Handlungsmöglichkeiten und Kenntnissen eines Benutzers in den verschiedenen Systemtypen zusammen. Systeme der Handlungsplanung verwenden bei der Plangenerierung lediglich ein Modell der Handlungsmöglichkeiten des Akteurs. Planerkennungssysteme können lediglich solche Aktionen beobachten, die der Akteur ausführen kann *und* auch kennt. Ihre Planbibliothek, auf der die Planerkennung beruht, entspricht damit dem Mengendurchschnitt der Modelle von Handlungsmöglichkeiten und Kenntnissen des Benutzers. Beide Systemtypen kommen damit ohne Differenzierung zwischen Handlungsmöglichkeiten und Kenntnissen aus.

Ein benutzerspezifisches Hilfesystem dagegen muß den Erklärungsbedarf aus der Differenz zwischen den für eine Problemlösung erforderlichen und den tatsächlichen Kenntnissen des Benutzers ermitteln. In dieser Arbeit wird zunächst das Handlungspotential des Benutzers eingesetzt, um einen Plan zu generieren, mit dem der Benutzer im Prinzip seine Ziele erreichen kann. Durch das Modell der Benutzerkenntnisse kann danach[12] der Erklärungsbedarf ermittelt werden.

3.1.4 Zusammenfassung

Das wichtigste Potential für die Unterstützung eines Computerbenutzers ergibt sich aus der Differenz zwischen den Handlungen, die dieser im Prinzip ausführen kann und darf und denjenigen, für die er auch die zur Ausführung erforderlichen Kenntnisse hat.

Diese Arbeit schlägt deshalb ein Benutzermodell vor, da sowohl die Handlungsmöglichkeiten eines Benutzers – die Handlungen, die er im Prinzip ausführen kann und darf – als auch seine Kennt-

[12] Auf Alternativen zur strikt sequentiellen Vorgehensweise wurde bereits hingewiesen; s.a. Kap. 4.5 *Optimierte Plangenerierung*.

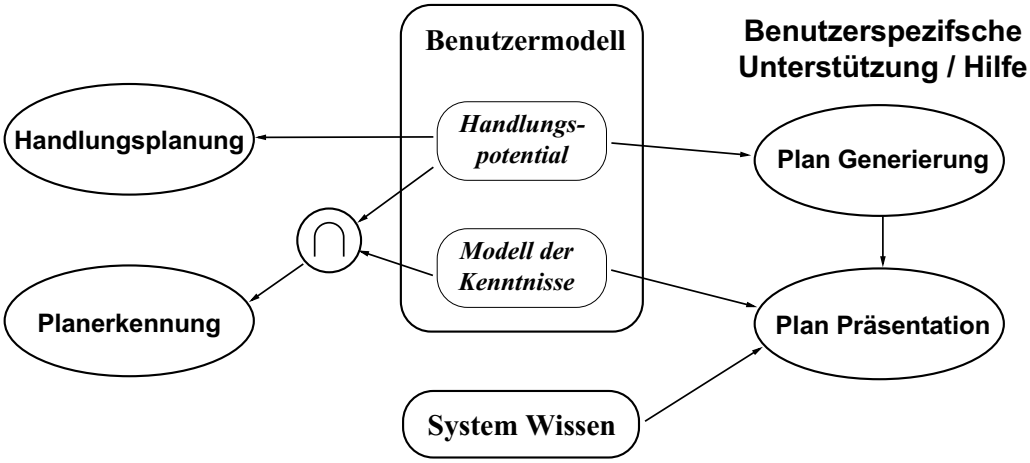

Abbildung 3.1: Einsatz von Handlungspotential und Modell der Kenntnisse eines Benutzers für unterschiedliche Anwendungen

nisse modelliert. Im Gegensatz zu bisherigen Systemen die Pläne eines Benutzers verarbeiten, werden beide Aspekte explizit modelliert und strikt getrennt. Die Handlungsmöglichkeiten werden wie in der Handlungsplanung durch Planoperatoren modelliert. Ein solches Modell wurde als Handlungspotential definiert und kann von einem Planungssystem genutzt werden, um einen Plan zu generieren, für den garantiert ist, daß der Benutzer ihn im Prinzip ausführen und damit seine Ziele erreichen kann[13]. Die konkrete Repräsentation eines solchen Benutzermodells und sein Einsatz bei der Plangenerierung sind Themen der folgenden Abschnitte.

3.2 Repräsentation von Planoperatoren

Im vorangegangenen Abschnitt wurde ein Benutzermodell vorgeschlagen, das sowohl die Kenntnisse als auch die Handlungsmöglichkeiten eines Benutzers repräsentiert. Zur Repräsentation von Benutzer*kenntnissen* liegen bereits umfangreiche Forschungsergebnisse vor. Viele Arbeiten (bspw. [Sleeman, 1985], [McCoy, 1989], [Sarner und Carberry, 1992], [Kobsa *et al.*, 1994]) konzentrieren sich auf das begriffliche Wissen, wobei sie die Begriffe des Anwendungsbereichs durch ein *beschreibungslogisches*[14] *Wissensrepräsentationssystem* definieren. Die Begriffe können dabei als Beschreibung von Objektklassen aufgefaßt werden. Bestimmende Merkmale für die Klassenzugehörigkeit sind dabei die Beziehungen zu anderen Objektklassen. Diese Art der Beschreibung hat ihre Wurzeln in *Frames* [Minsky, 1975] und *Semantischen Netzen* [Quillian, 1968], die als Wissensrepräsentationsformalismen für Systeme der Künstlichen Intelligenz und letztere insbesondere für die Verarbeitung natürlicher Sprache entwickelt wurden[15].

Die so definierten Begriffe bilden die Basis für eine Kommunikation zwischen Benutzer und System. Dabei setzt eine erfolgreiche Kommunikation voraus, daß beide Kommunikationspartner die gleiche Ansicht über die dabei verwendeten Begriffe haben – zumindest in den für die Kommunikation relevanten Aspekten. Das System kann diese Voraussetzung überprüfen, indem es seine eigenen

[13] unter den üblichen Voraussetzungen der Handlungsplanung (vgl. Kap. 2.1.3 *Voraussetzungen beim Planen*).

[14] Der Terminus *Beschreibungslogik* bzw. *Description Logic* wurde erst später eingeführt. Zuvor wurden beschreibungslogische Wissensrepräsentationssysteme oft *KL-ONE-artige* Systeme oder *terminologische Wissensrepräsentationssysteme* genannt.

[15] näheres s. nachfolgenden Abschnitt über Beschreibungslogiken.

Annahmen mit den in einem Benutzermodell repräsentierten (Systemannahmen über die) Annahmen des Benutzers vergleicht. Eventuelle Differenzen müssen ausgeräumt werden, sofern sie für die Kommunikation relevant sind. Dem Benutzer werden dazu die falschen Annahmen erklärt; d.h. es wird ihm eine korrekte Beschreibung der fraglichen Begriffe gegeben. Wie bereits erwähnt, ist die Einschätzung, ob Annahmen des Benutzers richtig oder falsch sind, grundsätzlich in Relation zu den Annahmen des Systems zu verstehen. Die korrigierenden Beschreibungen beruhen wiederum auf der Begrifflichkeit des Anwendungsbereichs und auch dabei muß wieder sichergestellt werden, daß beide Kommunikationspartner die gleichen Annahmen über die verwendeten Begriffe haben. Als Pionierarbeit für die Behandlung von falschen Benutzerannahmen über die Begrifflichkeit sei [McCoy, 1985] angeführt.

Für diese Arbeit ist interessant, ob sich die Erkenntnisse und Verfahren für die Beschreibung und Verarbeitung von Objekten bzw. Objektklassen auch für das Handeln eines Akteurs einsetzen lassen[16]. Dies betrifft sowohl die generellen Kenntnisse über die Möglichkeiten im Sachbereich zu Handeln als auch die spezifischen Handlungsmöglichkeiten eines bestimmten Akteurs. Für die Handlungsmöglichkeiten eines Akteurs wurde im vorangegangenen Abschnitt bereits vorgeschlagen, sie durch Planoperatoren zu modellieren, die in ihrer Gesamtheit das Handlungspotential des Benutzers darstellen. Wird auf Basis der Planoperatoren dieses Handlungspotentials ein Plan für einen Benutzer erzeugt, kann der Benutzer im Prinzip jeden einzelnen Planschritt des Plans ausführen. Damit der Benutzer den Plan auch tatsächlich ausführen kann, müssen ihm die Schritte des Plans genannt und – abhängig von seinen Kenntnissen – näher beschrieben werden. Es liegt also nahe, Kenntnisse *über* Handlungsmöglichkeiten in gleicher Weise zu modellieren wie die Handlungsmöglichkeiten selbst – d.h. durch Planoperatoren. Um an dem Ansatz festzuhalten, daß die Kommunikation zwischen System und Benutzer auf den Begriffen des Sachbereichs basiert, werden die Planoperatoren als Begriffe aufgefaßt. Welches die bestimmenden Merkmale für solche Begriffe sind und wie die konkrete Repräsentation in Beschreibungslogik und speziell im Wissensrepräsentationssystem SB-ONE [Kobsa, 1991] erfolgen kann, sind die Hauptthemen dieses Unterkapitels. Zuvor wird im nachfolgenden Abschnitt ein kurzer Überblick über die Technik der Wissensrepräsentation in Beschreibungslogik gegeben.

Der Ansatz, Begriffe für Handlungsmöglichkeiten zu definieren, eröffnet nicht nur die Möglichkeit die Forschungsergebnisse zu Begriffserklärungen und Korrektur von Fehlannahmen über Begriffe sowie generell zur Kommunikation zwischen System und Benutzer zu übernehmen. Darüber hinaus sind auch direkt die Techniken zum *Aufbau* eines Modells des begrifflichen Wissens des Benutzers einsetzbar; bspw. Interviewtechniken oder die Auswahl von Stereotypes als Quelle von a priori Annahmen über den Benutzer. Als technischer Vorteil ergibt sich, daß allgemein definierte Funktionen von Wissensrepräsentationssystemen – wie bspw. die Feststellung von Subsumption oder Disjunktheit zwischen zwei Begriffen – genutzt werden können, ohne auf ein bestimmtes System oder spezielle Algorithmen angewiesen zu sein.

3.2.1 Beschreibungslogiken

Beschreibungslogiken werden hier nur kurz und auf die für diese Arbeit relevanten Aspekte beschränkt dargestellt. Eine ausführlichere Darstellung mit vielen weiterführenden Hinweisen findet sich in [Baader, 1996]. Beschreibungslogiken sind Wissensrepräsentationsformalismen mit einer gegenüber der Prädikatenlogik erster Stufe eingeschränkten Ausdrucksfähigkeit, die am Zweck Wissensrepräsentation orientiert ist. Sie haben sich aus *Frames* [Minsky, 1975] und *semantischen Netzen* [Quillian, 1968] über Brachmans *strukturierte Vererbungsnetze* [Brachman, 1977] und *KL-ONE*

[16] Die damit implizierte Unterscheidung von Objekten (Dingen) und Handlungen wird in dieser Arbeit vorausgesetzt. Sie stützt sich auf die klassische Philosophie (Platon) und ist für diese Arbeit relevant, weil die charakteristischen Merkmale von Handlungen nicht nach dem gleichen Schema wie die von Objekten bestimmt und modelliert werden können.

[Brachman und Schmolze, 1985] entwickelt. Minskys Ziel war es, mithilfe von Frames zusammenhängende Information auch zusammenhängend zu repräsentieren. Er stellte seinen Formalismus in Kontrast zu einer (prädikaten-) logischen Wissensrepräsentation. Wichtig war auch die Zielsetzung einer – im Vergleich zur Semi-Entscheidbarkeit der Prädikatenlogik erster Stufe – geringen Berechnungskomplexität der Schlußfolgerungsalgorithmen. Eine ähnliche Zielsetzung verfolgt Quillian mit den semantischen Netzen, die für die Repräsentation der Semantik von natürlichen Sprachen entwickelt wurden. Beiden Ansätzen fehlt allerdings eine formal definierte Semantik (vgl. [Hayes, 1980] bzw. [Woods, 1975]). Dadurch wird die Bedeutung des damit dargestellten Wissens letztlich durch die Verarbeitungsalgorithmen bestimmt.

Brachman definiert für seine strukturierten Vererbungsnetze und deren Implementation KL-ONE eine feste Menge von – wie er es nennt – *epistemologischen Primitiven*. Die wichtigsten sind *Concept*[17], *Rolle* und die Operatoren *Spezialisierung* und *Restriktion*. Concepts beschreiben Mengen von Objekten durch charakteristische Merkmale, während durch Rollen Beziehungen zwischen zwei Concepts beschrieben werden können. Mithilfe der Operatoren können komplexere Concepts oder Rollen definiert werden. Der wichtige Punkt ist, daß die Verarbeitungsalgorithmen anhand der Definitionen der epistemologischen Primitiven entwickelt und auch hinsichtlich Korrektheit und Vollständigkeit beurteilt werden können. Nach Brachmans Veröffentlichung über KL-ONE wurden eine ganze Reihe von Wissensrepräsentationssystemen entwickelt, die diese Idee weiterführten[18]. Da diese Systeme zwar ähnlich sind, aber im Detail und insbesondere in der konkreten Notation Unterschiede aufweisen, spricht man auch von *KL-ONE-artigen* Systemen bzw. Wissensrepräsentationsformalismen.

Heutige Beschreibungslogiken basieren ebenfalls auf dieser Idee und unterscheiden sich hauptsächlich durch die Operatoren, mit denen komplexe Concepts und Rollen definiert werden können. Dies hat große Auswirkungen auf die Berechnungskomplexität der Schlußfolgerungsalgorithmen. Die *Worst-Case*-Komplexität für die – am besten untersuchte – Frage nach der Subsumption zweier Concepts reicht dabei von *polynominal* bis *unentscheidbar*.

In dieser Arbeit wird SB-ONE – als Wissensrepräsentationssystem, das sich "lose in das KL-ONE Paradigma einfügt" – für die Repräsentation der verschiedenen Teile des Benutzermodells und auch des Systemwissens eingesetzt. Die im weiteren verwendete Notation und Sprechweise entspricht allerdings der Spezifikation der *Knowledge Representation System Specification Group* (KRSS) der ARPA, die auf einem Vorschlag von DFKI-Forschern beruht [Baader *et al.*, 1990] und mit dem Zweck einer einheitlichen Darstellung beschreibungslogischen Wissens entwickelt wurde. Die für diese Arbeit wesentlichen Aspekte seien hier kurz wiedergegeben. Für eine vollständige Beschreibung sei auf [Patel-Schneider und Swartout, 1993] verwiesen. Auf die (für diese Arbeit relevanten) Unterschiede zwischen KRSS-Darstellung und SB-ONE sowie auf die speziellen Eigenschaften von SB-ONE wird im nachfolgenden Abschnitt eingegangen.

Grundbausteine der Repräsentation sind Concepts, Rollen, Attribute und Individuen. Sie sind entweder Namen oder Ausdrücke. Die syntaktisch zulässigen Concept-Ausdrücke die in dieser Arbeit verwendet werden sind in Tabelle 3.2 beschrieben. Dabei stehen (ggf. indizierte) C für Concept-Beschreibungen (Name oder Ausdruck), R für Rollen- und Attributbeschreibungen, A für Attributbeschreibungen und n für eine natürliche Zahl. Die Definition der Semantik erfolgt über Interpretationen I, die aus einem *Objektbereich* (auch *Interpretationsbereich* oder *Domain*) Δ^I und einer *Interpretationsfunktion* $.^I$ besteht. Die Interpretationsfunktion ordnet den Namen jeweils eine Extension zu: jedem Concept-Namen eine Teilmenge von Δ^I, jedem Rollen-Namen eine Relation ($\subseteq \Delta^I \times \Delta^I$), jedem Attribut-Namen eine rechtseindeutig Relation ($\subseteq \Delta^I \times \Delta^I$) und jedem Individuen-Namen ein

[17] In dieser Arbeit wird am englischen *Concept* festgehalten, weil die – häufig anzutreffende – Eindeutschung *Konzept* nicht in der Bedeutung des deutschen Begriffs *Konzept* verwendet wird (vgl. bspw. [Bro, 1996]). Andererseits ist die treffendere Übersetzung in *Begriff* in diesem Zusammenhang wenig verbreitet.

[18] vgl. [Heinsohn *et al.*, 1994] für eine Analyse implementierter Systeme.

Bezeichnung	KRSS-Ausdruck	Semantik
TOP	$\top$	Δ^I
Negation	$\neg C$	$\Delta^I \backslash C^I$
Konjunktion	$C_1 \sqcap \ldots \sqcap C_n$	$C_1^I \cap \ldots \cap C_n^I$
Disjunktion	$C_1 \sqcup \ldots \sqcup C_n$	$C_1^I \cup \ldots \cup C_n^I$
Werterestriktion	$\forall R : C$	$\{d \in \Delta^I \mid R^I(d) \subseteq C^I\}$
Werterestriktion[a] (ex.quant)	$\exists R : C$	$\{d \in \Delta^I \mid R^I(d) \cap C^I \neq \varnothing\}$
Anzahlrestriktion	$\geq nR$	$\{d \in \Delta^I \mid \lvert R^I(d)\rvert \geq n\}$
Anzahlrestriktion	$\leq nR$	$\{d \in \Delta^I \mid \lvert R^I(d)\rvert \leq n\}$
Role-Value-Map =	$R_1 = R_2 \sqcap \exists R_1$	$\{d \in \Delta^I \mid R_1^I(d) = R_2^I(d) \wedge R_1^I(d) \neq \varnothing\}$
Role-Value-Map $\neq$	$R_1 \neq R_2$	$\{d \in \Delta^I \mid R_1^I(d) \neq R_2^I(d)\}$
Role-Value-Map $\subseteq$	$R_1 \subseteq R_2$	$\{d \in \Delta^I \mid R_1^I(d) \subseteq R_2^I(d)\}$
Werterestriktion (Attribut)	$A : C$	$\{d \in \Delta^I \mid A^I(d) \in C^I\}$

[a] Unter Werterestriktion wird in der Literatur teilw. nur ein Ausdruck der Form $\forall R : C$ verstanden.

Tabelle 3.2: Syntax und Semantik von Concept-Ausdrücken (Auszug aus [Patel-Schneider und Swartout, 1993, Tab.1]).

Element aus Δ^I. Die Interpretationsfunktion wird auf Ausdrücke erweitert, wie in den o.g. Tabellen angegeben.

Die Extensionen von Rollen bzw. Attributen können auch als partielle Funktionen $\Delta^I \rightarrow 2^{\Delta^I}$ bzw. $\Delta^I \rightarrow \Delta^I$ aufgefaßt werden. Elemente der Extension eines Concepts werden als *Exemplare* des Concepts bezeichnet[19]. Oft wird ein Concept auch *Typ* eines Exemplars dieses Concepts genannt. Attribute können als Spezialfall von Rollen aufgefaßt werden und werden deshalb im weiteren nur dann gesondert behandelt, wenn ihre speziellen Eigenschaften relevant sind.

Von den Rollen- und Attribut-Ausdrücken, die KRSS vorsieht (s. [Patel-Schneider und Swartout, 1993, Tab. 2 u. 3]), ist für diese Arbeit lediglich die Komposition $R_1 \circ \ldots \circ R_n$ mit der Semantik $R_1^I \circ \ldots \circ R_n^I$ relevant, die für Rollen und Attribute gleich definiert ist.

Alternativ zur Angabe der Extensionen kann die Semantik auch durch Prädikatenlogik erster Stufe angegeben werden. Concepts bzw. Rollen entsprechen dann Formeln mit einer bzw. zwei freien Variablen. Der Zusammenhang zwischen beiden Alternativen ergibt sich dadurch, daß eine solche Formel bei gleichem Objektbereich genau für die Objekte bzw. Objektpaare "wahr" ist, die zur Extension des Concepts bzw. der Rolle gehören. Bspw. kann die Semantik des Ausdrucks $\forall R : C$ sowohl durch die Menge $\{d \in \Delta^I | R^I(d) \subseteq C^I\}$ als auch durch die prädikatenlogische Formel $\forall y\ R^I(x,y) \rightarrow C^I(y)$ mit der freien Variable x definiert werden. Die Gleichwertigkeit beider Definitionen wird weiter unten noch eine Rolle spielen, wenn eine (prädikatenlogische) Spezifikation des Planungssachbereichs aus der beschreibungslogischen Repräsentation des Benutzermodells abgeleitet wird (vgl. a. Kap. 3.4 *Planungsproblem und Planungssachbereich*).

Zur Definition von Concept-Namen durch komplexe Concept-Ausdrücke sehen Beschreibungslogiken *Statements* (oder *terminologische Axiome*) vor[20]. Die Ausdrücke werden dann als *Beschreibung* der Concept-Namen angesehen, da durch die Semantik der Ausdrücke bestimmt ist, welche Objekte

[19] In der Literatur findet sich teilw. auch die Eindeutschung *Instanz* (eines Konzepts) des englischen *instance*.

[20] KRSS sieht entsprechend auch die explizite Definition von Rollen- und Attribut-Namen vor, die in dieser Arbeit aber nicht verwendet wird.

durch den Ausdruck bezeichnet werden und welche nicht. Es wird zwischen einer *definierten* und einer *primitiven* Zuordnung unterschieden. Erstere hat die syntaktische Form *Concept-Name* $\doteq C$ und bedeutet, daß die Extension des Concept-Namen per Definition gleich derjenigen von C ist. Eine *primitive* Zuordnung *Concept-Name* $\sqsubseteq C$ bedeutet, daß die Extension des Concept-Namen per Definition eine Teilmenge derjenigen von C ist. Die durch solche Zuordnungen beschriebenen Concept-Namen werden auch *definierte* bzw. *primitive* Concepts genannt.

Der Ausdruck C einer *definierten* Zuordnung $CN \doteq C$ stellt eine *notwendige* und *hinreichende* Beschreibung von CN in dem Sinne dar, daß jedes Objekt $x \in \Delta^I$, das zu CN gehört ($x \in CN^I$), *notwendigerweise* auch der Beschreibung durch C entsprechen muß (d.h. $x \in C^I$) *und*, daß diese Beschreibung auch *hinreichend* ist, um ein Objekt als zu CN gehörig zu bestimmen (d.h. aus $x \in C^I$ folgt $x \in CN^I$). Demgegenüber werden durch C in einer *primitiven* Concept-Definition $CN \sqsubseteq C$ lediglich die *notwendigen* Bedingungen für die Zugehörigkeit eines Objektes zu CN angegeben; d.h. aus $x \in CN^I$ folgt $x \in C^I$, aber nicht umgekehrt.

Als Beispiel sei eine (vereinfachte Version der) Definition einer *Datei* (*fs-file*) angeführt, die im Kap. 5.2.2 *Der Beispielsachbereich* noch genauer behandelt wird:

$$\textit{fs-file} \quad \sqsubseteq \quad \forall \textit{named} : \textit{fs-name} \sqcap \textit{contents} : \textit{information} \sqcap \textit{file-type} : \textit{file-type-c}$$

Es handelt sich um eine *primitive Definition*, womit ausgedrückt werden soll, daß diese Beschreibung nicht vollständig ist – zur eindeutigen Identifikation eines Objekts kann sie also nicht herangezogen werden. Sie stellt lediglich eine Beschreibung der notwendigen Merkmale dar. Dies sind hier:

- eine Datei kann eine beliebige Anzahl Namen haben – ausgedrückt durch die Rolle *named* mit Werterestriktion *fs-name*
- eine Datei hat genau einen Inhalt – ausgedrückt durch das Attribut *contents* mit Werterestriktion *information*
- eine Datei hat genau einen Dateityp – ausgedrückt durch das Attribut *file-type* mit Werterestriktion *file-type-c*

Durch eine *Werterestriktion* (*Value Restriction*) wird der Wertebereich einer Rolle oder eines Attributs eingeschränkt. Damit wird bspw. definiert, daß ein Objekt, das zu *fs-file* gehört ($x \in \textit{fs-file}^I$), nur zu solchen Objekten in Relation $\textit{named}^I$ stehen kann, die zum Concept *fs-name* gehören. Solche Objekte werden auch *Rollenfüller* genannt; hier: Füller der Rolle *named*.

Ein Beispiel für ein *definertes* Concept ist *Text-Datei* (*text-file*):

$$\textit{text-file} \quad \doteq \quad \textit{fs-file} \sqcap \textit{file-type} : \textit{file-type-text}$$

Diese Definition beschreibt das Concept *text-file* als Spezialisierung von *fs-file*. Die Spezialisierung gegenüber *Datei* wird durch das zusätzliche Merkmal beschrieben, daß eine *Text-Datei* einen Dateityp mit Wertebereich *file-type-text* hat. Die Semantik eines Ausdrucks $C_1 \sqcap C_2$ besagt, daß alle Merkmale von C_1 oder C_2 auch Merkmale des Ausdrucks sind (Sprechweise: die Merkmale werden *vererbt*). Da eine *Datei* im Beispiel bereits einen Dateityp hat, hat *Text-Datei* sowohl das Merkmal *file-type : file-type-c* als auch *file-type : file-type-text*. Formal also (*file-type : file-type-c*) $\sqcap$ (*file-type : file-type-text*). Nach der o.g. Semantik ist dies gleichbedeutend mit *file-type :* (*file-type-c* $\sqcap$ *file-type-text*). Die Definition der *Text-Datei* besagt also, daß das bereits durch *Datei* ererbete Attribut *file-type* in seinem Wertebereich eingeschränkt wird. Diese Restriktion eines Wertebereichs gilt entsprechend für ererbte allquantifizierte Rollen.

Im Beispiel hier ist *file-type-text* eine Spezialisierung von *file-type-c*[21]; d.h. für jedes $x \in$ *file-type-text* gilt auch $x \in$ *file-type-c*. Der Wertebereich des Attributs *file-type* ist also demnach genau *file-type-text*.

Die Definition von *Text-Datei* bestimmt explizit, daß *Text-Datei* eine Spezialisierung von *Datei* ist. Es ist aber auch möglich, daß ein Concept C_1 spezieller als ein anderes C_2 ist, ohne daß dies explizit definiert ist. Spezieller meint dabei, daß jedes Objekt, das zu C_1 gehört, immer auch zu C_2 gehört – d.h. $C_1^I \subseteq C_2^I$. Bspw. *könnte* eine ASCII-Datei mit der Definition

$$\textit{ascii-file} \doteq \textit{fs-file} \sqcap \textit{file-type} : \textit{file-type-ascii}$$

spezieller als *Text-Datei* sein. Ob dies tatsächlich so ist, hängt hier von den jeweiligen Wertebereichen von *file-type* ab; d.h. ob *file-type-ascii* spezieller als *file-type-text* ist.

In Beschreibungslogiken drückt die Relation *Subsumption* $C_1 \Rightarrow C_2$ mit der Semantik $C_1^I \subseteq C_2^I$ bzw. $\forall x\, x \in C_1^I \rightarrow x \in C_2^I$ aus, daß C_1 spezieller als C_2 ist. Die Frage, ob die Relation gilt, wird *Subsumptionsproblem* genannt. Das Feststellen der Subsumption ist ein zentraler Dienst von beschreibungslogischen Wissensrepräsentationssystemen, dessen Berechnungskomplexität stark von den zulässigen Ausdrücken abhängt[22].

Besonders kritisch erweisen sich dabei sogenannte *Role-Value-Maps* (*RVMs*), die im nächsten Abschnitt für die Beschreibung von Planungswissen benötigt werden. Eine RVM besteht aus dem Concept-Ausdruck $R_a \oplus R_b$, wobei R_a und R_b für jeweils Rollenausdrücke der Form $R_1 \circ \ldots \circ R_n$ – sogenannten *Rollenketten* (*Role Chains*) – stehen. $\oplus$ ist ein Mengenvergleichsoperator, wobei KRSS $\subseteq$, = und $\neq$ zuläßt. Durch RVMs lassen sich Concepts beschreiben, deren Rollenfüller untereinander in einer bestimmten Relation stehen. Brachman und Schmolze definieren in [Brachman und Schmolze, 1985] als Beispiel eine *Wichtige Nachricht* als Nachricht an genau einen *Empfänger*, die vom *Chef des Empfängers* gesendet wurde. Abb. 3.2 zeigt das Beispiel (geringfügig vereinfacht) in KRSS-Syntax.

$$\begin{aligned}
\textit{Nachricht} &\sqsubseteq \textit{Inhalt} : \textit{Text} \sqcap \\
&\quad \forall \textit{Absender} : \textit{Person} \sqcap \geq 1\ \textit{Absender} \sqcap \\
&\quad \forall \textit{Empfänger} : \textit{Person} \sqcap \geq 1\ \textit{Empfänger} \\
\textit{Angestellter} &\sqsubseteq \textit{Person} \sqcap \textit{Chef} : \textit{Person} \\
\textit{Wichtige-Nachricht} &\doteq \textit{Nachricht} \sqcap \\
&\quad \textit{Empfänger} : \textit{Angestellter} \sqcap \\
&\quad \boldsymbol{\textit{Absender} = \textit{Empfänger} \circ \textit{Chef}}
\end{aligned}$$

Abbildung 3.2: Beschreibung einer *Wichtigen Nachricht* nach [Brachman und Schmolze, 1985, S.194]

In diesem Beispiel wird durch die RVM in der letzten Zeile gefordert, daß die Menge der Absender einer wichtigen Nachricht gleich der Menge der Chefs der Empfänger sein muß. Die Beschreibung besagt außerdem, daß eine wichtige Nachricht einen eindeutigen Empfänger, der Angestellter ist, haben muß und daß ein Angestellter einen eindeutigen Chef haben muß. Damit ergibt sich, (1) daß eine wichtige Nachricht auch einen eindeutigen Absender haben muß und (2) daß der (Füller der Rolle) *Empfänger* mit dem (Füller der Rolle) *Absender* in einer Relation steht, die durch das Attribut *Chef* repräsentiert wird.

[21] allerdings hier nicht explizit definiert. Für eine umfassendere Beschreibung s. Kap. 5.2.2 *Der Beispielsachbereich*.
[22] vgl. bspw. [Donini *et al.*, 1997]

In ihrer allgemeinen Form führen RVMs allerdings dazu, daß das Subsumptionsproblem von Beschreibungslogiken unentscheidbar wird ([Schmidt-Schauß, 1989]). Hollunder und Nutt weisen dagegen in [Hollunder und Nutt, 1990] nach, daß das Subsumptionsproblem entscheidbar ist, wenn anstelle von RVMs lediglich Vergleiche (=, $\neq$) sogenannter *Feature Chains* zulässig sind. *Feature Chains* unterscheiden sich von Rollenketten dadurch, daß anstelle von Rollen Attribute verknüpft werden. Die meisten Wissensrepräsentationssysteme lassen RVMs deshalb nur eingeschränkt zu und auch in dieser Arbeit werden sie nur eingeschränkt verwendet.

Bisher wurde lediglich die Definition von Concepts und (meist implizit) von Rollen behandelt. Eine Menge solcher Definitionen wird *Terminologie* oder *TBox* genannt. Beschreibungslogische Wissensrepräsentationssysteme, die sich auf die Definition und Verarbeitung einer Terminologie beschränken, nennt man deshalb *terminologische Wissensrepräsentationssysteme* (TWRS). Für diese Arbeit reicht eine solche Beschränkung aus. Lediglich der Vollständigkeit halber sei erwähnt, daß eine Komponente für die Definition und Verarbeitung von Individuen *assertionale Komponente* oder *ABox* genannt wird.

3.2.2 Planoperatoren in Beschreibungslogik

Zu Beginn des Kapitels über die Repräsentation von Planoperatoren wurde argumentiert, daß es für ein Benutzermodell wünschenswert ist, das Modell von Handlungsmöglichkeiten – in der gleichen Weise wie Kenntnisse über die Objekte des Anwendungsbereichs – als begriffliches Wissen zu repräsentieren. Voraussetzung dafür ist die Feststellung der identifizierenden Merkmale von Handlungsmöglichkeiten bzw. eines Planoperatoren. Natürlich hat ein *imaginäres* Gebilde wie eine Handlungsmöglichkeit oder eine Handlung keine *realen* Eigenschaften, wie bspw. die Farbe eines Objekts. Charakteristisch für eine Handlung ist dagegen ihre Wirkung, also die Veränderung die sie bewirkt. Im Modell der Handlungsplanung wird sie durch den Effekt eines Operators berücksichtigt und stellt den entscheidenden Grund für die Einbeziehung eines Operators in einen Plan dar. Aus Sicht des Akteurs ist der Effekt der Grund, warum er eine Aktion ausführt. Vergleichbar wichtig ist die Vorbedingung eines Operators, weil sie bestimmt, ob der Operator (an einer bestimmten Position) im Plan verwendet werden kann.

Eine Beschreibung von Planoperatoren durch Vorbedingung und Effekt läßt Entscheidungen über die Verwendung eines Operators für einen bestimmten Zweck zu, wie sie bei der Konstruktion eines Plans zu treffen sind. Eine andere Sicht von charakteristischen Eigenschaften ergibt sich, wenn eine zu einem Planoperator gehörende Aktion ausgeführt werden soll. Dann ist die Frage nach dem *wie* interessant; d.h. welche Subaktionen sind in welcher (zeitlichen) Anordnung für die Aktion auszuführen. Diese Sicht fragt also nach der inneren Struktur – den Bestandteilen und deren Anordnung. Die Sicht ist keineswegs auf Planoperatoren oder Handlungen beschränkt, sondern ist auch für Objekte relevant. Allerdings ist eine entsprechende Modellierung mithilfe von Beschreibungslogik nicht ohne weiteres möglich. Für die *Erzeugung* eines Plans ist sie nur bei hierarchischen Planungsverfahren erforderlich. Ihre Behandlung wird deshalb auf Kap. 4.6 *Die innere Struktur von Plan-Concepts* zurückgestellt und im Zusammenhang mit der Präsentation eines Plans behandelt.

Als Vorbereitung für die Entwicklung einer Repräsentation von Planoperatoren in Beschreibungslogik mit Vorbedingung und Effekt als identifizierende Merkmale, seien die Besonderheiten eines Planoperators (gegenüber Objekten) am Beispiel eines Operators für das Öffnen einer Datei gezeigt. Die Darstellung von Planoperatoren entspricht hier und im Rest dieser Arbeit der Syntax der *Planning Domain Definition Language* (PDDL) [Ghallab *et al.*, 1998], die bereits zu Beginn des Kapitels 2.1 über Handlungsplanung kurz besprochen wurde. Die Sprache wurde gewählt, weil sie eine planungssystemunabhängige Spezifikation von Planungssachbereich und Aufgabenstellung unterstützt. Nur ein Teil der Sprache betrifft die Definition von Planoperatoren. Abb. 3.3 zeigt den Ausschnitt der

planop-def	=	"(" ":action" *planop-name*
		[":parameters" "(" {*var*}$^+$ ")"]
		[":precondition" *plan-conjunction*]
		":effect" *plan-conjunction* ")" .
plan-conjunction	=	*literal-conjunction* | *literal* | "(" ")" .
literal-conjunction	=	"(" "and" {*literal*}$^+$ ")" .
literal	=	*predicate* | "(" "not" *predicate* ")" .
predicate	=	"(" *predicate-name var* [*var*] ")" .
name		Name nach LISP-Syntax
var		LISP-Name, der mit einem Fragezeichen beginnt
any-string		String nach LISP-Syntax

{X}$^+$ steht für ein oder beliebig viele Vorkommen von X

Abbildung 3.3: EBNF-Syntax für die Definition von Planoperatoren.

Syntax für Planoperatoren, wie sie in dieser Arbeit zulässig sind, in EBNF[23]; Startsymbol ist dabei *planop-def*. Parameter werden durch eine Liste von Variablennamen angegeben, Vorbedingung und Effekt jeweils als einzelnes ein- oder zweistelliges Literal oder als Konjunktion davon. Die Notation läßt den direkten Vergleich mit der Operatordarstellung in verschiedenen Planungssystemen zu und wird für den Aufruf des Planungssystems in dieser Arbeit verwendet (vgl. 3.4.2.7 *Erzeugung des aktiven Teils eines Planungssachbereichs*).

Abb. 3.4 zeigt den Beispiel-Operator *open-file-tpx*. Er dient dazu, eine Datei in einem Textverarbeitungssystem des Typs *tpx* zu öffnen; d.h. zum aktiven Dokument zu machen. Voraussetzung ist, daß es sich bei der Datei um eine Datei für dieses Textverarbeitungssystem handelt (*tpx-file*), daß sie sich im lokalen Dateisystem befindet (*local-fs-object*) und daß das Textverarbeitungssystem bereits gestartet wurde (*launched*). Die Notation für Vorbedingung und Effekt entspricht einer LISP-Darstellung prädikatenlogischer Formeln in Präfix-Notation. Dabei bezeichnet der erste Name innerhalb einer Klammer einen logischen Operator oder ein Prädikat; danach folgen die Argumente. Variablennamen ist grundsätzlich ein Fragezeichen vorangestellt. Freie Variablen für die Objekte, die für den Planoperator relevant sind, werden explizit als Parameter angegeben. In diesem Beispiel ist *?f* die Variable für die Datei und *?app* die für das Textverarbeitungssystem. Die Vorbedingung wird durch je zwei Prädikate, die für die beiden Objekte wahr sein müssen, definiert. Der Effekt wird durch das zweistellige Prädikat *active* beschrieben, das die Relation angibt, die *nach* Ausführung eines solchen Operators zwischen den beiden Objekten gilt.

```
(:action open-file-tpx
    :parameters   (?f ?app)
    :precondition (and (tpx-file ?f) (local-fs-object ?f)
                       (tpx ?app)    (launched ?app))
    :effect       (active ?app ?f))
```

Abbildung 3.4: Planoperator *open-file-tpx*.

Die ein- bzw. zweistelligen Prädikate einer solchen Beschreibung können in Beschreibungslo-

[23] vgl. [Jensen und Wirth, 1985]; die Beschreibung von EBNF findet sich auch im Anhang A.1 dieser Arbeit.

gik durch Concepts bzw. Rollen repräsentiert werden. Der zeitliche Aspekt, der sich dadurch ergibt, daß die Vorbedingung *vor* Ausführung des Planoperators gelten muß und der Effekt *danach*[24], läßt sich aber weder in Beschreibungslogik noch in Prädikatenlogik adäquat repräsentieren. Im Gegensatz zu [Devanbu und Litman, 1996][25], die diesen Aspekt durch die Einführung von *Zustand-Concepts* in gewisser Weise berücksichtigen, wird hier die Ansicht vertreten, daß eine Modellierung der Zeit für die Repräsentation von Planoperatoren nicht erforderlich ist. Auch die Repräsentation in PDDL sieht kein explizites Modell der Zeit vor. Daß Vorbedingung und Effekt eines Planoperators zu unterschiedlichen Zeiten gelten, ist lediglich eine *Vereinbarung*, die beim Planungsprozeß berücksichtigt wird. Eine beschreibungslogische Repräsentation, die Basis für eine Planerzeugung ist, kann deshalb ebenfalls auf die explizite Darstellung der Zeit verzichten. Sie muß lediglich sicherstellen, daß sich die Beschreibung der Vorbedingung eindeutig von der Beschreibung des Effekts trennen läßt.

Der Rest dieses Abschnitts behandelt die Entwicklung einer solchen Repräsentation von Planoperatoren. Für jeden Planoperator wird dabei ein *Plan-Concept* definiert. Das allgemeinste Plan-Concept einer Terminologie heißt *plan-concept*. Entsprechend der Unterscheidung zwischen Objekten und Handlungsmöglichkeiten wird *plan-concept* disjunkt[26] zu *domain-concept* definiert, wobei *domain-concept* das allgemeinste Concept ist, das ausschließlich Objekte modelliert[27]. Naheliegenderweise wird festgelegt, daß diese Menge von Objekten identisch ist zur Menge der Objekte des Planungssachbereichs, d.h. der Objekte, die für das Planungssystem modelliert werden (vgl. a. Kap. 3.4.2 *Der Planungssachbereich*). Die Beschreibung von Objekteigenschaften erfolgt im Planungssachbereich durch Prädikate. Mit Ausnahme der Prädikate *Gleich* und *Ungleich* gehört zu jedem Prädikat des Planungssachbereichs ein gleichnamiges Concept bzw. eine gleichnamige Rolle in der beschreibungslogischen Darstellung. Der Zusammenhang zwischen prädikatenlogischer und beschreibungslogischer Repräsentation ergibt sich dadurch, daß jedes Prädikat genau für die Elemente der Extension des zugehörigen Concepts bzw. Rolle "wahr" ist (vgl. die Definitionen 3.3 u. 3.4 S. 141).

Die PDDL-Definition eines Planoperators zeigt durch Parameter, welche Objekte an der modellierten Aktion beteiligt sind. Durch die Vorbedingung wird beschrieben, daß die Objekte vor der Ausführung des Operators bestimmte Eigenschaften haben müssen. Der Effekt bestimmt, welche Eigenschaften sich durch die Ausführung verändern. Die Eigenschaften betreffen dabei die gleichen Objekte; d.h. die Modellierung läßt zu, daß die Vorbedingung für ein Objekt eine Eigenschaft fordert, die vom Effekt für das gleiche Objekt negiert wird. Diese Vorstellung läßt sich in einer zeitunabhängigen Logik nicht modellieren, weil dadurch lediglich eine "Momentaufnahme" beschrieben werden kann. Das Negieren einer Eigenschaft würde dabei zu einem logischen Widerspruch führen.

Diese Schwierigkeit hat ihre Ursache in der Vorstellung, daß es sich bei den Objekten mit veränderten Eigenschaften tatsächlich um die gleichen Objekte wie vor der Ausführung eines Planoperators handelt. Genaugenommen widerspricht dies aber der Vorstellung, daß ein Objekt durch seine Eigenschaften oder einer Teilmenge davon identifiziert werden kann. Als alternative Sicht bietet sich an, die veränderten Objekte als *andere*, evtl. auch *neue* Objekte zu betrachten. Da mit dieser Sicht die o.g. Widersprüche durch eine Negierung von Objekteigenschaften nicht auftreten, bietet sie die Basis für die Repräsentation von Planoperatoren in einer zeitunabhängigen Beschreibungslogik.

Für ein Plan-Concept, das einen Planoperator repräsentiert, wird dazu für jeden Parameter jeweils ein Attribut definiert, das eine Funktion repräsentiert, die ein Exemplar des Plan-Concepts auf ein *Vorbedingungs-Objekt* abbildet und ein weiteres Attribut, das die entsprechende Funktion auf ein *Effekt-*

[24] Nach den Voraussetzungen der Handlungsplanung (vgl. Kap. 2.1.3) nimmt die Ausführung einer Aktion keine Zeit in Anspruch. Damit entfällt eine Festlegung, was *während* der Ausführung der Aktion gilt bzw. gelten muß.

[25] vgl. Kap. 2.2.3 *Planerkennung mit Beschreibungslogik*

[26] Zwei Concepts sind disjunkt, wenn ihre Extensionen disjunkt sind.

[27] Dies schließt "gedachte" Objekte ein – insb. auch solche, die erst durch das Ausführen einer Aktion erzeugt werden können (vgl. Kap. 3.4.2.5 *Erzeugen von Objekten* und 5.2.2 *Der Beispielsachbereich*).

Objekt repräsentiert. Ein Vorbedingungs- bzw. Effekt-Objekt steht dabei für ein Objekt – also ein Element von *domain-concept*^{I} –, das durch die Vorbedingung bzw. den Effekt des Operators beschrieben wird. Zur einfachen Unterscheidung wird in dieser Arbeit vereinbart, daß die Namen aller Attribute, die Funktionen auf Vorbedingungs- bzw. Effekt-Objekte repräsentierten, den Präfix *pre_* bzw. *eff_* erhalten und *Vorbedingungs-* bzw. *Effekt-Attribute* genannt werden. Eine solche Unterscheidung kann auch direkt in Beschreibungslogik ausgedrückt werden. Bspw. könnten alle Vorbedingungs- bzw. Effekt-Attribute als Spezialisierung einer Rolle[28] *precondition* bzw. *effect* definiert werden, wobei *precondition* und *effect* disjunkt und jeweils als *plan-concept* $\times$ *domain-concept* definiert sind. Davon wird in dieser Arbeit allerdings abgesehen, weil die Unterscheidung für Schlußfolgerungen des beschreibungslogischen Wissensrepräsentationssystems keine Rolle spielt.

Die einzelnen Vorbedingungskomponenten eines Planoperators können nun durch Restriktion der Vorbedingungs-Attribute und durch RVMs repräsentiert werden. Als Vorbedingungskomponenten sind ein- oder zweistellige Literale zulässig. Dabei sind in PDDL und auch dieser Arbeit die zweistelligen Prädikate *Gleich* (=) und *Ungleich* ($\neq$) vordefiniert. Da sie separat behandelt werden, können insgesamt die sechs in Tabelle 3.3 aufgeführten Fälle auftreten. Die Tabelle enthält für jede mögliche Form einer Vorbedingungskomponente den zugehörigen KRSS-Ausdruck und seine Realisierung durch ein SB-ONE-Konstrukt. Auf letzteres wird aber erst im folgenden Abschnitt eingegangen. P bzw. R stehen in der prädikatenlogischen Darstellung für Namen ein- bzw. zweistelliger Prädikate und in der beschreibungslogischen Darstellung für Concept- bzw. Rollen- oder Attribut-Namen.

	Komponente	KRSS-Ausdruck	SB-ONE-Konstrukt
(1)	$P(x)$	$pre_x : P$	$\mathrm{VR}(pre_x, P)$
(2)	$\neg P(x)$	$pre_x : \neg P$	*siehe nächster Abschnitt*
(3)	$= (x,y)$	$pre_x = pre_y \sqcap \exists pre_x$	$\mathrm{RVM}\ (=, (pre_x), (pre_y))$
(4)	$\neq (x,y)$	$pre_x \neq pre_y$	$\mathrm{RVM}\ (\otimes, (pre_x), (pre_y))$
(5)	$R(x,y)$	$pre_y \subseteq pre_x \circ R$	$\mathrm{RVM}\ (\subseteq, (pre_y), (pre_x, R))$
(6)	$\neg R(x,y)$	$\neg(pre_y \subseteq pre_x \circ R)$	$\mathrm{RVM}\ (\otimes, (pre_x, R), (pre_y))$

Tabelle 3.3: Zulässige Vorbedingungskomponenten eines Planoperators und die beschreibungslogische Entsprechung als KRSS-Ausdruck und SB-ONE-Konstrukt.

Der Effekt eines Planoperators wird analog behandelt, wobei in Tabelle 3.3 Vorbedingungs- durch Effekt-Komponenten zu ersetzen sind. Die Gleichheits- bzw. Ungleichheitsbedingungen (Zeilen (3) u. (4)) sind allerdings nur für Vorbedingungen zulässig.

Die Zeilen (3)-(6) setzen voraus, daß x und y Parameter des Planoperators sind und deshalb für das Plan-Concept die Attribute pre_x und pre_y definiert wurden. Die Zeilen (1), (2) und (4) verwenden direkt die Semantik der beschreibungslogischen Ausdrücke. In Zeile (3) ist der Teilausdruck $\exists pre_x$ redundant, weil pre_x ein Attribut ist und bereits damit die Existenz eines Rollenfüllers garantiert ist. KRSS sieht allerdings nur genau diesen zusammengesetzten Ausdruck (als RVM =) vor.

Der KRSS-Ausdruck von Zeile (5) soll Operatoren modellieren, deren Parameter x und y bei beliebiger Variablenbelegung in Relation R^I stehen. Das heißt, daß auch die Rollenfüller von pre_x und pre_y eines Exemplars d des zugehörigen Plan-Concepts in dieser Relation stehen müssen; anders ausgedrückt $pre_x^I(d)\ R^I\ pre_y^I(d)$. Da $pre_x^I(d)$ und $pre_y^I(d)$ eindeutig bestimmt sind, ist dies

[28] Eine Rolle ist erforderlich, weil mehrere verschiedene Attribute als Spezialisierung davon definiert werden können. Eine solche Spezialisierung wird auch *Differenzierung* genannt (vgl. [Brachman und Schmolze, 1985]), weicht aber von der Differenzierung in SB-ONE ab (vgl. [Kobsa, 1990b]).

gleichbedeutend mit $pre_y^I(d) \subseteq (pre_x \circ R)^I(d)$, was der Semantik einer RVM $pre_y \subseteq (pre_x \circ R)$ entspricht. Abb. 3.5 stellt den Zusammenhang durch eine Grafik dar. Die Überlegung gilt analog auch für Zeile (6) von Tabelle 3.3.

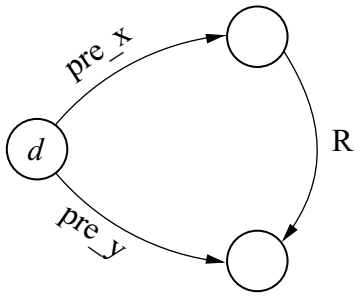

Abbildung 3.5: Relation R^I zwischen Rollenfüllern von *pre_x* und *pre_y*

Im vorangegangenen Abschnitt wurde darauf hingewiesen, daß die uneingeschränkte Verwendung von RVMs zur Unentscheidbarkeit des Subsumptionsproblems führt. In dieser Arbeit werden RVMs allerdings nur eingeschränkt verwendet. Für die in Tabelle 3.3 verwendeten RVMs läßt sich entweder nachweisen, daß sie die Entscheidbarkeit nicht beeinträchtigen, oder sie eignen sich zumindest nicht für den Unentscheidbarkeitsbeweis nach [Schmidt-Schauß, 1989]. In den Zeilen (3) und (4) werden lediglich die Werte von Attributen verglichen; d.h. es handelt sich um Feature Chains der Länge 1, die die Entscheidbarkeit des Subsumptionsproblems nicht beeinträchtigen [Hollunder und Nutt, 1990]. Schmidt-Schauß benötigt für seinen Unentscheidbarkeitsbeweis eine RVM der Form $R \circ R_i = R$, wobei ausgenutzt wird, daß R kein Attribut ist. Die RVMs in Zeile (5) und (6) haben die Struktur $A_1 \subseteq A_2 \circ R$; d.h. eine Rollenkette besteht aus einem einzigen Attribut und die zweite hat die Länge 2, wobei sie mit einem Attribut beginnt und mit einer Rolle endet. Dies ist für den Unentscheidbarkeitsbeweis ebensowenig ausreichend wie die RVM der Form $A \subseteq R$, die in Kap. 3.5.1 *Der Einfluß von Werkzeugen auf Handlungsmöglichkeiten* benötigt wird.

Zusammenfassend sei die beschreibungslogische Repräsentation eines Planoperators durch das unten angegebene Konstruktionsschema definiert, wobei von einer Beschreibung des Planoperators gemäß der Syntax aus Abb. 3.6 ausgegangen wird.

Der geringfügige Unterschied zur Syntax aus Abb. 3.3 ermöglicht, daß die Beschreibungen direkt als Argumente für die Konstruktor-Funktion der LISP-Structure *bgp-ms-plan-struc* genutzt werden können, aus der beim Aufbau der Benutzermodellierungswissensbasis die beschreibungslogische Re-

planstruc-def	=	":name" "'" *planconcept-name*
		[":supers" "'(" {*planconcept-name*}$^+$ ")"]
		[":precondition" "'" *plan-conjunction*]
		[":effect" "'" *plan-conjunction*]
		[":comment" *any-string*] .
plan-conjunction		s. Abb. 3.3 S. 111
name		Name nach LISP-Syntax (wie Abb. 3.3)
any-string		String nach LISP-Syntax (wie Abb. 3.3)

Abbildung 3.6: EBNF-Syntax für die Definition von Plan-Concepts.

präsentation gebildet wird. Hinzu kommt die Möglichkeit zur Angabe von Super-(Plan-)Concepts, womit (Plan-)Concepts – wie in Beschreibungslogiken üblich – als Spezialisierungen von allgemeinen Concepts definiert werden können. Im Beispielsachbereich dieser Arbeit wird dies u.a. für die Beschreibung des Planoperators bzw. Plan-Concepts *ftp-download-file-bin* genutzt, mit dem der Dateitransfer mit einem FTP-Programm im Binärmodus als Spezialisierung eines Dateitransfers, der vom Transfermodus abstrahiert, modelliert wird. (Vgl. das Bsp. in Abb. 5.3 S. 247 im Kapitel 5.2.1 *Vorgehensweise*, das auch die Erzeugung der LISP-Structure zeigt). Die explizite Angabe des allgemeinsten Plan-Concepts *plan-concept* als Super-Concept ist nicht erforderlich.

Konstruktionsanweisung 3.1 ***Beschreibungslogische Darstellung eines Planoperators***
Die Beschreibung des Plan-Concepts, das einen Planoperator repräsentiert, besteht aus einer Konjunktion:

a) (mindestens) eines allgemeineren Plan-Concepts.

b) je eines Vorbedingungs- bzw. Effekt-Attributs mit Werterestriktion *domain-concept* für jede freie Variable (Parameter), in der Vorbedingung bzw. im Effekt des Planoperators.

c) je einer beschreibungslogischen Entsprechung (gem. Tab. 3.3) für jede Vorbedingungs- und jede Effektkomponente des Planoperators.

Enthält eine daraus resultierende Beschreibung mehrere Werterestriktionen für das gleiche Attribut, so werden sie zu einer logisch äquivalenten Attributbeschreibung zusammengefaßt.
Redundante Faktoren – insbesondere solche, die sich durch Vererbung ergeben – müssen für die beschreibungslogische Darstellung nicht explizit angegeben werden.

Durch das Plan-Concept, das in Schritt a) zur Beschreibung hinzugefügt wird, kann gewährleistet werden, daß alle Plan-Concepts eine Spezialisierung des allgemeinsten Plan-Concepts *plan-concept* sind und damit von den Objekten des Sachbereichs (die Spezialisierungen von *domain-concept* sind) unterschieden werden können. Um auszudrücken, daß der Planoperator bzw. das Plan-Concept eine Spezialisierung bereits zuvor definierter Plan-Concepts ist, können außer bzw. anstelle des allgemeinsten Plan-Concepts auch andere Plan-Concepts als Super-Concepts angegeben werden.

Nach diesem Konstruktionsschema ergibt sich für das Beispiel des Planoperators *open-file-tpx* aus Abb. 3.4 die beschreibungslogische Repräsentation in Abb. 3.7.

$$
\begin{aligned}
\textit{open-file-tpx} \;\doteq\; & \textit{plan-concept} \sqcap \\
& \textit{pre_f} : (\textit{tpx-file} \sqcap \textit{local-fs-object}) \sqcap \textit{pre_app} : (\textit{tpx} \sqcap \textit{launched}) \sqcap \\
& \textit{eff_f} : \textit{domain-concept} \sqcap \textit{eff_app} : \textit{domain-concept} \sqcap \\
& (\textit{eff_f} \subseteq \textit{eff_app} \circ \textit{active})
\end{aligned}
$$

Abbildung 3.7: Plan-Concept *open-file-tpx*

Abb. 3.8 zeigt eine alternative Definition des Plan-Concepts *open-file-tpx* als Spezialisierung eines Plan-Concepts *open-file*, das das Öffnen einer beliebigen Datei (*fs-file*) durch ein beliebiges Anwendungsprogramm (*application*) repräsentiert.

3.2.2.1 Spezifika der Repräsentation in SB-ONE

In dieser Arbeit wird SB-ONE [Kobsa, 1991], [Kobsa, 1990b], [Profitlich, 1990] als Wissensrepräsentationsformalismus eingesetzt. SB-ONE ist einer der Wissensrepräsentationsformalismen, die

$$
\begin{aligned}
\textit{open-file} \;&\doteq\; \textit{plan-concept} \sqcap \\
&\quad \textit{pre_f} : \textit{fs-file} \sqcap \textit{pre_app} : (\textit{application} \sqcap \textit{launched}) \sqcap \\
&\quad \textit{eff_f} : \textit{domain-concept} \sqcap \textit{eff_app} : \textit{domain-concept} \sqcap \\
&\quad (\textit{eff_f} \subseteq \textit{eff_app} \circ \textit{active}) \\
\textit{open-file-tpx} \;&\doteq\; \textit{open-file} \sqcap \\
&\quad \textit{pre_f} : (\textit{tpx-file} \sqcap \textit{local-fs-object}) \sqcap \textit{pre_app} : \textit{tpx}
\end{aligned}
$$

Abbildung 3.8: Plan-Concept *open-file-tpx* als Spezialisierung von *open_file*

das ebenfalls in dieser Arbeit eingesetzte Benutzermodellierungsshellsystem BGP-MS[29] [Kobsa und Pohl, 1995] direkt unterstützt. Er steht in der Tradition "*KL-ONE-artiger*" Wissensrepräsentationsformalismen bzw. -systeme. Wie die meisten solcher Systeme (und insbesondere auch KL-ONE selbst [Brachman und Schmolze, 1985]) wird dabei von einem semantischen Netz mit formaler Semantik ausgegangen. Die Knoten des Netzes repräsentieren Concepts und Rollen, die durch verschiedenartige Kanten verbunden sind. Die Definition von Concepts erfolgt durch den Eintrag von *Concept-Knoten*. Die Beschreibung des Concepts erfolgt (im wesentlichen) durch das Hinzufügen von *ISA-Kanten* und Rollen-Beschreibungen[30], evtl. mit Werte- oder Anzahlrestriktion. Eine ISA-Kante macht eine Subsumptionsbeziehung explizit; d.h. verläuft eine ISA-Kante von Knoten CK_1 zu CK_2 und sind C_1 bzw. C_2 die zu diesen Knoten gehörenden Concepts, so gilt die Subsumptionsbeziehung $C_1 \Rightarrow C_2$. Eine ISA-Kante kann "von Hand" eingefügt werden – also vom Entwickler der Terminologie definiert werden. Sie entspricht dann einem Statement $C_1 \doteq C_2 \sqcap C_x$ bzw. $C_1 \sqsubseteq C_2 \sqcap C_x$, je nachdem, ob C_1 als *definiertes* oder *primitives* Concept vereinbart wird. C_x steht dabei für die weiteren Teile der Beschreibung von CK_1; d.h. alle weiteren Beschreibungen zu CK_1 werden als konjunktiv verknüpft aufgefaßt. Viele Wissensrepräsentationssysteme können ISA-Kanten auch von sich aus eintragen, indem ein Concept *klassifiziert* wird. Die *Klassifikation* eines Concepts C stellt die speziellsten Concepts fest, die C subsumieren und fügt dem Netz entsprechende ISA-Kanten hinzu. Oft wird ein Concept als allgemeinstes Concept ausgezeichnet und gefordert, daß von allen Knoten, die zu anderen Concepts gehören, zumindest *eine* ISA-Kante ausgehen muß. In SB-ONE heißt das allgemeinste Concept bspw. *THING*. Da so (außer *THING*) jedes Concept als Spezialisierung definiert ist, spricht man auch von einer *Abstraktionshierarchie* oder – indem die Subsumption als bestimmende Relation vorausgesetzt wird – von einer *Concept-* oder *Begriffshierarchie*.

Weitere Kanten verbinden Concept-Knoten mit Rollen-Knoten (*DA-Kante*) und umgekehrt (*VR-Kante*). Dabei wird das Concept, das zu dem Knoten gehört auf das eine VR-Kante zeigt, als Werterestriktion der zugehörigen *Rolle an dem Concept* aufgefaßt, von dessen Knoten eine DA-Kante ausgeht. Die Definition einer Rolle R mit Werterestriktion CV (Schreibw. VR(R, CV)) an einem Concept C entspricht dabei einer Komponente $\forall R : CV$ der Beschreibung von C. Die Komponente ist mit allen anderen Beschreibungs-Komponenten – zu denen auch die ISA-Kanten gehören – konjunktiv verknüpft. Zudem kann für jede *Rolle an einem Concept* eine Anzahlrestriktion angegeben werden. Dadurch wird definiert, daß jedes Exemplar des Concepts die angegebene Mindest- bzw. Maximalzahl von Füllern der Rolle hat. Eine Anzahlrestriktion [*min, max*] entspricht für eine *notwendige* (*necessary*) Rolle R an C dem KRSS-Ausdruck $(\geq \textit{min}\ R) \sqcap (\leq \textit{max}\ R)$. Darüber hinaus läßt SB-ONE die Definition *optionaler* Rollen zu. Bei einer optionalen Rolle R an C entspricht die Anzahlrestriktion [*min, max*] dem KRSS-Ausdruck $((\geq \textit{min}\ R) \sqcap (\leq \textit{max}\ R)) \sqcup (\leq 0\ R)$ bzw. $(\leq \textit{max}\ R)$ falls $\textit{min} = 1$.

[29] vgl. dazu auch Kap. 3.3 *Struktur eines Benutzermodells mit Handlungspotential und Wissen*

[30] Rollen werden in [Kobsa, 1991] als *Attribute-Descriptions* bezeichnet. Dies darf nicht mit den Attributen nach KRSS verwechselt werden.

Neben einer grafischen Oberfläche zur direkten Manipulation eines solchen Netzwerks durch den Entwickler des Sachbereichsmodells, sieht SB-ONE auch eine funktionale Schnittstelle vor, die in dieser Arbeit ausschließlich genutzt wird. Damit lassen sich bspw. Concepts als Sub-Concepts anderer Concepts definieren, wie dies für Schritt a) des Konstruktionsschemas für Plan-Concepts aus dem vorangegangenen Abschnitt benötigt wird. Weitere Schnittstellenfunktionen stehen für die Definition von Rollen, ggf. mit Werte- und/oder Anzahlrestriktionen sowie für Role-Value-Maps zur Verfügung. Einige der Konstrukte, die für das Konstruktionsschema benötigt werden, stehen allerdings nicht direkt zur Verfügung und müssen durch logisch äquivalente oder approximierte Beschreibungen ersetzt werden. Im einzelnen sind dies:

1. SB-ONE sieht keine *Attribute* vor.
2. für Werterestriktionen von Rollen sind keine Concept-*Ausdrücke* zulässig.
3. SB-ONE sieht nicht den Vergleichsoperator *Ungleich* ($\neq$) für RVMs vor[31].
4. SB-ONE sieht keine explizite *Negation* vor.

Ein KRSS *Attribut* ist logisch äquivalent zu einer *notwendigen* Rolle mit der Anzahlrestriktion [1,1] in SB-ONE. Solche Rollen seien deshalb im weiteren auch als Attribut bezeichnet. In diesem Sinne kann Schritt b) des Konstruktionsschemas für Plan-Concepts durch Hinzufügen von Attributbeschreibungen realisiert werden.

Concept-Ausdrücke für die Werterestriktion von Rollen sind für die beschreibungslogische Darstellung eines Planoperators notwendig, wenn die Vorbedingung (oder entsprechend der Effekt) mehrere verschiedene einstellige Prädikate mit der gleichen Variable als Argument enthält. Gemäß Schritt c) (1) des Konstruktionsschemas sind dann mehrere Werterestriktionen des gleichen Attributs erforderlich, die dann zu einer äquivalenten Attributbeschreibung zusammengefaßt werden. Die resultierende Werterestriktion ist dann eine Konjunktion von Concept-Namen. Eine solche Konjunktion von Concept-Namen $C_1 \ldots C_n$ läßt sich in SB-ONE durch ein definiertes Concept C ausdrücken, für das $C_1 \ldots C_n$ als Super-Concepts bestimmt werden. Dies entspricht dem KRSS-Statement $C \doteq C_1 \sqcap \ldots \sqcap C_n$. Für das Attribut wird C dann als Werterestriktion bestimmt. Wird das Concept C nur zur Realisierung des Concept-Ausdrucks definiert, wird es markiert, sodaß es bei der weiteren Verwendung – insbesondere bei der Erzeugung eines Planungssachbereichs (vgl. Kap. 3.4.2.7 *Erzeugung des aktiven Teils eines Planungssachbereichs*) – wieder in seine Bestandteile zerlegt werden kann.

Zeile (4) aus Tabelle 3.3 benötigt den Vergleichsoperator *Ungleich* ($\neq$) für RVMs, der in SB-ONE nicht zur Verfügung steht. Da es sich bei den Rollenketten um Feature Chains (der Länge 1) handelt, sind die beiden zu vergleichenden Mengen einelementig. Für das Ergebnis der RVM macht es deshalb keinen Unterschied, ob der Vergleichsoperator "$\neq$" oder der in SB-ONE zur Verfügung stehende Vergleichsoperator "$\otimes$" (disjunkt) verwendet wird.

Für die Realisierung des Konstruktionsschemas in SB-ONE verbleibt die Substitution der fehlenden Negation. Dabei können mehrere Fälle unterschieden werden.

Zeile (6) von Tabelle 3.3 erfordert die Negation einer RVM mit Vergleichsoperator "$\subseteq$". Diese Negation tritt aber nur genau in der Form $\neg(A_1 \subseteq A_2 \circ R)$ auf. Da A_1 ein Attribut ist, gibt es zu jedem Exemplar des Plan-Concepts genau einen Füller A_1. Dieser ist genau dann *nicht* in der Menge der Füller von $A_2 \circ R$ enthalten, wenn die beiden Füllermengen kein gemeinsames Element haben. Anstelle von $\neg(A_1 \subseteq A_2 \circ R)$ kann demnach die in SB-ONE zur Verfügung stehende RVM$(A_1 \otimes A_2 \circ R)$ verwendet werden.

[31] vgl. [Kobsa, 1991], [Profitlich, 1990, S.112]; als Sybmol bzw. Abkürzung wird dennoch teilw. "$\neq$" bzw. "neq" für "disjunkt" verwendet.

Die Negation von Concepts, wie sie Punkt (2) aus Tabelle 3.3 des Konstruktionsschemas für Plan-Concepts vorsieht, kann in SB-ONE nicht direkt ausgedrückt werden. Allerdings kann definiert werden, daß eine Menge $C_1, \ldots C_n$ von Concepts paarweise disjunkt ist. Dies bedeutet, daß kein Element aus Δ^I Exemplar von mehr als einem dieser Concepts sein darf; d.h. die Extensionen aller dieser Concepts sind disjunkt. Aus $d \in C_i^I$ folgt dann $\neg(d \in C_j^I)$ für alle $i, j \in \{1 \ldots n\}, i \neq j$; d.h. $C_i \Rightarrow \neg C_j$. Die Negation eines Concepts C_j läßt sich damit durch ein zu C_j disjunktes Concept annähern. Natürlich gilt nicht die Gleichheit von C_i und $\neg C_j$; d.h. eine Schlußfolgerung von $\neg C_j$ auf C_i wäre unzulässig. Dies ist (eingeschränkt) möglich, wenn ein Concept C in *genau zwei* Sub-Concepts C_0, C_1 partitioniert wird. Dabei sei die Semantik einer *Partitionierung eines Concepts* in speziellere Sub-Concepts dadurch definiert, daß die Extensionen der Sub-Concepts eine Partitionierung der Extension des allgemeineren Concepts darstellen. In SB-ONE läßt sich eine Partitionierung eines Concepts C in Sub-Concepts $C_1, \ldots C_n$ ausdrücken, indem $C_1, \ldots C_n$ als disjunkt und als *Überdeckung* von C definiert werden. Wird ein Concept C in genau zwei Sub-Concepts C_0, C_1 partitioniert, sind C_0, C_1 bezüglich C *komplementär* ; d.h. für jedes Exemplar d von C gilt sowohl $d \in C_i^I \rightarrow \neg(d \in C_{(i+1)mod2}^I)$ als auch $\neg(d \in C_i^I) \rightarrow d \in C_{(i+1)mod2}^I$ für $i \in \{0, 1\}$; d.h. $C_0 = \neg C_1 \sqcap C$ und $C_1 = \neg C_0 \sqcap C$. Für Exemplare von C kann also sowohl von C_0 auf $\neg C_1$ als auch von $\neg C_0$ auf C_1 geschlossen werden.

Anstelle einer (in SB-ONE unzulässigen) Verwendung von Concept-Ausdrücken mit Negation wird für die Realisierung von Punkt (2) aus Tabelle 3.3 von diesen Möglichkeiten Gebrauch gemacht die Negation implizit auszudrücken. Bei Planoperatoren wird die Negation einstelliger Prädikate in folgenden Fällen benötigt:

1. eine Objekt-Eigenschaft, die zwei Werte annehmen kann, wechselt ihren Wert.
2. eine Objekt-Eigenschaft, die mehr als zwei Werte annehmen kann, bekommt einen neuen Wert und verliert damit den alten.
3. ein Objekt wird erzeugt oder vernichtet.

Charakteristisch für die ersten beiden Fälle ist, daß eine Eigenschaft durch die Ausführung des Planoperators einen neuen Wert bekommt und *dadurch* den alten verliert. Ersteres wird in der prädikatenlogischen Repräsentation des Planungssachbereichs (meist) durch eine nichtnegierte Atomformel im Effekt des Planoperators ausgedrückt, während der Verlust eines Wertes (meist) durch eine negierte Atomformel ausgedrückt wird. Ähnlich verhält es sich im dritten Fall. Bei einer Objekterzeugung[32] bekommt ein *potentielles Objekt* den Status *Objekt* und verliert *dadurch* den Status eines *nicht-existierenden Objekts*; bei einer Vernichtung ist es umgekehrt. In allen drei Fällen enthält der Effekt sowohl nichtnegierte als auch negierte Komponenten, die voneinander abhängen. Wegen dieser Abhängigkeit ist eine explizite Repräsentation aller derartiger Komponenten redundant. Für eine Beschreibung eines Planoperators reicht es aus, die nichtnegierte Komponente des Effekts explizit zu repräsentieren, wenn sich die Wirkung des Planoperators, die sich aus den negierten Komponenten ergibt, logisch folgern läßt. Dies läßt sich für die ersten beiden Fälle erreichen, wenn die Concepts, die die Eigenschaftswerte repräsentieren, disjunkt definiert werden. Für dem dritten Fall werden *Objekte* und *nicht-existierende Objekte* disjunkt definiert. Die Verwendung negierter Concepts, wie es für Punkt (2) aus Tabelle 3.3 notwendig wäre, kann damit also vermieden werden. Allerdings muß bei der Verwendung der Plan-Concepts bzw. Planoperatoren sichergestellt werden, daß die notwendigen Schlußfolgerungen auch tatsächlich gezogen werden. Dies betrifft insbesondere die Planerzeugung, da Planungssysteme selbst normalerweise nur über eingeschränkte logische Schlußfolgerungsfähigkeiten verfügen. Kap. 3.4.2 *Der Planungssachbereich* geht im Abschnitt 3.4.2.3 noch näher auf darauf ein und behandelt dann auch die Frage, wie sichergestellt werden kann, daß bei der Plangenerierung die notwendigen Schlußfolgerungen gezogen werden können.

[32] vgl. Kap. 3.4.2.5 S. 153

Kann eine Objekt-Eigenschaft *genau zwei* Werte annehmen, ist es manchmal wünschenswert dafür nur *einen* Begriff zu definieren, der dann negiert und nichtnegiert verwendet wird. Zwar müssen in SB-ONE Concepts für beide Möglichkeiten definiert werden, um die Negation wie oben beschrieben zu modellieren. Eines davon kann allerdings – vergleichbar zu den Concepts zur Substitution von Concept-Ausdrücken (s.o.) – markiert werden. Damit wird für weitere Verarbeitungsschritte erkennbar, daß das Concept weniger als eigenständiger Begriff sondern vielmehr als Komplement zu einem anderen Concept aufzufassen ist. Im Beispielsachbereich wurde dies u.a. für die Eigenschaft, ob ein Verzeichnis (für Mount-Operationen) freigegeben ist, genutzt.

Abschließend sei in Abb. 3.9 das Beispiel des Planoperators *open-file-tpx* aus Abb. 3.4 in graphischer SB-ONE Darstellung nach [Profitlich, 1990][33] gezeigt. Die Concepts, die mit "*tpx-file* ⊓ *local-fs-object*" bzw. "*tpx* ⊓ *launched*" bezeichnet sind, wurden als gemeinsame Sub-Concepts von *tpx-file* und *local-fs-object* bzw. von *tpx* und *launched* definiert, weil SB-ONE keine Concept-Ausdrücke für die Werterestriktion von Rollen zuläßt (s.o.). Um die Übersichtlichkeit nicht zu beeinträchtigen sind die dazu erforderlichen ISA-Kanten nicht in der Abbildung enthalten. Gleiches gilt für die ISA-Kanten, die von *plan-concept* und *domain-concept* ausgehen.

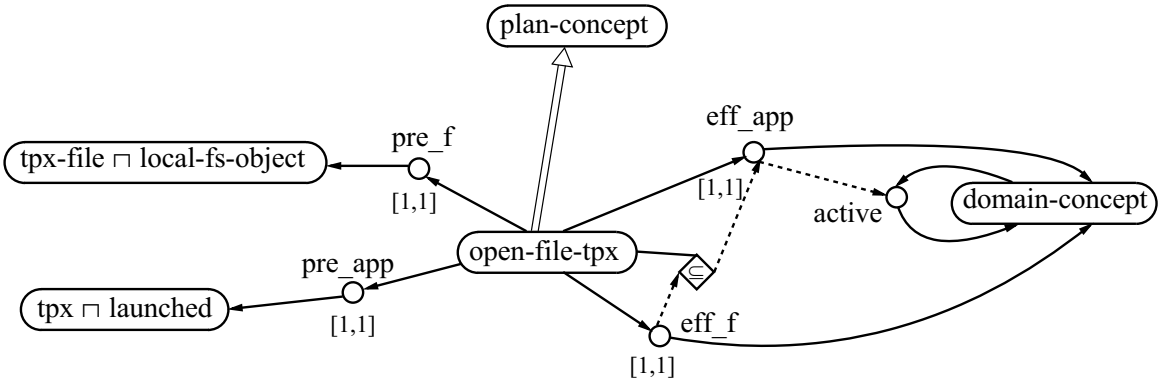

Abbildung 3.9: Graphische SB-ONE Darstellung des Plan-Concepts *open-file-tpx*

3.3 Struktur eines Benutzermodells mit Handlungspotential und Wissen

Zu Beginn dieses Kapitels über die Modellierung- und Plangenerierungsaspekte wurde argumentiert, daß ein Benutzermodell eines Hilfe- oder Unterstützungssystems (kurz: Systems) sowohl ein Modell der Kenntnisse des Benutzers als auch ein Modell seiner Handlungsmöglichkeiten enthalten muß. Es wurde gezeigt, daß beide Teilmodelle widersprüchliche Informationen enthalten können und strikt getrennt werden müssen. Die Inhalte der Teilmodelle werden durch das System bestimmt und repräsentieren damit die Annahmen des Systems *über* die Kenntnisse und Handlungsmöglichkeiten des Benutzers. Diese können sich wiederum von den eigenen Kenntnissen des Systems – in der Sprechweise von [Kobsa, 1989] den *einfachen Annahmen* (*Basic Beliefs*) des Systems – unterscheiden und müssen auch davon getrennt werden. Diese Annahmen des Systems sind keine *Aspekte des Benutzers* und gehören damit – streng nach der Definition aus [Wahlster und Kobsa, 1989] – nicht zum Benutzermodell. Gleiches gilt für die Annahmen über *typische* Benutzer oder Benutzergruppen, die für den einzelnen Benutzer nicht zutreffen müssen. Die Gesamtheit dieser für die Benutzermodellierung

[33] Eine Zusammenfassung der graphischen Darstellung findet sich auch im Anhang A.3 dieser Arbeit.

eingesetzten Annahmen sei im weiteren als *Benutzermodellierungswissensbasis* (*BMWB*) bezeichnet (vgl. [Pohl, 1997]).

Wenngleich die genannten Komponenten einer solchen Benutzermodellierungswissensbasis getrennt werden müssen, sind sie keineswegs unabhängig voneinander. Dieser Abschnitt befaßt sich deshalb mit dem Modell einer solchen Trennung, aber auch damit, wie der Zusammenhang der einzelnen Teile gewährleistet wird. Es wird zunächst ein kurzer Überblick über die wichtigsten Ideen zur Akteurmodellierung gegeben. Es folgt die Behandlung der Repräsentation eines Modells eines einzelnen Benutzers und danach eines Modells, das auch die Annahmen über *typische* Benutzer oder Benutzergruppen beinhaltet. Abschließend wird noch darauf eingegangen, inwieweit Annahmen des Systems oder des Benutzers über die Handlungsmöglichkeiten anderer Akteure genutzt werden können und welchen Einfluß sie auf die Repräsentation der BMWB haben.

3.3.1 Akteurmodelle

Für die Repräsentation verschiedener Arten von Wissen verschiedener Akteure sind eine ganze Reihe unterschiedlicher Formalismen entwickelt worden. In diesem Abschnitt sei lediglich kurz auf die grundlegenden Ideen eingegangen, die dem in dieser Arbeit eingesetzten Ansatz zugrunde liegen. Für eine ausführliche Behandlung und weitere Referenzen sei auf [Kobsa, 1985] und [Taylor *et al.*, 1996] verwiesen. Die Arbeiten können grob in *modallogische* und *partitionsbasierte* Ansätze eingeteilt werden.

In modallogischen Ansätzen werden für die verschiedenen Arten des Wissens Modaloperatoren definiert, bspw. B_A für die *Annahmen* (*Beliefs*), W_A für die *Ziele* (*Wants*) eines Akteurs A. Beispiele für modallogische Formeln, die Annahmen über einen Sachverhalt repräsentieren, der sich durch die logische Formel p beschreiben läßt, sind:

$B_S\ p$	für: das System (Akteur S) glaubt p
$B_S W_U\ p$	für: das System glaubt, daß der Benutzer (Akteur U) das Ziel p hat

Durch modallogische Formeln lassen sich die verschiedenen Teile des Wissens in Beziehung setzen. Bspw. kann durch $B_S\ p \rightarrow B_S \neg W_U\ q$ ausgedrückt werden, daß das System annimmt, daß der Benutzer nicht das Ziel hat q zu erreichen, wenn es annimmt, daß p gilt.

Das gesamte Wissen über alle Akteure wird bei einem solchen Ansatz durch eine Menge modallogischer Formeln in *einer* Wissensbasis repräsentiert.

Demgegenüber wird das Wissen bei partitionsbasierten Ansätzen in Partitionen untergliedert. Das Ziel dieser Aufteilung ist, für die einzelnen Teile des Wissens (im Vergleich zu Modallogik) einfachere Algorithmen für Schlußfolgerungen und Verarbeitung des Wissens einsetzen zu können. Die Idee geht auf Hendrix zurück, der für die Erweiterung der Ausdrucksfähigkeit einer auf *Semantischen Netzen* basierenden Wissensrepräsentation *Partitionierte Semantische Netze* vorschlug [Hendrix, 1975]. Cohen [Cohen, 1978] setzt Wissens-Partitionen zur Repräsentation von Annahmen und Zielen von Akteuren ein. Dabei geht er von einer Interaktion zwischen einem System und einem Benutzer aus. Die Akteure sind demnach *System* (S) und *Benutzer* (U) und das Modell stellt die Sicht des Systems dar. Um auszudrücken, daß ein Akteur Annahmen über Annahmen oder Ziele eines (evtl.) anderen Akteurs hat, können Partitionen geschachtelt werden. Da das gesamte Modell die Ansicht des Systems darstellt, ist die Partition für die Annahmen des Systems (SB) die äußere Partition, die alle anderen enthält. Abb. 3.10 zeigt Cohens Beispiel (Fig.4.2b), in dem zu den Systemannahmen die Annahmen über die eigenen Ziele (SBSW) und diejenigen über die Annahmen des Benutzers (SBUB) gehören. In SBUB sind wiederum die Partitionen für die Systemannahmen über die Annahmen des Benutzers über seine eigenen Ziele (SBUBUW) und die Annahmen des Systems (SBUBSB) enthalten. Die Schachtelungsstruktur wird zum einen durch das Benamungsschema der Partitionen durch

eine Folge von <Akteur>-<Modalität>-Paaren deutlich, wobei die Modalität in Cohens Beispielen entweder “B” (*Belief*) für Annahmen oder “W” (*Want*) für Ziele ist. Diese Notation hat inzwischen – in dieser oder verallgemeinerter Form – eine große Verbreitung gefunden und wird auch dieser Arbeit eingesetzt. Die Bedeutung einer Partition ergibt sich allerdings durch ein Netzwerk, das u.a. spezielle WANT- und BELIEF-Knoten vorsieht. Diese Knoten haben jeweils eine Agent-Kante (agt) zu einem der beiden Akteure und eine Object-Kante (obj) zu einer Partition. Damit wird ausgedrückt, daß die Partition auf die die Object-Kante zeigt (je nach Knotentyp) die Ziele bzw. die Annahmen des Akteurs beschreibt, auf den die Agent-Kante zeigt. Aus welcher Sicht diese Ziele oder Annahmen beschrieben werden wird dadurch bestimmt, in welcher Partition sich der zugehörige WANT bzw. BELIEF-Knoten befindet. Im Beispiel enthält die äußere Partition SB einen WANT-Knoten, der Kanten zu S und zu SBSW hat und drückt damit aus, daß SBSW die Ziele des Systems aus Sicht (also gemäß der Annahmen) des Systems repräsentiert. SB enthält auch einen BELIEF-Knoten, der nicht wie die anderen eine Object-Kante zu inneren Partitionen hat, sondern zu der Partition SB, in der der Knoten selbst enthalten ist. Damit werden reflexive Annahmen ausgedrückt, d.h. daß das System auch weiß, was es weiß. Eine Aussage, die in SB enthalten ist, gilt demnach ebenfalls für SBSB.

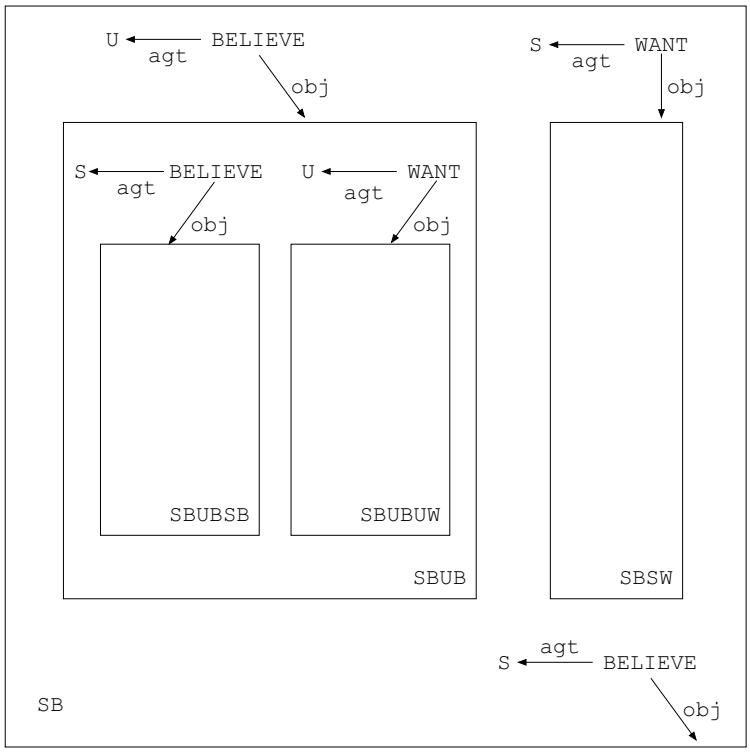

Abbildung 3.10: Geschachtelte Kontexte zur Repräsentation von Systemannahmen (aus [Cohen, 1978, Fig.4.2b])

Im Vergleich zum modallogischen Ansatz liegt der Vorteil des Partitionsansatzes vor allem in der Trennung von Partitionsstruktur und Partitionsinhalt. Das heißt, daß innerhalb einer Partition (abgesehen von den WANT- und BELIEF-Knoten) ein einfacherer Formalismus verwendet werden kann, der nicht zu berücksichtigen braucht, daß es sich bei dem repräsentierten Wissen um Annahmen oder Ziele eines bestimmten Akteurs handelt. Damit ist für Partitionsinhalte eine an das Anwendungsgebiet angepaßte Repräsentation möglich und es können spezielle Verarbeitungsalgorithmen eingesetzt

werden, die effizienter arbeiten können als allgemeine modallogische Schlußfolgerungsmechanismen. Dem stehen als Nachteile eine geringere Ausdrucksstärke und das Fehlen einer formalen Semantik gegenüber (vgl. bspw. [Hustadt, 1995]).

Kobsa vergleicht in [Kobsa, 1985] die Stärken und Schwächen verschiedener Ansätze und kommt dabei zu dem Schluß, daß eine Wissensrepräsentation auf Basis des Partitionsansatzes für die Benutzermodellierung vorteilhaft ist. Er entwickelt auf dieser Basis eine Repräsentation für das Benutzermodell des natürlichsprachlichen Dialogsystem VIE-LANG [Buchberger *et al.*, 1982], wobei die Partitionen allerdings nicht wie bei Cohen geschachtelt, sondern durch spezielle Kanten verbunden sind.

In dieser Arbeit wird für die Repräsentation der verschiedenen Teile des Benutzermodells das Benutzermodellierungsshellsystem BGP-MS [Kobsa und Pohl, 1995] eingesetzt, das ebenfalls den Partitionsansatz nutzt. Auch hierbei sind die Partitionen nicht geschachtelt. Der Zusammenhang zwischen den einfachen Annahmen des Systems und den Annahmen über die Annahmen oder Ziele eines anderen Akteurs, werden damit nicht explizit repräsentiert. Für natürlichsprachliche Dialogsysteme – zu deren Unterstützung BGP-MS ursprünglich entwickelt wurde – ist allerdings ein anderer Zusammenhang wichtig: die Frage, welche Annahmen die am Dialog beteiligten Akteure gemeinsam haben. Wie bereits gesagt, ist die Frage nicht nur für den natürlichsprachlichen Dialog, sondern für jede Kommunikation zwischen Akteuren wichtig, weil dabei eine gemeinsame Auffassung über die verwendete Begrifflichkeit bestehen muß. BGP-MS unterstützt die Repräsentation und Verwaltung solcher gemeinsamer Annahmen durch die Möglichkeit, Partitionen in einer Vererbungshierarchie anzuordnen. Der Inhalt einer Partition steht (durch den Vererbungsmechanismus) allen dazu gehörenden Sub-Partitionen zur Verfügung. Umgekehrt unterstützt BGP-MS auch das Propagieren von Teilen des Inhalts, der in allen Sub-Partitionen einer Partition gleich ist, in die übergeordnete Partition. Abb. 3.11 zeigt als Beispiel ein einfaches Modell gemeinsamer Annahmen. Die Partitionen sind als Rechtecke gezeichnet, die durch Pfeile von Sub- zu Super-Partitionen verbunden sind. Die Partition ShB repräsentiert die Annahmen, die von System und Benutzer geteilt werden. Der Inhalt dieser Partition wird an die Sub-Partitionen SB und SBUB vererbt, die entsprechend der oben vereinbarten Notation das Systemwissen bzw. die Annahmen des Systems über den Benutzer repräsentieren. Durch den Propagationsmechanismus kann erreicht werden, daß ShB auch dann alle gemeinsamen Annahmen von System und Benutzer repräsentiert, wenn sich die Annahmen über das Benutzerwissen – bspw. durch den Dialog oder sonstige Beobachtung – ändern. Der Propagationsmechanismus prüft bei einem neuen Eintrag in eine Sub-Partition (in diesem Falle SBUB), ob der Eintrag auch in allen Geschwister-Partitionen[34] enthalten ist und propagiert ihn im positiven Fall in die direkt übergeordnete Partition. Ein weitergehendes Modell gemeinsamer Annahmen, das ausdrückt, daß die beteiligten Akteure auch gegenseitig wissen, daß sie gemeinsame Annahmen haben (→ *Mutual Beliefs*), läßt sich ebenfalls durch eine Partitionenhierarchie repräsentieren (vgl. bspw. [Kobsa, 1990a]).

Eine technische Besonderheit sei noch kurz erwähnt, weil sie zur Einführung des Begriffs *View* geführt hat. Normalerweise wird nicht der gesamte Inhalt einer Partition (im o.g. Sinne) auch tatsächlich in der Datenstruktur gespeichert, die die Partition realisiert. Um Redundanz und damit die Möglichkeit von Inkonsistenzen zu vermeiden, werden die einzelnen Einträge, die die Annahmen repräsentieren, so allgemein wie möglich abgespeichert. D.h. die Vererbung eines Eintrags wird nicht durch Kopieren in die Sub-Partition realisiert, sondern lediglich der Abfragemechanismus stellt beim Zugriff sicher, daß ererbte Einträge mit berücksichtigt werden. Umgekehrt werden propagierte Einträge nur in der Super-Partition gespeichert und aus den Sub-Partitionen gelöscht. Wenn der Unterschied relevant ist, wird deshalb zwischen der *Partition* als Container der tatsächlich gespeicherten Einträge und der *View* unterschieden. Eine View trägt die gleiche Bezeichnung wie eine Partition und umfaßt

[34] Geschwister-Partitionen einer Partition P sind alle weiteren direkten Sub-Partitionen einer Partition, die P direkt übergeordnet ist. Im Beispiel sind SB und SBUB jeweils gegenseitig Geschwister-Partitionen.

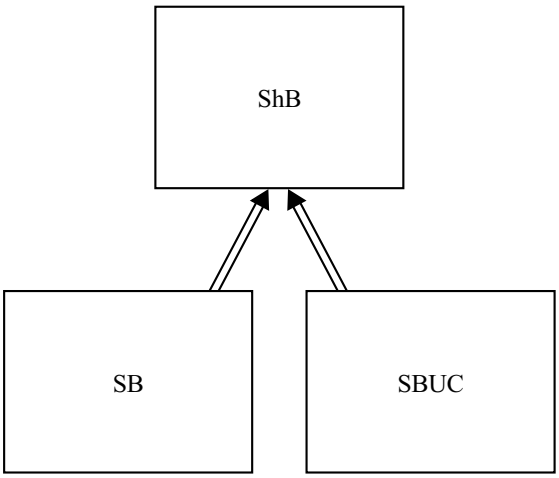

Abbildung 3.11: BGP-MS Partitionenhierarchie, bei der *Shared-Beliefs* (ShB) an die einfachen Systemannahmen (SB) und Systemannahmen über die Benutzerannahmen (SBUB) vererbt werden

alle Einträge dieser Partition und aller Partitionen, die dieser direkt oder indirekt übergeordnet sind. Im Beispiel von Abb. 3.11 enthält die View SB also alle Einträge, die in den Partitionen SB und ShB gespeichert sind.

Das Propagieren von Einträgen ist nicht immer erwünscht und läßt sich deshalb verhindern, indem einer Partition das Attribut *primitive* zugewiesen wird. Ein Beispiel dafür sind Stereotype-Partitionen, die die typischen Annahmen von charakteristischen Benutzergruppen enthalten[35]. Der BGP-MS Vererbungsmechanismus ermöglicht eine besonders einfache Realisierung der Zuordnung eines Benutzers zu einem Stereotype. Dazu wird die entsprechende Stereotype-Partition als Super-Partition von SBUB definiert. SBUB erbt damit alle Annahmen, die für die Benutzergruppe modelliert sind. Umgekehrt soll normalerweise verhindert werden, daß Inhalte vom Modell eines speziellen Benutzers an das einer Benutzergruppe propagiert werden. Stereotype-Partitionen erhalten damit immer das Attribut *primitive*. Dieser Mechanismus wird auch in dieser Arbeit genutzt – Abb. 3.16 zeigt ein Beispiel.

Allerdings reichten für diese Arbeit Stereotypes, die – wie in [Kobsa und Pohl, 1995] beschrieben – lediglich das *Wissen* oder *Präferenzen* von Benutzergruppen repräsentieren, nicht aus. Es kommt deshalb eine erweiterte Version von BGP-MS zum Einsatz, die in [Pohl, 1997] beschrieben ist und als eine Implementation des von Pohl entwickelten *Assumption Type Representation* Formalismus bzw. kurz AsTRa-Formalismus angesehen werden kann. Die Erweiterungen und Verallgemeinerungen sind ein Ergebnis der Arbeit zur Verbindung der Vorteile der modallogischen und partitionsbasierten Ansätze zur Akteurmodellierung. Pohl führt *Assumption Types* (ATs) ein, die – vergleichbar zu Partitionen – Annahmen eines bestimmten Typs über Akteure als abgegrenzte Teile des Gesamtwissens eines Systems modellieren. Ein AT besteht aus einem *Label*, der den AT bezeichnet, und dem *Inhalt* – einer Wissensbasis zur Repräsentation der Annahmen. Das Label ist – vergleichbar zu Cohens Notation – eine Folge von Akteur-Modalität-Paaren, wobei als Akteure S, U und M zugelassen sind (vgl. [Pohl, 1997, Def. 5.1]). M steht als gedachter Akteur für gemeinsame Annahmen (*Mutual Beliefs*) von System und Benutzer. Als Modalität ist “B” für Belief vorgegeben, ansonsten können Modalitäten frei definiert werden. Stereotypes werden als Mengen von Wissensbasen, die beliebigen ATs zugeordnet werden können, angesehen.

[35] vgl. [Rich, 1979], [Rich, 1989] für den Einsatz von Stereotypes in der Benutzermodellierung.

Es ist vorgesehen, daß der Inhalt eines ATs durch mehrere Logik-Formalismen repräsentiert wird, um die Stärken verschiedener Formalismen parallel nutzen zu können. In der Implementierung durch BGP-MS entspricht der Inhalt einer *View* dem Inhalt eines *AT*s; beide Begriffe werden im weiteren austauschbar verwendet. Als Formalismen zur Repräsentation des Inhalts stehen die Beschreibungslogik SB-ONE [Kobsa, 1991] und Prädikatenlogik erster Stufe zur Verfügung.

Diese Arbeit folgt Pohls Ansatz, ohne allerdings alle Möglichkeiten auszuschöpfen. An erster Stelle sei die Möglichkeit genannt, ATs für beliebige Wissensarten definieren zu können, die auch ihre Entsprechung bei der Definition von Stereotypes haben. Für die Repräsentation des Handlungspotentials wird die Modalität "C" (als Abkürzung für *Capability*) definiert. Für ATs (Views) bzw. Partitionen, die ein Handlungspotential repräsentieren, seien als bequeme Sprechweise die Begriffe *Fähigkeitsview* bzw. *Fähigkeitspartition* vereinbart. Als Formalismus für die Repräsentation der Inhalte der ATs kommt ausschließlich SB-ONE zum Einsatz. Da das Propagieren von Einträgen für diese Arbeit nicht sinnvoll ist, wird im weiteren grundsätzlich von Partitionen mit dem Attribut *primitive* ausgegangen.

3.3.2 Repräsentation eines Benutzermodells mit Handlungspotential und Wissen im AsTRa-Formalismus

Wir beschränken uns hier zunächst auf ein Modell eines einzelnen Benutzers. Die Ergänzung um Stereotypes folgt im nächsten Abschnitt. Für ein solches Benutzermodell müssen drei Typen von Annahmen differenziert werden:

- Die einfachen Annahmen des Systems (SB)
- Die Annahmen des Systems über die Annahmen des Benutzers (SBUB)
- Die Annahmen des Systems über die Handlungsmöglichkeiten des Benutzers (SBUC)

SB und SBUB repräsentieren begriffliches Wissen und zwar sowohl über die Objekte als auch über die Handlungsmöglichkeiten im Anwendungssachbereich. Wie in Kap. 3.2.2 *Planoperatoren in Beschreibungslogik* vorgeschlagen, werden Handlungsmöglichkeiten durch eine beschreibungslogische Repräsentation von Planoperatoren (den *Plan-Concepts*) modelliert. Die Beschreibung der Plan-Concepts basiert auf Beschreibungen der Objekte, die durch Ausführen der zugehörigen Planoperatoren verändert werden. Zur Kenntnis der Handlungsmöglichkeiten ist demnach die Kenntnis der beteiligten Objekte bzw. Objektklassen (den *Domain-Concepts*) unverzichtbar. Bspw. ist die Kenntnis über das *Öffnen einer Datei in einem Textverarbeitungssystem des Typs tpx* wie in Kap. 3.2.2 beschrieben, ohne gleichzeitige Kenntnis der Begriffe *Textverarbeitungssystem* und *Datei* nicht möglich.

Dies gilt analog auch für die Handlungsmöglichkeiten eines Akteurs, die hier durch den Assumption Type (AT) SBUC modelliert sind und zwar in gleicher Weise wie in SB oder SBUB. Zu beachten ist allerdings, daß für die Beschreibung der Plan-Concepts nicht die Kenntnisse des *modellierten* Akteurs herangezogen werden können, sondern diejenigen des Akteurs, zu dessen Annahmen die Beurteilung der Handlungsmöglichkeiten gehört[36]. Dies gilt nicht nur für die Concepts, die für die Beschreibung der Plan-Concepts eingesetzt werden, sondern auch für Plan-Concepts selbst, denn die Annahmen eines Akteurs A_i über die Handlungsmöglichkeiten eines anderen Akteurs A_j müssen konform zu A_is Annahmen darüber sein, welche Handlungen in dem Sachbereich überhaupt möglich sind. Generell gilt also, daß alle Beschreibungen von Plan-Concepts und alle dazu notwendigen

[36] Sofern es sich um verschiedene Akteure handelt. Die folgenden Überlegungen gelten allerdings auch dann, wenn ein Akteur Annahmen über seine eigenen Handlungsmöglichkeiten hat. Die im weiteren verwendeten Indizes für den modellierten und den beobachtenden Akteur dürfen also durchaus gleich sein.

Domain-Concepts eines ATs {XB}*A_iBA_jC, auch Teil von {XB}*A_iB sind. Dabei steht {XB}* für eine beliebige, auch leere Folge von Akteur-Modalität-Paaren, wobei die Modalität ausschließlich "B" sein darf[37]. Im Falle von SBUC ist {XB}* leer, A_i=S und A_j=U.

Domain-Concepts dienen in einer Fähigkeitspartition ausschließlich dazu, Plan-Concepts näher zu beschreiben. Eine Fähigkeitspartition muß also alle für die Beschreibung der Plan-Concepts notwendigen Domain-Concepts enthalten. Zusätzlich darf eine Fähigkeitspartition weitere, zu *plan-concept* disjunkte Concepts enthalten, weil ein Modell von Handlungsmöglichkeiten nur auf den Plan-Concepts beruht. Da das in einer Fähigkeitspartition enthaltene Wissen konform zum Wissen des beobachtenden Akteurs sein muß, sei vereinbart, daß sich die entsprechenden ATs – allgemein {XB}*A_iBA_jC und {XB}*A_iB – nur bezüglich der Plan-Concepts unterscheiden dürfen. Zwar wird eine Fähigkeitspartition dann i.a. mehr (Domain-) Concepts enthalten als unbedingt notwendig, es entfällt aber die Notwendigkeit, die tatsächlich erforderlichen Domain-Concepts zu ermitteln. Stattdessen reicht es aus, das objektbezogene Wissen eines Akteurs einmal in einer eigenen Partition zu modellieren. Es kann dann sowohl für das Modell des Planungswissens dieses Akteurs als auch für das Modell seiner Annahmen über die Handlungsmöglichkeiten anderer Akteure genutzt werden. Unter objektbezogenem Wissen eines Akteurs wird dabei sein Wissen ohne die Kenntnisse über Handlungen verstanden – im Modell also sein Wissen ohne Plan-Concepts. Eine Partition, die (ggf. Annahmen über) objektbezogenes Wissen eines Akteurs A_i repräsentiert, sei mit {XB}*A_i^{Dom}B bezeichnet[38]. Wird sie als Super-Partition von {XB}*A_iBA_jC und {XB}*A_iB definiert, stellt der Vererbungsmechanismus von BGP-MS den Inhalt auch diesen Partitionen zur Verfügung. Abb. 3.12 zeigt eine solche Partitions-Struktur sowohl für den allgemeinen Fall als auch konkret für die einfachen objektbezogenen Systemannahmen (S^{Dom}B), die gesamten (einfachen) Systemannahmen (SB) und die Systemannahmen über die Handlungsmöglichkeiten des Benutzers (SBUC). Es sei angemerkt, daß die Modellierung des objektbezogenen Wissens in einer eigenen Partition natürlich nur dann vorteilhaft ist, wenn das Wissen mehrfach benötigt wird – insbesondere, wenn für den modellierten Akteur auch seine Annahmen über das Handlungspotential anderer Akteure berücksichtigt werden.

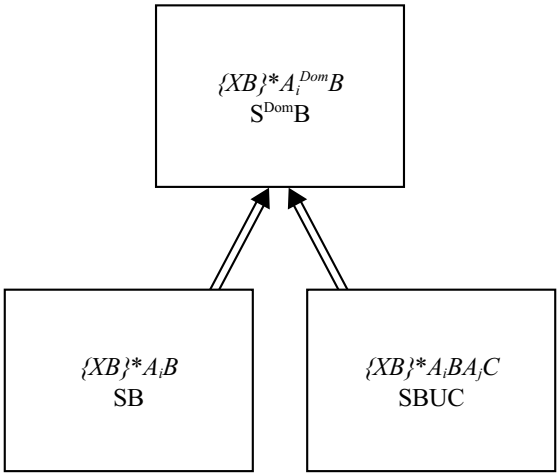

Abbildung 3.12: Partitionenhierarchie mit einer eigenen Partition {XB}*A_i^{Dom}B für die objektbezogenen Annahmen eines Akteurs

[37] vgl. den folgenden Abschnitt über geschachtelte Annahmen über Handlungsmöglichkeiten.

[38] Der hochgestellte Präfix *Dom* vor der Bezeichung B für eine Wissenspartition soll ausdrücken, daß alle Concepts dieser Partition *Dom*ain-Concepts sind.

Es liegt nahe zu versuchen, den Vererbungsmechanismus auch zu nutzen, um die o.g. Konformität der einfachen Annahmen eines Akteurs mit denen über die Handlungsmöglichkeiten eines anderen Akteurs im Modell zu gewährleisten. Ein Ansatz wäre, die Partition für die einfachen Annahmen eines Akteurs ({XB}*A_iB) als Sub-Partition der Partition(en) für die Annahmen über die Handlungsmöglichkeiten anderer Akteure ({XB}*A_iBA_jC) zu definieren. Die Einträge dieser Fähigkeitspartitionen stünden dann automatisch auch in der Partition für die einfachen Annahmen des Akteurs zur Verfügung.

Das folgende Beispiel mit Systemannahmen über die Handlungsmöglichkeiten von zwei[39] Akteuren A_1 und A_2 zeigt allerdings, daß dies nicht zu dem gewünschten Ergebnis führt. Es würde sich die Partitionsstruktur nach Abb. 3.13 ergeben. In dem Beispiel kennt das System den Begriff *Textverarbeitungssystem* (*tp*) mit den beiden disjunkten Spezialisierungen *tpx* und *tpy*. Die Begriffe werden an alle anderen Partitionen vererbt, was aus Gründen der Übersichtlichkeit in der Abbildung nicht dargestellt ist. Der Partitionsmechanismus von BGP-MS stellt sicher, daß Concepts mit gleichem Namen in allen Partitionen gleich bezeichnet werden; d.h. daß sie den gleichen *Bezeichner* (*Identifier*) haben. Das heißt aber nicht, daß sie im beschreibungslogischen Sinne gleich sein müssen – sie können durchaus unterschiedliche Beschreibungen und damit unterschiedliche Extensionen haben. Durch gleich bezeichnete Concepts, die aber in verschiedenen Belief-Partitionen unterschiedliche Extensionen haben können, lassen sich die (mglw.) unterschiedlichen Ansichten verschiedener Akteure über *einen* Begriff modellieren. Concepts, die die Ansichten der verschiedenen Akteure über *einen* Begriff modellieren, werden durch die Concept-Bezeichnungen identifiziert. Zu Beginn dieses Kapitels wurde eine (in den wesentlichen Aspekten) übereinstimmende Begrifflichkeit kommunizierender Akteure gefordert. Dies kann anhand des Benutzermodells (bzw. Akteurmodells) überprüft werden, indem für jeden an einem Kommunikationsakt beteiligten Begriff festgestellt wird, inwieweit die dazu gehörenden Concepts übereinstimmen.

Die Identifikation durch die Concept-Bezeichnungen ist auch für Fähigkeitspartitionen relevant. Es kann so bspw. geprüft werden, ob ein Benutzer ein Plan-Concept, das zu seinem Handlungspotential gehört, auch gut genug *kennt*, sodaß der zugehörige Begriff ohne weitere Erläuterungen in einem Ratschlag an den Benutzer verwendet werden kann. Zusätzlich zu den unterschiedlichen Ansichten verschiedener Akteure ist es hier allerdings auch möglich, daß die Akteure eine Handlung auf unterschiedliche Weise ausführen können; daß sich also auch gleich bezeichnete Plan-Concepts in den Fähigkeitspartitionen der verschiedenen Akteure unterscheiden. Abb. 3.13 zeigt dies für eine vereinfachte Version der Handlung *Datei in einem Textverarbeitungssystem öffnen* (*open-file-tp*), die eine Verallgemeinerung der bisher schon verwendeten *Datei in einem Textverarbeitungssystem des Typs tpx zu öffnen* bzgl. des Typs des Textverarbeitungssystems darstellt. Aus Gründen der Übersichtlichkeit stellt die Abbildung die zugehörigen Plan-Concepts nur mit diesem einen Vorbedingungs-Attribut dar. Im Beispiel geht das System davon aus, daß die beiden Akteure A_1 und A_2 die Handlung unter spezifischen Einschränkungen ausführen können – bspw. weil sie Lizenzen für die unterschiedlichen Textverarbeitungssysteme *tpx* bzw. *tpy* haben. Das System, das notwendigerweise diese beiden Varianten kennt, kennt auch die Verallgemeinerung, die in Partition SB dargestellt ist.

Ein Modell, das die Gegebenheit, daß die Plan-Concepts der beiden Fähigkeiten jeweils Spezialisierungen des Plan-Concepts in SB sind, korrekt wiedergibt, ist in Abb. 3.14 dargestellt. Dies widerspricht allerdings der durch den Partitionsmechanismus realisierten Vorstellung, daß gleich bezeichnete Concepts den gleichen Begriff modellieren. Dementsprechend führt die Vererbung von SBA_1C und SBA_2C nach SB zu der unerfüllbaren Beschreibung *plan-concept* $\sqcap$ *pre_app : (tpx* $\sqcap$ *tpy* $\sqcap$ *tp)*, die in Abb. 3.15 dargestellt ist. Eine Möglichkeit, diese falsche Modellierung zu vermeiden, wäre die Erweiterung des Partitionsmechanismus, sodaß die geerbten Concepts wie in Abb. 3.14 dargestellt

[39] Die Berücksichtigung mehrerer Akteure dient hier lediglich der Verdeutlichung; ausführlicher wird in Kap. 3.5 *Das Umfeld des Benutzers: Werkzeuge und andere Akteure* darauf eingegangen.

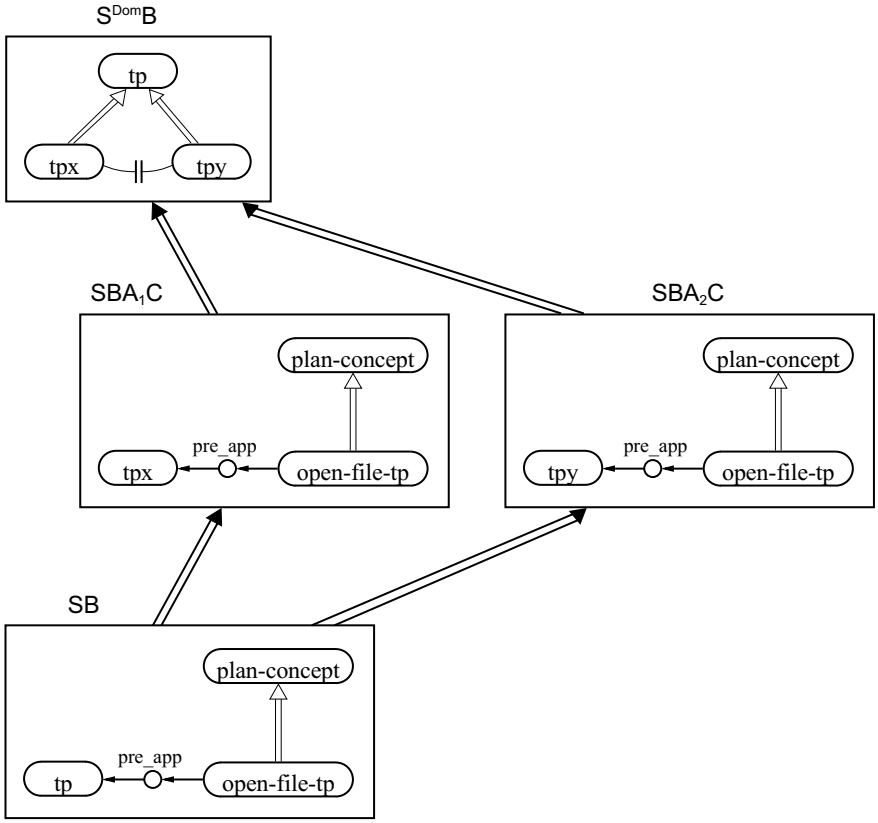

Abbildung 3.13: Partitionenhierarchie, bei der SB Sub-Partition aller SBA_iC Partitionen ist. (Geerbte Inhalte sind nicht dargestellt.)

als neue Sub-Concepts von *open-file-tp* eingetragen werden. Aber abgesehen davon, daß dies keine allgemeine Lösung ist, entspricht ein solches Modell auch nicht der Intention über die Annahmen des Systems.

Es soll hier die Annahme modelliert werden, daß das System die allgemeine Handlung *Datei in einem Textverarbeitungssystem öffnen* kennt, aber gleichzeitig weiß, daß diese Handlung von einigen Akteuren nur eingeschränkt ausgeführt werden kann. Dazu ist es aber nicht notwendig, alle daraus resultierenden Spezialisierungen explizit als neue Concepts zu modellieren. Vielmehr reicht es aus, wenn SB nur die allgemein definierten Concepts enthält. Ob und inwiefern ein Akteur die Handlungen eingeschränkt ausführen kann, läßt sich durch Vergleich mit den Concepts in der View für die Annahmen über die Handlungsmöglichkeiten des Akteurs (im Bsp. SBA_1C bzw. SBA_2C) feststellen. Die weitere Behandlung der Benutzermodell-Struktur geht deshalb von der in Abb. 3.12 gezeigten Partitionenhierarchie aus. Die Partition für die einfachen Annahmen eines Akteurs wird also in Bezug auf die Plan-Concepts unabhängig von den Annahmen des Akteurs über die Handlungsmöglichkeiten anderer Akteure gesehen. Gemeinsamkeiten bestehen lediglich in Hinblick auf das objektbezogene Wissen.

Der Aspekt, daß eine allgemein definierte Handlung durch äußere Einflüsse eingeschränkt wird – im Bsp. die verschiedenen Textverarbeitungssysteme – wird noch in Kap. 3.5 *Das Umfeld des Benutzers: Werkzeuge und andere Akteure* behandelt, wenn es um den Einfluß von Werkzeugen oder anderer Akteure auf die Handlungsmöglichkeiten eines Benutzers geht.

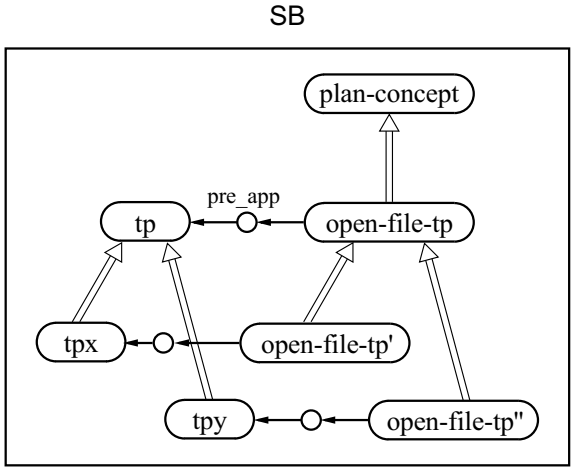

Abbildung 3.14: Wünschenswerter Inhalt von SB bei Partitionenhierarchie nach Abb. 3.13 (ohne Domain-Concepts; Rollenvererbung nicht dargestellt).

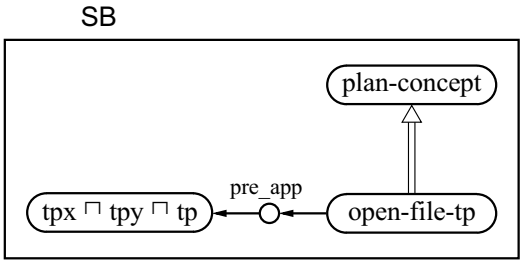

Abbildung 3.15: Geerbter Inhalt von SB bei Partitionenhierarchie nach Abb. 3.13 (ohne Domain-Concepts).

3.3.3 Stereotypes

In der Einführung zu diesem Kapitel wurde als Argument für die Modellierung des Handlungspotentials eines Benutzers durch Begriffe, die Möglichkeit der Nutzung von Forschungsergebnissen zum Aufbau eines Benutzermodells angeführt. Eine Möglichkeit, die schnell – also auch bei einer kurzen Interaktion mit einem Benutzer – zu einem relativ ausführlichen Bild über den Benutzer führt, ist der Einsatz von *Stereotypes* [Rich, 1979]. Dabei wird ein Benutzer a priori oder auf Grund seines Verhaltens nach wenigen Interaktionen einer oder mehreren Benutzergruppen mit typischen Eigenschaften zugeordnet. Die gemeinsamen, typischen Eigenschaften der Benutzergruppe werden durch den Stereotype modelliert. Wird ein Benutzer einer solchen Gruppe zugeordnet, so werden ihm alle typischen Eigenschaften der Gruppe zugeschrieben – zumindest solange die direkte Beobachtung des Benutzers nicht dagegen spricht. Ein inzwischen bereits klassisches Beispiel sind die Benutzergruppen *Anfänger* und *Experte* im Hilfesystem UNIX-Consultant [Wilensky *et al.*, 1988]. Bspw. wird ein Benutzer, der eine Frage zu einem sehr grundlegenden UNIX-Kommando stellt, als Anfänger eingestuft, während Fragen zu komplexen Kommandos zu einer Einstufung als Experte führen. Chin beschreibt in [Chin, 1989] weitere Differenzierungen und geht auch auf die Frage ein, auf Grund welcher Beobachtungen ein Benutzer einer Benutzergruppe zugeordnet werden kann.

In komplexen Anwendungsbereichen sind Benutzer in verschiedenen Teilbereichen oft unterschiedlich kompetent. Dem kann durch Stereotypes Rechnung getragen werden, deren Modellierung

sich auf solche Teilbereiche beschränkt. Ein Benutzer kann dann bspw. in einem Bereich als Experte und in einem anderen als Anfänger eingestuft werden, indem er mehreren Stereotypes zugeordnet wird. Kobsa beschreibt in [Kobsa, 1990a] verschiedene Möglichkeiten von Stereotype-Strukturen und auch den BGP-MS-Mechanismus für die Zuordnung von Benutzern zu Stereotypes – der sogenannten *Stereotype-Triggerung*.

Jeder Stereotype wird in BGP-MS durch eine View modelliert; d.h. Stereotypes können selbst eine Partitionenhierarchie bilden, wodurch bspw. gemeinsame Inhalte nur in der allgemeinsten Partition repräsentiert werden müssen. Wird ein Benutzer einem Stereotype zugeordnet, so wird die dazu gehörende Partition Super-Partition von SBUB, der Partition, die die Annahmen des Systems über die Benutzerkenntnisse repräsentiert. Durch den BGP-MS Vererbungsmechanismus enthält damit die *View* SBUB automatisch auch den Inhalt des Stereotypes. Dem Benutzer werden also alle Kenntnisse zugeschrieben, die durch den Stereotype modelliert sind.

In der in [Kobsa und Pohl, 1995] beschriebenen Version von BGP-MS werden nur Stereotypes unterstützt, die sich auf die Kenntnisse der Benutzer beschränken. Pohl verallgemeinert diese Sicht in [Pohl, 1997, Def.5.2]. Demnach sind Stereotypes Mengen von Paaren aus Wissensbasis und Assumption-Type-Bezeichnungen. Wird ein Benutzer einem Stereotype zugeordnet, so erben die ATs, deren Kennung mit einer der Kennungen des Stereotypes übereinstimmen, die Inhalte der Wissensbasen, die im Stereotype zu dieser Kennung gehören. Für die Anwendung in dieser Arbeit sind vor allem die Kennungen SBUB und SBUC relevant. Stereotypes, die ein Wissensbasis-Kennung-Paar mit Kennung SBUB enthalten, repräsentieren Kenntnisse einer Benutzergruppe, während ein Paar mit Kennung SBUC die typischen Handlungsmöglichkeiten einer Benutzergruppe repräsentiert. Pohl und die Realisierung in BGP-MS überlassen dem Entwickler der Stereotype-Modelle die Entscheidung, ob *ein* Stereotype tatsächlich mehrere Wissensbasis-Kennung-Paare enthalten soll. Das folgende Beispiel wird zeigen, daß sich für die Modellierung typischer Kenntnisse und Handlungsmöglichkeiten unterschiedliche Strukturen für Stereotypes ergeben und deshalb zumeist die Beschränkung eines Stereotypes auf ein Wissensbasis-Kennung-Paar sinnvoll ist.

Das Beispiel geht von folgender Vorstellung aus. Bzgl. der Kenntnisse ergeben sich in dem Sachbereich dieser Arbeit die Teilbereiche *Betriebssystem* (hier konkret UNIX und Windows) sowie *Netzwerk*, in denen die Benutzer unterschiedlich kompetent sein können. Im Beispiel sollen deshalb neben *allgemeinen Mindestkenntnissen*, *UNIX-Kenntnisse*, die *Kenntnisse eines Windows-Experten* sowie *Kenntnisse von Netzwerk-Anfängern* und *-Experten* modelliert werden. Während bei Windows-Kenntnissen lediglich das Experten-Wissen explizit modelliert wird, sind bei Netzwerk-Kenntnissen sowohl Anfänger als auch Experten explizit modelliert. Damit lassen sich auch Fehlannahmen modellieren, die ein Experte üblicherweise *nicht* hat. Abb. 3.16 zeigt die Struktur der Stereotype-Partitionen zusammen mit den fünf gedachten Benutzern *Oscar*, *Trixie*, *Nick*, *Theo* und *Sue*. Die Abbildung zeigt ausschließlich Partitionen, die Kenntnisse modellieren.

In Anlehnung an die Bezeichnung durch eine Folge von Akteur-Modalität-Paaren enden die Namen aller Partitionen mit “B” für *Belief*. Stereotype-Partitionen sind durch fett gedruckte Rechtecke dargestellt. Ihre Bezeichnungen setzen sich aus dem Namen des Stereotypes und der Modalität zusammen, also *All*B für die allgemeinen Mindestkenntnisse, *NetNov*B, *NetExp*B und *WinExp*B für die Kenntnisse von Netzwerk-Anfängern, -Experten bzw. Windows-Experten sowie *Ux*B für typische UNIX-Kenntnisse. Die verbleibenden Partitionen repräsentieren die Annahmen des Systems über die gedachten Benutzer und tragen dementsprechend die Bezeichnungen SB<*Benutzername*>B. Von diesen Partitionen, die hier zu Vergleichszwecken gemeinsam dargestellt sind, berücksichtigt BGP-MS *zu einem Zeitpunkt* nur eine, weil BGP-MS von *einem* ausgezeichneten Benutzer ausgeht. Diese Partition heißt dann jeweils SBUB.

Die dargestellten Benutzer stellen Beispiele von Mitarbeitern dar, die in dem Anwendungsbereich vorkommen können. Dazu ein grobe Charakterisierung:

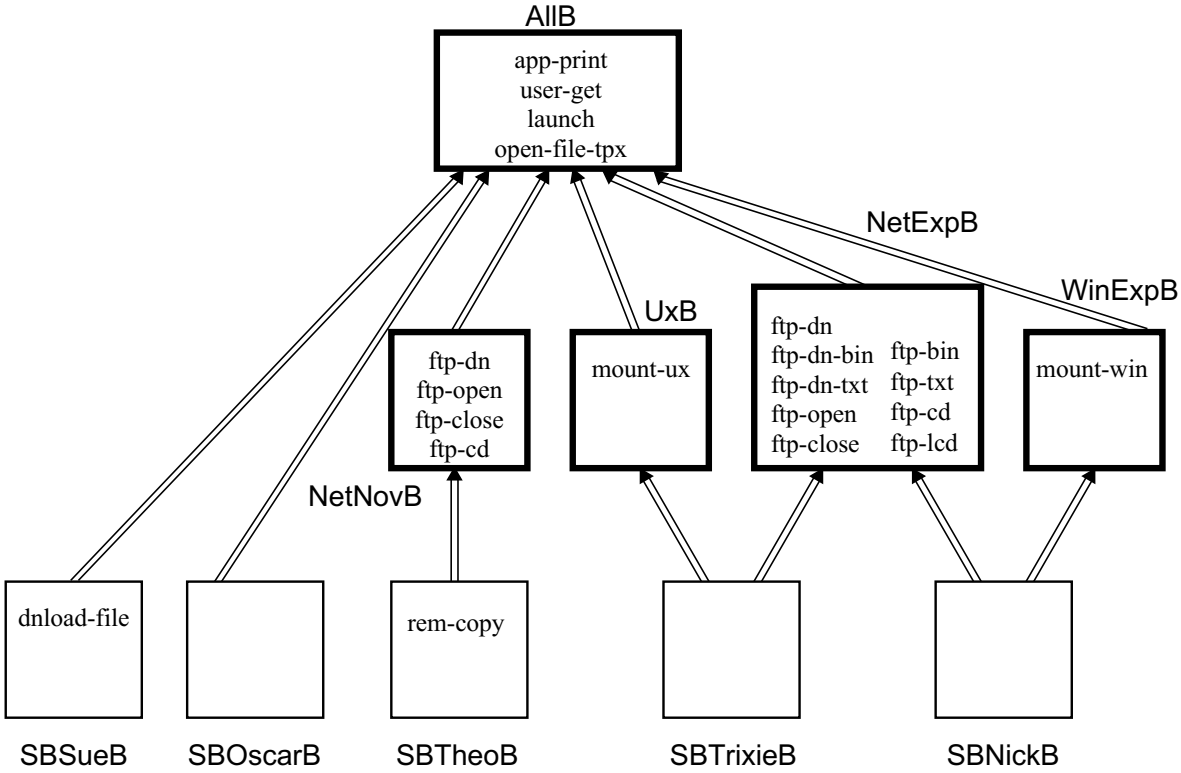

Abbildung 3.16: Beispiel einer Partitionsstruktur mit Stereotype-Partitionen für einfache Annahmen und zugeordneten Benutzer-Partitionen. (Geerbte und Domain-Concepts sind nicht dargestellt.)

Oscar	ist Büroangestellter und entspricht weitgehend dem Klischee eines typischen Benutzers von Office-Programmen, ohne daß er über seine tägliche Arbeit hinaus besondere Kenntnisse hätte.
Trixie	ist Gastangestellte, deren normale Arbeitsumgebung UNIX-basiert ist.
Nick	ist ein Informatik-Student, der neben seinem Studium regelmäßig in dem Büro arbeitet und sich mit der technischen Infrastruktur des Büros gut auskennt.
Theo	ist ebenfalls Student, der aber nur gelegentlich in dem Büro aushilft und sich nicht besonders gut auskennt.
Sue	ist eine Person, über die das System nur geringe Kenntnisse hat.

Dementsprechend ist Oscar dem Stereotype *All* zugeordnet; ihm werden also lediglich die allgemeinen Mindestkenntnisse zugetraut. *Theo* ist *NetNov* zugeordnet, ihm werden damit außer den allgemeinen Kenntnissen auch Anfänger-Kenntnisse bzgl. Netzwerken zugetraut. *Trixie* ist als UNIX- und Netzwerk-Expertin den beiden Stereotypes *Ux* und *NetExp* zugeordnet. Auch *Nick*, der sich sowohl mit Windows als auch mit Netzwerken gut auskennt, ist ebenfalls zwei Stereotypes zugeordnet: *NetExp* und *WinExp*. Über *Sue* weiß das System nur wenig, ihr werden deshalb nur die Standardannahmen aus *All*B zugebilligt sowie ein sehr allgemeines Plan-Concept *download-file* (*dnload-file*), mit dem eine fremde Datei auf irgend eine Art und Weise auf den lokalen Computer gebracht werden kann.

Die Abbildung zeigt für jede Partition die (abgekürzten) Namen der Plan-Concepts, die direkt für die Partition definiert sind. Anhang B.2 enthält eine Liste der verwendeten Abkürzungen (B.2.4)

und eine kurze Beschreibung der Plan-Concepts (B.2.1). Aus Gründen der Übersichtlichkeit sind ererbte Concepts nicht dargestellt. Gleiches gilt für Domain-Concepts, die hier nicht im Mittelpunkt des Interesses stehen und deshalb für das System und alle Benutzer gleich angenommen werden[40]. Dementsprechend sind die meisten Partitionen die Benutzerkenntnisse modellieren leer dargestellt. Zu erwähnen bliebe noch das Plan-Concept *remote-copy* (*rem-copy*), das nach *Theos* Ansicht dazu geeignet ist, Dateien von beliebigen fremden Computern auf den lokalen Computer zu kopieren. Ein Vergleich mit den Annahmen des Systems in Abb. 3.17 zeigt, daß das System dieses Concept *nicht* kennt. Nach vereinbarter Sprechweise handelt es sich dabei also um eine Fehlannahme von *Theo*.

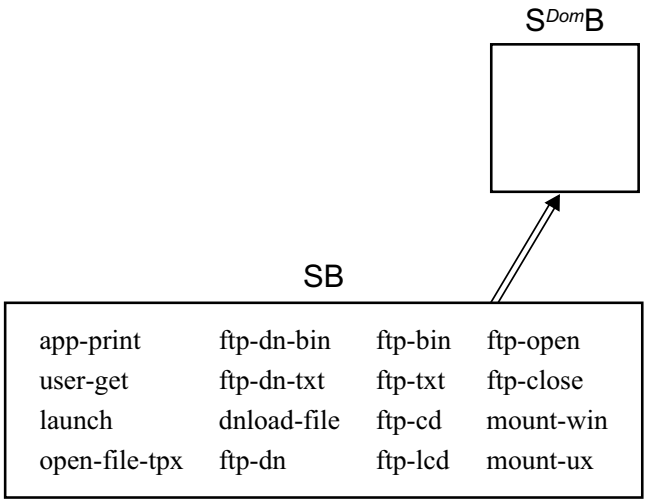

Abbildung 3.17: Partitionsstruktur der Systemannahmen mit einer eigenen Partition für objektbezogene Annahmen. Domain-Concepts und ererbte Concepts sind nicht dargestellt.

Die Stereotype-Partitionen haben folgenden Inhalt: als Mindestkenntnisse werden das Starten eines Anwendungsprogramms (*launch*), das Drucken aus einem Anwendungsprogramm (*app-print*), das Öffnen einer Datei in dem Textverarbeitungssystem, das für das Beispiel zur Verfügung steht, (*open-file-tpx*) sowie das Holen eines Gegenstandes (*user-get*) angesehen. Netzwerk-Anfänger kennen darüber hinaus einige Funktionen zum Dateitransfer via FTP (*ftp-download-file*, *ftp-open-connection*, *ftp-close-connection*, *ftp-cd*). Es handelt sich dabei allerdings nur um allgemeine Funktionen, die für die korrekte Arbeit nicht ausreichen. Netzwerk-Experten wissen darüber hinaus, daß ein Dateitransfer nur mit den spezialisierten Funktionen *ftp-download-file-bin*, *ftp-download-file-text* erfolgreich ist und können den zugehörigen Programm-Modus durch *ftp-chmode-binary* bzw. *ftp-chmode-text* auch einstellen. Außerdem wissen sie, wie man das lokale Dateiverzeichnis (das Verzeichnis, in das die Dateien transferiert werden) ändern kann (*ftp-lcd*). Die zusätzlichen Kenntnisse typischer UNIX-Anwender und Windows-Experten sind hier nur auf das Beispiel eingeschränkt dargestellt. Demnach kennen beide Benutzergruppen die Funktion *mount* – allerdings spezialisiert für das jeweilige Betriebssystem (*mount-ux*, *mount-win*).

Im Gegensatz zu den Kenntnissen hängen die Handlungsmöglichkeiten von Büro-Mitarbeitern weniger von persönlichen Eigenschaften, als vielmehr vom Status innerhalb der jeweiligen Organisation ab, weil sie in erster Linie durch Befugnisse bestimmt werden. Für das Beispiel wird zwischen *Angestellten* und *studentischen Hilfskräften* unterschieden. Angestellte unterliegen dabei keinen Einschränkungen. Insbesondere dürfen sie sich durch das Mounten von Verzeichnissen fremder Compu-

[40]Eine Aufstellung der Begriffshierarchie mit allen Domain-Concepts findet sich im Anhang B.1 *Domain-Concepts*.

ter Zugriff zu allen freigegebenen Daten innerhalb der Organisation verschaffen. Da von Windows als Betriebssystem ausgegangen wird, steht ausschließlich die Windows-Variante des Mountens (*mount-win*) zur Verfügung. Genau dies ist studentischen Hilfskräften allerdings aus Sicherheitsgründen verwehrt. Einem Dateiaustausch via FTP steht allerdings nichts im Wege, weil davon ausgegangen wird, daß ein Datei-Transfer aus dem Internet unkritisch ist[41] und innerhalb der Organisation nur sehr gezielt einige wenige Daten für den FTP-Transfer zugänglich sind. Eine Sonderstellung nehmen *Gäste* ein. Ihnen wird zwar für ihre Arbeit innerhalb der Organisation Zugang zu allen freigegebenen Daten gewährt, aber ein Transfer nach außerhalb, der mit FTP möglich wäre, ist ihnen nicht gestattet.

Dieser Vorstellung folgt die Struktur der Stereotypes für die Handlungsmöglichkeiten von Benutzern, die in Abb. 3.18 dargestellt ist. Die Abbildung ist analog zu Abb. 3.16. Dementsprechend enden Namen von Fähigkeits-Partitionen mit "C" für *Capability*. Außer den Handlungsmöglichkeiten, die allen Benutzer zugeschrieben werden (*All*C), sind die speziellen Befugnisse der Angestellten, Gäste und studentischen Hilfskräfte durch die Stereotypes *Staff*C, *Guest*C bzw. *Stud*C modelliert. Die Modelle der Handlungsmöglichkeiten der Beispiel-Benutzer entsprechen hier weitgehend den Stereotypes, denen sie zugeordnet sind. Lediglich *Sue* und *Nick* weichen davon ab. *Sue* wird zugebilligt, daß sie die sehr allgemeine Form des Dateitransfers die sie kennt auch ausführen kann. *Nick* hat als langjähriger, vertrauenswürdiger Mitarbeiter die zusätzliche Befugnis für den Zugriff auf interne, freigegebene Verzeichnisse mit Hilfe von *mount-win*.

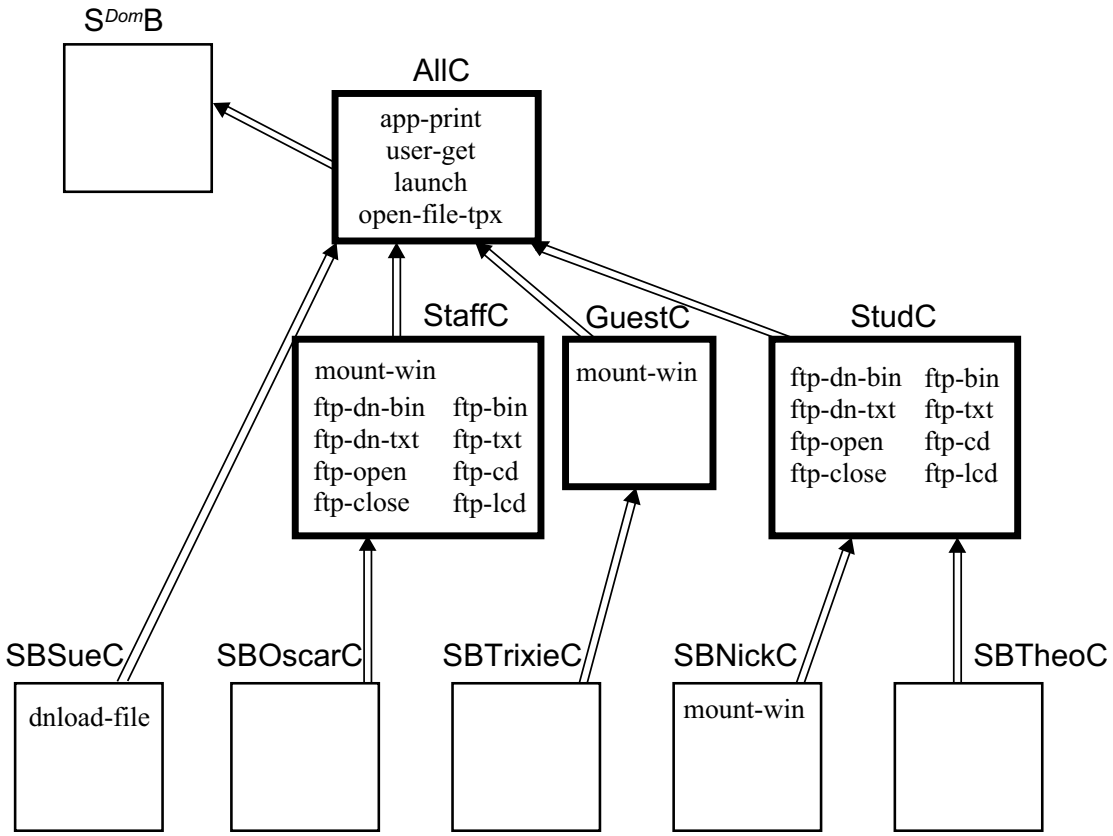

Abbildung 3.18: Partitionsstruktur mit Stereotype-Partitionen für Handlungspotentiale und zugeordneten Benutzer-Partitionen. Domain-Concepts und ererbte Concepts sind nicht dargestellt.

[41] Von Sicherheitsrisiken, die in der Praxis bestehen, sei für dieses Beispiel abgesehen.

Alle Fähigkeits-Partitionen sind direkt oder indirekt Sub-Partitionen von $S^{Dom}B$ und erben damit die Kenntnisse des Systems über die Objekte des Sachbereichs. Die Struktur verdeutlicht, daß nicht nur die Systemannahmen über die Handlungsmöglichkeiten der einzelnen Benutzer, sondern auch diejenigen über die typischen Handlungsmöglichkeiten von Benutzergruppen zu den Annahmen des Systems gehören.

Diese Arbeit folgt der BGP-MS-Konvention, Stereotypes implizit zu den Systemannahmen über Benutzergruppen anzusehen und keine davon abweichenden Annahmen anderer Akteure über Benutzergruppen durch Stereotypes zu unterstützen (vgl. allerdings den folgenden Abschnitt 3.3.4 *Geschachtelte Annahmen über Handlungsmöglichkeiten*). Die Benamung der Stereotype-Partitionen folgt deshalb wie oben dem Muster <Stereotype-Name><Modalität>.

Eine Definition von Stereotypes orientiert sich an einer Gruppierung der potentiellen Benutzer eines Systems, sodaß die Mitglieder der Gruppen bzgl. vorgegebener Eigenschaften möglichst homogen sind. Die Gruppen, die aufgrund der Kenntnisse gebildet werden können, reflektieren die fachlichen Themenbereiche des Anwendungsgebiets – hier Betriebssysteme und Netzwerke. Sie unterscheiden sich deutlich von solchen, die anhand der Handlungsmöglichkeiten gebildet werden können. Im Beispielsachbereich werden sie durch Befugnisse bestimmt, die sich am Status innerhalb einer Organisation orientieren – hier Angestellte, Gäste und studentische Hilfskräfte.

Die daraus abgeleiteten Stereotypes modellieren deshalb *entweder* Kenntnisse *oder* Handlungsmöglichkeiten, i.d.Regel aber nicht beides. Einzige Ausnahme ist in dem Beispiel der Stereotype *All*, der Annahmen über die Kenntnisse und Handlungsmöglichkeiten repräsentiert, die allen potentiellen Benutzern zugeschrieben wird. Nach Pohls Notation ergeben sich aus den Abb. 3.16 und 3.18 die Stereotypes in Abb. 3.19.

$$
\begin{aligned}
\text{All} &= \{<KB_{SBAllB}, SBUB>, <KB_{SBAllC}, SBUC>\} \\
\text{NetNov} &= \{<KB_{SBNetNovB}, SBUB>\} \\
\text{Ux} &= \{<KB_{SBUxB}, SBUB>\} \\
\text{NetExp} &= \{<KB_{SBNetExpB}, SBUB>\} \\
\text{WinExp} &= \{<KB_{SBWinExpB}, SBUB>\} \\
\text{Staff} &= \{<KB_{SBStaffC}, SBUC>\} \\
\text{Guest} &= \{<KB_{SBGuestC}, SBUC>\} \\
\text{Stud} &= \{<KB_{SBStudC}, SBUC>\}
\end{aligned}
$$

Abbildung 3.19: Stereotypes aus Abb. 3.16 und 3.18 in der Notation des AsTRa-Formalismus [Pohl, 1997, Kap.6.2.3]

Abb. 3.16 und 3.18 stellen die Inhalte der Benutzermodellteile lediglich durch die Namen der Plan-Concepts dar. Dies ist aber lediglich ein Zugeständnis an eine übersichtliche Darstellung und bedeutet *nicht*, daß Plan-Concepts nur durch ihren Namen bestimmt werden oder daß alle Plan-Concepts gleichen Namens auch im beschreibungslogischen Sinne gleich sind. (Vgl. dazu auch das Bsp. im vorangegangenen Abschnitt.) Allerdings sei die Konvention vereinbart, daß gleich bezeichnete Concepts auch gleich sind, sofern Differenzen nicht explizit erwähnt werden.

3.3.4 Geschachtelte Annahmen über Handlungsmöglichkeiten

Für die Zielsetzung dieser Arbeit sind vor allem die Annahmen des Systems über die Kenntnisse und Handlungsmöglichkeiten des Benutzers von Bedeutung. Dennoch soll kurz auf die mögliche Anwen-

dung und Konsequenzen geschachtelter Annahmen – also Annahmen über Annahmen usw. – eingegangen werden. Kobsa behandelt in [Kobsa, 1990a] geschachtelte Annahmen und weist dabei auch darauf hin, daß die Schachtelungstiefe in einer Dialogsituation (wie sie auch für diese Arbeit vorausgesetzt wird) begrenzt ist. Es ist bspw. schwer vorstellbar, welchen Beitrag Annahmen der Schachtelungstiefe 6 (die nach der obigen Notation mit SBUBSBUBSBUBSB bezeichnet werden können) für einen Dialog leisten können. [Taylor *et al.*, 1996] analysieren die Verwendung geschachtelter Annahmen im Rahmen eines kooperativen Dialogs und kommen zu dem Schluß, daß sie maximal bis zur Schachtelungstiefe drei sinnvoll sind.

Sollen bei geschachtelten Annahmen auch Annahmen über Handlungsmöglichkeiten berücksichtigt werden, so können diese nur am Ende eine Schachtelung auftreten. Bspw. sind Annahmen eines Akteurs über die Annahmen eines weiteren über die Handlungsmöglichkeiten eines dritten denkbar. Gehen wir wiederum davon aus, daß das gesamte Modell Teil der Systemannahmen ist, können diese geschachtelten Annahmen als SBA_iBA_jC bezeichnet werden. Für SBA_iCA_jB oder SBA_iCA_jC fehlt dagegen eine sinnvolle Bedeutungszuordnung.

Annahmen des Typs SBA_iBA_jC lassen sich von SB vor allem für die Interpretation von A_is Äußerungen an A_js Adresse nutzen. Im Falle eines System-Benutzer-Dialogs[42] ist A_i = U und A_j = S; die Annahmen können also für die Interpretation der Äußerung des Benutzers genutzt werden. Die Vorstellung ist dabei, daß jeder Akteur im wesentlichen die gleichen grundlegenden Kommunikationsprinzipien beim Erzeugen einer Äußerung beachtet wie auch das System. Das heißt insbesondere, daß ein Akteur nur solche Begriffe verwendet, von denen annimmt, daß der Adressat sie kennt und daß er nur solche Aktionen als Reaktion auf die Äußerung erwartet, die der Adressat auch ausführen kann. Dies kann bspw. als Kriterium für eine Disambiguierung von Äußerungen des Benutzers herangezogen werden. Es brauchen nur solche Interpretationen berücksichtigt zu werden, die

1. ausschließlich Begriffe enthalten, von denen das System annimmt, daß der Benutzer annimmt, daß das System sie kennt, und
2. vom System nur solche Aktionen – als Reaktion auf die Äußerung – fordern, von denen das System annimmt, daß der Benutzer annimmt, daß das System sie ausführen kann.

Es sei angemerkt, daß viele Dialogsysteme zur Einschränkung der Interpretationsmöglichkeiten einer Benutzer-Äußerung – meist implizit – den sogenannten Dialograhmen berücksichtigen, der durch die Situation bestimmt wird, in der der Dialog stattfindet. Bspw. können Fragen an einen Bahnbediensteten in einer Bahnhofs-Schalterhalle als Fragen zum Zugreiseverkehr interpretiert werden. Eine Formalisierung des Dialograhmens fehlt allerdings. Durch die explizite Berücksichtigung der (angenommenen) Kenntnisse und Handlungsmöglichkeiten eines Dialogpartners läßt sich der Einfluß der Situation auf sinnvolle Interpretationen einer Äußerung wesentlich präziser fassen.

3.3.4.1 Auswirkungen geschachtelter Annahmen auf die Partitionsstruktur

Weiter oben wurde bereits darauf hingewiesen, daß die Annahmen eines Akteurs über die Handlungsmöglichkeiten eines anderen von den eigenen Vorstellungen geleitet werden. Das gilt natürlich auch für Annahmen des Benutzers über die Handlungsmöglichkeiten des Systems. Sinnvollerweise wird dann auch das objektbezogene Wissen des Benutzers in einer eigenen Partition SBU^{Dom}B repräsentiert, die Super-Partition von SBUB und SBUBSC ist. Dies wiederum hat Auswirkungen auf die Stereotype-Struktur, denn auch ein Stereotype muß dann das typische Wissen von Benutzergruppen bzgl. objektbezogenem und allgemeinem Wissen differenzieren. Dazu wird für jede Stereotype-Partition, die typisches Wissen repräsentiert, eine Super-Partition definiert, die nur das objektbezogene

[42] In der verallgemeinerten Interpretation als (nicht notwendigerweise natürlichsprachliche) Kommunikation zwischen zwei Akteuren.

Wissen enthält. Abb. 3.20 zeigt das Beispiel aus Abb. 3.16 mit diesen Erweiterungen. Um die Übersichtlichkeit zu erhöhen, ist die Relation zwischen Sub- und Super-Partition lediglich als einfacher Pfeil gezeichnet. Wie in Abb. 3.16 sind Domain-Concepts nicht dargestellt. Alle ^{Dom}B-Partitionen

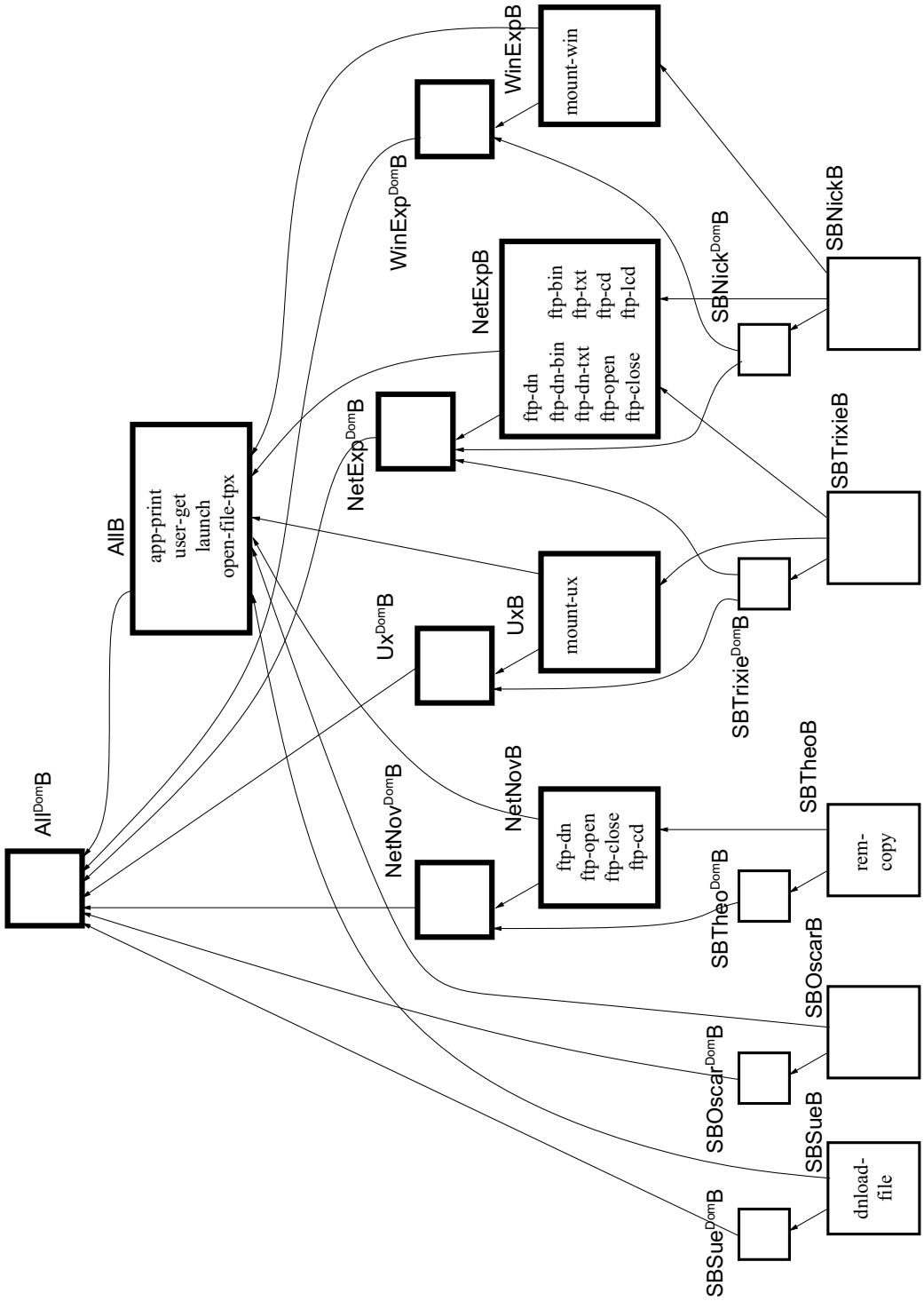

Abbildung 3.20: Partitionsstruktur aus Abb. 3.16 mit eigenen Partitionen für objektbezogenes Wissen

sind dementsprechend leer. Im Sinne des AsTRa-Formalismus stellt $SBU^{Dom}B$ einen Assumption Type und ^{Dom}B eine Modalität dar.

Um rasch ein Bild von den Annahmen des Benutzers über Annahmen und Handlungsmöglichkeiten des Systems zu bekommen, bietet sich wiederum der Einsatz von Stereotypes an. Da hierbei typische Annahmen von Benutzergruppen modelliert werden, werden die Stereotypes sinnvollerweise anhand von Benutzergruppen mit möglichst homogenen *Kenntnis*ständen gebildet. Analog zur Einteilung des Benutzerwissens, werden auch die typischen Annahmen von Benutzergruppen aufgeteilt in:

- einfache objektbezogene Annahmen
- einfache Annahmen, die das gesamte Wissen betreffen
- Annahmen über die (einfachen) Annahmen des Systems
- Annahmen über die Handlungsmöglichkeiten des Systems

Im AsTRa-Formalismus heißt dies, daß die dazu gehörenden Stereotypes (zumindest) vier Wissensbasen enthalten, die Beiträge zu den Assumption Types $SBU^{Dom}B$, SBUB, SBUBSB und SBUBSC leisten. Abb. 3.21 zeigt das Beispiel aus Abb. 3.19 mit den Erweiterungen für die Benutzerannahmen über Wissen und Handlungsmöglichkeiten des Systems.

$$
\begin{array}{lcl}
\text{All} & = & \{<KB_{SBAllB}, SBUB>, <KB_{SBAllDomB}, SBU^{Dom}B>, \\
 & & <KB_{SBAllBSB}, SBUBSB>, <KB_{SBAllBSC}, SBUBSC>, \\
 & & <KB_{SBAllC}, SBUC>\} \\
\text{NetNov} & = & \{<KB_{SBNetNovB}, SBUB>, <KB_{SBNetNovDomB}, SBU^{Dom}B>, \\
 & & <KB_{SBNetNovBSB}, SBUBSB>, <KB_{SBNetNovBSC}, SBUBSC>\} \\
\text{Ux} & = & \{<KB_{SBUxB}, SBUB>, <KB_{SBUxDomB}, SBU^{Dom}B>, \\
 & & <KB_{SBUxBSB}, SBUBSB>, <KB_{SBUxBSC}, SBUBSC>\} \\
\text{NetExp} & = & \{<KB_{SBNetExpB}, SBUB>, <KB_{SBNetExpDomB}, SBU^{Dom}B>, \\
 & & <KB_{SBNetExpBSB}, SBUBSB>, <KB_{SBNetExpBSC}, SBUBSC>\} \\
\text{WinExp} & = & \{<KB_{SBWinExpB}, SBUB>, <KB_{SBWinExpDomB}, SBU^{Dom}B>, \\
 & & <KB_{SBWinExpBSB}, SBUBSB>, <KB_{SBWinExpBSC}, SBUBSC>\} \\
\text{Staff} & = & \{<KB_{SBStaffC}, SBUC>\} \\
\text{Guest} & = & \{<KB_{SBGuestC}, SBUC>\} \\
\text{Stud} & = & \{<KB_{SBStudC}, SBUC>\}
\end{array}
$$

Abbildung 3.21: Stereotypes unter Berücksichtigung von Annahmen des Benutzers über das System

Dabei ist wiederum zu berücksichtigen, daß die Annahmen über Domain-Concepts für die Beschreibung der Plan-Concepts in den Wissensbasen, die zu SBUBSC beitragen, durch den beobachteten Akteur – hier also den Benutzer – bestimmt werden. Eine Wissensbasis $KB_{SB<St>BSC}$ kann dabei nur genau die Domain-Concepts einsetzen, die dieser Benutzergruppe <*St*> auch zugeschrieben werden; d.h. die in $KB_{SB<St>DomB}$ enthalten sind. Die zugehörigen Stereotype-Partitionen werden deshalb als Sub-Partitionen der Partitionen für die Domain-Concepts der jeweiligen Benutzergruppe definiert. Abb. 3.22 zeigt die Gesamtstruktur der Stereotype-Partitionen für die Benutzergruppen *All*, *NetNov*,

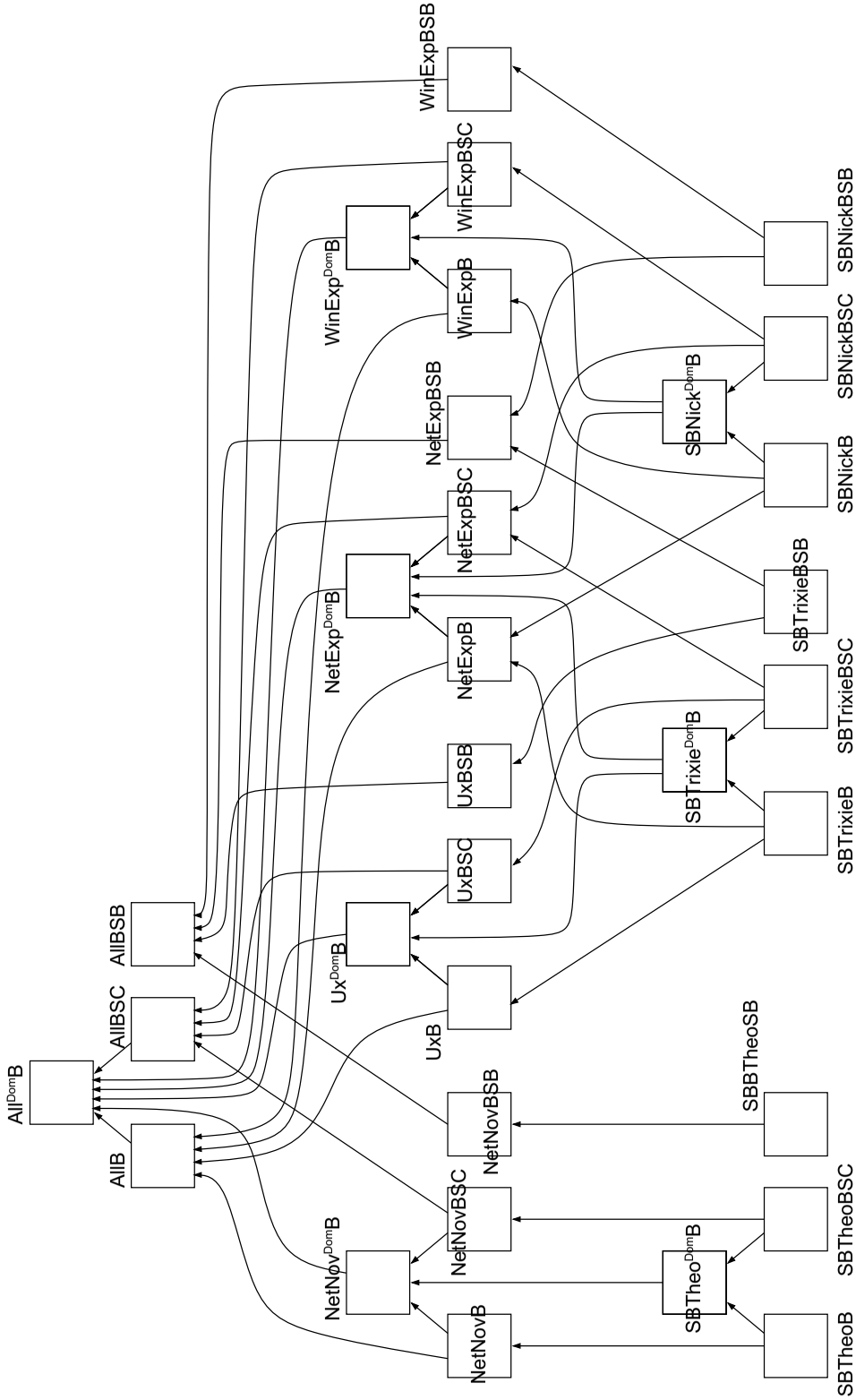

Abbildung 3.22: Partitionsstruktur aus Abb. 3.20 (ohne *Sue* und *Oscar*) mit zusätzlicher Berücksichtigung von Benutzerannahmen über Annahmen und Handlungsmöglichkeiten des Systems.

Ux, *Winexp* und die Zuordnung der gedachten Benutzern *Theo*, *Trixie* und *Nick* wie in Abb. 3.16. Aus Gründen der Übersichtlichkeit blieben die Benutzerpartitionen für *Sue* und *Oscar* unberücksichtigt. Außerdem sind die Inhalte der Partitionen nicht dargestellt und die Vererbungspfeile wie in Abb. 3.20 nur dünn gezeichnet. Wiederum ist zu beachten, daß BGP-MS von *einem* ausgezeichneten Benutzer ausgeht. Anstelle der in der Abbildung dargestellten Benutzerpartitionen werden zu einem Zeitpunkt demnach nur die vier Partitionen SBUB, SBU^{Dom}B, SBUBSC, SBUBSB für den aktuellen Benutzer berücksichtigt. Diese ergeben sich aus den dargestellten, wenn der Name des Benutzers durch "U" ersetzt wird.

Die Vorgehensweise kann im Prinzip auch für Annahmen des Systems über die Annahmen des Benutzers über dritte Akteure weitergeführt werden. Sollen für a priori Annahmen über solche dritte Akteure wiederum Stereotypes eingesetzt werden, muß berücksichtigt werden, daß verschiedene Benutzer bzw. Benutzergruppen auch über Stereotypes sowie deren Inhalt und auch der Triggerbedingungen unterschiedliche Ansichten haben können. Es dürfte schwierig sein, zuverlässige und ausreichend vollständige Informationen für ein solches Modell zu bekommen (vgl. bspw. [Ballim und Wilks, 1991]). Da dies vom Kern dieser Arbeit wegführt, wird der Weg hier nicht weiter verfolgt.

3.4 Planungsproblem und Planungssachbereich

In den vorangegangenen Abschnitten wurde gezeigt, wie Planungswissen – d.h. Annahmen über Veränderungsmöglichkeiten in einem Anwendungssachbereich und über die Handlungsmöglichkeiten von Akteuren – in einer Benutzermodellierungswissensbasis (BMWB) organisiert und repräsentiert werden kann. Dieser Abschnitt widmet sich nun der Frage, wie ein Planungssystem dieses Wissen nutzen kann, um dann einen Plan spezifisch für den jeweiligen Benutzer zu erzeugen, sodaß dieser damit seine Ziele im Sachbereich erreichen kann. Der hier gewählte Ansatz besteht darin, aus den gegebenen Zielen und dem Benutzermodell eine Gesamt-Aufgabenspezifikation für das Planungssystem zu erzeugen, die zur Generierung eines solchen benutzerspezifischen Plans führt.

Eine Gesamt-Aufgabenspezifikation für ein Planungssystem untergliedert sich in eine Beschreibung des *Planungsproblems* und des *Planungssachbereichs*. Das Planungsproblem stellt die Aufgabenbeschreibung im engeren Sinne dar. Es besteht aus der Startsituation, die die Gegebenheiten beschreibt, wie sie vor Ausführung des Plans gelten, sowie dem Ziel, also den Gegebenheiten, die mit Hilfe des zu erzeugenden Plans erreicht werden sollen. Der Planungssachbereich modelliert die prinzipiellen Zusammenhänge im Sachbereich, die unabhängig von der konkreten Aufgabe bestehen. Hierzu gehören insbesondere die Planoperatoren[43], die die Veränderungsmöglichkeiten durch Handlungen des vorgesehenen Akteurs beschreiben, und die Abhängigkeiten im Sachbereich.

Die Informationen über den Sachbereich sind in der BMWB enthalten. Planungssysteme können die Abhängigkeiten im Sachbereich i.a. nur eingeschränkt berücksichtigen. Bei der Erzeugung eines Planungssachbereichs muß deshalb entschieden werden, inwiefern Schußfolgerungen, die aus den (in der BMWB) modellierten Abhängigkeiten resultieren, dem Planungssystem überlassen werden können. Alternativ kann der Beschreibung des Planungssachbereichs (an sich) redundante Information hinzugefügt werden, die solche Schußfolgerungen durch das Planungssystem zumindest teilweise obsolet machen. Im Verlauf dieses Unterkapitels wird dieses Thema noch ausführlich behandelt.

Wie bereits im Abschnitt 3.3 *Struktur eines Benutzermodells mit Handlungspotential und Wissen* erwähnt, wird in dieser Arbeit der Inhalt der verschiedenen Teile einer BMWB im beschreibungslogischen SB-ONE-Formalismus repräsentiert, während die meisten Planungssysteme und insb. auch das in dieser Arbeit eingesetzte Systems UCPOP [Penberthy und Weld, 1992], [Barrett *et al.*, 1995] eine

[43] Manche Autoren (vgl. [Russell und Norvig, 1995], [McAllester und Rosenblitt, 1991]) zählen die Planoperatoren mit zum Planungsproblem.

prädikatenlogische Notation für Planungsproblem und Planungssachbereich erwarten. Der Zusammenhang der beiden Repräsentationsformen wird weiter unten in naheliegender Weise definiert. Für beide Bereiche wird jeweils eine systemunabhängige Notation verwendet. Die beschreibungslogische Notation entspricht der *Description-Logic Knowledge Representation System Specification* (KRSS) (s. [Patel-Schneider und Swartout, 1993] u. Abschnitt 3.2.1 *Beschreibungslogiken*). Für Planungsproblem und Planungssachbereich wird entweder direkt eine mathematische Notation der prädikatenlogischen Formeln oder die systemunabhängige *Planning Domain Definition Language* (PDDL)[44] eingesetzt, die der UCPOP-Notation sehr nahe kommt. Wo notwendig und/oder sinnvoll wird auch auf die Spezifika der eingesetzten Systeme SB-ONE und UCPOP sowie deren Beschreibungssprachen eingegangen. Die im Demonstrations-Prototyp realisierte Erzeugung der Gesamt-Aufgabenspezifikation verwendet ausschließlich die Syntax für das eingesetzte Planungssystem UCPOP.

3.4.1 Das Planungsproblem

Im engeren Sinne beschreibt das Planungsproblem die eigentliche Aufgabe; d.h. die Beschreibung von welchen Gegebenheiten ausgegangen werden soll – die Startsituation – und welche Veränderungen demgegenüber erreicht werden sollen – das Ziel. In UCPOP und PDDL wird die Startsituation durch eine Menge implizit konjunktiv verknüpfter, variablenfreier Atomformeln beschrieben. Viele Planungssysteme – und auch UCPOP – gehen von der *Closed-World*-Annahme aus; d.h. alle Aussagen, die nicht aus der (als Konjunktion aufgefaßten) Startsituation abgeleitet werden können, werden als falsch angenommen. Eine explizite Angabe von negativen Aussagen ist deshalb überflüssig. Gleichwohl läßt PDDL für die Beschreibung der Startsituation eine Menge von Literalen, also auch negierte Atomformeln zu, um auch Planungssysteme zu unterstützen, die *nicht* von der *Closed-World*-Annahme ausgehen.

Mit der Startsituation ist durch die Verwendung der Konstanten implizit auch die Angabe aller für das Planungsproblem relevanten Objekte verbunden. PDDL erlaubt darüber hinaus eine explizite Deklaration der Konstanten für die Definition eines Planungssachbereichs. Die Erzeugung von Objekten bzw. das Hinzufügen von neuen Konstanten durch Planoperatoren wird durch PDDL im Gegensatz zu UCPOP nicht unterstützt[45]. Dieses Thema wird in Abschnitt 3.4.2.5 *Erzeugen von Objekten* noch weiter behandelt.

Im Gegensatz zur Startsituation, die per Definition eine vollständige Beschreibung darstellt, stellt die Zielbeschreibung nur Anforderungen an die Situation, die nach Ausführung des Plans erreicht werden soll. Es wird also die *Menge* der Situationen beschrieben, die den Anforderungen genügen. Als Zielbeschreibung ist eine prädikatenlogische Formel ohne freie Variablen zulässig. In PDDL kann dabei von dem vordefinierten Prädikat "=" Gebrauch gemacht werden, während UCPOP explizit die Gleichheit und Ungleichheit zweier Konstanten oder Variablen als zusätzliches syntaktisches Konstrukt vorsieht.

Sowohl Startsituation als auch Ziel sind Beschreibungen von Momentaufnahmen; d.h. sie gelten für genau einen Zeitpunkt und enthalten weder explizite noch implizite Aussagen zum Zeitbezug. Durch eine Zielbeschreibung alleine kann deshalb nicht ausgedrückt werden, daß etwas *verändert* werden soll – bspw., daß ein *bisher* roter Klotz *nach* Ausführung des Plans grün sein soll. Solche Veränderungen sind nur durch die *Kombination von Startsituation und Ziel* möglich; z.B. (bei offensichtlicher Interpretation):

[44] s. [Ghallab *et al.*, 1998], S. 16 in Abschnitt 2.1 sowie Abschnitt 3.2.2

[45] Das steht im Widerspruch zur Einleitung des PDDL-Manuals [Ghallab *et al.*, 1998], in dem es heißt "The language supports the following syntactic features: Universal quantification over dynamic universes (i.e., object creation and destruction) ...". Der Punkt wird aber im Manual nicht weiter behandelt. Eine Nachfrage beim Kontakt-Autor Drew McDermott ergab, daß nicht klar ist wie der Punkt in das Manual gelangte. Die Vermutung liegt nahe, daß er vom UCPOP-Manual stammt, auf dem das PDDL-Manual aufgebaut wurde.

Startsituation:	rot (Klotz1)
Ziel:	¬rot (Klotz1) ∧ grün (Klotz1)

Wegen der – in der realen Welt bestehenden – Abhängigkeit, daß ein Klotz nicht gleichzeitig rot und grün sein kann, erscheint die Zielbeschreibung redundant. Eine Erörterung der Abhängigkeiten im Sachbereich findet sich in den nachfolgenden Abschnitten. Sehr vergleichbar zu gewünschten Veränderungen ist ein Ziel, *neue* Objekten zu erzeugen. Bspw. das Ziel, daß *ein Akteur einen* neuen *Ausdruck einer Datei besitzt* (und nicht etwa dadurch in den Besitz eines Dateiausdrucks gelangt, daß er einem Kollegen den Ausdruck wegnimmt). Darauf wird im Zusammenhang mit der Erzeugung von Objekten noch näher eingegangen.

Sowohl die Startsituation als auch das Ziel können (sinnvollerweise) nur Gegebenheiten beschreiben, die in dem Sachbereich prinzipiell möglich sind. Sie müssen also mit der Sachbereichsbeschreibung konform sein. Insbesondere müssen die verwendeten Prädikate in der Sachbereichsbeschreibung definiert sein. Näheres zu Startsituationen und Zielen des gewählten Beispielsachbereichs findet sich in Kap. 5.3.1 *Beispiel-Planungsprobleme*. In Kapitel 5.2.2 *Der Beispielsachbereich* werden einzelne Aspekte dieser Beschreibungen noch näher erläutert.

3.4.2 Der Planungssachbereich

Für den Begriff *Planungssachbereich* (*Domain*, *Application Domain*) fehlt in der Literatur eine einheitliche Definition. Wenn dieser Begriff verwendet wird, zählt dazu immer die Menge der Operatoren die dem Planungssystem zur Verfügung stehen. Lt. PDDL-Manual ([Ghallab *et al.*, 1998]) soll damit die "Physik" des Sachbereichs beschrieben werden, d.h. "what predicates there are, what actions are possible, what the structure of compound actions is, and what the effects of actions are". Dabei sei angemerkt, daß *Actions* in PDDL Planoperatoren entsprechen und auf die übliche Unterscheidung zwischen einer *Action*, die keine Variablen enthalten darf, und einem *Action Schema* (*Operator*), das Variablen enthalten darf (vgl. bspw. [Weld, 1994], [Russell und Norvig, 1995]) verzichtet wird.

Die syntaktische Definition von *Domain* in PDDL zeigt, daß zur Beschreibung eines Planungssachbereichs auch die explizite Deklaration von Prädikaten und (Objekt-)Typen gehört, sowie die Beschreibung von Abhängigkeiten im Sachbereich, die durch *Axiome* definiert werden können.

Der Planungssachbereich stellt also ein Modell des Weltausschnitts dar, das dessen prinzipielle Gegebenheiten, Zusammenhänge und Veränderungsmöglichkeiten in einer für ein Planungssystem geeigneten Weise beschreibt. Die Beschreibung läßt sich in einen *aktiven* und einen *passiven* Anteil untergliedern. Zum *aktiven Teil des Planungssachbereichs* gehört die Beschreibung, unter *welchen Umständen* sich *was wie* ändern kann. Dies wird durch Planoperatoren definiert. Sie enthalten zwar keine expliziten Angaben zum Zeitbezug, allerdings gelten Vorbedingung und Effekt zu unterschiedlichen Zeitpunkten: die Vorbedingung eines Planoperators muß *vor* der Ausführung gelten, der Effekt gilt *danach*. Der aktive Teil des Sachbereichs berücksichtigt damit implizit den Zeitaspekt und unterscheidet sich genau dadurch vom *passiven Teil des Planungssachbereichs*, dessen Aussagen zu jedem Zeitpunkt bzw. in jeder Situation gültig sind. Zum passiven Teil gehören sowohl die Begrifflichkeit des Sachbereichs als auch die Abhängigkeiten zwischen Objektklassen.

Da auch die verschiedenen Bereiche einer BMWB, wie sie in Kap. 3.3 *Struktur eines Benutzermodells mit Handlungspotential und Wissen* beschrieben wurden, Modelle des gleichen Weltausschnitts repräsentieren, stellt sich die Frage nach dem Zusammenhang der Modelle. Ein wichtiger Unterschied ist zunächst, daß es vom Standpunkt der Benutzermodellierung keine ausgezeichnete korrekte Beschreibung des Sachbereichs gibt, sondern lediglich Annahmen über den Sachbereich. Diese können (und werden normalerweise) nach Akteur (hier: *System* und *Benutzer*) und Modalität (hier: *Wissen* und *Handlungspotential*) verschieden sein und können sich – bspw. durch Beobachtung des Benutzers – auch ändern.

Die für eine Aufgabenspezifikation an ein Planungssystem erforderliche Beschreibung des Planungssachbereichs muß also unter Zugrundelegung der geeigneten Sichten (*Views*) aus der BMWB abgeleitet werden. Soll simuliert werden, welche Pläne der Benutzer selbst ermitteln kann, sind das die *Annahmen des Systems über die Annahmen des Benutzers* (SBUB) bzw. wenn er die Pläne auch selbst ausführen will oder soll, die *Annahmen des Systems über die Annahmen des Benutzers über seine eigenen Handlungsmöglichkeiten* (SBUBUC). Im Zentrum des Interesses dieser Arbeit stehen allerdings Pläne, die *der Benutzer nach Ansicht des Systems ausführen kann*. Die relevante Sicht hierfür sind die *Annahmen des Systems über das Handlungspotential des Benutzers*, die in der View SBUC modelliert sind. Sofern nicht anders angegeben, beruht die Erzeugung der verschiedenen Teile des Planungssachbereichs deshalb auf dem Inhalt dieser View.

Der Zusammenhang zwischen prädikatenlogischer Repräsentation des Planungssachbereichs und beschreibungslogischer Repräsentation des Benutzermodells wurde für den aktiven Teil des Sachbereichs – d.h. für Planoperatoren bzw. Plan-Concepts – bereits in Kap. 3.2.2 *Planoperatoren in Beschreibungslogik* beschrieben. Da Planoperatoren keine Elemente des prädikatenlogischen Objektbereichs (O_{PS}) bezeichnen, unterscheiden sich die Objektbereiche von prädikatenlogischer und beschreibungslogischer Repräsentation. Der prädikatenlogische Objektbereich enthält gerade die Objekte, die für Planoperatoren bzw. Aktionen benötigt und/oder von ihnen verändert werden. In der beschreibungslogischen Repräsentation sind dies gerade die Elemente der Extension des allgemeinsten Domain-Concepts *domain-concept*[46]; d.h. O_{PS} = *domain-concept*^{I}. Wie in Kap. 3.2.1 *Beschreibungslogiken* beschrieben, wird die Semantik von Beschreibungslogiken meist durch eine Interpretationsfunktion $.^I$ definiert, die Concepts und Concept-Ausdrücke auf Teilmengen des Objektbereichs Δ^I und Rollen, Attribute und entsprechende Ausdrücke auf Teilmengen von $\Delta^I \times \Delta^I$ abbildet. Dementsprechend wird die Zuordnung zwischen prädikatenlogischer und beschreibungslogischer Repräsentation des passiven Teils des Sachbereichs wie folgt definiert:[47]

Definition 3.3
Zu jedem modellierten einstelligen Prädikat P_{PS} des Planungssachbereichs gehört ein gleichnamiges Domain-Concept P_{BM} mit $P_{PS} = P^I_{BM}$. Umgekehrt gehört zu jedem *relevanten* Domain-Concept P_{BM} ein gleichnamiges einstelliges Prädikat P_{PS} des Planungssachbereichs mit $P_{PS} = P^I_{BM}$.

Definition 3.4
Zu jedem modellierten zweistelligen Prädikat R_{PS} des Planungssachbereichs gehört eine gleichnamige Rolle[48] R_{BM} mit $R_{PS} = R^I_{BM}$.

Da das Modell des Planungssachbereichs aus der beschreibungslogischen Repräsentation abgeleitet wird, gibt es zu jedem Prädikat ein zugehöriges Domain-Concept. Die Umkehrung gilt nur eingeschränkt, weil i.a. nicht alle Domain-Concepts und Rollen oder Attribute für die Erzeugung eines Plans gebraucht werden. Dies gilt insbesondere für solche Concepts, die nur aus technischen Gründen definiert wurden (s.u.), sowie für Vorbedingungs- und Effekt-Attribute.

Einstellige Prädikate bzw. Concepts werden oft auch als (Objekt-) *Typen* aufgefaßt, die zu einer *Typisierung des Sachbereichs* führen. Dementsprechend seien die Sprechweisen *x gehört zum Typ P*, *x hat Typ P*, *x ist von Typ P* und *x gehört zum Concept P* vereinbart, wenn $P(x)$ bzw. $x \in P^I$. Bspw. gehören (implizit) alle Objekte aus O_{PS} zum Typ *domain-concept*.

Der Rest des Abschnitts behandelt die Einzelheiten der Erzeugung eines Planungssachbereichs aus dem Benutzermodell – konkret aus der View SBUC. Außer der direkten Umsetzung der verwendeten Repräsentationsformalismen muß berücksichtigt werden, welche der Schlußfolgerungen,

[46] vgl. Kap. 3.2.2 *Planoperatoren in Beschreibungslogik*.

[47] Die in den Definitionen verwendeten Indizes "*PS*" und "*BM*" dienen hier lediglich der Verdeutlichung und entfallen im weiteren.

[48] Oder ein Attribut als eingeschränkte Rolle.

die sich aus der beschreibungslogischen Repräsentation ergeben, durch das Planungssystem gezogen werden können und welche durch Hinzufügung (an sich) redundanter Information zur Beschreibung des Planungssachbereichs explizit gemacht werden müssen.

3.4.2.1 Typen im Planungssachbereich

Während die Typisierung des Sachbereichs die wesentliche Aufgabe einer beschreibungslogischen Wissensrepräsentation ist, spielen explizit definierte Typen im Bereich Handlungsplanung nur eine untergeordnete Rolle. Wird dennoch die explizite Vereinbarung von Typen unterstützt, liegt der Nutzen – im Gegensatz zu Beschreibungslogiken – nicht in Schlußfolgerungen, über die Typzugehörigkeit oder die Eigenschaften von Objektklassen. Dies wäre schon deshalb nicht gerechtfertigt, weil die klassischen Planungssysteme keine Definition der Semantik von Typen vorsehen (vgl. bspw. [Ghallab *et al.*, 1998] oder [Barrett *et al.*, 1995]). Der Nutzen von Typen für ein Planungssystem liegt darin, daß ein Typ die Menge der Werte die eine Variable annehmen kann einschränkt. Dies gilt insbesondere dann, wenn Typinformation *nicht geändert* werden kann; d.h. wenn kein Effekt eines Planoperators Typinformation enthalten darf. Man spricht dann von einem *Static Universe*, ansonsten von einem *Dynamic Universe*. Nicht jedes einstellige Prädikat kann also (in diesem Sinne) als Typ aufgefaßt werden, denn für viele Objekt-Eigenschaften, die durch einen Plan geändert werden *sollen*, ist es üblich und angemessen sie durch einstellige Prädikate zu modellieren. In der klassischen Beispielwelt der Handlungsplanung ist dies etwa das Prädikat *clear*, das für einen Block gilt, auf dem keine weiteren Blöcke sind, oder das Prädikat *lauched* des Demonstrationsbeispiels, das für Anwendungsprogramme gilt, die *aktiviert* bzw. im Zustand *bereit* sind.

Während UCPOP selbst ermittelt, welche Prädikate in diesem Sinne als Typ aufgefaßt werden können, ermöglicht PDDL die explizite Deklaration von Typinformation.

Eine explizite Typisierung des Sachbereichs und ein *Static Universe* ist für ein Planungssystem vor allem nützlich:

1. bei Parametern von Planoperatoren
2. bei allquantifizierten Variablen von Vorbedingungen
3. bei Planungssystemen, deren Vorgehen nach der Idee von Blum und Furst [Blum und Furst, 1995] auf dem Aufbau von Plangraphen basiert.

Hat ein Planoperator bspw. eine Vorbedingungskomponente $T(x)$ und ist T ein unveränderlicher Typ, so müssen nur die Konstanten berücksichtigt werden, die bereits in der Startsituation zu dem Typ gehören. Ist T dagegen ein veränderliches Prädikat, muß das Planungssystem – wenn es vollständig sein soll – auch Konstanten des Sachbereichs berücksichtigen, für die T nicht gilt und dann als neues Teilziel zu erreichen versuchen, *daß* das Prädikat für diese Konstante gilt.

Vorbedingungen mit einer allquantifizierten Variablen eines unveränderlicher Typs T können durch eine quantorenfreie äquivalente Formel ersetzt werden. Die neue Formel besteht aus einer Konjunktion, die für jede mögliche (durch T definierte) Belegung der allquantifizierten Variablen einen Faktor enthält, der aus der ursprünglichen Formel durch Belegung dieser Variablen hervorgeht. Fordert eine Vorbedingung bspw., daß für eine allquantifizierte Variable x vom Typ T $Q(x)$ gelten muß, so wird aus

$$\forall x\, T(x) \wedge Q(x)$$

durch Einsetzen aller $x \in T$

$$Q(A) \wedge Q(B) \wedge Q(C)$$

wenn T aus der Menge $\{A, B, C\}$ besteht. Das Verfahren kann für mehrere Quantoren fortgeschrieben werden, wobei auch eine Mischung von All- und Existenzquantoren zulässig ist. (vgl. [Weld, 1994]). Von der Anzahl der Elemente die zu T gehören hängt direkt ab, wie groß der Aufwand zur Berechnung der quantorenfreien Formel – der sogenannten *Universal Base* – ist und natürlich auch der Aufwand für die Feststellung, ob die Vorbedingung erfüllbar ist. Es sei angemerkt, daß [Weld, 1994] einen Vorschlag enthält, das Verfahren so zu erweitern, daß T ein beliebiges einstelliges Prädikat sein darf. Allerdings wird das Verfahren erheblich aufwendiger und muß insbesondere um eine Funktion für die Buchführung über die Veränderungen von Typzugehörigkeiten erweitert werden.

Planungsverfahren, die nach der Idee von Blum und Furst Plangraphen aufbauen, haben zu sehr effizienten Planungssystemen geführt (vgl. z.B. [Weld, 1999]). Der Plangraph wird von der Startsituation aus aufgebaut, indem alle Planoperatoren angewendet werden deren Vorbedingung erfüllt ist, wobei deren Parameter mit allen zulässigen Konstanten belegt werden. Es ist offensichtlich, daß die Größe des Plangraphs entscheidend davon abhängt, ob die Parameter typisiert sind und wie viele Konstanten zu den jeweiligen Typen gehören. Eine Änderung der Typzugehörigkeit würde zumindest Teile des Plangraphen obsolet machen und evtl. den Neuaufbau anderer Teile erfordern. Dies ist daher (z.Zt.) bei allen Planungssystemen die nach diesem Verfahren arbeiten unzulässig.

Um die Information über unveränderliche Typen (der aufgabenrelevanten View) des Sachbereichs aus dem Benutzermodell zu entnehmen, müßte diese Information über die Veränderbarkeit der Concept-Zugehörigkeit enthalten. Zur Repräsentation der Benutzermodell-Views wird mit SB-ONE eine Standard-Beschreibungslogik eingesetzt, die ebenso wie Standard-Logik keine Zeit modelliert. Veränderlichkeit kann also keine Rolle spielen. Natürlich ist es möglich die Concepts in zwei disjunkte Klassen einzuteilen – solche mit veränderlicher und solche mit konstanter Objektmenge. Darauf wurde aber verzichtet, da das eingesetzte Planungssystem die explizite Angabe von Typinformation nicht zuläßt.

3.4.2.2 Abhängigkeiten im Sachbereich

Abhängigkeiten zwischen Objekten oder Objekteigenschaften gehören zu den Eigenheiten eines Sachbereichs, die sich – eine adäquate Modellierung vorausgesetzt – dazu nutzen lassen den Modellierungsaufwand zu verringern und damit die Möglichkeit von Modellierungsfehlern zu verringern. Außerdem können damit Wirkungen von Planoperatoren modelliert werden, die sich nicht direkt durch deren Effekte ausdrücken lassen – bspw. wenn ein Effekt eine Eigenschaft einer Menge von Objekten ändern soll, die nicht von vorn herein bestimmt oder begrenzt ist. Im Anwendungssachbereich dieser Arbeit läßt sich damit bspw. modellieren, daß sämtliche Dateisystem-Objekte zu *lokalen Objekten* werden, wenn das Verzeichnis, in dem sie enthalten sind, durch eine *Mount*-Operation *lokal* wird.

Die für die Plangenerierung relevanten Abhängigkeiten lassen sich in verschiedene Gruppen einteilen, die sinnvollerweise unterschiedlich modelliert werden. Die daraus resultierenden Schlußfolgerungsmöglichkeiten können je nach Art der Abhängigkeit einem Planungssystem überlassen werden oder müssen von anderen Komponenten des Gesamtsystems gezogen werden. Folgende Arten von Abhängigkeiten sind für die Plangenerierung relevant:

1. die Abhängigkeit der Typzugehörigkeit zu verschieden abstrakten Typen
2. die Abhängigkeit eines Objektmerkmals vom Typ des Objekts
3. die Abhängigkeit des Typs eines Objekts von einem Objektmerkmal
4. Ausschließende Eigenschaften bzw. Typzugehörigkeiten
5. Komplementäre Typzugehörigkeiten

Die erste Abhängigkeit ist typisch in hierarchischen Terminologien – ein Objekt, das einem speziellen Typ zugeordnet wird, gehört automatisch auch allen abstrakteren Typen an. Eine Binär-Datei ist gleichzeitig auch eine Datei, ein Objekt usw.

Ohne Ausnutzung der Abhängigkeit müßten alle Typzugehörigkeiten explizit berücksichtigt werden. Damit verbunden wäre Mehraufwand für Spezifikation und auch für die Lösung des Planungsproblems. Entscheidender ist allerdings, daß die Möglichkeit verloren geht, Objekte aus verschieden abstrakten Sichten zu betrachten und zu verarbeiten. Im Standardbeispiel[49] dieser Arbeit muß eine Textverarbeitungs-Datei zum Ausdruck in das lokale Dateisystem gebracht werden. Eine Möglichkeit dazu ist das Herunterladen der Datei via FTP und dann der Ausdruck unter Zuhilfenahme eines passenden Textverarbeitungssystems. Während für das Herunterladen der Datei nur bekannt sein muß, ob es sich um eine Text- oder Binär-Datei handelt, läßt sich eine Datei nur dann im Textverarbeitungssystem öffnen, wenn es sich um eine spezielle Textverarbeitungs-Datei handelt. Die einfachste Lösung für das Übertragen der Typzugehörigkeit von der Original-Datei auf die Kopie, die durch das Herunterladen entsteht, wäre ein Planoperator, der die Typzugehörigkeit als Vorbedingung hat und als Effekt eine neue Datei mit genau derselben Typzugehörigkeit erzeugt. Damit hätte die Datei-Kopie aber nur den Typ *Binär-Datei* und nicht auch *Textverarbeitungsdatei*. Um bei diesem Schema sicherzustellen, daß alle Typinformation auf die Kopie übertragen wird, müßten Planoperatoren für alle Kombinationen von Typzugehörigkeiten definiert werden. Abgesehen von einem erheblichen Mehraufwand werden so Planoperatoren modelliert, die in der Realität nicht vorkommen. Das ist insbesondere dann inakzeptabel, wenn es um die Modellierung des Benutzerwissens oder um die Erklärung von Sachverhalten für den Benutzer geht.

Eine Alternative ist das Ableiten der Typzugehörigkeit aus Objektmerkmalen – unter Ausnutzung der dritten Art von Abhängigkeiten – oder aus der Zugehörigkeit zu spezielleren Typen. Im Demonstrationsbeispiel wird der Typ *Textverarbeitungs-Datei* anhand eines Objektmerkmals festgestellt und durch Verallgemeinerung auf den abstrakteren Typ *Binär-Datei* geschlossen.

Die Abhängigkeit eines Objektmerkmals vom Typ des Objekts kann genutzt werden, wenn die Veränderung einer Eigenschaft eines Objekts, die Veränderung von Eigenschaften bei abhängigen Objekten nach sich zieht. Ein Beispiel dafür ist die Eigenschaft *lokal* für ein Dateiverzeichnis, denn jede in diesem Verzeichnis enthaltene Datei oder Dateiverzeichnis ist ebenfalls lokal. Für Details der Modellierung sei auch hier wieder auf das Beispielkapitel verwiesen.

Die verbleibenden beiden Arten von Abhängigkeiten betreffen (wechselseitig) ausschließende Eigenschaften oder Zustände, die durch (wechselseitig) ausschließende Typzugehörigkeiten modelliert werden können. Im allgemeinen Fall ausschließender Typzugehörigkeiten oder Zustände kann ein Objekt genau oder höchstens einem von mehreren festgelegten Typen angehören bzw. in einem von mehreren Zuständen sein. Ein Beispiel dafür ist ein Drucker, der in genau einem der drei Betriebszustände *bereit*, *aktiv* oder *ausgeschaltet* sein kann, oder der eingangs erwähnte Klotz, der nur eine Farbe haben kann. Befindet sich ein Objekt eines allgemeinen Typs dagegen grundsätzlich in genau einem von zwei disjunkten Zuständen, seien die Sub-Typen, die diese Zustände modellieren, *komplementäre* Typen genannt. Bspw. kann ein FTP-Programm nur entweder im Transfer-Zustand *Binär-Modus* oder *Text-Modus* sein, womit der Typ *FTP-Programm* in zwei komplementäre Sub-Typen (bspw. *FTP-Programm im Binär-Modus* und *FTP-Programm im Text-Modus*) aufgeteilt werden kann.

[49] Auf dieses Beispiel wird in Kap. 5.2 *Eine Beispiel-Benutzermodellierungswissensbasis* noch genauer eingegangen.

3.4.2.3 Nichtatomare Formeln im Planungssachbereich

Bereits im STRIPS-System [Fikes und Nilsson, 1971] wurde die Beschreibung von Abhängigkeiten im Planungssachbereich dazu genutzt, den Modellierungsaufwand zu reduzieren. Bspw. sollte die Formel

$$\forall x, y, z\ connects(x, y, z) \rightarrow connects(x, z, y)$$

ausdrücken, daß jede Tür x, die Raum y mit Raum z verbindet, auch Raum z mit Raum y verbindet. Dadurch halbierte sich der Aufwand für die Beschreibung der räumlichen Gegebenheiten in denen sich der Roboter, für den der Plan bestimmt war, bewegen sollte. Wichtig ist dabei, daß das Prädikat *connects* in dem Beispiel niemals als Effekt eines Operators vorkommt; d.h. welche Tür welche Räume verbindet und insbesondere auch die Formel selbst ist unveränderlich; d.h. sie gilt in allen Situationen der Planausführung. [Lifschitz, 1986] weist darauf hin, daß diese Eigenschaft entscheidend für die Korrektheit eines STRIPS-Systems ist. Die Feststellungen von Lifschitz gelten allerdings nur für den STRIPS-Algorithmus. Eine weitergehende Berücksichtigung von Abhängigkeiten, insbesondere die Behandlung impliziter Effekte, ist zwar möglich, erfordert aber komplexere Planungsalgorithmen. Unter *impliziten Effekten* werden Änderungen der Situation verstanden, die durch einen Planoperator hervorgerufen werden können, aber nicht explizit durch dessen Effekt modelliert sind. Im Bereich Handlungsplanung wird die Problematik unter dem Namen *Ramification Problem* behandelt (s. bspw. [Georgeff, 1987]).

Abgesehen von der Notwendigkeit komplexerer Planungsalgorithmen kann die uneingeschränkte Verwendung von Formeln zur Beschreibung von Abhängigkeiten auch zu unerwünschten Resultaten führen, wie etwa *nichtdeterministische Effekte* eines Planoperators. Bspw. läßt sich die gesamte Wirkung eines Operators, dessen Effekt $\neg P(x)$ enthält, nicht bestimmen, wenn die Abhängigkeit $\forall x\ Q(x) \wedge R(x) \rightarrow P(x)$ modelliert ist. Wird $P(x)$ durch den Effekt des Planoperators falsch, so muß das Planungssystem auch sicherstellen, daß $Q(x)$ *oder* $R(x)$ falsch ist. Es gibt aber kein Kriterium, welche der beiden Formeln ausgewählt werden soll.

Viele Nachfolger von STRIPS verzichten ganz auf die Behandlung von Abhängigkeiten; bspw. sind sie nicht in der ansonsten recht ausdrucksstarken Planbeschreibungssprache ADL [Pednault, 1989] enthalten. PDDL sieht für die Definition von Abhängigkeiten im Sachbereich *Axiome* vor; allerdings nur, wenn (via *Requirement* :domain-axioms) explizit angegeben wird, daß das Planungssystem sie behandeln kann. Dann ist die Definition von Axiomen in gleicher Weise wie in UCPOP zulässig; d.h. es müssen die folgenden Einschränkungen beachtet werden:

- Axiome müssen die Form $\forall x_1, \ldots, x_n\ G \rightarrow P$ haben, wobei G eine Formel ist, wie sie auch für Vorbedingungen von Planoperatoren zulässig ist und P ist eine atomare Formel[50]. Beide Formeln dürfen keine freien Variablen enthalten.

- Das Prädikat von P darf weder in einem Effekt eines Planoperators noch in der Startsituation auftreten.

Daraus ergibt sich, daß nur von G auf P, aber nicht umgekehrt geschlossen werden kann. Der Aufwand für daraus resultierende Schlußfolgerungen kann so in Grenzen gehalten werden. Lt. UCPOP-Manual [Barrett *et al.*, 1995] werden die im Sachbereich vorkommenden Prädikate dementsprechend in zwei disjunkte Mengen aufgeteilt:

- *primitive Prädikate*, die in der Startsituation und in Effekten vorkommen dürfen und so direkt durch die Planausführung beeinflußt werden können.

[50] PDDL läßt auch negierte Atomformeln zu, wenn im Gegensatz zur Konvention *Closed-World Annahme*, die Behandlung echter Negation (via *Requirement* :true-negation) vereinbart wurde.

- *abgeleitete Prädikate*, die in der Konklusion von Axiomen vorkommen dürfen und nur indirekt durch die Planausführung beeinflußt werden können.

Die Möglichkeiten, Schlußfolgerungen durch Axiome an ein Planungssystem zu delegieren, sind also begrenzt. Der folgende Abschnitt befaßt sich mit der Frage, welche der eingangs genannten Abhängigkeiten durch Axiome ausgedrückt werden können und wie sie aus dem Benutzermodell abgeleitet werden können. Auf jeden Fall muß der Entwickler des Sachbereichsmodells sorgsam abwägen welche Prädikate abgeleitet werden, weil diese dann nicht mehr für Effekte von Planoperatoren eingesetzt werden können.

3.4.2.4 Ableitung von Abhängigkeiten aus dem Benutzermodell

Mit den Annahmen eines Akteurs über den Sachbereich werden auch seine Ansichten über Abhängigkeiten modelliert. Auch diese Ansichten sind also akteurspezifisch. Das folgende Beispiel zeigt, daß verschiedene Ansichten über Abhängigkeiten zu unterschiedlichen Ergebnissen des Planungsprozesses führen können. Daraus resultiert die Notwendigkeit, nicht nur die Menge der Planoperatoren, sondern den gesamten Planungssachbereich – spezifisch für den Aufgabentyp und den jeweiligen Benutzer – anhand des Benutzermodells zu erzeugen.

Es sei angemerkt, daß für die Berücksichtigung von Abhängigkeiten beim Planungsprozeß grundsätzlich die Ansichten des planenden Akteurs maßgeblich sind. Es handelt sich dabei um Abhängigkeiten zwischen den Objekten (bzw. Objektklassen) des Sachbereichs. Sie ergeben sich aus den Annahmen des Akteurs über Objekte; d.h. über Domain-Concepts. Nach den Vorgaben aus Abschnitt 3.3.2 *Repräsentation eines Benutzermodells mit Handlungspotential und Wissen im* AsTRa-*Formalismus* ist für deren Repräsentation eine BGP-MS Partition vorgesehen, die ihren Inhalt an die Partition für die Annahmen dieses Akteurs über die Handlungsmöglichkeiten anderer Akteure vererbt. Konstruiert ein Akteur A_1 also einen Plan für einen anderen Akteur A_2, so ist die View, die Annahmen von A_1 über die Handlungsmöglichkeiten von A_2 repräsentiert, nicht nur für die Erzeugung der Planoperatoren, sondern auch für das Ableiten der Abhängigkeiten angemessen. Für die Erzeugung eines Plans, den der Benutzer ausführen soll, durch das System ist dies die View SBUC.

Als Beispiel greifen wir auf das Ziel aus Abschnitt 3.4.2.2 *Abhängigkeiten im Sachbereich* zurück, daß der Benutzer einen Ausdruck einer Textverarbeitungs-Datei haben will, die sich auf einem fremden Computer befindet und deshalb vor dem Ausdruck ins lokale Dateisystem gebracht werden muß. Weiter oben wurde argumentiert, daß dies möglich ist, indem die Datei via FTP im *Binär-Modus* heruntergeladen und dann mit dem passenden Textverarbeitungssystem ausgedruckt werden kann. Der Planoperator für FTP im *Binär-Modus* (*ftp-download-file-bin*) war anwendbar, weil aus den Merkmalen der Datei und dann via Verallgemeinerung geschlossen wurde, daß die Datei eine Binär-Datei ist (Dateityp *bin-file*). Es wurde u.a. ausgenutzt, daß jede Datei für diese Textverarbeitung eine Binär-Datei ist; d.h. daß die Abhängigkeit $\forall x\ \textit{tpx-file}(x) \rightarrow \textit{bin-file}(x)$ gilt. Hat der Benutzer nun bspw. die Annahme, daß alle Dateien für das Textverarbeitungssystem Text-Dateien sind, muß der Planungssachbereich (den der Benutzer für seine Plangenerierung verwendet) anstelle der o.g. Abhängigkeit $\forall x\ \textit{tpx-file}(x) \rightarrow \textit{text-file}(x)$ berücksichtigen. Der Planoperator für FTP im *Binär-Modus* ist nun nicht mehr anwendbar, weil nicht mehr geschlossen werden kann, daß die Datei eine Binär-Datei ist, sondern daß sie eine Text-Datei ist. Stattdessen kann der Operator *ftp-download-file-text* eingesetzt werden. Damit wird aber ein Plan erzeugt, der sein Ziel nicht erreicht, weil die Datei dann fälschlicherweise im Textmodus heruntergeladen wird.

Will der Benutzer wissen *warum* ein entsprechender Plan *nicht* funktioniert, kann das System einen Plan generieren, der von den Annahmen des Benutzers ausgeht. Der Planungssachbereich für die Aufgabenspezifikation an das Planungssystem muß dann gemäß der Annahmen des Benutzers

erzeugt werden. Das System kann so die Planerzeugung des Benutzers simulieren und damit die Ursache für den falschen Plan in der unzutreffenden Abhängigkeit bzw. Zuordnung dieser speziellen Datei erkennen und eine spezifische Hilfestellung geben. Hätte das System nur einen Plan unter Berücksichtigung eigenen Annahmen erzeugt, so könnte es aus der Differenz zu dem falschen Plan des Benutzers lediglich feststellen, daß der Plan den falschen Planoperator *ftp-download-file-text* enthält und daß stattdessen *ftp-download-file-bin* verwendet werden muß. Hat der Benutzer korrekte Annahmen über die *Plan-Concepts*, muß ihm die Hilfestellung unlogisch erscheinen, weil Text-Dateien *nicht* im Binär-Modus transferiert werden.

Es sei noch einmal darauf hingewiesen, daß in dieser Arbeit die Beurteilung von Annahmen als *korrekt* bzw. *falsch* grundsätzlich in Relation zu anderen Annahmen getroffen wird und daß – sofern nichts anderes angegeben wird – die Annahmen des Systems den Bezug darstellen. Als *falsche Annahmen* werden also Annahmen bezeichnet, die nicht zu den Annahmen des Systems konform sind (vgl. Kap. 3.1 *Handlungspotential und Wissen von Benutzern*).

Für die Aufgabenspezifikation an das Planungssystem müssen die Abhängigkeiten im Planungssachbereich also in gleicher Weise wie die Planoperatoren aus dem Benutzermodell abgeleitet werden. Ein direktes Ableiten, wie bei den Planoperatoren, ist für Abhängigkeiten allerdings nicht sinnvoll. Der Grund liegt darin, daß (praktisch) jeder Sachbereich eine Vielzahl von Abhängigkeiten enthält, die zwar zumindest teilweise eine Entsprechung in der beschreibungslogischen Repräsentation haben, die aber zum großen Teil für die Plangenerierung irrelevant sind. Für den Planungssachbereich ist eine Beschränkung auf die relvanten Abhängigkeiten zweckmäßig.

Es bietet sich an, den Entwickler des Sachbereichsmodells, der bspw. bereits über die Relevanz von Vorbedingungs-Komponenten von Planoperatoren entscheiden soll (vgl. Kap. 2.1 *Handlungsplanung*), auch entscheiden zu lassen, welche *Abhängigkeiten* für die Plangenerierung relevant sind. Für die Erzeugung des Planungssachbereichs müssen *davon* dann diejenigen Abhängigkeiten berücksichtig werden, die sich auch aus dem Benutzermodell ableiten lassen. Da die Ansichten über Abhängigkeiten unterschiedlich sein können, muß der Entwickler des Sachbereichsmodells auch solche Abhängigkeiten berücksichtigen, die er für falsch hält, von denen er aber annimmt, daß irgend ein Akteur sie für richtig halten *könnte*. Hierfür sei die Bezeichnung *Menge der relevanten potentiellen Abhängigkeiten* oder kurz *potentielle Abhängigkeiten* eingeführt. Im Beispiel oben muß die Menge also sowohl $\forall x\ \mathit{tpx\text{-}file}(x) \rightarrow \mathit{bin\text{-}file}(x)$ als auch $\forall x\ \mathit{tpx\text{-}file}(x) \rightarrow \mathit{text\text{-}file}(x)$ enthalten. Wird (gem. o.g. Beispiel) aus der Sicht des Systems geplant, wird nur die erste, bei einer Planung aus Benutzersicht nur die zweite Abhängigkeit berücksichtigt und als Teil der Aufgabenspezifikation an das Planungssystem übergeben.

Im Rest des Abschnitts wird für jede der fünf Arten von Abhängigkeiten gezeigt, wie sie aus der beschreibungslogischen Repräsentation des Benutzermodells abgeleitet werden kann, wie der Entwickler des Sachbereichsmodells sie beschreiben kann und wie sie dann für den Planungssachbereich berücksichtigt wird. Als Möglichkeit zur Beschreibung von Abhängigkeiten stehen die bereits genannten UCPOP-Axiome zur Verfügung, die – sofern sie durch das Benutzermodell verifiziert werden können – ohne Änderung in den Planungssachbereich übernommen werden. Abhängigkeiten durch ausschließende Typzugehörigkeit lassen sich nicht durch Axiome ausdrücken. Hierfür besteht die Möglichkeit *implizite Effekte* zu definieren. Deren Komponenten werden – wiederum sofern durch das Benutzermodell verifizierbar – dem Effekt der zugehörigen Planoperatoren hinzugefügt (s.u).

3.4.2.4.1 Abhängigkeit der Typzugehörigkeit zu verschieden abstrakten Typen In Abstraktionshierarchien gehört jedes Element eines Typs S auch zu allen Verallgemeinerungen dieses Typs. Ist A eine solche Verallgemeinerung von S, läßt sich dies in Beschreibungslogik durch Concepts S und A ausdrücken, für die die Subsumptionsbeziehung

$$S \Rightarrow A$$

gilt, da $S \Rightarrow A$ gdw. $S^I \subseteq A^I$.

Dementsprechend gilt die Teilmengenbeziehung auch für die zugehörigen Prädikate des Planungssachbereichs; d.h.

$$\forall x\, S(x) \rightarrow A(x)$$

Ist A kein primitives Prädikat, kommt A also nicht in einem Effekt oder der Startsituation vor, kann die Abhängigkeit durch folgendes Axiom beschrieben werden:

PDDL	UCPOP

`(:axiom`
`  :vars (?x - S)`
`  :context (S ?x)`
`  :implies (A ?x))`
	`(:axiom <Name des Axioms>`
	`  :context (S ?x)`
	`  :implies (A ?x))`

Variablen, die in einem Axiom nicht explizit gebunden sind (hier ?x), sind per Vereinbarung in beiden Sprachen implizit allquantifiziert. PDDL fordert die explizite Angabe solcher Variablen nach dem Schlüsselwort `:vars`.

3.4.2.4.2 Abhängigkeit eines Objektmerkmals vom Typ des Objekts Ein Objektmerkmal kann in Beschreibungslogik durch eine Rolle, d.h. durch eine Beziehung zu einer anderen Objektklasse, modelliert werden. Der Wertebereich der Rolle wird dann auf diese Klasse, d.h. auf Objekte dieser Klasse bzw. dieses Typs, beschränkt. Der beschreibungslogische Ausdruck:

$$\forall R : T_2$$

bezeichnet die Menge der Objekte, die ausschließlich zu Objekten des Typs T_2 in Relation R^I stehen. Subsumiert ein solcher Ausdruck einen Typ T_1, so kann für ein Objekt y geschlossen werden, daß es den Typ T_2 hat, wenn ein Objekt x vom Typ T_1 in Relation R^I zu y steht. Bspw. kann geschlossen werden, daß es sich bei einem Dateinamen in einem *lokalen Verzeichnis* um einen *lokalen Namen* handelt, weil *lokale Verzeichnisse* ausschließlich *lokale Namen* für Dateien oder Verzeichnisse enthalten (d.h. in Relation *contains* stehen; vgl. a. Kap. 5.2.2.1.3 *Lokale Dateien*).

Eine solche Schlußfolgerung kann wieder dem Planungssystem überlassen werden, wenn T_2 kein primitives Prädikat im Planungssachbereich ist. Die prädikatenlogische Beschreibung der Abhängigkeit:

$$\forall x, y\, (T_1(x) \wedge R(x,y)) \rightarrow T_2(y)$$

kann aus der beschreibungslogischen Repräsentation abgeleitet werden, wenn

$$T_1 \Rightarrow \forall R : T_2\ .$$

In diesem Falle ist jedes $x \in T_1^I$ auch Element von $\{d \mid R^I(d) \subseteq T_2^I\}$. In der (gem. Def. 3.3 und 3.4 S. 141) zugehörigen prädikatenlogischen Darstellung bedeutet dies:

$$\forall x\, T_1(x) \rightarrow (\forall y\, R(x,y) \rightarrow T_2(y))$$

was äquivalent zur gewünschten Formel ist.

Eine direkte Modellierung der Abhängigkeit als UCPOP-Axiom ist allerdings nicht möglich, weil in der Prämisse eines UCPOP-Axioms keine Variable ungebunden sein darf. Die Abhängigkeit kann aber durch das Axiom

PDDL	UCPOP

`(:axiom :vars (?y)`
`  :context (`
`  exists (T1 ?x) (R ?x ?y))`
`  :implies (T2 ?y))`
	`(:axiom <Name des Axioms>`
	`  :context (`
	`  exists (T1 ?x) (R ?x ?y))`
	`  :implies (T2 ?y))`

beschrieben werden, das der Formel

$$\forall y\ (\exists x\ T_1(x) \wedge R(x,y)) \rightarrow T_2(y)$$

entspricht, die zur o.g. Beschreibung der Abhängigkeit äquivalent ist.

3.4.2.4.3 Abhängigkeit des Typs eines Objekts von einem Objektmerkmal Eine Schlußfolgerung von einem Objektmerkmal auf den Typ des Objekts ist möglich, wenn das Merkmal den Typ hinreichend genau kennzeichnet. Dies läßt sich in Beschreibungslogiken durch definierte Concepts ausdrücken. Ein solches Concept kann als Spezialisierung eines Basistyps durch zusätzliche, kennzeichnende Merkmale definiert werden. Bspw. definiert

$$S \doteq B \sqcap \exists R : AT$$

S als Spezialisierung von B. Kennzeichnendes Merkmal ist die werterestringierte Rolle, weil per Definition $\exists R : AT \sqcap B \Rightarrow S$ gilt und damit ein Objekt x vom Basistyp B, das in Relation R^I zu einem Objekt vom Typ AT steht, eindeutig als Objekt des spezielleren Typs S identifiziert werden kann. Im Beispiel können wir schließen, daß es sich bei einer Datei um eine Text-Datei handelt, wenn sie zu einem Objekt vom Typ *file-type-text* in Relation *file-type* steht.

Eine prädikatenlogische Beschreibung dieser Abhängigkeit im Planungssachbereich kann aus der beschreibungslogischen Repräsentation abgeleitet werden, wenn $\exists R : AT \sqcap B \Rightarrow S$ gilt und wieder der durch Def. 3.3 und 3.4 festgelegte Zusammenhang der beiden Repräsentationsformen zugrunde gelegt wird. $\exists R : AT \sqcap B \Rightarrow S$ bedeutet, daß jedes $x \in \{d \mid R^I(d) \cap AT^I \neq \varnothing\} \cap B^I$ auch Element von S^I sein muß. Dabei enthält die Menge $\{d \mid R^I(d) \cap AT^I \neq \varnothing\}$ genau die Elemente, für die mindestens ein Element aus AT^I existiert, zu dem sie in Relation R^I stehen. Durch Einsetzen der zu den Concepts und Rollen gehörenden Prädikate des Planungssachbereichs ergibt sich also

$$\forall x\ (\exists y\ R(x,y) \wedge AT(y) \wedge B(x)) \rightarrow S(x)$$

Die zugehörigen Axiome lauten:

PDDL	UCPOP

`(:axiom`
`  :vars (?x - B)`
`  :context (`
`  exists (?y - AT) (R ?x ?y))`
`  :implies (S ?x))`
	`(:axiom <Name des Axioms>`
	`  :context (`
	`  exists (AT ?y)`
	`  (and (R ?x ?y) (B ?x)))`
	`  :implies (S ?x))`

Die Ableitung der prädikatenlogischen Beschreibung und der Axiome kann auch angewendet werden, wenn die Typzugehörigkeit nur anhand mehrerer Merkmale festgestellt werden kann. Aus der Subsumption

$$B \sqcap \exists R_1 : AT_1 \sqcap \ldots \sqcap \exists R_n : AT_n \Rightarrow S$$

die grundsätzlich gilt, wenn

$$S \doteq B \sqcap \exists R_1 : AT_1 \sqcap \ldots \sqcap \exists R_n : AT_n \ ,$$

kann

$$\forall x\ (\exists y_1\ R_1(x,y_1) \wedge AT_1(y_1) \wedge \ldots \wedge \exists y_n\ R_n(x,y_n) \wedge AT_n(y_n) \wedge B(x)) \rightarrow S(x)$$

abgeleitet werden. In dieser Arbeit wird von der Typzuordnung anhand mehrerer Merkmale allerdings kein Gebrauch gemacht.

3.4.2.4.4 Ausschließende Eigenschaften bzw. Typzugehörigkeiten Die Ausführungen in diesem Abschnitt beschränken sich auf die Behandlung ausschließender Typzugehörigkeiten, weil ausschließende Eigenschaften darauf zurückgeführt werden können. Bspw. können die möglichen Betriebszustände eines Druckers aus dem Beispiel oben durch eine Relation *im-Zustand* zwischen einem *Drucker* und einem *Betriebszustand* modelliert werden, wobei ein Betriebszustand genau einen der drei Typen *bereit*, *aktiv* oder *ausgeschaltet* haben kann.

In Beschreibungslogiken können ausschließende Typzugehörigkeiten durch disjunkte Concepts modelliert werden. Um, wie im Beispiel des Betriebszustandes, auszudrücken, daß jedes Exemplar eines (Basis-)Concepts B auch Exemplar *genau* eines von wechselseitig disjunkten Sub-Concepts $T_1 \ldots T_n$ von B ist, kann B als definiertes Concept und Disjunktion[51] der Sub-Concpets definiert werden:

$$B \doteq T_1 \sqcup \ldots \sqcup T_n$$

In der zugehörigen prädikatenlogischen Darstellung kann durch die Formel

$$(\forall x\ T_1(x) \rightarrow (\neg T_2(x) \wedge \ldots \wedge \neg T_n(x))) \quad \wedge$$
$$\ldots \quad \wedge$$
$$(\forall x\ T_n(x) \rightarrow (\neg T_1(x) \wedge \ldots \wedge \neg T_n - {}_1(x)))$$

ausgedrückt werden, daß ein Element des Planungssachbereichs *höchstens einen* der Typen $T_1 \ldots T_n$ haben kann. Die Formel läßt sich aber nicht durch Axiome ausdrücken, wenn die Einschränkungen berücksichtigt werden, die im vorangegangenen Abschnitt für Axiome in Planungssystemen angegeben wurden, da dabei die Konklusion einer Implikationen eine atomare Formel sein muß. Gehen wir von einer korrekt modellierten Startsituation aus, in der keine Abhängigkeit verletzt wird, kann die korrekte Behandlung von wechselseitig ausschließenden Typen $T_1 \ldots T_n$ aber dennoch gewährleistet werden, wenn jeder Planoperator, der eine Effektkomponente $T_i(x)$ mit $i \in \{1,\ldots,n\}$ enthält, gleichzeitig auch für jedes $j \in \{1,\ldots,n\} \backslash \{i\}$ die Komponente $\neg T_j(x)$ enthält.

Soll ausgedrückt werden, daß ein Element des Planungssachbereichs mit Typ B auch *genau einen* der (Sub-)Typen $T_1 \ldots T_n$ hat, läßt sich das analog zur beschreibungslogischen Darstellung durch die zusätzliche Formel

$$\forall x\ B(x) \rightarrow T_1(x) \vee \ldots \vee T_n(x)$$

ausdrücken. Auch diese Formel kann nicht als UCPOP-Axiom dargestellt werden, kann aber durch folgende Bedingung an die Planoperatoren sichergestellt werden: Ein Planoperator mit einer Effektkomponente $\neg T_i(x)$ mit $i \in \{1,\ldots,n\}$ muß auch ein $T_j(x)$ mit $j \in \{1,\ldots,n\} \backslash \{i\}$ oder $\neg B(x)$ enthalten. Im ersten Fall ändert sich die Typzugehörigkeit von x lediglich im Rahmen der T_i, während im zweiten Fall auch die Zughörigkeit zum Basistyp B verneint wird.

[51] Der in dieser Arbeit eingesetzte SB-ONE-Formalismus sieht zwar keine explizite Disjunktion vor. Eine logisch äquivalente Beschreibung ergibt sich, wenn $T_1, \ldots T_n$ jeweils als Sub-Concepts von B definiert und als *überdeckend* (exhaustive) markiert werden (vgl. [Profitlich, 1990, Kap. 2.2.1.2 u. 3.2.2]).

Die zusätzlichen Effektkomponenten, die die korrekte Behandlung ausschließender Typzugehörigkeiten gewährleisten, gleichen die begrenzten Schlußfolgerungsfähigkeiten von Planungssystemen aus, sind aber aus logischer Sicht redundant. In Hinblick auf die eingangs genannte Zielsetzung, den Modellierungsaufwand durch Ausnutzen von Abhängigkeiten im Sachbereich zu verringern, werden solche Effektkomponenten in dieser Arbeit nicht explizit modelliert, sondern den Planoperatoren erst bei der Erzeugung des Planungssachbereichs hinzugefügt. Um diese Erzeugung zu steuern, können sogenannte *implizite Effekte* definiert werden. Abb. 3.23 beschreibt die syntaktische Form in EBNF[52].

impl-effect-def	=	"(" ":conseq" *predicate-name* ":implies" "(" {*lit*}$^+$ ")" ")" .
lit	=	"(" "not" *predicate-name* ")" .
predicate-name		Name nach LISP-Syntax

Abbildung 3.23: EBNF-Syntax für die Definition von impliziten Effekten.

Die Notation steht für eine Implikation der Form

atomare Formel $\rightarrow$ Konjunktion von negierten Atomformeln

Alle Teilformeln dürfen nur einstellige Prädikate enthalten und müssen das gleiche Argument haben. Auf die explizite Angabe der Argumente wird deshalb verzichtet. Es muß auch nicht explizit angegeben werden, daß es sich bei der Konklusion um eine Konjunktion handelt. In der Konjunktion sind nur negierte Atomformeln zulässig, wie es für die Modellierung der ausschließenden Typzugehörigkeit notwendig ist. Sollen auch nichtnegierte Atomformeln zugelassen werden, kann die Syntax von *lit* wie folgt geändert werden:

lit = *predicate-name* | "(" "not" *predicate-name* ")" .

Implizite Effekte *könnten* dann auch anstelle von Axiomen für die erste Art von Abhängigkeiten, der Typzugehörigkeit zu verschieden abstrakten Typen, eingesetzt werden. In dieser Arbeit wird von dieser Möglichkeit aber kein Gebrauch gemacht.

Analog zur Vorgehensweise bei Axiomen, werden die *möglichen* und für die Plangenerierung *sinnvollen* impliziten Effekte *benutzerunabhängig* definiert. Bei der Erzeugung des Planungssachbereichs wird dann für jeden impliziten Effekt geprüft, ob die dazu gehörende Implikation aus der für die jeweilige Aufgabe relevanten View des Benutzermodells abgeleitet werden kann. Im negativen Falle wird der implizite Effekt ignoriert. Im positiven Fall wird jedem Planoperator, der aus dem Benutzermodell erzeugt wurde und der eine Effektkomponente hat, die gleich der Prämisse des impliziten Effekts ist, die Konklusion des impliziten Effekts hinzugefügt. Dabei wird jeweils die Variable der Effektkomponente verwendet. Da ein impliziter Effekt der Form:

$$T_1 \rightarrow \neg T_2 \wedge \ldots \wedge \neg T_n$$

lediglich eine bequemere Schreibweise für:

$$(T_1 \rightarrow \neg T_2) \wedge \ldots \wedge (T_1 \rightarrow \neg T_n)\ ,$$

d.h. für mehrere implizite Effekte mit jeweils nur einem negierten Prädikatsymbol ist, wird jeder Faktor der Konklusion einzeln überprüft und ggf. dem Effekt eines Planoperators hinzugefügt.

Eine Implikation der Form:

$$\forall x\ T_1(x) \rightarrow \neg T_2(x)$$

[52] vgl. [Jensen und Wirth, 1985] und die Beschreibung von EBNF im Anhang A.1 S. 300 dieser Arbeit.

kann aus der beschreibungslogischen Repräsentation abgeleitet werden, wenn die Disjunktheit der beiden zugehörigen Concepts festgestellt werden kann.

Im Beispielsachbereich dieser Arbeit werden implizite Effekte nur im Zusammenhang mit der Erzeugung von Objekten eingesetzt, auf die weiter unten noch in einem eigenen Abschnitt (3.4.2.5) eingegangen wird.

Es sei angemerkt, daß durch implizite Effekte lediglich sichergestellt werden kann, daß ein Objekt nur *höchstens* einen von gegenseitig ausschießenden Typen hat. Einen sinnvollen Automatismus, der sicherstellt, daß ein Objekt *genau* einen Typ aus einer Menge von Typen hat, kann es aber nicht geben. Verneint ein Planoperator die Zugehörigkeit eines Objekts zu einem solchen Typ, kann nicht automatisch festgelegt werden, *welchem* der alternativen Typen das Objekt zugeordnet werden soll. Diese Entscheidung muß der Entwickler des Sachbereichsmodells treffen. Die explizite Negation der Typzugehörigkeit durch den Planoperator kann dann allerdings wieder entfallen. Als wünschenswerter Nebeneffekt ergibt sich, daß die Repräsentation des Effekts von Planoperatoren in allen Fällen, die ausschließende Typzugehörigkeiten betreffen, ohne explizite Negation auskommt (vgl. a. Abschnitt 3.2.2.1 *Spezifika der Repräsentation in SB-ONE*).

3.4.2.4.5 Komplementäre Typzugehörigkeiten Im vorangegangen Abschnitt wurde vom allgemeinen Fall ausschließender Typzugehörigkeiten ausgegangen, bei dem ein Objekt höchstens oder genau einen von mehreren alternativen Typen haben konnte. Die Vorgehensweise ist zwar auch anwendbar, wenn ein Objekt eines Basistyps immer einen von genau zwei alternativen Sub-Typen hat. Da die beiden Sub-Typen dann (in Bezug zum Basistyp) komplementär sind, *kann* aber auf die explizite Modellierung eines der beiden Typen verzichtet werden. Im Beispielsachbereich sind die Transfermodi eines FTP-Programms *Binär-Modus* und *Text-Modus* sowie die Status *freigegeben* und *nicht-freigegeben* für Dateiverzeichnisse solche komplementären Typen. Im Planungssachbereich wird lediglich die Entsprechung für den *Binär-Modus* – das Prädikat *ftp-binary-mode* – bzw. für die Freigabe – das Prädikat *fs-name-for-mount* – verwendet. Ist *fs-name-for-mount* für ein Verzeichnis wahr, so ist es freigegeben, ansonsten nicht. Ein Beenden einer Freigabe muß demnach durch das Negieren des Prädikats *fs-name-for-mount* für das Verzeichnis durch den Effekt des zugehörigen Planoperators modelliert werden.

Der Vorteil einer solchen Vorgehensweise besteht darin, daß die Zahl der modellierten Prädikate und der Effektkomponenten der beteiligten Planoperatoren verringert wird. Allerdings kann der Verzicht auf die Modellierung beider Typen nachteilig sein, wenn die zugehörigen Begriffe im Sachbereich tatsächlich beide vorkommen und der Benutzer dementsprechend auch falsche Vorstellungen davon haben kann. Bspw. ist eine Erklärung, *daß Binär-Modus* und *Text-Modus* komplementär sind, nicht ohne weiteres möglich, wenn das Sachbereichsmodell den Begriff *Text-Modus* gar nicht enthält. Die gewählte Modellierung ist also lediglich als Beispiel für die Verwendung negierter einstelliger Prädikate in Effekt oder Vorbedingung von Planoperatoren zu verstehen. Der Entwickler des Sachbereichsmodells kann frei darüber entscheiden, welche Art der Modellierung für den Anwendungszweck angemessen ist.

Für die konkrete Modellierung mit SB-ONE kann aber nicht auf die Modellierung jeweils beider komplementärer Concepts verzichtet werden. Allerdings kann eines davon markiert werden, sodaß es nicht allgemein verwendet wird (vgl. Abschnitt 3.2.2.1 *Spezifika der Repräsentation in SB-ONE*). Kommt ein so markiertes Concept als Werterestriktion eines Vorbedinungs- oder Effekt-Attributs vor, wird bei der Erzeugung der Planoperatoren aus dem Benutzermodell nicht das zugehörige Prädikat erzeugt (vgl. Abschitt 3.4.2.7 *Erzeugung des aktiven Teils eines Planungssachbereichs*), sondern die Negation des Prädikats, das zu seinem Komplementär-Concept gehört. Im Sinne von Definition 3.3 S. 141 ist das Concept also irrelevant.

3.4.2.4.6 Zusammenfassung Abhängigkeiten Die folgende Tabelle 3.4 zeigt die vier Arten von Abhängigkeiten des Sachbereichs, die als Axiom oder impliziter Effekt definiert werden können, als prädikatenlogische Formel und in der Syntax für die Definition der Abhängigkeiten (konkrete Form); dazu die Bedingung, unter denen die Abhängigkeit aus der beschreibungslogischen Repräsentation einer BGP-MS-View ableitbar ist.

	Prädikatenlogische Formel	Konkrete Form	Aus BGP-MS-View ableitbar wenn

1	$\forall x\, S(x) \rightarrow A(x)$	`(:axiom <Name des Axioms>`
		`  :context (S ?x)`
		`  :implies (A ?x))`
			$S \Rightarrow A$

2	$\forall y\, (\exists x\, T_1(x) \wedge R(x,y)) \rightarrow T_2(y)$	`(:axiom <Name des Axioms>`
		`  :context (`
		`  exists (T1 ?x)`
		`  (R ?x ?y))`
		`  :implies (T2 ?y))`
			$T_1 \Rightarrow \forall R : T_2$

3	$\forall x\, (\exists y\, R(x,y) \wedge AT(y) \wedge B(x)) \rightarrow S(x)$	`(:axiom <Name des Axioms>`
		`  :context (`
		`  exists (AT ?y)`
		`  (and (R ?x ?y) (B ?x)))`
		`  :implies (S ?x))`
			$\exists R : AT \sqcap B \Rightarrow S$

4	$\forall x\, T_1(x) \rightarrow (\neg T_2(x) \wedge \ldots \wedge \neg T_n(x))$	`(:conseq T1 :implies` `  ((not T2)...(not Tn)))`	$T_1 \parallel T_2$ und ... und $T_1 \parallel T_n$

Dabei entsprechen bzw. bedeuten

Dabei stehen	in Formel und konkreter Form für	im Concept-Ausdruck für
$S, A, T_1, T_2, T_n, AT, B$	einstellige Prädikatensymbole	Concept-Namen
R	zweistelliges Prädikatensymbol	Rollen- oder Attribut-Name
x, y	Variablensymbole	-
$\parallel$	-	Disjunktheit zweier Concepts

Die konkrete Form der Axiome (1-3) folgt den syntaktischen Regeln von UCPOP; die des impliziten Effekts (4) der Syntax aus Abb. 3.23 des Abschnitts 3.4.2.4.4.

Tabelle 3.4: Zusammenfassung Abhängigkeiten

3.4.2.5 Erzeugen von Objekten

Eine übliche Modellierung der Erzeugung von Objekten ist die Zuordnung einer vorher typlosen Konstante zu einem Typ für existierende Objekte. Die Verarbeitung vereinfacht sich, wenn anstelle der Verwendung einer *typlosen* Konstante vereinbart wird, daß ein bestimmter Typ für "nicht existierende Objekte" steht. Im Beispielsachbereich ist dafür der Typ bzw. das Concept *non-existent* vorgesehen, während *object* das allgemeinste Concepts ist, das existierende Objekte bezeichnet. Soll ein Planoperator ein Objekt erzeugen, benötigt er ein "nicht existierendes Objekt"; d.h. seine Vorbedingung muß

eine atomare Formel mit dem Prädikat *non-existent* enthalten. Nach Ausführung des Operators muß für die gleiche Variable das Prädikat *object* wahr sein. Da i.a. nicht irgend ein Objekt erzeugt werden soll, sondern ein spezifisches, enthält der Effekt dann ein Prädikat eines Subtyps von *object*. Für das Ausdrucken eines Dokuments (*app-print*) ist dies bspw. *paperdoc* für *Papierdokument* (vgl. Abb. 3.27 u. Kap. 5.2.2.2 *Erzeugen von Objekten*). Außerdem darf *non-existent* dann *nicht* mehr gelten. Im Beispielsachbereich ist deshalb für jeden Typ, von dem Objekte erzeugt werden können, ein impliziter Effekt definiert, der genau dies bewirkt (vgl. a. Kap. 5.2.2.2 *Erzeugen von Objekten*).

Nachteilig ist, daß für ein Planungsproblem ausreichend Konstanten des Typs *non-existent* im Voraus definiert werden müssen. Das Planungssystem UCPOP bietet die Möglichkeit, solche Konstantensymbole *bei Bedarf* zu erzeugen. UCPOP ermöglicht dies über *Facts*, die eine Schnittstelle zwischen Planungs- und LISP-System realisieren. Die Lisp-Form

```
(:fact (new-var ?x)
       (when (variable:variable? ?x) (list (setb ?x (gensym "obj-"))))
```

erzeugt in der Beispielmodellierung ein neues Symbol, wenn das Planungssystem das Ziel hat, daß *new-var* für eine Variable gilt. Da die Erzeugung neuer Konstantensymbole für den Planungsprozeß rein programmtechnischer Natur ist, findet sich dafür keine Entsprechung im Benutzermodell. Die Erzeugung von Variablen steht dem Planungsprozeß unabhängig davon zur Verfügung, welche View des Benutzermodells für die Plangenerierung verwendet wird. Darüber hinaus sind die mit der Erzeugung von Objekten verbundenen Concepts *object*, *non-existent*, *to-create* (s.u) und *new-var* mit den genannten Eigenschaften in allen Views der Benutzermodellierungswissensbasis (BMWB) enthalten.

Zu Beginn dieses Kapitels wurde gesagt, daß das Ziel einer Veränderung nur durch die Differenz zwischen Startsituation und Ziel ausgedrückt werden kann und als Beispiel angegeben, daß ein neues Objekt erzeugt werden soll. In diesem Fall muß die Startsituation doch eine Konstante für das "zu erzeugende" Objekt enthalten. Hierfür ist im Beispielsachbereich der Typ *to-create* vorgesehen. Die Typen *to-create* und *new-var* sind beides Subtypen von *non-existent* (vgl. a. Abb. 5.7 in Kap. 5.2.2 *Der Beispielsachbereich*). Für den Planungssachbereich wird dies durch die Axiome $\forall x\ \textit{to-create}(x) \rightarrow \textit{non-existent}(x)$ und $\forall x\ \textit{new-var}(x) \rightarrow \textit{non-existent}(x)$ modelliert.

3.4.2.6 Vereinfachte beschreibungslogische Inferenzen

Im Abschnitt 3.4.2.4 *Ableitung von Abhängigkeiten aus dem Benutzermodell* wurde vorgeschlagen, die *möglichen* Abhängigkeiten eines Sachbereichs unabhängig vom Benutzermodell durch Axiome und implizite Effekte zu modellieren. Erst bei Bedarf – d.h. bei der Erzeugung eines Planungssachbereichs – sollte überprüft werden, welche der Abhängigkeiten sich aus der beschreibungslogischen Repräsentation der verwendeten View der BMWB ableiten lassen. Die Überprüfung wurde auf beschreibungslogische Inferenzen – den Subsumptionstest und die Feststellung von Disjunktheit – zurückgeführt. Insgesamt waren die folgenden Test erforderlich (vgl. Abschnitt 3.4.2.4.6 *Zusammenfassung Abhängigkeiten*):

(1) $S \Rightarrow A$

(2) $T_1 \Rightarrow \forall R : T_2$

(3) $\exists R : AT \sqcap B \Rightarrow S$

(4) $T_1 \parallel T_2$

Die Überprüfung der Subsumptionsbeziehungen zwischen zwei Concept-Ausdrücken ist neben Klassifikation und Vererbung die wichtigste Inferenz eines beschreibungslogischen Wissensrepräsentationssystems. Allerdings stellen die realisierten Systeme einen Kompromiß zwischen Ausdrucksfähigkeit und Vollständigkeit der Inferenzen dar. Bspw. ist das Subsumptionsproblem für die in

[Patel-Schneider und Swartout, 1993] vorgeschlagene Sprache unentscheidbar[53]. Das in dieser Arbeit eingesetzte Wissensrepräsentationssystem SB-ONE stellt keine direkten Funktionen zur Überprüfung der Disjunktheit oder der Subsumptionsbeziehung zwischen zwei Concept-Ausdrücken zur Verfügung (vgl. [Profitlich, 1989], [Profitlich, 1990]). Allerdings kann ein Concept klassifiziert werden und die Subsumption dann anhand der Sub-Concept-Relation ermittelt werden. Die Fälle (2) und (3) lassen sich auf diese Weise nur überprüfen, wenn für die Concept-Ausdrücke Hilfs-Concepts definiert werden. Da die Überprüfung der Subsumptionsbeziehungen bei jeder Erzeugung einer Spezifikation an das Planungssystem und für jede definierte Abhängigkeit erforderlich ist, führt dieses Vorgehen zu einem inakzeptabel hohen Rechenaufwand.

Als Alternative seien für alle o.g. Tests Bedingungen angegeben, unter denen die Subsumptionsbeziehung bzw. die Disjunktheit besteht, wobei die Ausdrucks- und Inferenzfähigkeit von SB-ONE berücksichtigt wird. Die Beziehungen können allerdings auch bestehen, wenn die Bedingungen *nicht* erfüllt sind; d.h. ein Test auf dieser Basis ist zwar korrekt, aber unvollständig. Allerdings werden damit viele praktisch relevante Fälle abgedeckt.

3.4.2.6.1 $S \Rightarrow A$ Die Beziehung besteht, sofern $S = A$ oder S ein Sub-Concept von A ist. Sofern beide Concepts korrekt klassifiziert sind, ist S *genau dann* ein Sub-Concept von A, wenn $S \Rightarrow A$.

3.4.2.6.2 $T_1 \Rightarrow \forall R : T_2$ Hat T_1 eine Rolle R mit Werterestriktion VR und gilt $VR \Rightarrow T_2$, so besteht die Subsumptionsbeziehung. Da SB-ONE Rollen automatisch an Sub-Concepts vererbt, ist es nicht erforderlich, die Überprüfung auch für Super-Concepts von T_1 durchzuführen.

3.4.2.6.3 $\exists R : AT \sqcap B \Rightarrow S$ Die Beziehung besteht, wenn $B \sqcap \exists R : AT$ eine hinreichende Bedingung für die Zugehörigkeit zu S darstellt, also S als *definiertes* Concept (*Defined Concept*) vereinbart wurde[54], bspw. durch $S \doteq B \sqcap \exists R : AT$. *Existenzquantifizierte* Rollen stellt SB-ONE allerdings nicht uneingeschränkt zur Verfügung. Typangaben bzw. Werterestriktionen sind nur im Zusammenhang mit *Allquantifizierung* möglich (vgl. [Kobsa, 1991]). Wird S gleich $B \sqcap \exists R \sqcap \forall R : AT$ definiert, kann dies aber wiederum eine zu starke Einschränkung darstellen, weil dazu nur solche Objekte gehören, die für R *ausschließlich* Rollenfüller vom Typ AT haben. Ob diese Einschränkung für die Definition von S zu stark ist, hängt vom Concept B ab. Ist für B bspw. nur ein Rollenfüller für R zulässig ($B \Rightarrow$=1R), so sind die Concept-Ausdrücke

$B \sqcap \exists R \sqcap \exists R : AT$ und

$B \sqcap \exists R \sqcap \forall R : AT$ und auch

$B \sqcap R : AT$

äquivalent. Beim letzten Ausdruck steht R für ein Attribut (bzw. in SB-ONE für eine notwendige Rolle mit Anzahlrestriktion [1 1]) mit Werterestriktion AT.

Der Ausdruck $\exists R : AT \sqcap B$ wird demnach von S subsumiert, wenn

a) S definiertes Sub-Concept von B ist

b) S sich von B höchstens durch die Werterestriktion VR der Rolle R unterscheidet

c) R ein Attribut ([1 1] anzahlbeschränkte Rolle) von B ist

d) AT von VR subsumiert wird.

[53] Bspw. läßt die KRSS-Spezifiaktion uneingeschränkte Role-Value-Maps zu, die nach [Schmidt-Schauß, 1989] zur Unentscheidbarkeit des Subsumptionsproblems führen.

[54] vgl. Kap. 3.2.1 *Beschreibungslogiken*.

$\exists R : AT \sqcap B \Rightarrow S$ gilt (unter der Bedingung $B \Rightarrow$=1R) auch dann, wenn S als definiertes Conept $S \doteq Z \sqcap \forall R : AT$ vereinbart ist und sich Concept Z von Concept B nur durch eine Werterestriktion für R unterscheidet, die allgemeiner ist als AT; d.h. $Z = B \sqcap \forall R{:}AT_x$ und $AT \Rightarrow AT_x$. In diesem Fall sind die Ausdrücke $B \sqcap \forall R{:}AT$ und $Z \sqcap \forall R{:}AT$ äquivalent und können bei der Definition von S alternativ verwendet werden.

Dieser Spezialfall kann im Beispielsachbereich genutzt werden, um das Axiom

$$\forall f\ \exists t\ \textit{file-type-tpx}(t) \wedge \textit{file-type}(f,t) \wedge \textit{fs-file}(f)) \rightarrow \textit{tpx-file}(f)$$

zu verifizieren, wozu nachgewiesen werden muß, daß *tpx-file* den Ausdruck

$$\exists \textit{file-type} : \textit{file-type-tpx} \sqcap \textit{fs-file}$$

subsumiert. *tpx-file* ist durch

$$\textit{tpx-file} \doteq \textit{bin-file} \sqcap \textit{file-type} : \textit{file-type-tpx}$$

definiert (vgl. Anhang B.1 *Domain-Concepts*), wobei

$$\textit{bin-file} \doteq \textit{fs-file} \sqcap \textit{file-type} : \textit{file-type-bin} \qquad \text{und}$$
$$\textit{fs-file} \sqsubseteq \ldots \sqcap \textit{file-type} : \textit{file-type-c} \qquad \text{und}$$
$$\textit{file-type-tpx} \Rightarrow \textit{file-type-bin} \Rightarrow \textit{file-type-c}$$

3.4.2.6.4 $T_1 \parallel T_2$ T_1 und T_2 sind disjunkt, wenn T_1 und T_2 oder Super-Concepts davon als disjunkte Concepts definiert sind.

3.4.2.7 Erzeugung des aktiven Teils eines Planungssachbereichs

Für ein Planungssystem sind die Planoperatoren (bzw. in PDDL die Actions) die Basisbausteine, die Veränderungen *in der modellierten Welt* bewirken können. In Kapitel 3.2.2 *Planoperatoren in Beschreibungslogik* wurde gezeigt, wie ein Planoperator durch ein Plan-Concept in Beschreibungslogik und insbesondere in SB-ONE repräsentiert werden kann. Für die Spezifikation einer Aufgabe an ein Planungssystem muß nun der umgekehrte Weg beschritten werden. Aus der Menge der Plan-Concepts, die in der für den jeweiligen Aufgabentyp relevanten BGP-MS View enthalten sind, muß eine Menge von Planoperatoren erzeugt werden. Diese Menge bildet dann den aktiven Teil des für diese Aufgabe zu definierenden Planungssachbereichs.

Im Demonstrationsprototyp übernimmt die Funktion *knpp-view-2-op-set* die Erzeugung der Planoperatoren für die Funktion *knpp-view-2-planning-domain*, die einen Planungssachbereich mit Planoperatoren, Axiomen und Facts für das Planungssystem erzeugt. Da in dieser Arbeit das *nichthierarchische* Planungssystem UCPOP eingesetzt wird, brauchen nur solche Plan-Concepts berücksichtigt zu werden, für die keine Spezialisierungen mehr definiert sind[55]. Die Erzeugung eines einzelnen Planoperators in PDDL- bzw. UCPOP-Syntax aus der beschreibungslogischen Darstellung des Benutzermodells gliedert sich in vier Schritte.

Im ersten Schritt werden

- die Menge der ggf. typisierten Parameter (freien Variablen), die im Planoperator vorkommen,
- die Menge der Vorbedingungs-Literale *und*
- die Menge der Effekt-Literale

[55] Dabei weden *Dekompositions-Concept*, die in Kap. 4.6.2 eingeführt werden, *nicht* beachtet.

des Planoperators erzeugt, die direkt – d.h. durch Umkehrung des Konstruktionsschemas 3.1 aus Abschnitt 3.2.2 S. 115 – aus der beschreibungslogischen Darstellung gewonnen werden können.

Der zweite Schritt ergänzt die impliziten Effekte, wie im Abschnitt 3.4.2.4.4 *Ausschließende Eigenschaften bzw. Typzugehörigkeiten* beschrieben.

Im nächsten Schritt werden die Vorbedingungs-Literale anhand von *Vorrang-Mustern* in eine geeignete Reihenfolge gebracht (s.u.).

Abschließend wird aus der Ergebnis-Datenstruktur eine LISP-Liste erzeugt, die der UCPOP-Syntax für Planoperatoren entspricht und als Teil der Spezifikation des Planungsproblems an UCPOP übergeben werden kann.

Die Menge der Parameter, die für den ersten Erzeugungsschritt benötigt wird, ergibt sich aus den Vorbedingungs- und Effekt-Attributen des Plan-Concepts, da sie die Beziehungen zu den für den Operator benötigten Objekten modellieren. Eine Typeinschränkung der Parameter kann aus der Werterestriktion des zugehörigen Vorbedingungs-Attributs übernommen werden. Da die Semantik der Typisierung von Parametern in PDDL nicht spezifiziert wird und UCPOP keine Typisierung unterstützt[56], wird nur eine typen*lose* Liste von Parameter für den Planoperator erzeugt.

Die Werterestriktionen der Attribute führen stattdessen zur Erzeugung von Vorbedingungs- und Effekt-Literalen. Nach Konstruktionsschema 3.1 *Planoperatoren in Beschreibungslogik* ist eine solche Werterestriktion ein einzelner, evtl. negierter Concept-Name oder eine Konjunktion davon. Dabei sei angemerkt, daß sowohl die Beschreibung der Plan-Concepts, die initial in das Benutzermodell eingetragen werden, als auch diejenigen, die durch eventuelle Änderungen im Verlauf einer Benutzer-Interaktion auftreten können, mithilfe des Konstruktionsschemas erzeugt werden. D.h. jedes Plan-Concept in der Benutzermodellierungswissensbasis ist grundsätzlich nach diesem Schema aufgebaut. In Umkehrung des Schemas führt jeder Faktor CN bzw. $\neg CN$ der Werterestriktion eines Vorbedingungs- oder Effekt-Attributs A zu einem Vorbedingungs- bzw. Effekt-Literal $CN(var(A))$ bzw. $\neg CN(var(A))$. CN steht dabei für einen Concept-Namen in der beschreibungslogischen Repräsentation und wird gem. Definition 3.3 S. 141 als Prädikatsymbol für die prädikatenlogische Darstellung genutzt. Hat ein solches Attribut lediglich die Werterestriktion *domain-concept*, wird dafür kein Literal erzeugt, weil jedes Objekt, das für einen Planoperator verwendet wird, Exemplar von *domain-concept* ist. Das entsprechende Prädikat ist also immer "wahr" und kann deshalb entfallen. $var(A)$ gibt die zum Attribut A gehörende Variable an.

Für den ersten Erzeugungsschritt fehlen noch die zweistelligen Prädikate, die aus den Role-Value-Maps (RVMs) des Plan-Concepts erzeugt werden. Sind A und B Vorbedingungs-Attribute und $A = B$ oder $A \neq B$ RVMs des Plan-Concepts, wird damit ausgedrückt, daß die Rollenfüller identisch bzw. verschieden sein müssen. In der prädikatenlogischen Darstellung müssen die Werte der zugehörigen Variablen demnach gleich bzw. ungleich sein. Dementsprechend werden für jede solche RVM eine Gleichheits- $(eq(var(A), var(B)))$ bzw. Ungleichheitsbedingung $(neq(var(A), var(B)))$ zur Menge der Vorbedingungs-Literale hinzugefügt. Die verbleibenden Möglichkeiten von RVMs haben nach Tabelle 3.3 S. 113 die Form $A \subseteq B \circ R$ oder $\neg(A \subseteq B \circ R)$, wobei A und B entweder beide Vorbedingungs- oder beide Effekt-Attribute sind und R eine Rolle ist. Die RVMs drücken aus, daß die Rollenfüller von B und A in der durch R modellierten Relation stehen bzw. nicht stehen. Diese Bedingung kann durch Formeln $R(var(B), var(A))$ bzw. $\neg R(var(B), var(A))$ repräsentiert werden, die dementsprechend zur Menge der Vorbedingungs- bzw. Effekt-Literale hinzugefügt werden.

Die obigen Beschreibungen der Plan-Concepts gingen von der Notation und den Möglichkeiten von KRSS aus. Kap. 3.2.2.1 *Spezifika der Repräsentation in SB-ONE* behandelte die Besonderheiten der Repräsentation, die jetzt wieder die für die Erzeugung der Vorbedingungs- bzw. Effekt-Literale

[56] UCPOP erlaubt zwar die Angabe eines Typ-Prädikats für einen Parameter. Die Angabe ist aber gleichbedeutend mit der Angabe dieses Prädikats als Vorbedingungskomponente. Eine darüber hinausgehende, besondere Bedeutung ist damit nicht verbunden.

berücksichtigt werden müssen. Dies heißt insbesondere, daß anstelle der Concept-Ausdrücke $A \neq B$ und $\neg(A \subseteq B \circ R)$ die RVMs $A \otimes B$ bzw. $A \otimes B \circ R$ verwendet werden. Außerdem müssen die speziell markierten Concepts, die zur Realisierung eines Concept-Ausdrucks (Konjunktion und/oder Negation) definiert wurden, aufgelöst werden; d.h. es wird hier wieder der ursprüngliche (zu dem Concept äquivalente) Ausdruck verwendet.

Der dritte Schritt der Erzeugung eines Planoperators bringt die Vorbedingungs-Literale in eine geeignete Reihenfolge. Dieser Schritt ist notwendig, weil UCPOP – wie viele andere Planungssysteme (vgl. [Blum und Furst, 1995]) – mit sehr unterschiedlichem Zeit- und/oder Speicherplatzverbrauch auf diese Reihenfolge reagieren kann. Dies beeinträchtigt zwar theoretisch nicht die Vollständigkeit von UCPOP; d.h. falls es eine Lösung des Planungsproblems gibt, so findet UCPOP sie theoretisch auch. Allerdings kann die Berechnung einer Lösung bei ungeschickter Anordnung so viel Zeit und/oder Speicherplatz beanspruchen, daß sie letztlich doch nicht gefunden wird. Für jeden Planoperator *können* deshalb *Vorrang-Muster* angegeben werden, anhand derer die Vorbedingungs-Komponenten angeordnet werden können. Abb. 3.24 zeigt den syntaktischen Aufbau eines Vorrang-Musters. Es besteht aus einer Liste von Vorbedingungs-Literalen und der Angabe der Planoperatoren, für die es angewendet werden soll. Der Entwickler des Sachbereichsmodells kann die Vorbedingungs-Literale eines Planoperators, die nach seiner Kenntnis einen relevanten Einfluß auf den Verlauf des Planungsprozesses haben, in einer geeigneten Reihenfolge angeben. Er muß aber nicht alle angeben und er darf auch Vorbedingungs-Literale angeben, die der Planoperator gar nicht hat. Vergleichbar zu Axiomen und impliziten Effekten, werden auch hier für die Erzeugung des Planungssachbereichs lediglich solche Komponenten herangezogen, die aus der jeweils relevanten View der Benutzermodellierungswissensbasis abgeleitet werden können. Damit können die Vorrang-Muster nicht nur für alle Views verwendet werden, sondern lassen sich auch für mehrere Planoperatoren mit ähnlichen Vorbedingungen einsetzen. Letzteres wird im Beispielsachbereich ausgenutzt, indem bspw. nur *ein* Muster für alle Varianten von *ftp-download*-Planoperatoren definiert wird.

prec-pattern-def = “(” “:prec-pattern” {*literal*}$^+$ “:for” “(” {*planop-name*}$^+$ “)” “)” .
literal = *predicate* | “(” “not” *predicate* “)” . (wie Abb. 3.3 S. 111)

Abbildung 3.24: EBNF-Syntax für die Definition eines Vorrang-Musters.

Die Umordnung der Vorbedingungs-Literale eines Planoperators erfolgt in der Weise, daß zunächst alle Vorbedingungs-Literale, die auch in der Literal-Liste des zum Planoperator gehörenden Vorrang-Musters enthalten sind, in die Reihenfolge gebracht werden, die das Vorrang-Muster angibt. Die restlichen Vorbedingungs-Literale folgen anschließend in der ursprünglichen Reihenfolge. Die Umordnung stellt aus logischer Sicht eine Äquivalenzumformung dar, hat also keinen Einfluß auf die Logik des Planoperators. Für Planungssysteme, die nicht auf die Reihenfolge der Vorbedingungs-Prädikate reagieren (bspw. Graphplan-basierte Systeme [Blum und Furst, 1995]), entfällt die Umordnung sinnvollerweise.

Abb. 3.27 zeigt ein Beispiel für die Erzeugung des Planoperators *app-print* aus dem gleichnamigen Plan-Concept, so wie es im Handlungspotential des Beispielbenutzers Oscar (vgl. Kap. 3.3.3 *Stereotypes*) modelliert ist. Die Abb. 3.25 und 3.26 stellen das Plan-Concept in KRSS- und linearer SB-ONE-Notation dar. Abb. 3.27 zeigt im ersten Teil die durch Schritt 1 erzeugten Parameter und Literal-Mengen. Der nächste Teil zeigt die Slots der LISP-Structure für den Planoperator, nachdem implizite Effekte (hier einer) hinzugefügt und die Vorbedingungs-Literale nach dem zu *app-print* gehörenden Vorrang-Muster angeordnet wurden. Der dritte Teil enthält das Ergebnis in UCPOP-Syntax, wie es für die Spezifikation des Planungsproblems eingesetzt wird.

$$\begin{aligned}
&\textit{app-print} \sqsubseteq \textit{plan-concept} \sqcap \textit{pre_app} : \textit{application} \sqcap \textit{pre_doc} : \textit{domain-concept} \sqcap \\
&\quad \textit{pre_doc-c} : \textit{domain-concept} \sqcap \textit{pre_pap-doc} : \textit{non-existent} \sqcap \textit{pre_pr} : \textit{printer} \sqcap \\
&\quad \textit{eff_doc-c} : \textit{domain-concept} \sqcap \textit{eff_pap-doc} : \textit{paperdoc} \sqcap \textit{eff_pr} : \textit{domain-concept} \sqcap \\
&\quad \textit{pre_doc} \subseteq \textit{pre_app} \circ \textit{active} \sqcap \textit{pre_doc-c} \subseteq \textit{pre_doc} \circ \textit{fso-contents} \sqcap \\
&\quad \textit{eff_pap-doc} \subseteq \textit{eff_pr} \circ \textit{has} \sqcap \textit{eff_doc-c} \subseteq \textit{eff_pap-doc} \circ \textit{pd-contents}
\end{aligned}$$

Abbildung 3.25: KRSS-Darstellung des Plan-Concepts *app-print* in SBUC (Benutzer ist Oscar)

```
app-print (GC58) IS-A ((plan-concept (GC1)))
    pre_app (GR1917) : application (GC37) [(1 1), NEC]
    pre_doc (GR1922) : domain-concept (GC2) [(1 1), NEC]
    pre_doc-c (GR1919) : domain-concept (GC2) [(1 1), NEC]
    pre_pap-doc (GR1916) : non-existent (GC6) [(1 1), NEC]
    pre_pr (GR1918) : printer (GC42) [(1 1), NEC]
    eff_doc-c (GR1923) : domain-concept (GC2) [(1 1), NEC]
    eff_pap-doc (GR1921) : paperdoc (GC36) [(1 1), NEC]
    eff_pr (GR1920) : domain-concept (GC2) [(1 1), NEC]
    (RVM217 SUB ((GR1922 . pre_doc)) ((GR1917 . pre_app) (GR1924 . active)))
    (RVM218 SUB ((GR1919 . pre_doc-c)) ((GR1922 . pre_doc) (GR1939 . fso-contents)))
    (RVM219 SUB ((GR1921 . eff_pap-doc)) ((GR1920 . eff_pr) (GR1995 . has)))
    (RVM220 SUB ((GR1923 . eff_doc-c)) ((GR1921 . eff_pap-doc) (GR42 . pd-contents)))
```

Abbildung 3.26: SB-ONE-Darstellung des Plan-Concepts *app-print* in SBUC (Benutzer ist Oscar)

Parameter:	{app, doc, doc-c, pap-doc, pr}
Precondition-Literale:	{application (app), non-existent (pap-doc), printer (pr), active (app, doc), fso-contents (doc, doc-c)}
Effekt-Literale:	{paperdoc (pap-doc), has (pr, pap-doc), pd-contents (pap-doc, doc-c)}

1: Parameter, Precondition und Effekt-Literale des Plan-Concepts *app-print*

```
:operator     app-print
:parameters   (?app ?doc ?doc-c ?pap-doc ?pr)
:precondition (and ((fso-contents ?doc ?doc-c) (application ?app) (active ?app ?doc) (printer ?pr)
                    (non-existent ?pap-doc) ))
:effect       (and (paperdoc ?pap-doc) (not (to-create ?pap-doc)) (has ?pr ?pap-doc)
                    (pd-contents ?pap-doc ?doc-c))
```

2: Plan-Operator *app-print* mit impliziten Effekten und gemäß Vorrang-Muster

```
(:action      app-print
:parameters   (?app ?doc ?doc-c ?pap-doc ?pr)
:precondition (and ((fso-contents ?doc ?doc-c) (application ?app) (active ?app ?doc) (printer ?pr)
                    (non-existent ?pap-doc) ))
:effect       (and (paperdoc ?pap-doc) (not (to-create ?pap-doc)) (has ?pr ?pap-doc)
                    (pd-contents ?pap-doc ?doc-c)))
```

3: endgültiger Plan-Operator *app-print* in UCPOP-Syntax

Abbildung 3.27: Erzeugung des Planoperators *app-print* aus Plan-Concept *app-print* nach Abb. 3.26

3.5 Das Umfeld des Benutzers: Werkzeuge und andere Akteure

Das bis zu diesem Punkt entwickelte Modell von Handlungsmöglichkeiten eines Benutzers geht davon aus, daß der Benutzer "aus eigener Kraft" handelt und daß das Handlungspotential demnach den Stellenwert einer persönlichen Eigenschaft hat. Allerdings unterliegen die Handlungsmöglichkeiten eines jeden Akteurs äußeren Einflüssen. Beispiele dafür sind Handlungen, die nur mithilfe von Werkzeugen und/oder der Unterstützung weiterer Akteure durchgeführt werden können. Die Handlungen der bisherigen Beispiele nutzten zwar Werkzeuge, der Einfluß, den sie auf die Handlungen nahmen, wurde aber lediglich implizit durch das Handlungspotential des Benutzers berücksichtigt. Dies ist insbesondere dann nachteilig, wenn ein Werkzeug ausfällt oder neu hinzukommt, da Werkzeuge viele einzelne Handlungen beeinflussen können – man denke etwa an ein Textverarbeitungssystem.

In diesem Abschnitt wird das Modell der Handlungsmöglichkeiten deshalb insofern differenziert, als die Einflüsse von Werkzeugen explizit berücksichtigt werden. Die damit verbundene bessere Modularisierung des Wissens über Objekte und Handlungsmöglichkeiten des Anwendungssachbereichs führt zu einer größeren Flexibilität und besseren Wartbarkeit der Benutzermodellierungswissensbasis. Als Nebeneffekt ergibt sich dadurch die Möglichkeit der Klassifikation von Objekten anhand ihrer *Verwendungsmöglichkeiten*. Darüber hinaus kann ein solches Modell so erweitert werden, daß damit einige Aspekte der Zusammenarbeit von mehreren Akteuren beschrieben werden können.

Die Beschreibungen im ersten Teil gehen vom Hauptanwendungsbereich dieser Arbeit aus, bei dem das System einen für den Benutzer spezifischen Plan generiert und dazu ein Modell der Annahmen und der Handlungsmöglichkeiten des Benutzers einsetzt. Die Vorgehensweise kann – in gleicher Weise wie in Kap. 3.3.2 *Repräsentation eines Benutzermodells mit Handlungspotential und Wissen im* AsTRa-*Formalismus* – auf einen beliebigen planenden Akteur und einen Akteur, für den geplant wird, verallgemeinert werden.

3.5.1 Der Einfluß von Werkzeugen auf Handlungsmöglichkeiten

Die bisherige Behandlung des Handlungspotentials hat sich den Fragen gewidmet, wie das Handlungspotential genutzt wird und wie es für einen speziellen Benutzer aufgebaut werden kann. Als Möglichkeiten wodurch es bestimmt wird, wurden die physischen Eigenschaften und die Befugnisse eines Benutzers angeführt. Die Beispiele haben aber gezeigt, daß auch die Werkzeuge oder Hilfsmittel, die dem Benutzer zur Verfügung stehen, seine Handlungsmöglichkeiten beeinflussen. Die Erklärung zum in Abb. 3.13 in Kap. 3.3.2 dargestellten Beispiel des *Öffnens einer Datei in einem Textverarbeitungssystem* besagte, daß zwei Akteure – bspw. wg. unterschiedlicher Lizenzen – unterschiedliche Varianten der Aktion *open-file-tp* ausführen konnten. Das Handlungspotential wurde dabei durch die Verfügbarkeit unterschiedlicher Werkzeuge – hier der Textverarbeitungssysteme – beeinflußt. Gleiches gilt für das in Abb. 3.18 in Kap. 3.3.3 *Stereotypes* dargestellte Beispiel. Hier war die Möglichkeit *ein Dateiverzeichnis zu mounten* auf die Windows-Variante eingeschränkt, weil der zur Verfügung stehende Computer unter diesem Betriebssystem arbeitete.

Ziel dieses Abschnitts ist die Differenzierung des bisherigen Modells von Handlungsmöglichkeiten, wobei der Einfluß von Werkzeugen explizit repräsentiert wird. Das vorrangige Ziel ist dabei die Verbesserung der Organisation des repräsentierten Wissens. Abgesehen von einer Erhöhung der Flexibilität (falls die Menge der zur Verfügung stehenden Werkzeuge veränderlich ist) bleibt die Adaption der Hilfeleistung oder Unterstützung an die Eigenschaften des Benutzers davon unberührt.

Die Organisation des Wissens ist für die praktische Einsetzbarkeit eines jeden KI-Projekts zumindest mitentscheidend, weil der Aufwand für die Modellierung und Wartung des Anwendungssachbereichs oft sehr hoch ist. Generelle Prinzipien um den Aufwand möglichst gering zu halten sind das Vermeiden von Redundanz und die zusammenhängende Repräsentation von zusammenhängendem

Wissen. Damit kann erreicht werden, daß geringfügige Änderungen im Sachbereich nur lokale Auswirkungen auf die Repräsentation haben. Eine direkte Konsequenz für die Repräsentation des Handlungspotentials ist die Trennung der Einflüsse, die von den persönlichen Eigenschaften des Benutzers ausgehen, von denjenigen, die von Werkzeugen herrühren. Erst wenn aus dem Benutzermodell die Spezifikation einer Planungsaufgabe abgeleitet werden soll (vgl. Kap. 3.4.2.7 *Erzeugung des aktiven Teils eines Planungssachbereichs*), müssen diese Einflüsse zusammengeführt werden. Daraus können dann die Handlungsmöglichkeiten abgeleitet werden, die der Benutzer *unter Berücksichtigung der zur Verfügung stehenden Werkzeuge* hat. Eine so skizzierte Modularisierung des Wissens erleichtert die Modifikation des Sachbereichsmodells, indem die Handlungsmöglichkeiten der potentiellen Benutzer durch das Einbringen neuer Werkzeugtypen erweitert werden können, ohne daß damit die Notwendigkeit verbunden wäre, sämtliche Plan-Concepts im Handlungspotential eines Benutzers oder der Stereotypes zu modifizieren. Denkbar sind auch Sammlungen von Modellen von Werkzeugtypen, die für verschiedene Anwendungsgebiete eingesetzt werden können und die bei Bedarf mit sehr geringem Aufwand in ein Sachbereichsmodell eingebracht werden können.

Der spezifische Einfluß eines Benutzers auf allgemeine Handlungsmöglichkeiten ist bereits durch sein Handlungspotential modelliert. Dort werden die Handlungen genau mit den Einschränkungen und Besonderheiten beschrieben, die mit der Ausführung durch den jeweiligen Benutzer verbunden sind. Diese Vorgehensweise kann auch für Werkzeuge eingesetzt werden, indem die Handlungen modelliert werden *für die das Werkzeug verwendet werden kann*. Allerdings sollen Werkzeuge nicht individuell behandelt, sondern zu Klassen zusammengefaßt werden. Im weiteren wird deshalb von Werkzeug*typen* die Rede sein, die durch Werkzeug-Concepts repräsentiert werden (s.u.). Analog zu Handlungsmöglichkeiten und Handlungspotential eines Benutzers seien *Verwendungsmöglichkeiten* und *Verwendungspotential eines Werkzeugtyps* definiert:

Definition 3.5 ***Verwendungsmöglichkeiten eines Werkzeugtyps***
Die *Verwendungsmöglichkeiten eines Werkzeugtyps* umfaßt die Menge der Handlungen, die mit einem Werkzeug dieses Typs *ausgeführt werden können*.

Definition 3.6 ***Verwendungspotential eines Werkzeugtyps***
Das *Verwendungspotential eines Werkzeugtyps* ist ein Modell seiner Verwendungsmöglichkeiten, die durch Werkzeuge des Typs *spezifisch beeinflußt werden können*.

Analog zum Handlungspotential wird das *Verwendungspotential eines Werkzeugtyps* auch als Menge von Plan-Concepts repräsentiert. Die Forderung, daß nur solche Handlungen als Verwendungspotential modelliert werden müssen, die durch Werkzeuge des Typs *spezifisch beeinflußt werden können*, grenzt Handlungen aus, bei denen ein Werkzeug lediglich als passiver Gegenstand verwendet wird. Bspw. kann eine Computertastatur als Briefbeschwerer verwendet werden. Es ist aber nicht sinnvoll eine solche passive Verwendung als Verwendungsmöglichkeit einer Computertastatur zu modellieren; schließlich kommt es hier nur auf das Gewicht des Gegenstandes an. Für eine Aktion *Brief beschweren* reicht also das *Vorhandensein* eines schweren Gegenstandes – der Gegenstand übt aber keinen spezifischer Einfluß auf die Handlung aus.

Der spezifische Einfluß eines Werkzeugs auf eine Handlung kann sich durch spezifische Bedingungen an die Ausführung und/oder spezifische Effekte auswirken. Dies kann – analog zur Modellierung von Handlungsmöglichkeiten eines Benutzers – durch Plan-Concepts mit spezifischen Vorbedingungen und Effekten modelliert werden. Die Möglichkeiten eines Benutzers, eine *Handlung mithilfe eines Werkzeugs auszuführen*, ergibt sich durch Zusammenführen der Einflüsse des Benutzers und des Werkzeugs. Dies entspricht in der beschreibungslogischen Repräsentation der Konjunktion der zugehörigen Plan-Concepts aus dem Handlungspotential des Benutzers und den werkzeugspezifischen Einflüssen auf die Plan-Concepts (s.u.). Die so gebildeten Plan-Concepts seien *werkzeugspezifische*

Plan-Concepts genannt. Die Menge aller werkzeugspezifischen Plan-Concepts, die sich unter Berücksichtigung aller dem Benutzer zur Verfügung stehenden Werkzeugtypen bilden lassen, sei das *werkzeugspezifische Handlungspotential des Benutzers* genannt. Die in Kap. 3.4.2.7 *Erzeugung des aktiven Teils eines Planungssachbereichs* beschriebene Spezifikation des Planungssachbereichs, die Basis für die benutzerspezifische Planerzeugung ist, wird – wenn der Werkzeugeinfluß, wie hier vorgeschlagen, explizit modelliert ist – aus dieser Menge werkzeugspezifischer Plan-Concepts erzeugt.

3.5.1.1 Die Repräsentation eines Verwendungspotentials

Das Handlungspotential eines Benutzers wird in einer Fähigkeitspartition des Benutzermodells repräsentiert. Für die Verwendungspotentiale von Werkzeugtypen erscheinen eigene Partitionen allerdings nicht sinnvoll. Die Vorteile, die sich durch die Gleichbehandlung des Handlungspotentials und des Modells der Benutzerkenntnisse ergeben, sind für Werkzeuge ohne Belang. Auch haben Werkzeuge im Modell weder die herausragende Position wie der Benutzer selbst noch spielen sie als Individuum eine Rolle. Vielmehr sind die Ansichten der verschiedenen Akteure über *Werkzeugklassen* normalerweise bereits durch (Domain-) Concepts in den jeweiligen Wissenspartitionen modelliert. Das Verwendungspotential stellt eine Möglichkeit dar, diese Modelle zu präzisieren. Dazu werden die Werkzeugklassen zu ihren Verwendungsmöglichkeiten in Relation gesetzt. In der beschreibungslogischen Repräsentation des Benutzermodells heißt dies, daß für jedes Concept das einen Werkzeugtyp repräsentiert eine Rolle *usable* definiert wird. Solche Concepts seien *Werkzeug-Concepts* genannt.

Das Verwendungspotential eines Werkzeugtyp wird dann modelliert, indem die Rolle *usable* auf die Plan-Concepts eingeschränkt wird, für die das Werkzeug-Concept eingesetzt werden kann. Zusätzlich können für jedes dieser Plan-Concepts weitere Einschränkungen angegeben werden, die dann den spezifischen Einfluß des Werkzeugtyps auf das Plan-Concept modellieren. Wird ein Typ nur für ein Plan-Concept als Werkzeug eingesetzt, wird *usable* auf einen Ausdruck der folgenden Form eingeschränkt:

$$(\text{<}\textit{Plan-Concept-Name}\text{>} \sqcap \\ \text{<}\textit{Beschreibung des werkzeugspezifischen Einflusses auf}\ \text{<}\textit{Plan-Concept-Name}\text{>}\ \text{>})$$

Soll der Werkzeugtyp für mehrere Plan-Concepts verwendet werden, müssen die Werterestriktionen disjunktiv verknüpft werden[57].

Im o.g. Beispiel des *Öffnens einer Datei in einem Textverarbeitungssystem* (*open-file-tp*) aus Kap. 3.3.2 ist ein Textverarbeitungssystem als Werkzeug erforderlich. Der Typ dieses Werkzeugs stellt die zusätzliche Vorbedingung, daß der Typ der zu öffnenden Datei zum Textverarbeitungssystem passen muß. Das Verwendungspotential der im genannten Beispiel modellierten Textverarbeitungssystemtypen *tpx* und *tpy* enthält demnach *open-file-tp* mit dieser zusätzlichen Vorbedingung. Die beschreibungslogische Repräsentation lautet dann:

$$\begin{aligned} tpx &\doteq tp \sqcap \forall usable : (open\text{-}file\text{-}tp \sqcap pre_f : \boldsymbol{tpx\text{-}file}) \\ tpy &\doteq tp \sqcap \forall usable : (open\text{-}file\text{-}tp \sqcap pre_f : \boldsymbol{tpy\text{-}file}) \end{aligned}$$

Die beiden Concept-Definitionen gehen von einem Concept *tp* als Super-Concept für alle Textverarbeitungssysteme aus und berücksichtigen für das Verwendungspotential nur das eine Plan-Concept *open-file-tp*. Die spezifischen Einflüsse der Werkzeuge auf das Plan-Concept werden durch die Teilausdrücke *pre_f : tpx-file* bzw. *pre_f : tpy-file* beschrieben. Eine explizite Repräsentation von spezielle Plan-Concepts *open-file-tpx* oder *open-file-tpy* ist nun nicht mehr erforderlich.

[57] Auf die konkrete Repräsentation in SB-ONE wird weiter unten noch eingegangen.

Ein Beispiel für die Verwendung von Objekten, die als Werkzeuge angesehen werden können, aber keinen spezifischen Einfluß auf die Aktion ausüben, ist das Plan-Concept *app-print*. *app-print* modelliert das Drucken aus einem Anwendungsprogramm (s.u. und Anhang B.2 *Plan-Concepts*). Die in den Beispielen modellierten Anwendungsprogramme – wozu auch Textverarbeitungssysteme zählen – haben keinen spezifischen Einfluß auf die durch *app-print* modellierte Aktion; d.h. keines dieser Anwendungsprogramme stellt gegenüber der allgemeinen Definition des Plan-Concepts zusätzliche Bedingungen oder bewirkt zusätzliche Effekte. *app-print* braucht deshalb nicht im Verwendungspotential der Anwendungsprogramme oder der Textverarbeitungssysteme modelliert zu werden.

Eine andere Sicht ergibt sich, wenn eine möglichst aussagekräftige Beschreibung von Werkzeugen im Vordergrund steht. Für die Beschreibung von Werkzeugen sind die Verwendungsmöglichkeiten von primärem Interesse – im Gegensatz bspw. zu sonstigen Merkmalen, wie äußeres Erscheinungsbild, physikalische Eigenschaften oder Bestandteile, die oft als Klassifizierungsmerkmale von Gegenständen verwendet werden. Eine beschreibungslogische Repräsentation von Handlungen, wie sie in dieser Arbeit eingesetzt wird, ermöglicht die Modellierung der Verwendungsmöglichkeiten in gleicher Weise wie andere Merkmale von Objekten. Der Wert einer solchen Beschreibung ist aber unabhängig davon, ob für den Werkzeugtyp ein spezifischer Einfluß auf die Handlungen, für die er eingesetzt werden kann, modelliert ist. Aus diesem Grund *können* auch Handlungen als Verwendungsmöglichkeit eines Werkzeugtyps modelliert werden, ohne daß gleichzeitig ein spezifischer Einfluß des Werkzeugs auf die Handlungen modelliert sein muß.

Für die Spezifikation des Planungssachbereichs aus dem Benutzermodell muß das werkzeugspezifische Handlungspotential zunächst erzeugt werden. Dazu muß für jedes Plan-Concept erkennbar sein, ob und ggf. welche der beteiligten Objekte Werkzeuge sind und dementsprechend einen Einfluß auf das Plan-Concept haben können. Dies wird dadurch erreicht, daß ein Plan-Concept zu den benötigten Werkzeugtypen in Beziehung gesetzt wird. In der beschreibungslogischen Repräsentation sei dafür eine Rolle *tool* für Plan-Concepts definiert. Ein für ein Plan-Concept benötigtes Objekt wird als Werkzeug ausgezeichnet, wenn es nicht nur Füller eines Vorbedingungs-Attributs, sondern zusätzlich Füller der Rolle *tool* sein muß. Das heißt, daß für das Plan-Concept eine Role-Value-Map "$\subseteq$" zwischen dem Vorbedingungs-Attribut und der Rolle *tool* definiert wird. Solche Attribute werden im weiteren *Werkzeug-Attribute* genannt. Um im Beispiel auszudrücken, daß das Textverarbeitungsprogramm ein Werkzeug für das Plan-Concept *open-file-tp* ist, wird die Beschreibung des Plan-Concepts um den Faktor $pre_app \subseteq tool$ ergänzt, womit *pre_app* ein *Werkzeug-Attribut* wird.

Sind mehrere Werkzeuge für ein Plan-Concept erforderlich, können sie sich auch gegenseitig beeinflussen, indem die Verwendung eines Werkzeugtyps die Auswahl der anderen einschränkt. Ein Beispiel ist das *Drucken aus einem Anwendungsprogramm unter dem Betriebssystem DOS* (*dos-app-print*). Im Gegensatz zum Vorgehen in neueren Betriebssystemen, sind unter DOS die Anwendungsprogramme selbst für die Ansteuerung eines Druckers zuständig. Die Anwendungsprogramme unterstützen meist nur eine begrenzte Menge von Druckertypen und die oft auch nur mit Einschränkungen. Sowohl das Anwendungsprogramm als auch der Druckertyp beeinflussen damit das Druckergebnis. Um auszudrücken, daß der Einsatz eines bestimmten Werkzeugtyps t_1 für ein Plan-Concept die Verwendung eines anderen Werkzeugtyps t_2 einschränken kann, muß die Einschränkung für t_2 für das Plan-Concept im Verwendungspotential von t_1 definiert werden. Als Beispiel möge *dos-app-print* als Spezialisierung von *app-print* mit dem Anwendungsprogramm (*application*) und dem Drucker (*printer*) als Werkzeuge definiert sein:

$$dos\text{-}app\text{-}print \quad \sqsubseteq \quad app\text{-}print \sqcap pre_app \subseteq tool \sqcap pre_pr \subseteq tool$$

Zum Vergleich die Definition von *app-print* aus Abb. 3.25:

$$\begin{aligned}
\textit{app-print} \sqsubseteq\ & \textit{plan-concept} \sqcap \textit{pre_app} : \textit{application} \sqcap \textit{pre_doc} : \textit{domain-concept} \sqcap \\
& \textit{pre_doc-c} : \textit{domain-concept} \sqcap \textit{pre_pap-doc} : \textit{non-existent} \sqcap \textit{pre_pr} : \textit{printer} \sqcap \\
& \textit{eff_doc-c} : \textit{domain-concept} \sqcap \textit{eff_pap-doc} : \textit{paperdoc} \sqcap \textit{eff_pr} : \textit{domain-concept} \sqcap \\
& \textit{pre_doc} \subseteq \textit{pre_app} \circ \textit{active} \sqcap \textit{pre_doc-c} \subseteq \textit{pre_doc} \circ \textit{fso-contents} \sqcap \\
& \textit{eff_pap-doc} \subseteq \textit{eff_pr} \circ \textit{has} \sqcap \textit{eff_doc-c} \subseteq \textit{eff_pap-doc} \circ \textit{pd-contents}
\end{aligned}$$

app-print erzeugt ein neues Papierdokument (*paperdoc*), das durch das Effekt-Attribut *eff_pap-doc* angegeben wird (vgl. Anhang B.2 *Plan-Concepts* für weitere Erläuterungen). Für das Beispiel hier mögen Papierdokumente in Farb- (*color-doc*) und Schwarz-Weiß-Dokumente (*black-white-doc*) eingeteilt sein.

Als Typen von Anwendungsprogrammen seien definiert:

$$\begin{aligned}
\textit{app1} &\sqsubseteq \textit{application} \sqcap \forall \textit{usable} : (\textit{dos-app-print} \sqcap \textit{pre_pr} : \textit{laser-printer}) \\
\textit{app2} &\sqsubseteq \textit{application} \sqcap \forall \textit{usable} : (\textit{dos-app-print} \sqcap \textit{eff_pap-doc} : \textit{black-white-doc})
\end{aligned}$$

Beide Anwendungstypen können als Werkzeuge für das Plan-Concept *dos-app-print* eingesetzt werden. Der erste schränkt ein (Werkzeug-)Attribut des Plan-Concepts auf *laser-printer* ein; d.h. Anwendungsprogramme dieses Typs können nur mit Laserdruckern drucken. Ansonsten gehen von dem Anwendungstyp keine spezifischen Bedingungen oder Effekte aus. Der zweite Anwendungstyp kann zwar mit allen Druckern umgehen, kann Drucker aber nur so ansteuern, daß sich ein Schwarz-Weiß-Druck ergibt.

Außerdem geht das Beispiel davon aus, daß die beiden folgenden Druckertypen zur Verfügung stehen:

$$\begin{aligned}
\textit{bw-laser-printer} &\sqsubseteq \textit{laser-printer} \sqcap \forall \textit{usable} : (\textit{app-print} \sqcap \textit{eff_pap-doc} : \textit{black-white-doc}) \\
\textit{ink-jet-printer} &\sqsubseteq \textit{printer} \\
&\text{wobei} \\
\textit{printer} &\sqsubseteq \textit{domain-concept} \sqcap \forall \textit{usable} : \textit{app-print} \\
\textit{laser-printer} &\sqsubseteq \textit{printer} \\
\textit{laser-printer} &\parallel \textit{ink-jet-printer}.
\end{aligned}$$

Während Laserdrucker und Tintenstrahl-Drucker (*ink-jet-printer*) keine spezifischen Einschränkungen haben, kann ein Schwarz-Weiß-Laserdrucker (*bw-laser-printer*) nur schwarz-weiß drucken.

Durch die Kombination der Einflüsse der verschiedenen Werkzeuge, auf die im nächsten Abschnitt noch genauer eingegangen wird, können aus dem Plan-Concept *dos-app-print* insgesamt drei werkzeugspezifische Varianten abgeleitet werden. In diesem Beispiel ist dennoch nur ein Schwarz-Weiß-Druck möglich. Der zweite Anwendungstyp kann ohnehin nur schwarz-weiße Ausgaben erzeugen. Der erste könnte zwar auch in Farbe drucken, der zur Verfügung stehende Laser-Drucker-Typ ist dazu aber nicht in der Lage. In einem Fall bringt also der Typ des Anwendungsprogramms den Faktor *eff_pap-doc : black-white-doc* in das resultierende werkzeugspezifische Plan-Concept ein, im anderen der Druckertyp. Ständen auch Farb-Laser-Drucker zur Verfügung, gäbe es weitere zulässige Kombinationen der Werkzeuge, wodurch sich dann auch ein werkzeugspezifisches Plan-Concept ergeben würde, mit dem ein Farb-Ausdruck möglich wäre (s. Bsp. im nächsten Abschnitt).

Zusammenfassend kann also festgehalten werden, daß a) für Plan-Concepts definiert werden kann, welche der beteiligten Objekte als Werkzeuge berücksichtigt werden müssen, indem die zugehörigen Vorbedingungs-Attribute als Werkzeug-Attribute ausgezeichnet werden und, daß b) das

Verwendungspotential eines Werkzeugs durch Werterestriktion der Rolle *usable* eines Werkzeug-Concepts repräsentiert wird. Die Restriktion erfolgt auf Plan-Concepts, für die der Werkzeugtyp eingesetzt werden kann. Dabei kann der Werkzeugtyp das Plan-Concept durch spezifische Bedingungen direkt beeinflussen oder die Typen weiterer, für das Plan-Concept erforderlicher Werkzeugtypen einschränken. Durch diese Repräsentation unterscheidet sich das Verwendungspotential nicht von anderen Attributen eines Werkzeugs im Modell des begrifflichen Wissens eines Akteurs und gehört damit zu den *einfachen Annahmen* dieses Akteurs.

3.5.1.2 Ermittlung des werkzeugspezifischen Handlungspotentials

Für die Erzeugung des werkzeugspezifischen Handlungspotentials eines Benutzers muß das werkzeug*un*spezifische Handlungspotential des Benutzers mit den Annahmen des Systems (bzw. allgemein: des planenden Akteurs) über den Einfluß, den die (für den jeweiligen Benutzer) verwendbaren Werkzeuge auf die Plan-Concepts dieses Handlungspotentials haben, kombiniert werden. Nach dem Modellierungsansatz aus Kap. 3.3.2 *Repräsentation eines Benutzermodells mit Handlungspotential und Wissen im* AsTRa-*Formalismus* modellieren die Plan-Concepts im Handlungspotential des Benutzers die Auffassung des Systems über allgemeine Handlungsmöglichkeiten im Anwendungssachbereich, die aber durch die Eigenschaften des jeweiligen Benutzers eingeschränkt sein können. Der Partitionsmechanismus von BGP-MS gewährleistet die Korrespondenz der Plan-Concepts in den einfachen Annahmen des Systems (SB) und dem Handlungspotential des Benutzers (SBUC) durch gleiche *Bezeichner*. Bisher wurde die Ansicht vertreten, daß die *Domain-Concepts* in beiden Views gleich beschrieben sein können. Die Beschreibungen wurden deshalb von der gemeinsamen Super-Partition $S^{Dom}B$ übernommen. Diese Sichtweise kann für das Verwendungspotential von Werkzeug-Concepts nicht aufrecht gehalten werden, obwohl es zur Beschreibung von Werkzeug-Concepts gehört, die eine Teilmenge der Domain-Concepts sind. Der Grund liegt darin, daß das Verwendungspotential von Werkzeug-Concepts *benutzerunabhängig* ist und auch Plan-Concepts enthalten kann, die der Benutzer *nicht* ausführen kann oder darf. BGP-MS sieht ausdrücklich vor, daß einzelne Partitionen nur *Teile* der Definitionen von Concepts enthalten. Dadurch kann die Beschreibung von Domain-Concepts, abgesehen von Verwendungspotential und werkzeugspezifischen Einflüssen, unverändert (in Partition $S^{Dom}B$) bleiben. Lediglich die zusätzlichen Definitionen für das Verwendungspotential und die werkzeugspezifischen Einflüsse erfolgen ausschließlich in SB. Die Korrespondenz der Concepts wird wiederum durch BGP-MS gewährleistet. Wichtig ist, daß nicht nur SB, sondern auch SBUC (korrekte) Systemannahmen modellieren. Damit können die Inhalte aus beiden Partitionen verknüpft werden, um daraus das werkzeugspezifische Handlungspotential eines Benutzers zu gewinnen. Im weiteren wird ohne weitere Erwähnung davon ausgegangen, daß die Plan-Concepts, aus denen das werkzeugspezifische Handlungspotential eines Benutzers gewonnen wird, im Handlungspotential des Benutzers (SBUC) repräsentiert sind. Da es – wenngleich in den Beispielen nicht dargestellt – durchaus denkbar ist, daß verschiedene Benutzer zur Ausführung von Aktionen unterschiedliche Werkzeuge benötigen, enthält das Handlungspotential eines Benutzers auch diese Information – ggf. durch geeignete Restriktion der Werkzeugattribute von Plan-Concepts. Die Information über die Verwendungsmöglichkeiten von Werkzeugtypen ist aber benutzerunabhängig und deshalb durch die Concepts in SB (bzw. in der Wissenspartition des planenden Akteurs) repräsentiert.

Es bleibt noch die Festlegung einer geeigneten Repräsentation des Wissens darüber, welche Werkzeugtypen für den aktuellen Benutzer bzw. allgemein für einen Akteur zur Verfügung stehen und demnach für das werkzeugspezifische Handlungspotential berücksichtigt werden müssen. Zwar ist im Handlungspotential eines Akteurs angegeben, welche Plan-Concepts der Akteur mit welchen Werkzeugtypen ausführen kann. Aber damit ist *nicht* ausgesagt, daß der Akteur auch *jede denkbare Variante* einer Handlung ausführen kann, die durch ein Plan-Concept in seinem Handlungspotential

beschrieben ist. Insbesondere wird nicht ausgesagt, daß der Akteur auch mit *allen Spezialisierungen* eines evtl. angegebenen Werkzeugtyps umgehen kann oder darf. Als Beispiel wurden eingangs zwei Varianten von Textverarbeitungssystemen genannt, für die jeweils eigene Lizenzen erforderlich waren.

Um festzulegen, welche Typen von Werkzeugen ein Akteur verwenden darf, bietet sich eine explizite Aufzählung an. Dies eröffnet auch die Möglichkeit, im Handlungspotential eines Benutzers oder von Benutzergruppen Handlungsmöglichkeiten zu modellieren, die nur dann genutzt werden können, wenn bestimmte Werkzeugtypen zur Verfügung stehen. Das Beispiel aus Kap. 3.3.3 *Stereotypes* enthielt eine Mitarbeiterin, die normalerweise mit UNIX arbeitet, der aber lediglich ein Windows-Computer zur Verfügung stand und deren Handlungspotential deshalb nur Windows-Funktionen enthielt. Bei einer expliziten Modellierung der Verfügbarkeit von Werkzeugen könnte das Handlungspotential auch die UNIX-Funktionalität enthalten, die aber nur dann für die Planerzeugung verwendet wird, wenn dem Benutzer ein UNIX-Computer als Werkzeug zur Verfügung steht. Trotz eines erweiterten, flexibleren Modells der Handlungsmöglichkeiten bleibt dem Planungsprozeß damit eine aufwendige Berücksichtigung von Planoperatoren, die letztlich nicht zum Ziel führen, erspart (vgl. a. Kap. 3.1.1 *Alternativen zum Verarbeitungsschema*).

Die Annahmen darüber, welche Werkzeugtypen ein Akteur verwenden kann, sind analog zu den Annahmen über die Handlungsmöglichkeiten eines Akteurs zu sehen. Entsprechend wird in BGP-MS zu jeder Fähigkeitspartition eine zugehörige Partition für die *verwendbaren Werkzeugtypen* definiert. Solche *Werkzeug-Partitionen* sind lediglich Container, die die entsprechenden Werkzeug-Concepts ohne jegliche weitere Beschreibung enthalten. Schlußfolgerungen innerhalb einer solchen Partition oder View sind damit weder möglich noch erwünscht. Die Beschreibung der Werkzeuge findet sich in der Wissens-Partition des planenden Akteurs – hier also des Systems. Die Korrespondenz der Concepts wird wiederum durch den BGP-MS Partitionsmechanismus gewährleistet, sodaß Concepts aus verschiedenen Views in Beziehung gesetzt werden können. Die einzige Inferenz, die sich auf Werkzeug-Partitionen bezieht, ist die Vererbung der Inhalte von Super- an Sub-Partitionen. Sie wird wie in Kap. 3.3.3 *Stereotypes* für den Aufbau und die Zuordnung von Stereotypes genutzt. Im AsTRa-Formalismus[58] entsprechen die Werkzeug-Partitionen Assumption Types (ATs), für die eine neue Modalität "T" (*Tool*) definiert werden muß. Zu jedem AT und jedem Wissensbasis-AT-Label-Paar eines Stereotypes mit Modalität "C" wird dabei ein AT bzw. ein Wissensbasis-AT-Label-Paar für den gleichen Stereotype mit Modalität "T" definiert.

Ausgehend von den Plan-Concepts des Handlungspotentials eines Benutzers ergibt sich sein werkzeugspezifisches Handlungspotential dann aus folgendem Konstruktionsschema:

1. Jedes Plan-Concept, für das der Benutzer keine Werkzeuge benötigt – für das also *keine* Werkzeug-Attribute definiert sind –, gehört unverändert auch zu seinem werkzeugspezifischen Handlungspotential.
2. Für jedes Plan-Concept *pc mit* Werkzeug-Attributen gehören alle erfüllbaren *werkzeugspezifischen Plan-Concepts von pc* zum werkzeugspezifischen Handlungspotential des Benutzers.

$\mathrm{TSPC}(pc)$ – die *Menge der werkzeugspezifischen Plan-Concepts von pc* – ergibt sich, wenn die Werkzeug-Attribute von *pc* auf die für diesen Benutzer verwendbaren Werkzeugtypen eingeschränkt werden und gleichzeitig deren werkzeugspezifischer Einfluß berücksichtigt wird. $\mathrm{TSPC}(pc)$ kann als Menge von beschreibungslogischen Ausdrücken beschrieben werden:

$$\mathrm{TSPC}(pc) := \{\ pc \quad \sqcap \quad \Pi_{ta \in \mathrm{TA}(pc)} \big((ta : tc) \sqcap tci(tc, pc) \big) \mid tc \in \mathrm{TAC}(ta, pc) \}$$

[58] [Pohl, 1997]; vgl. auch Kap. 3.3.2 *Repräsentation eines Benutzermodells mit Handlungspotential und Wissen im* AsTRa-*Formalismus*

Dabei ist $\mathrm{TA}(pc)$ die Menge der Werkzeug-Attribute von *pc*, $tci(tc, pc)$ der werkzeugspezifische Einfluß von *tc* auf *pc*. $\mathrm{TAC}(ta, pc)$ ist die Menge der für das Attribut *ta* von *pc* einsetzbaren Werkzeugtypen; d.h. die Concepts aus dem Verwendungspotential des Benutzers, die von der Werterestriktion von *ta* an *pc* ($VR(ta, pc)$) subsumiert werden:

$$\mathrm{TAC}(ta, pc) := \{tc \in \mathrm{SBUT} \mid tc \Rightarrow VR(ta, pc)\}$$

Im Beispiel des vorangegangenen Abschnitts standen die Werkzeuge *app1, app2, bw-laser-printer* und *ink-jet-printer* zur Verfügung. Das Plan-Concept *dos-app-print* hatte die Werkzeug-Attribute *pre_app* und *pre_pr* mit den Werterestriktionen *application* und *printer*, die jeweils von *app-print* ererbt waren. Damit ergeben sich zwei Mengen von einsetzbaren Werkzeugtypen:

$$\begin{aligned}
\mathrm{TAC}(\textit{pre_app}, \textit{dos-app-print}) &= \{\textit{app1}, \textit{app2}\} \\
\mathrm{TAC}(\textit{pre_pr}, \textit{dos-app-print}) &= \{\textit{bw-laser-printer}, \textit{ink-jet-printer}\}
\end{aligned}$$

Daraus läßt sich die folgende Menge *werkzeugspezifischer Plan-Concepts von pc* ableiten:

$$\begin{aligned}
\{\textit{dos-app-print} \;&\sqcap\; \big((\textit{pre_app} : \textit{app1}) \sqcap (\textit{pre_pr} : \textit{laser-printer})\big) \\
&\sqcap\; \big((\textit{pre_pr} : \textit{bw-laser-printer}) \sqcap (\textit{eff_pap-doc} : \textit{black-white-doc})\big), \\
\textit{dos-app-print} \;&\sqcap\; \big((\textit{pre_app} : \textit{app1}) \sqcap (\textit{pre_pr} : \textit{laser-printer})\big) \\
&\sqcap\; \big((\textit{pre_pr} : \textit{ink-jet-printer}) \sqcap (\top)\big), \\
\textit{dos-app-print} \;&\sqcap\; \big((\textit{pre_app} : \textit{app2}) \sqcap (\textit{eff_pap-doc} : \textit{black-white-doc})\big) \\
&\sqcap\; \big((\textit{pre_pr} : \textit{bw-laser-printer}) \sqcap (\textit{eff_pap-doc} : \textit{black-white-doc})\big), \\
\textit{dos-app-print} \;&\sqcap\; \big((\textit{pre_app} : \textit{app2}) \sqcap (\textit{eff_pap-doc} : \textit{black-white-doc})\big) \\
&\sqcap\; \big((\textit{pre_pr} : \textit{ink-jet-printer}) \sqcap (\top)\big)\}
\end{aligned}$$

Das zweite Element dieser Menge ist allerdings unerfüllbar, weil *laser-printer* und *ink-jet-printer* disjunkt sind. Die Bedingungen der beiden Werkzeuge (*app1* und *ink-jet-printer*) schließen also eine gemeinsame Verwendung für *dos-app-print* aus. Die drei anderen Concept-Ausdrücke sind erfüllbar und gehören deshalb zum werkzeugspezifischen Handlungspotential dieses Beispielbenutzers. Dennoch ist dem Benutzer kein Farb-Druck mit *dos-app-print* möglich, weil jede Kombination von Werkzeugen nur einen Schwarz-Weiß-Druck ermöglicht. Stünde dem Benutzer auch ein Farb-Laserdrucker (*c-laser-printer* $\sqsubseteq$ *laser-printer*) zur Verfügung, für den kein spezieller werkzeugspezifischer Einfuß auf *dos-app-print* definiert ist, so kämen zu seinem werkzeugspezifischen Handlungspotential die beiden folgenden Beschreibungen hinzu:

$$\begin{aligned}
\textit{dos-app-print} \;&\sqcap\; \big((\textit{pre_app} : \textit{app1}) \sqcap (\textit{pre_pr} : \textit{laser-printer})\big) \\
&\sqcap\; \big((\textit{pre_pr} : \textit{c-laser-printer}) \sqcap (\top)\big) \\
\textit{dos-app-print} \;&\sqcap\; \big((\textit{pre_app} : \textit{app2}) \sqcap (\textit{eff_pap-doc} : \textit{black-white-doc})\big) \\
&\sqcap\; \big((\textit{pre_pr} : \textit{c-laser-printer}) \sqcap (\top)\big)
\end{aligned}$$

Die Kombination von *app1* und *c-laser-printer* bewirkt keine weiteren Restriktionen gegenüber *dos-app-print* und läßt so auch einen Farb-Ausdruck zu.

Durch die Berücksichtigung der verschiedenen Werkzeugtypen können aus einem Plan-Concept des Handlungspotentials eines Benutzers mehrere werkzeugspezifische Plan-Concepts erzeugt werden – im Beispiel aus *dos-app-print* drei bzw. fünf. Für die Spezifikation des Planungssachbereichs aus einem Benutzermodell müssen in einem solchen Fall neue Namen für die Planoperatoren bestimmt werden, um sie für den Planungsprozeß unterscheidbar zu machen.

Die werkzeugspezifischen Plan-Concepts werden normalerweise nicht dauerhaft im Benutzermodell gespeichert, sondern stehen nur temporär für die Generierung und für die unmittelbar darauf folgende Präsentation eines Plans zur Verfügung. In Anwendungsbereichen, in denen sich Ansichten über die Verfügbarkeit von Werkzeugen nur selten ändern, kann es allerdings sinnvoll sein das werkzeugspezifische Handlungspotential eines Benutzers zumindest solange zu speichern, bis neue Beobachtungen eine Neuberechnung oder Korrektur erforderlich machen. Damit würde sich der Aufwand für die Berechnung des werkzeugspezifische Handlungspotentials verringern, ohne daß der Vorteil eines besser strukturierten und wartbaren Benutzermodells verloren ginge. Nachteilig ist der höhere Aufwand für Verwaltung und Buchführung über die Abhängigkeiten der werkzeugspezifischen Plan-Concepts.

3.5.1.3 Realisierungsaspekte in SB-ONE

In Kap. 3.2.2.1 *Spezifika der Repräsentation in SB-ONE* wurde darauf hingewiesen, daß das in dieser Arbeit eingesetzte Wissensrepräsentationssystem SB-ONE [Kobsa, 1991], [Kobsa, 1990b] gegenüber der allgemeinen Notation aus [Patel-Schneider und Swartout, 1993] einige Einschränkungen aufweist, die durch alternative Beschreibungen ersetzt, oder angenähert werden können. Für die Repräsentation des Verwendungspotentials und der werkzeugspezifischen Einflüsse wurde in diesem Abschnitt vorgeschlagen, den Wertebereich einer Rolle (*usable*) durch eine Disjunktion von Concept-Ausdrücken zu festzulegen. Die in SB-ONE unzulässige Disjunktion ist dann erforderlich, wenn das Verwendungspotential eines Werkzeug-Concepts mehrere Plan-Concepts enthält. Als Modellierungsalternative kommt eine Differenzierung der Rolle *usable* in Frage. Jedes Disjunktionsglied des ursprünglichen Concept-Ausdrucks wird dann Werterestriktion einer der Sub-Rollen. Unschön ist dabei, daß die Zuordnung der einzelnen Teil-Ausdrücke zu bestimmten Sub-Rollen eine willkürliche Festlegung ist. Dadurch ist die SB-ONE Darstellung des KRSS-Ausdrucks nicht eindeutig. Allerdings können die Rollendifferenzierungen auch bewußt bei der Modellierung eingesetzt werden, da mit den verschiedenen Verwendungsmöglichkeiten eines Werkzeugs oft unterschiedliche Bedeutungen verbunden sind.

Eine weitere Einschränkung von SB-ONE liegt darin, daß für Werterestriktionen von Rollen grundsätzlich nur ein Concept-Name angegeben werden kann und kein allgemeiner Concept-Ausdruck, wie dies für die Definition des Verwendungspotentials von Werkzeug-Concepts erforderlich ist. Wie bereits in Kap. 3.2.2.1 *Spezifika der Repräsentation in SB-ONE* vorgeschlagen, werden für die Concept-Ausdrücke äquivalente Concepts definiert und deren Namen als Werterestriktion für Rollen verwendet. Die Concepts selbst können wieder als besondere Concepts markiert werden, um bspw. zu vermeiden, daß sie für die Kommunikation mit dem Benutzer direkt verwendet werden.

3.5.2 Unterstützung durch andere Akteure

Der Schwerpunkt dieser Arbeit liegt in der Unterstützung *eines* Benutzer bei der Arbeit mit dem Computer, sodaß der Benutzer in die Lage versetzt wird, seine Ziele durch *eigenes* Handeln zu erreichen. Dennoch wird dies nicht immer möglich sein, bspw. wenn der Benutzer notwendige Aktionen nicht ausführen kann oder darf oder eine notwendige Erklärung unpraktikabel langwierig würde. In vielen Fällen kann aber ein anderer Akteur bei der Ausführung einer Aktion helfen oder einen Teil eines Plans ausführen. Wenngleich eine solche Kooperation mehrerer Akteure nicht Thema dieser Arbeit ist, sollen einige Aspekte einer Erweiterung des Ansatzes auf die Zusammenarbeit mehrerer Akteure kurz skizziert werden. Dabei muß unterschieden werden, ob die Akteure gemeinsam handeln oder ob jeder jeweils nur Teile eines Gesamtplans ausführt.

3.5.2.1 Gemeinsames Handeln

Beim gemeinsamen Handeln führen die beteiligten Akteure zu *einem* Zeitpunkt eine Handlung aus, die – zumindest im Modell – *unteilbar* ist, wobei sich die Fähigkeiten und Eigenschaften der Akteure ergänzen. Beispiele sind der Transport eines Konzertflügels durch mehrere Personen oder der Zugriff auf ein streng vertrauliches Dokument, der nur möglich ist, wenn zwei vertrauenswürdige Personen dies (quasi) gleichzeitig genehmigen.

Die Kombination von Fähigkeiten und Eigenschaften wurde bereits im vorangegangenen Abschnitt behandelt – dort die Fähigkeiten des Benutzers und die Eigenschaften von Werkzeugen. Wird ein Akteur ausgezeichnet, während die anderen die Rolle von Werkzeugen einnehmen, könnte das im vorangegangenen Abschnitt beschriebene Vorgehen direkt für das gemeinsame Handeln eingesetzt werden. Dazu müßten dann Akteurklassen und ihre "Verwendungsmöglichkeiten" modelliert werden. Eine Teilmenge aller denkbaren Akteure – nämlich die potentiellen Benutzer – ist bereits in Klassen mit weitgehend übereinstimmenden Handlungsmöglichkeiten eingeteilt und zusammen mit ihren typischen Handlungsmöglichkeiten durch Stereotypes modelliert. Es bietet sich damit an, dieses bereits vorhandene Wissen zu nutzen und die Modellierung auch auf solche Akteurklassen zu übertragen, die nicht als potentielle Benutzer vorgesehen sind.

Die Feststellung der Eigenschaften von Handlungen, die ein Benutzer mithilfe anderer Akteure ausführen kann, kann analog zur Berechnung eines werkzeugspezifischen Plan-Concepts erfolgen. Auch hierbei wird von einem Plan-Concept im Handlungspotential des Benutzers ausgegangen und dann der Einfluß auf dieses Plan-Concept ermittelt, der hier von den kooperierenden Akteuren ausgeht. Es muß allerdings beachtet werden, daß der Benutzer und die Mitglieder einer Benutzergruppe bzw. Akteurklasse prinzipiell gleichwertig sind. Die für eine Benutzergruppe modellierten Handlungsmöglichkeiten können sowohl für den aktuellen Benutzer verwendet werden (wenn dieser der Benutzergruppe angehört) als auch für die Berechnung des Einflusses von kooperierenden Akteuren. Daß eine Handlung nur mithilfe anderer Akteure durchgeführt werden kann, wird deshalb nicht durch eine Rolle *tool*, sondern durch ein Rolle *req-assistant* des zugehörigen Plan-Concepts modelliert. Durch eine Anzahlrestriktion dieser Rolle kann definiert werden, wie viele Akteure für die Durchführung der zum Plan-Concept gehörenden Handlung notwendig sind. Da hier von der Gleichwertigkeit der Akteure ausgegangen wird, ist dies die Gesamtzahl, also einschließlich des Benutzers bzw. des Akteurs für den geplant wird. Entsprechend können durch Werterestriktionen dieser Rolle auch Anforderungen an *alle* Akteure gestellt werden. Dazu ist eine beschreibungslogische Repräsentation der Akteurklassen erforderlich, die den durch Stereotypes modellierten Benutzergruppen entspricht. Es ist dafür ausreichend, für jeden Stereotype, der eine Fähigkeitspartition enthält, ein primitives (Domain-)Concept zu definieren. Die Vererbungsrelation zwischen Stereotypes wird dabei durch die Spezialisierungsrelation zwischen den zugehörigen Concepts modelliert. Die für die Kombination der Fähigkeiten relevante Information besteht aus den Mengen von Plan-Concepts in den Fähigkeitspartitionen der Stereotypes. Aus diesen Mengen kann der Einfluß auf eine gemeinsame Handlung bestimmt werden, indem die Konjunktion der zugehörigen Plan-Concepts gebildet wird.

3.5.2.2 Verteiltes Handeln

Begrenzte Handlungsmöglichkeiten eines Benutzer können dazu führen, daß ein Ziel nicht ausschließlich durch das Handeln des Benutzers erreicht werden kann, sondern daß dazu ein oder mehrere andere Akteure erforderlich sind, die eigenständig Teilaufgaben übernehmen, die der Benutzer nicht selbst ausführen kann. Ein Beispiel wäre die Bearbeitung einer Datei, die sich auf einem fremden Rechner befindet, zu dem der Benutzer keinen Zugriff hat. Die Gesamtaufgabe kann durchgeführt werden, wenn ein anderer Akteur, der die notwendigen Rechte hat, die Datei kopiert und der Benutzer sie dann bearbeitet.

Aus der Sicht der Plangenerierung ist dies kein besonderes Problem, da hierfür ohnehin nicht berücksichtigt wird *welcher* Akteur einen Planschritt ausführt, solange davon ausgegangen werden kann, *daß* es einen Akteur gibt der dies tun kann. Allerdings müssen die Akteure normalerweise dazu bewegt werden, eine Teilaufgabe zu übernehmen. Die dafür erforderliche Kommunikation ist ein Thema der Forschung zu *Multi-Agent-Systemen*[59], die sich aber überwiegend mit der Interaktion künstlicher Akteure befaßt, und der Forschung zum gemeinsamen Planen[60], die meist von der Interaktion eines Menschen und künstlicher Akteure ausgeht. Für ein System, das einen Benutzer unterstützen soll und bei dem davon ausgegangen werden kann, daß auch die Akteure die dem Benutzer helfen Menschen sind, erscheint es sinnvoll, dem Benutzer selbst die Planung und Ausführung der Kommunikation mit anderen Akteuren zu überlassen. Für den Benutzer ist es dafür allerdings wichtig zu wissen, welche Teilaufgabe zu welchem Zweck von welchem Akteur übernommen werden kann und soll. Die Ausgabe eines Planungsprozesses, der mehrere Akteure berücksichtigt, muß damit zu jedem Schritt auch einen Akteur oder eine Klasse von Akteuren vermerken, von dem er ausführt werden soll. Für *Klassen* von kooperierenden Akteuren zu planen, ist dabei aus mehreren Gründen die sinnvollere Lösung. Erstens dürfte es aufwendig sein, die Eigenschaften aller einzelnen Akteure eines Anwendungsbereich explizit zu repräsentieren, um daraus ein verläßliches Bild der Handlungspotentiale zu gewinnen. Zweitens würde das Planungssystem eine Vielzahl gleichartiger Planoperatoren verarbeiten müssen, wenn der Planungssachbereich aus den Handlungspotentialen *aller einzelnen* Akteure gebildet würde. Auf die negativen Auswirkungen auf das Laufzeitverhalten des Planungsprozesses wurde in Kap. 3.1.1 *Alternativen zum Verarbeitungsschema* hingewiesen. Dazu kommt, daß mit den Akteurklassen, die durch gleichartige typische Handlungsmöglichkeiten gebildet werden, oft auch typische Aufgabenbereiche und Zuständigkeiten verbunden sind. Es ist für den Benutzer erfolgversprechender, sich mit der Bitte um Übernahme einer Teilaufgabe an einen Akteur zu wenden, in dessen *Zuständigkeitsbereich* die Aufgabe fällt, als einen beliebigen Akteur zu bitten – selbst wenn dieser die Aufgabe ebenfalls ausführen könnte. Als Beispiel möge das Einrichten eines neuen Benutzerkontos auf einem Computer dienen. Die Bitte um Übernahme einer solchen Aufgabe dürfte bei einem Akteur aus der Gruppe der Systemadministratoren erfolgversprechender sein, als bei einem sonstigen Angestellten, der – bspw. um eine Systemwartung durchzuführen – ebenfalls Administrator-Rechte hat und damit diese Aufgabe ausführen könnte.

Ein weiterer Aspekt bei der Erzeugung eines Plans für mehrere Akteure liegt darin, daß der Aufwand für die Beauftragung weiterer Akteure nicht vernachlässigt werden kann. Der Planungsprozeß muß demnach Pläne, die der Benutzer selbst durchführen kann, gegenüber solchen bevorzugen, für die er Hilfestellung durch andere benötigt. Dies gilt selbst dann, wenn die Folge der Planschritte dadurch (etwas) länger oder umständlicher wird. Nehmen wir für das Beispiel der Bearbeitung einer Datei die sich auf einem fremden Rechner befindet an, daß der Benutzer keine fremden Verzeichnisse mounten darf, daß er die Datei aber via FTP auf den lokalen Rechner kopieren kann. Obwohl das Kopieren mit FTP (mit den meisten FTP-Programmen) etwas aufwendiger ist als das Mounten eines Verzeichnisses, wäre es unangemessen einen anderen Akteur für das Mounten um Hilfe zu bitten. Generell muß der Planungsprozeß den Benutzer als handelnden Akteur bevorzugen. Wenn eine Hilfestellung durch andere Akteure unvermeidlich ist, müssen Pläne bevorzugt werden, die möglichst wenige Akteure erfordern, da jede Beauftragung eines Akteurs mit zusätzlichem Aufwand verbunden ist. Solche Präferenzen bei der Plangenerierung lassen sich bei vielen Planungssystemen – und auch bei dem in dieser Arbeit eingesetzten System UCPOP [Penberthy und Weld, 1992] – durch Kostenfunktionen erreichen, die jede Auswahl von Planoperatoren bewerten (vgl. dazu auch Kap. 4.5.2 *Steuerung des Planungsprozesses*).

[59] vgl. bspw. [Kraus, 1997], [desJardins *et al.*, 1999]

[60] vgl. Kap. 2.3 *Gemeinsames Planen (mit einem Benutzer)*

Kapitel 4

Präsentation eines Plans

Während das vorangegangene Kapitel die Aspekte der Generierung eines Plans behandelte, den der Benutzer *im Prinzip* ausführen kann, wenden wir uns nun der Frage zu, welche Informationen der Benutzer benötigt, um einen solchen Plan auch tatsächlich erfolgreich ausführen zu können. An erster Stelle benötigt der Benutzer eine Beschreibung des Plans. Im Gegensatz zu einem physischen Objekt, das ein Benutzer anhand seiner Eigenschaften erkennen kann, handelt es sich bei einem Plan um ein abstraktes Objekt, das sich in der Vorstellung des Benutzer bilden muß und das der Benutzer oft noch weiterentwickeln muß. Ziel der Beschreibung ist also nicht die Identifikation eines Objekts aus einer Menge vorgegebener Objekte, sondern eine Art Konstruktionsanleitung, um diese Vorstellung bilden zu können. Eine Beschreibung kann sich deshalb nicht nur auf die Nennung der Schritte eines Plans beschränken, sondern muß auch Informationen über ihre funktionalen Zusammenhänge enthalten. Ein weiterer wichtiger Aspekt bei der Beschreibung eines Plans ist, daß der Akteur der den Plan ausführen soll – also hier der Benutzer – wissen muß, *wie* die Schritte des Plans auszuführen sind. Da die Planschritte Exemplare von Plan-Concepts sind, die auch dem Benutzer bekannt sein können, *kann* der Benutzer auch wissen *wie* die davon abgeleiteten Schritte ausgeführt werden, wobei dieses Wissen (nach Ansicht des Systems) auch unvollständig oder falsch sein kann. Je nach Benutzerwissen muß eine Planpräsentation hier also mehr oder weniger Erklärungen zu Ausführungsmöglichkeiten enthalten. Unter Ausführungsmöglichkeiten eines Plan-Concepts sind hier die möglichen Folgen von Teil-Schritten zu verstehen, die ein Plan-Concept wie beim Planen mit Operatorabstraktion[1] realisieren. Solche Ausführungsmöglichkeiten können quasi als Rezepte für ein Plan-Concept im Modell des Wissens von System und Benutzer abgelegt sein. Auf die dafür notwendigen Erweiterungen des bisherigen Benutzermodells und die Beurteilung dieses Wissens wird im zweiten Unterkapitel noch genauer eingegangen. Die Entscheidungen, ob Erklärungen für den jeweiligen Benutzer erforderlich sind und welche Ausführungsmöglichkeiten eines Plan-Concepts sich für ihn eignen sind Themen des dritten Unterkapitels. Kriterien sind dabei eine Bewertung der Erklärung und ob ein Benutzer gemäß der Erklärungen handeln kann – also ob er die darin enthaltenen Teil-Schritte ausführen kann. Als Bewertung wird hier zunächst eine Abschätzung des Aufwands für das Verstehen der Erklärung durch den Benutzer verwendet. Auf weitere Aspekte wie Benutzerpräferenzen oder Erfolgswahrscheinlichkeiten wird ebenfalls kurz eingegangen. Die Kriterien für eine Entscheidung zwischen einzelnen Ausführungsmöglichkeiten lassen sich auch für die Bewertung des gesamten Plans verwenden. Eine solche Gesamtbewertung kann dann als Grundlage für die Entscheidung dienen, ob der mit dem Plan verbundene Aufwand für den Benutzer noch akzeptabel ist oder nicht. Anstelle einer schwierig zu bestimmenden "Akzeptanzschwelle" wird im fünften Unterkapitel beschrieben, wie der Planungs-

[1] vgl. Kap. 2.1.2.2 *Operatorabstraktion*

prozeß beeinflußt werden kann, damit er die Erzeugung von Plänen bevorzugt, für die der Benutzer möglichst wenige Erklärungen benötigt.

Der Umfang einer Planpräsentation läßt sich reduzierten – und damit normalerweise die Verständlichkeit erhöhen –, wenn Komponenten des Plans, die der Benutzer selbst leicht schließen kann, *nicht explizit* genannt werden. Diese auf Grice' Relevanzkriterium beruhende Idee wird für die Generierung von natürlichsprachlichen Texten (bspw. in ([Zukerman und McConachy, 1993b] und [Horacek, 1997]) genutzt und wird in [Young, 1999] für die Beschreibung von Plänen eingesetzt. Das vierte Unterkapitel setzt sich mit dieser Arbeit auseinander und stellt dem einen Ansatz gegenüber, der die Benutzerkenntnisse berücksichtigt und eine Abwägung zwischen dem Aufwand für das Verstehen einzelner Komponenten einer Präsentation und dem Aufwand für das Schlußfolgern einer solchen Komponente vornimmt. Das Kapitel endet mit den technischen Fragen, die mit der Dekomposition von Plan-Concepts verbunden sind. Dies sind insbesondere die Behandlung von Teil-Ganzes-Relationen in Beschreibungslogik und die Realisierung in BGP-MS sowie eine Gegenüberstellung mit einem Ansatz aus dem Bereich Planerkennung.

Insgesamt hat dieses Kapitel zum Ziel die Inhalte für eine Planpräsentation festzulegen, die für den speziellen Benutzer möglichst gut geeignet sind. Das heißt, daß er mit möglichst geringem eigenem Aufwand in die Lage versetzt wird einen Plan, den er *im Prinzip* ausführen kann, auch tatsächlich in der Realität ausführen zu können.

Dabei wird aber nicht auf die konkrete Darstellung dieser Inhalte eingegangen, weil dies über den Rahmen dieser Arbeit hinausgehen würde. Die Bestimmung des Inhalts ist auch nicht an eine bestimmte Darstellungsform gebunden. Denkbar sind dafür bspw. natürlichsprachliche Anweisungstexte (s.bspw. [McKeown *et al.*, 1990], [Rösner und Stede, 1992], [Moore und Paris, 1993], [Paris und Linden, 1996]), graphische Darstellungen (bspw. [Miksch *et al.*, 1998]) sowie zur Veranschaulichung der zeitlichen Abfolge des Plans auch animierte Darstellungen (bspw. [Wahlster *et al.*, 1993]).

Ebenfalls nicht enthalten ist die Beschreibung der Objekte, die von den Planschritten verwendet werden. Hier sei auf die umfangreiche Literatur zur benutzerspezifischen Beschreibung und Referenzierung von Objekten verwiesen, bspw. [Kobsa, 1985], [Schmauks und Reithinger, 1988], [Sarner und Carberry, 1992].

4.1 Planstruktur

Die einfachste Möglichkeit einen Plan zu beschreiben, besteht in der Nennung der einzelnen Planschritte in einer geeigneten, linearen Reihenfolge. Nach den Standard-Annahmen des Planens (vgl. Kap. 2.1.3 *Voraussetzungen beim Planen*) wird durch die Ausführung einer solchen Folge von Planschritten das Ziel erreicht. Wird – wie in der Handlungsplanung üblich – von einem unintelligenten Roboter als ausführenden Akteur ausgegangen, so reicht eine solche Vorgehensweise aus. Die Planschritte sind aus Operatoren abgeleitet, die die funktionale Schnittstelle des Roboters modellieren und können damit direkt oder nach einem festen Programm von dem Roboter ausgeführt werden. Dieser Ansatz ist aber nicht angemessen, wenn ein Mensch den Plan ausführen soll.

Der erste Grund dafür liegt darin, daß Menschen keine funktionale Schnittstelle haben und Aktionen, die ein Mensch direkt ausführen kann – bspw. das Bewegen eines Armes oder eines Fingers – zu detailliert sind um eine sinnvolle Basis für eine Plangenerierung darstellen zu können. Neben einem sehr hohen Aufwand für die Modellierung des Planungssachbereiches, muß damit gerechnet werden, daß selbst einfache Aufgabenstellungen zu Plänen führen, deren Länge für ein Planungssystem nicht mehr handhabbar wäre[2]. Zudem wären solche Pläne sowohl wegen der großen Länge als auch wegen des sehr niedrigen Abstraktionsniveaus für einen Menschen kaum mehr akzeptabel. Sollen die Pläne

[2] Der Zeitbedarf von Planungsverfahren hängt exponentiell von der Planungslänge ab.

von einem Benutzer eines Hilfesystems ausgeführt werden, bietet es sich vielmehr an, dessen Fähigkeiten als intelligenter Akteur zu nutzen. Damit ist insbesondere die Fähigkeit gemeint, komplexe Planschritte zu verfeinern und dementsprechend zu handeln.

Außerdem können unerwartete Hindernisse die erfolgreiche Ausführung eines Plans in der realen Welt verhindern. Solche Hindernisse können von *unvorhersehbaren Ereignissen* herrühren oder durch die Plangenerierung unberücksichtigt sein, weil sie Sachverhalte betreffen, die *nicht modelliert* sind. Dabei sei angemerkt, daß es prinzipiell nicht möglich ist, *alle* Bedingungen zu modellieren, die in der realen Welt für eine erfolgreiche Planausführung erfüllt werden müssen[3]. Im Beispielsachbereich sind *ein Fenster der Bedienoberfläche, das durch eine wichtige Nachricht des Systemadministrators hervorgerufen wurde und ein anderes Fenster überdeckt* bzw. *eine Kaffeetasse auf dem Maus-Pad* Beispiele für die beiden Typen von unerwarteten Hindernissen. Oft wird der Benutzer solche Hindernisse durch zusätzliche Planschritte beseitigen können und so dennoch seine Ziele erreichen.

Beide Aspekte stützen die Ansicht, daß der Benutzer den generierten Plan nicht einfach als unumstößliches Rezept ansehen kann, sondern lediglich als Ausgangspunkt für eine Weiterentwicklung (vgl. auch [Webber *et al.*, 1995]). Um die Funktionsfähigkeit des Plans nicht zu beeinträchtigen, darf der Benutzer durch eine Verfeinerung oder Modifikation des Plans weder bereits erreichte Teilergebnisse zerstören, noch neue Hindernisse für die weitere Planausführung aufbauen. Die Aufgabe ist so eine *Fortsetzung des Planungsprozesses durch den Benutzer*. Dazu bedarf es nicht nur der Kenntnis über die Voraussetzung und den Effekt der einzelnen Planschritte, sondern auch der Kenntnis der Beiträge, die sie für den Gesamtplan leisten. Durch diese Notwendigkeit läßt sich auch erklären, warum ein Benutzer normalerweise wissen will, wozu welcher Schritt im Plan ausgeführt werden muß (vgl. [Gilbert und Jirotka, 1990], [Kidd, 1985]).

Die Information über den Beitrag der einzelnen Planschritte für den Plan wird bereits bei der Generierung des Plans verwendet und in vielen Planungssystemen – wie auch im in dieser Arbeit eingesetzten UCPOP – explizit als Datenstruktur repräsentiert und als *Causal Link* bezeichnet (vgl. Kap. 2.1.1 *Least Commitment Planning*). Ein Causal Link (e, l, v) ist eine Relation zwischen zwei Planschritten e und v und dem Literal l, das Effektkomponente von e und Vorbedingungskomponente von v ist. Anders ausgedrückt erzeugt e die Bedingung l für v. Für einen (vollständigen) Plan muß es für jede Vorbedingungskomponente vk_i eines Planschritts v einen Causal Link zu einem Planschritt e, der die Vorbedingungskomponente erzeugt, geben[4]. Ein solcher Causal Link gibt damit den[5] Beitrag von e für den Plan an, der gleichzeitig auch der Grund für die Existenz von e im Plan darstellt: das Erzeugen von vk_i. In einem vollständigen Plan muß zu jeder Vorbedingungskomponente eines jeden Planschritts ein Causal Link führen. Damit impliziert die Information über alle Causal Links des Plans für jeden Planschritt einzeln die Information über seine Vorbedingungskomponenten sowie über die Effektkomponenten, die im Plan eine Rolle spielen. In Kap. 3.2.2 *Planoperatoren in Beschreibungslogik* wurden Vorbedingungs- und Effektkomponenten als identifizierende Attribute eines Plan-Concepts definiert. Die Gesamtheit der Causal Links stellt damit nicht nur eine Beschreibung der Struktur des Plans dar, sondern vermittelt gleichzeitig die wichtigsten beschreibenden Eigenschaften der Planschritte. Auf die Präsentation darüber hinausgehender Eigenschaften – also Effektkomponenten, die im Plan nicht gebraucht werden – kann verzichtet werden, weil sie für die Ausführung des Plans irrelevant sind. Eine Beschränkung von Beschreibungen auf die Eigenschaften, die für das Ausführen eines Plans benötigt werden, findet sich bspw. auch in [Sarner und Carberry, 1992] – dort speziell für Begriffsdefinitionen.

[3] vgl. das Qualifikationsproblem [McCarthy, 1977] u. Einleitung zu Kap. 2.1 *Handlungsplanung*.

[4] Um dabei auch die Startsituation und das Ziel des Plans zu berücksichtigen, wird von der Existenz eines Start-Schritts und eines Ziel-Schritts ausgegangen, der alle Literale der Startsituation erzeugt bzw. das Ziel als Vorbedingung hat (vgl. Kap. 2.1.1 *Least Commitment Planning*).

[5] bzw. *einen* Beitrag, falls e Bedingungen für mehr als einen Planschritt erzeugt.

Die Kenntnis der Causal Links ist für ein Weiter- oder Umplanen aus dem gleichen Grund wichtig wie bei der Plangenerierung selbst. Für den vollständigen Plan der ausgeführt werden soll darf kein Causal Link bedroht sein. D.h. es muß sichergestellt sein, daß zwischen den Planschritten e und v eines Causal Links (e, l, v) kein Planschritt ausgeführt werden darf der l negiert, weil ansonsten nicht mehr garantiert werden kann, daß Planschritt v – und damit der gesamte Plan – ausführbar ist.

Die Notwendigkeit der Berücksichtigung von Causal Links sei anhand eines Plans für das Beispielziel *einen Ausdruck einer Datei bekommen, die sich auf einem fremden Computer befindet* demonstriert[6]. Das Ziel läßt sich erreichen, indem die Datei zunächst ins lokale Dateisystem gebracht wird, dann im passenden Textverarbeitungssystem geöffnet und dann damit ausgedruckt wird. Abb. 4.1 zeigt einen Ausschnitt des Plans, bei dem FTP eingesetzt wird, um die Datei ins lokale Dateisystem zu bringen. Ein solcher Plan könnte von den Beispielbenutzern *Oscar*, *Nick* und *Theo* ausgeführt werden (vgl. Kap. 3.3.3 *Stereotypes*). Die Abbildung stellt die Planschritte als Rechtecke dar und Causal Links als Pfeile, die mit den Literalen beschriftet sind. Dabei sind nur die hier interessierenden Causal Links eingezeichnet. Die Planschritte können von links nach rechts ausgeführt werden, also *Starten des FTP-Programms* und dann *des Textverarbeitungssystems*, *Einschalten des Binärmodus des FTP-Programms*, *Herunterladen der Datei* und dann *Öffnen der* inzwischen lokalen *Datei mit dem Textverarbeitungssystem*. Dies ist eine der möglichen Ausführungsreihenfolgen, die den Zeitbedingungen entspricht, die durch die Causal Links gegeben sind.

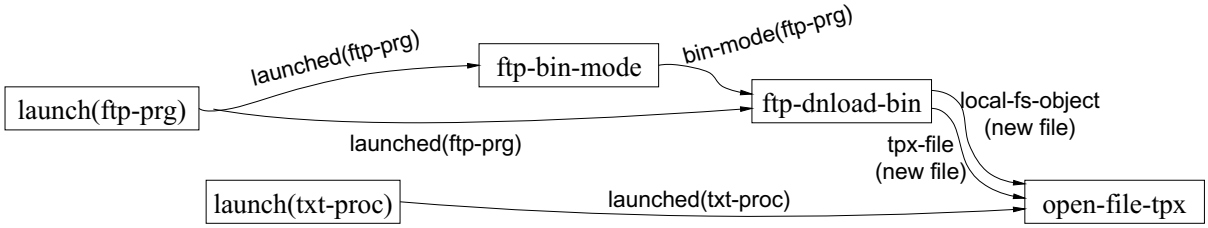

Abbildung 4.1: Vereinfachter Planausschnitt für das Ziel *Einen Ausdruck einer Datei bekommen, die sich auf einem fremden Computer befindet.*

Als Beispiel für ein Hindernis, das der Planausführung entgegensteht, sei von einer graphischen Bedienoberfläche ausgegangen und angenommen, daß unmittelbar bevor der Planschritt *ftp-bin-mode* ausgeführt werden soll, ein Fenster der Bedienoberfläche das Fenster des FTP-Programms verdeckt und damit die Bedienung des FTP-Programms verhindert. Ein solches Fenster kann bspw. von einer wichtigen Meldung des Systemadministrators hervorgerufen sein oder durch das Starten des Textverarbeitungssystems unmittelbar zuvor. Eine Möglichkeit dieses Hindernis zu beseitigen besteht darin alle Anwendungen zu beenden, deren Fenster das Fenster des FTP-Programms verdecken – möglicherweise also auch das Textverarbeitungsprogramm. Dies würde aber eine Bedrohung des Causal Links zwischen dem Starten des Textverarbeitungssystems und dem Öffnen der Datei darstellen und damit die Ausführbarkeit des Plans gefährden. Weiß der Benutzer um den Causal Link, kann er eine andere Möglichkeit wählen, um das Hindernis zu beseitigen; bspw. das störende Fenster verkleinern oder in den Hintergrund versetzen. Oder er kann die Negation der Vorbedingung für das Öffnen der Datei bewußt in Kauf nehmen und sie durch ein späteres erneutes Starten des Textverarbeitungssystems wieder herstellen.

Dieses Beispiel zeigt sowohl die Notwendigkeit des Benutzers den Planungsprozeß fortzusetzen

[6]Kap. 5.3.2 *Benutzerspezifische Pläne* zeigt die verschiedenen Pläne, mit denen die Beispielbenutzer dieses Ziel erreichen können.

– im konkreten Fall, weil die genaue Lage und Größe von Bildschirm-Fenstern nicht modelliert ist – als auch die Wichtigkeit der Kenntnis der Causal Links für diese Aufgabe. Diese Kenntnis ist auch für eine Dekomposition eines Planschritts in Teilschritte von Bedeutung, auf die im nächsten Abschnitt noch eingegangen wird. Der Grund liegt darin, daß die Teilschritte (Neben-) Effekte haben können, die einen Causal Link des Plans bedrohen *können*. Wenn der Benutzer den Plan weiterentwickelt, muß er also darauf achten, daß solche Bedrohung vermieden werden.

Andererseits ist es normalerweise nicht notwendig, dem Benutzer *alle* Causal Links zu nennen. Dies trifft insbesondere auf die Causal Links zu, die vom Start-Schritt ausgehen und deren Literal nicht vom Benutzer negiert werden kann. Solche Causal Links existieren, ohne daß der Benutzer (oder sonst ein Akteur) etwas tun muß und sie können damit auch nicht den Grund für das Auftreten eines Planschritts im Plan anzeigen. Da sie nicht bedroht werden können, entfällt auch die Notwendigkeit sie bei einer evtl. Weiterplanung zu berücksichtigen. Solche Causal Links werden deshalb im weiteren als *irrelevant* betrachtet und nicht weiter berücksichtigt. Das heißt, daß grundsätzlich nur die *relevanten* Causal Links gemeint sind, wenn im Kontext der Planpräsentation von *den Causal Links* bzw. *der Menge der Causal Links* eines Plans gesprochen wird. Beispiele für irrelevante Causal Links ergeben sich aus den (in Abb. 4.1 nicht dargestellten) Vorbedingungen der *launch*-Schritte im o.g. Plan. Das entsprechende Plan-Concept fordert als Vorbedingung, daß es sich bei dem Objekt das gestartet wird um ein Anwendungsprogramm handeln muß (für das das Prädikat *application* wahr sein muß). Diese Bedingung ist für ein Objekt im Beispielsachbereich entweder wahr oder falsch, sie kann aber nicht verändert werden. Die zugehörigen Causal Links fehlen in Abb. 4.1 nicht nur aus Gründen der Übersichtlichkeit, sondern auch weil sie irrelevant sind.

Bei der Präsentation von Causal Links ist auch zu beachten, daß es für die Definition des Planungssachbereichs möglich ist, Prädikate als Folgerung aus anderen Prädikaten durch Axiome zu definieren (vgl. Kap. 3.4.2.3 *Nichtatomare Formeln im Planungssachbereich*). Dies kommt einer übersichtlicheren Modellierung zugute und ermöglicht eine adäquate(re) Modellierung des Benutzerwissens (s.a. Kap. 3.4.2.4 *Ableitung von Abhängigkeiten aus dem Benutzermodell*). Im Beispielsachbereich wurde dies u.a. genutzt, um das Prädikat *local-fs-object*[7] zu definieren, das für ein Datei-Objekt wahr ist, dessen Name sich in einem lokalen Verzeichnis befindet. Das Prädikat kann nicht direkt durch einen Planoperator bzw. Plan-Concept verändert werden, sondern lediglich indirekt. Im Beispielplan aus Abb. 4.1 bewirkt der Schritt *ftp-dnload-bin*, daß ein neues Datei-Objekt erzeugt wird, das durch einen (ebenfalls neuen) Datei-Namen bezeichnet wird, der *einem lokalen Verzeichnis* zugeordnet wird. Dadurch werden mehrere Prädikate wahr, die zusammen für Schritt *open-file-tpx* die Vorbedingungskomponente, daß *local-fs-object* für das Datei-Objekt wahr sein muß, implizieren. Im Gegensatz zu den während des Planungsprozesses aufgebauten Causal Links für jedes einzelne der Prädikate reicht es aus, dem Benutzer nur einen Link für das Prädikat *local-fs-object* zu präsentiert, da die relevante Information so zusammengefaßt wird. Causal Links einer Planpräsentation sind deshalb immer in diesem Sinne zu verstehen.

Zusammenfassend sei festgehalten, daß es im Gegensatz zur Beschreibung von konkreten Objekten oder Begriffen, weniger darauf ankommt ausreichend viele identifizierende Eigenschaften zu nennen, um ein Objekt in einer Menge wiederzuerkennen oder als Exemplar einer Klasse zu identifizieren. Vielmehr ist der Plan ein abstraktes Objekt, das der Benutzer meist noch konkretisieren und teilweise auch modifizieren muß. Dabei darf die ursprüngliche Funktion des Plans nicht bzw. nur insoweit verändert werden, daß unberücksichtige Hindernisse für eine erfolgreiche Planausführung beseitigt werden. Für die Präsentation eines Plans steht also die Vermittlung der Kenntnisse über das Zusammenspiel der Komponenten und damit die Struktur des Plans im Vordergrund, die den Benutzer bei seiner Fortsetzung bzw. Revision des Planungsprozesses unterstützen.

[7] hier genau: *local-named-fs-object* (s.a. Anhang B.3 *Potentielle Axiome*)

4.2 Wissen wie ein Plan-Concept ausgeführt wird

Für die Aufgabe der Planpräsentation, einem Benutzer einen Plan in der Weise zu präsentieren, daß dieser in die Lage versetzt wird, mit diesem Plan seine Ziele zu erreichen, reichen die Kenntnisse über die Struktur des Plans und die Wirkung der Planschritte allerdings nicht aus. Der Benutzer muß auch wissen *wie* er die Planschritte ausführen kann[8]. Dies stellt eine weitere Besonderheit gegenüber der Beschreibung von konkreten Objekten oder Begriffen dar.

Während die Information über die Struktur des durch den Planungsprozeß neu generierten Plans für den Benutzer unbekannt sein *muß*, kann der Benutzer aber durchaus Kenntnisse darüber haben *wie* die Planschritte ausgeführt werden. Da die Planschritte Exemplare von Plan-Concepts sind, kann der Benutzer diese Kenntnis aus seinem generellen Planungswissen ableiten, sofern ihm die entsprechenden Plan-Concepts bekannt sind. Die Präsentation eines Plans muß deshalb berücksichtigen, welche Annahmen über Ausführungsmöglichkeiten der Benutzer selbst hat und welche Informationen er dazu noch zusätzlich benötigt, um den Plan ausführen zu können. Dabei ist zu beachten, daß die Annahmen des Benutzers auch unvollständig oder falsch sein können. Die Beurteilung der Benutzerannahmen wird – entsprechend des Rahmens dieser Arbeit – in Bezug auf das Wissen des Systems anhand des Benutzermodells vorgenommen.

Dieser Abschnitt behandelt die Frage, wie Annahmen über Ausführungsmöglichkeiten von Plan-Concepts im Benutzermodell repräsentiert werden können, die für die Abschätzung der Information benötigt werden, die der Benutzer noch zusätzlich für eine erfolgreiche Planausführung benötigt. Zum Ende des Abschnitts wird kurz darauf eingegangen, inwiefern dieses Modell auch für andere Kombinationen eines Akteurs, der beobachtet und beurteilt, und eines Akteurs, der beurteilt wird und einen Plan ausführen soll, geeignet ist.

Aus Gründen der einfacheren Sprechweise wird im weiteren nicht streng zwischen Plan-Concepts und den davon abgeleiteten Exemplaren – den Planschritten – unterschieden. In diesem Sinne sind die Kenntnisse eines Akteurs über die Eigenschaften und insbesondere über die Ausführungsmöglichkeiten eines Plan*schritts* als Kenntnisse zu verstehen, die von seinem generellen Planungswissen über das zugehörige Plan-*Concept* geerbt sind.

4.2.1 Dekomposition eines Plan-Concepts

Für die Beurteilung der Ausführungsmöglichkeiten eines Planschritts bzw. eines Plan-Concepts muß unterschieden werden, ob der Benutzer den Planschritt direkt in einem Schritt ausführen kann, oder ob der Planschritt zunächst in Teilschritte zerlegt werden muß. Allerdings ist dies keine (beobachtbare) Eigenschaft des Planschritts bzw. Plan-Concepts, sondern grundsätzlich eine Festlegung, die vom Detaillierungsgrad abhängt, mit dem der Planschritt betrachtet werden soll. Solange der physikalische Makrokosmos betrachtet wird, könnte jede Handlung in Teilschritte zerlegt werden. Selbst eine Armbewegung, die im vorangegangenen Abschnitt als Beispiel für direkt ausführbare Handlungen angeführt wurde, könnte in eine Folge von Muskelanspannungen zerlegt werden.

Für die Planpräsentation ist die Festlegung einer direkten Ausführbarkeit bezüglich des Wissens über Ausführungsmöglichkeiten sinnvoll. Dazu sei festgelegt, daß das System dem Benutzer die *atomare* Kenntnis eines Plan-Concepts zuspricht, wenn es annimmt, daß der Benutzer für eine erfolgreiche Ausführung des Plan-Concepts keine zusätzlichen Informationen benötigt. Damit kann dem Benutzer die atomare Kenntnis eines Plan-Concepts unabhängig davon zugesprochen werden, ob (dem Benutzer oder dem System) Zerlegungen des Plan-Concepts bekannt sind. Es ist offensichtlich, daß die atomare Kenntnis benutzerspezifisch ist. Bspw. kann ein Benutzer atomare Kenntnis über das *Starten eines Anwendungsprogramms* haben, während ein anderer eine Erklärung benötigt, wie

[8] vgl. bspw. auch den Unterschied zwischen *Beliefs* und *Skills* in [Zukerman und McConachy, 1993a].

er eine solche Aktion durchführen kann. Als erste Erweiterung der bisher vorgeschlagenen Benutzermodellierung wird deshalb ein Prädikat $atomar_P$ für Plan-Concepts in einer beliebigen Wissens-Partition P eingeführt, auf dessen konkrete Repräsentation noch in Kap. 4.6.2 *Repräsentation von Dekompositionen in BGP-MS* eingegangen wird. Dieses Prädikat ist in Partition SBUB[9] genau dann für ein Plan-Concept wahr, wenn das System dem Benutzer die atomare Kenntnis des Plan-Concepts zuspricht. Das Prädikat wird auch für Planschritte verwendet, wobei der jeweilige Wert von dem zugehörigem Plan-Concept geerbt wird. Da das Prädikat in dieser Arbeit nur in der Partition SBUB verwendet wird, sei die folgende vereinfachte Sprechweise vereinbart: ist für ein Plan-Concept bzw. einen Planschritt *pc* $atomar_{\mathrm{SBUB}}(pc)$ wahr, wird *pc* als *atomares* Plan-Concept bzw. *atomarer* Planschritt bezeichnet; ansonsten als *nicht-atomares* Plan-Concept bzw. *nicht-atomarer* Planschritt.

Es sei angemerkt, daß die atomare Benutzerkenntnis eines Plan-Concepts *nicht* impliziert, daß der Benutzer das Plan-Concept auch tatsächlich ausführen kann. Dafür ist zusätzlich notwendig, daß er das Plan-Concept auch prinzipiell ausführen kann. Ein Beispiel für den Unterschied ist wieder das *Starten eines Anwendungsprogramms*, das ein Benutzer atomar kennen kann, ohne gleichzeitig die dafür notwendigen Rechte zu besitzen. Nur falls $atomar_{\mathrm{SBUB}}(pc)$ (und damit auch $pc \in$ SBUB) und $pc \in$ SBUC nimmt das System an, daß der Benutzer das Plan-Concept *pc* ohne Erklärungen ausführen kann.

Für ein nicht-atomares Plan-Concept muß der Benutzer eine geeignete Zerlegung in Teilschritte kennen, die das Plan-Concept realisiert; d.h. die er anstelle des Plan-Concepts bzw. eines davon abgeleiteten Planschritts ausführen kann. Wie bereits erwähnt, können Handlungen generell weiter zerlegt werden. Relevant ist dies für eine Planpräsentation aber nur bis zu Teilschritten, die der Benutzer atomar kennt, weil der Benutzer darüber hinaus keinerlei Erklärungen benötigt.

Die Ähnlichkeit einer solchen Zerlegung von Plan-Concepts mit der Dekomposition von Planoperatoren beim Planen mit Operatorabstraktion[10] ist unverkennbar. Dort sind komplexen Planoperatoren ein oder mehrere Dekompositionen zugeordnet, die die Struktur eines Plans haben. Im Laufe des Planungsprozesses wird jeder komplexe Planoperator durch eine geeignete Dekomposition im zu entwickelnden Plan ersetzt. Eine Dekomposition eines Planoperators hat i.a. mehr Vorbedingungskomponenten und mehr Effektkomponenten als der Planoperator selbst. Deshalb erfordert die Ersetzung eines Operators durch eine Dekomposition die Prüfung, ob dadurch Causal Links bedroht werden. Ggf. müssen Maßnahmen dagegen ergriffen werden. Außerdem muß geprüft werden, ob das Einsetzen weiterer Planschritte für das Erfüllen der zusätzlichen Bedingungen erforderlich ist. Dieser Prozeß endet mit sogenannten *primitiven Planoperatoren*, denen keine Dekompositionen zugeordnet sind. Erst diese Planoperatoren berücksichtigen alle Gegebenheiten des Sachbereichs, die für dessen Modellierung als relevant eingeschätzt wurden.

Dies ist ein wesentlicher Unterschied zu den Gegebenheiten bei der Präsentation eines vollständigen Plans. Ein solcher Plan berücksichtigt *alle relevanten* Gegebenheiten des Sachbereichs[11]. Die Teilschritte, die ein Plan-Concept realisieren, müssen deshalb zwar alle Effektkomponenten des Plan-Concepts erzeugen, aber keine – zumindest keine relevanten – die darüber hinausgehen. Entsprechendes gilt für die Vorbedingungskomponenten. Sind diese Bedingungen erfüllt, kann das Plan-Concept bzw. ein davon gebildeter Planschritt durch die zugehörigen Teilschritte ersetzt werden, ohne daß dies Auswirkungen auf den Gesamtplan hat – insbesondere ohne daß Causal Links bedroht oder zusätzliche Planschritte notwendig werden. Im weiteren sei von dieser Bedingung ausgegangen. Da bereits durch die ursprüngliche Plangenerierung sichergestellt wird, daß der Gesamtplan (durch den Benutzer) ausführbar ist und daß damit die vorgegebenen Ziele erreicht werden, kann nun auf eine explizite

[9] Strenggenommen in *View* SBUB; vgl. Kap. 3.3.1 *Akteurmodelle*

[10] vgl. [Sacerdoti, 1977], [Erol *et al.*, 1994a] und Kap. 2.1.2.2 *Operatorabstraktion*

[11] Dies ist offensichtlich eine Abstraktion der Realität, die aber wegen des unlösbaren Qualifikationsproblems beim Planen unvermeidlich ist (vgl. Kap. 1 Einleitung).

Modellierung von Vorbedingungen und Effekten für die Teilschritte bzw. den Plan-Concepts, von denen sie abgeleitet wurden, verzichtet werden.

Dementsprechend lehnt sich die Modellierung des Wissens *wie ein Plan-Concept ausgeführt werden kann* zwar an die Repräsentation komplexer Planoperatoren beim Planen mit Operatorabstraktion an, allerdings mit folgenden Vereinfachungen. Es sollen auch hier jedem Plan-Concept eine oder mehrere Dekompositionen zugeordnet werden können. Die Planschritte dieser Dekompositionen werden aber nur bzgl. der zeitlichen Anordnung in Relation gesetzt. Vorbedingungen und Effekte werden nicht berücksichtigt und damit entfällt auch die Notwendigkeit zur Repräsentation von Causal Links oder Parameterbindungen. Für die Planpräsentation im Rahmen dieser Arbeit sei ausschließlich von solchen vereinfachten Dekompositionen ausgegangen. Sie können als *vorgefertigte Rezepte* für die Realisierung von Plan-Concepts aufgefaßt werden.

Durch eine solche Modellierung werden die Plan-Concepts in zwei Gruppen eingeteilt. Die erste Gruppe modelliert explizit die Veränderungsmöglichkeiten im Anwendungssachbereich. Bezogen auf das gewählte Abstraktionsniveau und das (ggf. angenommene) Wissen des planenden Akteurs beschreiben die Vorbedingungen und Effekte der Plan-Concepts vollständig die Bedingungen unter denen die Plan-Concepts ausgeführt werden können und die Veränderungen, die ihre Ausführung bewirkt. Wird auf Basis solcher Plan-Concepts (bzw. der zugehörigen Planoperatoren) mithilfe eines korrekten Planungsverfahrens ein Plan erzeugt, so wird durch die Ausführung des Plans – wiederum bezogen auf das gleiche Abstraktionsniveau und Wissen – garantiert[12] das Ziel des Plans erreicht. Diese Garantie kann unabhängig von der Art des Planungsverfahren (bspw. hierarchisch oder nicht-hierarchisch) gegeben werden. Dies gilt auch unabhängig davon, ob die Plan-Concepts noch in Teil-Plan-Concepts dekomponiert werden können. Da die Vorbedingungen und Effekte der Plan-Concepts vollständig modelliert sind, können eventuelle Teil-Schritte weder zusätzliche Bedingungen erfordern noch zusätzliche Effekte hervorrufen. Aus der Sicht der Plangenerierung können die Plan-Concepts deshalb als unteilbar oder – im Sinne von Operatorabstraktion – als primitiv angesehen werden.

Dies muß aber nicht die Sicht des ausführenden Akteurs sein, für den es durchaus notwendig sein kann einzelne Schritte des generierten Plans in Teil-Schritte zu dekomponieren, um sie ausführen zu können. Bei der hier vorgeschlagenen Modellierung gehören solche Teil-Schritte bzw. die zugehörigen Teil-Plan-Concepts zur zweiten Gruppe von Plan-Concepts, für die normalerweise[13] weder Vorbedingungen noch Effekte modelliert sind. Solche Plan-Concepts werden als *unvollständig* markiert[14] und von vorn herein vom Planungsprozeß ausgeschlossen. Das heißt allerdings nicht, daß ihre Ausführung in der Realität keinerlei Wirkung hat. Da sie aber nur in den o.g. *vorgefertigten Rezepten* für die Realisierung von übergeordneten Plan-Concepts eingesetzt werden, sind die Auswirkungen ihrer Vorbedingungen und Effekte bereits durch die Plan-Concepts die sie realisieren modelliert. Das Wissen über die Rezepte wird dabei wie beim Planen mit Operatorabstraktion vom Entwickler des Sachbereichsmodells eingebracht[15]. Er bestimmt damit auch die Einteilung der Plan-Concepts in die beiden Gruppen und mögliche Abstraktionsniveaus für die Plangenerierung. Diese können für verschiedene Benutzer unterschiedlich sein, weil die Plan-Concepts der Handlungspotentiale verschiedener Benutzer unterschiedlich modelliert sein können. Die Wahl eines für den Benutzer angemessenen Abstraktionsniveaus ist von Bedeutung, weil der Benutzer für darüber hinausgehende Bedingungen und Effekte keine Unterstützung vom System bekommen kann. In den Beispielen dieser Arbeit wird allerdings nicht von der Möglichkeit *benutzerspezifischer* Abstraktionsniveaus Gebrauch

[12] mit den im Kap. 2.1.3 *Voraussetzungen beim Planen* genannen Einschränkungen

[13] Allerdings *können* solche Vorbedingungen oder Effekte – ggf. zuammen mit Bindungsbedingungen (s.u.) – zur Beschreibung der an den Planschritten beteiligten Objekte genutzt werden (vgl. bspw. die Dekomposition des Plan-Concepts *ftp-download-file-bin* in Abb. 4.10).

[14] für die konkrete Repräsentation s. Kap. 4.6.2 *Repräsentation von Dekompositionen in BGP-MS*

[15] vgl. Kap. 2.1.2.2 *Operatorabstraktion*

gemacht. Bspw. wird das Erscheinen und Verschwinden, sowie die genaue Position von Fenstern einer Computer-Bedienoberfläche nicht modelliert. Damit wird allen Benutzern zugemutet, falls erforderlich Fenster geeignet zu verschieben oder ein Zeigegerät so zu positionieren, daß eine Schaltfläche betätigt werden kann.

Die bewußt unvollständige Modellierung von Plan-Concepts, die nicht für die Plangenerierung eingesetzt werden sollen (und können), eignet sich besonders für Plan-Concepts, die ein sehr niedriges Abstraktionsniveau haben. Damit sind solche Plan-Concepts gemeint, die – einzeln betrachtet – keine *relevanten* Vorbedingungen und Effekte haben und deren Bedeutung sich erst in einem bestimmten Ausführungskontext ergibt. Im Beispielsachbereich sind dies vor allem die elementaren Bedienmöglichkeiten des Computers, also ein *Maus-Klick* oder das *Betätigen einer Taste* oder einer *"OK"-Schaltfläche* auf einer graphischen Bedienoberfläche. Sie können für die Realisierung sehr vieler Plan-Concepts eingesetzt werden und haben je nach Ausführungskontext unterschiedliche Effekte. Bspw. hängt das Resultat der Betätigung einer "OK"-Schaltfläche in der Regel vom Fenster ab, in dem sich die Schaltfläche befindet und oft zusätzlich vom Zustand des Programms, zu dem das Fenster gehört. Für das Fenster "Windows beenden" der (deutschen) Bedienoberfläche von Microsoft-Windows®, hängt das Resultat bspw. von der Auswahl ab, die in diesem Fenster getroffen wurde. Eine Modellierung aller möglichen Resultate einer *Betätigung einer "OK"-Schaltfläche* in beliebigen Fenstern würde sowohl bei der Modellierung als auch beim Planungsprozeß zu einem sehr hohen Aufwand führen.

Der Verzicht auf die explizite Berücksichtigung der Bedingungen für unterschiedliche Effekte von Plan-Concepts wird möglich, wenn der Effekt durch den Ausführungskontext eindeutig wird. Genau dieses ist der Zweck von vorgefertigten Rezepten. Lautet bspw. ein solches Rezept für das Herunterfahren eines Windows-Computers: *Betätigen der "Start"-Schaltfläche*, *Auswahl von "Beenden"*, *Auswahl von "Herunterfahren"*, *Betätigen der "OK"-Schaltfläche*, so wird durch den Ausführungskontext in dem die *"OK"-Schaltfläche* betätigt wird bestimmt, daß der Computer heruntergefahren wird.

Eine formale Verifikation der Korrektheit von Dekompositionen eines Plan-Concepts ist wegen der unvollständigen Modellierung der darin enthaltenen Teil-Plan-Concepts weder möglich noch sinnvoll. Da verschiedene Akteure unterschiedliche *Annahmen* über die Dekompositionen eines Plan-Concepts haben können, muß es ohnehin möglich sein, auch *falsche Dekompositionen* zu repräsentieren; d.h. Dekompositionen, die nicht die Effekte der dazu gehörenden Plan-Concepts erzielen. Maßgeblich für die Beurteilung, *ob* eine Dekomposition eines Plan-Concepts korrekt ist oder nicht, ist – wie auch bei anderen Sachverhalten[16] – das Wissen des Systems, das als korrekt vorausgesetzt wird.

Die Repräsentation des Wissens über Ausführungsmöglichkeiten von Plan-Concepts erfolgt analog zur Einführung des Prädikats *atomar* für Plan-Concepts. Dekompositionen können Plan-Concepts in beliebigen Wissens-Partitionen zugeordnet werden, um die Annahmen des zugehörigen Akteurs über die Ausführungsmöglichkeiten des Plan-Concepts zu modellieren. Als Zugriffsfunktionen seien partitionsspezifische Funktionen $dekomp_{\mathrm{P}}(pc)$ definiert, die die Menge aller Dekomposition liefern, die einem Plan-Concept *pc* in einer Partition P zugeordnet sind. Eine Dekomposition eines Plan-Concepts wird in Anlehnung an [Russell und Norvig, 1995] und [Erol *et al.*, 1994a] als Paar (*steps*, *constraints*) definiert, wobei *steps* eine Menge $\{s_1{:}pc_1, \ldots, s_n{:}pc_n\}$ von Planschritten s_i mit den zugehörigen Plan-Concepts pc_i darstellt. Die Unterscheidung zwischen Planschritten und Plan-Concepts ist hier notwendig, weil eine Dekomposition mehrere Exemplare des gleichen Plan-Concepts enthalten kann; die pc_i sind also nicht notwendigerweise (alle) verschieden. *Constraints* ist eine Menge von Bedingungen an die Komponenten der Dekomposition. Dazu zählen Reihenfolgebedingungen

[16] vgl. Kap. 3.1 *Handlungspotential und Wissen von Benutzern*

für die Dekompositionsschritte sowie Bindungsbedingungen für die Parameter der Schritte und des zu dekomponierenden Plan-Concepts. Bindungsbedingungen können für die Beschreibung der an den Planschritten beteiligten Objekte genutzt werden (vgl.a. Fußnote 13). Da dies nicht Thema dieser Arbeit ist, seien sie hier nur der Vollständigkeit halber erwähnt. Reihenfolgebedingungen sind Tupel einer Ordnungsrelation auf den Planschritten, die Bedingungen für die Ausführungsreihenfolge der Schritte bestimmt. Die Relation selbst, die im weiteren *before* genannt sei, läßt sich durch transitiven Abschluß gewinnen. Für den Zugriff auf die Komponenten einer Dekomposition seien die Funktionen *steps*, *plan-concepts*, *constraints* und *before* definiert. Für eine Dekomposition liefern sie die Mengen der Planschritte, Plan-Concepts, aller Bedingungen sowie der angegebenen Reihenfolgebedingungen. Analog werden diese Funktionen auch für den Zugriff auf die Komponenten eines Plans verwendet. Für die konkrete Realisierung der Repräsentation in BGP-MS sei wiederum auf den Abschnitt 4.6.2 *Repräsentation von Dekompositionen in BGP-MS* verwiesen.

4.2.2 Beurteilung von Wissen, wie ein Plan-Concept ausgeführt wird

Da das Wissen eines Benutzers auch unvollständig oder falsch sein kann, muß bei der Präsentation eines Plans berücksichtigt werden, ob das vorhandene Benutzerwissen zu den einzelnen Planschritten hinsichtlich Qualität und Quantität ausreicht, damit der Benutzer damit seine Ziele erreichen kann oder ob er hierfür noch Erklärungen benötigt. Ziel dieses Abschnitts ist es, Kriterien zu bestimmen, die aufgrund des Benutzermodells eine solche Entscheidung erlauben. Dafür werden sogenannte *Beurteilungsprädikate* für das Wissens des Benutzers definiert. Das im vorangegangenen Abschnitt eingeführte Prädikat *atomar* für ein Plan-Concept ist damit sehr vergleichbar. Allerdings werden die Werte der Beurteilungsprädikate nicht explizit im Benutzermodell repräsentiert, sondern aus verschiedenen Komponenten des Benutzermodells abgeleitet.

Zur Beurteilung des Benutzerwissens über die Ausführungsmöglichkeiten eines Plan-Concepts wird das Beurteilungsprädikat *verläßlich* definiert, das sich für die Planpräsentation nur dadurch vom Prädikat *atomar* unterscheidet, daß es nicht explizit im Benutzermodell repräsentiert ist. Zur Beurteilung einer Dekomposition wird zwischen *falschem*, *nutzbarem* und *selbständig nutzbarem* Wissen unterschieden. Alle diese Beurteilungen stellen die Sicht des beobachtenden Akteurs – hier das System – in Bezug auf einen Akteur der einen Plan ausführen soll – hier der Benutzer – dar. Sie beruhen auf dem Wissen des beobachtenden Akteurs sowie den Annahmen beobachtenden Akteurs über Wissen *und* Handlungspotential des Akteurs der den Plan ausführen soll. Die Beurteilungsprädikate werden deshalb durch einen Index gekennzeichnet, der sich aus der Bezeichnung des Wissenskontextes des beobachtenden Akteurs – hier also SB – plus der Kennung für den zu beurteilenden Akteur der einen Plan ausführen soll – hier also B – zusammensetzt. Bspw. bezeichnet $\mathit{verläßlich}_{\mathrm{SBU}}(pc)$ die Annahme des Systems, daß der Benutzer verläßliches Wissen über die Ausführungsmöglichkeiten des Plan-Concepts *pc* hat (s.u.).

Die folgenden Definitionen der Beurteilungskriterien setzen voraus, daß die Dekompositionshierarchie, die sich in einer Partition P durch die Relation

$$\mathit{Part\text{-}Step}_{\mathrm{P}} = \{(p,q) | p \in \mathrm{P} \wedge q \in \bigcup_{d \in \mathit{dekomp}_P(p)} \mathit{plan\text{-}concepts}(d)\}$$

ergibt, zyklenfrei ist; d.h. ein Plan-Concept darf nicht wieder in einer seiner Ausführungsmöglichkeiten vorkommen. Dies muß auch dann gewährleistet sein, wenn ein Plan-Concept in mehreren Schritten dekomponiert wird.

4.2.2.1 Korrekte und falsche Annahmen über Ausführungsmöglichkeiten von Plan-Concepts

Wie bereits erwähnt, ist das Wissen des Systems die Referenz für die Entscheidung über die Korrektheit der Annahmen des Benutzers. Für die Modellierung von Ausführungs- bzw. Realisierungsmöglichkeiten von Plan-Concepts wurden in Abschnitt 4.2.1 *Dekomposition eines Plan-Concepts* Dekompositionen als partiell geordnete Menge von Planschritten eingeführt. Jede der daraus ableitbaren Linearisierungen[17] beschreibt eine alternative Ausführungsmöglichkeit für das Plan-Concept. Eine Beurteilung, ob eine Dekomposition eines Plan-Concepts im Modell des Benutzerwissens korrekt ist, muß also alle Linearisierungen berücksichtigen. Wir definieren daher, daß das System genau dann annimmt, daß eine Dekomposition d eines Plan-Concepts pc im Modell des Benutzerwissens *für den Benutzer korrekte Ausführungsmöglichkeiten des Plan-Concepts* beschreibt, wenn jede Linearisierung von d auch nach Ansicht des Systems eine zulässige Ausführungsmöglichkeit von pc darstellt. Diese Annahme sei das formal durch das Beurteilungsprädikat $\textit{korrekte-dekomp}_{\mathrm{SBU}}(pc, d)$ definiert.

Definition 4.1 ***Korrektheit einer Dekomposition eines Plan-Concepts***

$$\begin{aligned}
&\textit{korrekte-dekomp}_{\mathrm{SBU}}(pc, d) :\Leftrightarrow d \in \textit{dekomp}_{\mathrm{SBUB}}(pc) \wedge \\
&\quad (\forall s_1, \ldots, s_n, pc_1, \ldots, pc_n \;\; (s_1 : pc_1, \ldots, s_n : pc_n) \in \textit{lin}(d) \Rightarrow \\
&\qquad\qquad \exists d', s'_1, \ldots, s'_n \;\; d' \in \textit{dekomp}_{\mathrm{SB}}(pc) \wedge (s'_1 : pc_1, \ldots, s'_n : pc_n) \in \textit{lin}(d')) \wedge \\
&\quad (\forall pc', d'' \;\; (pc' \in \textit{plan-concepts}(d) \wedge d'' \in \textit{dekomp}_{\mathrm{SBUB}}(pc')) \Rightarrow \textit{korrekte-dekomp}_{\mathrm{SBU}}(pc', d''))
\end{aligned}$$

Dabei sei $\textit{lin}(d)$ eine Funktion, die die Menge aller Folgen von $s_i{:}pc_i \in \textit{steps}(d)$ ermittelt, die gemäß $\textit{before}(d)$ zulässig ist.

Die Definition berücksichtigt, daß die Bezeichnungen der Plan*schritte* ohne Bedeutung sind. Für den Vergleich von Plänen ist vielmehr relevant, von welchen Plan-Concepts die Planschritte abgeleitet sind. In Kap. 3.3.2 *Repräsentation eines Benutzermodells mit Handlungspotential und Wissen im* AsTRa-*Formalismus* wurde bereits darauf eingegangen, daß ein Plan-Concept in verschiedenen Partitionen des Benutzermodells durch seinen Bezeichner identifiziert wird. Die beschreibungslogischen Definitionen können sich in verschiedenen Partitionen aber durchaus unterscheiden, um bspw. verschiedene Auffassungen von System und Benutzer von einem Plan-Concept auszudrücken. Für die Beurteilung, ob eine Dekomposition für den Benutzer korrekte Ausführungsmöglichkeiten eines Plan-Concepts beschreibt, muß deshalb nicht nur festgestellt werden, ob das System Dekompositionen des Plan-Concepts kennt, deren Planschritte mit denen der zu beurteilenden Dekomposition insofern übereinstimmen, daß sie jeweils Exemplare von gleich bezeichneten Plan-Concepts sind. Es muß auch geprüft werden, ob die Auffassung des Benutzers über deren Ausführungsmöglichkeiten – soweit vorhanden – korrekt sind, da eine korrekte Ausführung der Schritte einer Dekomposition Voraussetzung für die korrekte Ausführung der Dekomposition selbst ist.

Da es in diesem Kapitel im wesentlichen um die Ausführungsmöglichkeiten geht, die der Benutzer nach Ansicht des Systems hat, sei die abkürzende Sprechweise vereinbart, daß eine Dekomposition d eines Plan-Concepts pc *korrekt* bzw. *falsch* ist, wenn $\textit{korrekte-dekomp}_{\mathrm{SBU}}(pc,d)$ wahr bzw. falsch ist. Entsprechendes gilt für die Ausführungsmöglichkeiten, die durch die Dekomposition repräsentiert werden.

Nach der Definition ist eine Dekomposition eines Plan-Concepts nur dann korrekt, wenn sie selbst keine falsche Ausführungsmöglichkeiten impliziert und der Benutzer auch keine falschen Ausführungsmöglichkeiten für die darin vorkommenden Plan-Concepts kennt. Die korrekte Ausführung selbst kann zwar auch in einem solchen Fall nicht garantiert werden, weil dies außerhalb der Kontrolle des Systems liegt. Allerdings kann eine falsche Ausführung unter der Annahme ausgeschlossen

[17] vgl. Kap. 2.1.1 *Least Commitment Planning*

werden, daß der Benutzer nur anhand solcher Ausführungsmöglichkeiten handelt die er kennt und daß das Benutzermodell diese Kenntnis ausreichend genau modelliert.

Durch falsche Kenntnisse des Benutzers über die Ausführungsmöglichkeiten von Plan-Concepts können – vergleichbar zu Tutor- oder aktiven Hilfesystemen[18] – *typische Fehler* modelliert werden, die bei bestimmten Benutzergruppen häufig vorkommen. Abgesehen davon, daß diese Hilfesysteme auf eine benutzerspezifische Modellierung verzichten, besteht ein wichtiger Unterschied darin, daß die Klassifizierung als falsche Ausführungsmöglichkeit dort explizit im Sachbereichsmodell repräsentiert und nicht veränderlich ist. Die hier vorgeschlagene Modellierung sieht eine solche explizit repräsentierte Differenzierung zwischen korrekten Ausführungsmöglichkeiten und typischen Fehlern nicht vor. Annahmen über falsche Benutzerkenntnisse über die Ausführungsmöglichkeiten eines Plan-Concepts können damit auf die gleiche Weise ins Benutzermodell gelangen, wie anderes Wissen auch. Insbesondere durch die Zuordnung von Benutzern zu bestimmten Benutzergruppen, deren typischen Kenntnisse durch Stereotypes (vgl. Kap. 3.3.3 *Stereotypes*) repräsentiert werden, sowie durch Schlußfolgerungen aus Beobachtungen des jeweiligen Benutzers, wie dies bspw. von Pollack [Pollack, 1990] vorgeschlagen wurde. Die Beurteilung, ob eine Ausführungsmöglichkeit eines Plan-Concepts korrekt oder falsch ist, erfolgt dann durch Vergleich mit dem Wissen des Systems. Wichtig ist dabei, daß nicht nur die Dekompositionen eines Plan-Concepts im Benutzermodell beurteilt werden, sondern auch die Kenntnisse des Benutzers über die Ausführungsmöglichkeiten der einzelnen Komponenten der Dekompositionen. Es ist dabei durchaus möglich, daß eine Ausführungsmöglichkeit d eines Plan-Concepts, die für viele Benutzer korrekt ist, für einen speziellen Benutzer falsch ist, weil er (u.a.) falsche Vorstellungen über die Ausführungsmöglichkeiten eines Plan-Concepts hat, das in d vorkommt. Da eine solche Vorstellung – bspw. durch eine Erklärung des Systems – korrigiert werden kann, ist die Beurteilung der Korrektheit von Ausführungsmöglichkeiten im Laufe einer System-Benutzer Interaktion veränderlich.

Die Beurteilung der Korrektheit einer Dekomposition eines Plan-Concepts durch das System hat Ähnlichkeiten mit der Erkennung eines Plans durch das System. Auf die Unterschiede wird noch ausführlich in Abschnitt 4.6.3 *Beurteilung von Ausführungsmöglichkeiten im Vergleich zur Planerkennung* eingegangen.

Es sei angemerkt, daß zur Beurteilung einer Dekomposition eines Plan-Concepts ein Vergleich mit einer einzelnen Dekomposition des Systemwissens nicht ausreicht, weil sich eine Menge von Ausführungsmöglichkeiten auch dann nicht immer durch eine einzelne Dekomposition beschreiben läßt, wenn die gleichen Planschritte verwendet werden. Als Beispiel dafür sei von den Planschritten a, b, c ausgegangen, die jeweils Exemplare von verschiedenen Plan-Concepts sind und für die der Benutzer keine (und damit insbesondere keine *falschen*) Dekompositionen kennt. Sowohl das System als auch der Benutzer nehmen an, daß sich ein Plan-Concept *pc* durch die Ausführung dieser drei Planschritte realisieren läßt. Geht das System davon aus, daß die Schritte in beliebiger Reihenfolge außer der Folge b, c, a ausgeführt werden können, läßt sich dies nicht mit einer, aber bspw. mit den beiden Dekomposition d_1 = ({a,b,c}, {a<c}) und d_2 = ({a,b,c}, {c<b}) ausdrücken (wobei hier auf die Angabe der Plan-Concepts verzichtet wird). Nimmt der Benutzer an, daß die Planschritte in der Reihenfolge a, b, c sowie a, c, b oder c, a, b ausgeführt werden können, läßt sich dies durch die Dekomposition d_3 = ({a,b,c}, {a<b}) ausdrücken. Nach der obigen Definition wird d_3 vom System als korrekte Dekomposition von *pc* beurteilt. Ein einzelner Vergleich von d_3 mit d_1 und d_2 würde aber jeweils ergeben, daß d_3 Ausführungsmöglichkeiten beschreibt, die *nicht* in d_1 bzw. d_2 enthalten sind; d.h. der Benutzer würde demnach Ausführungsmöglichkeiten kennen, die das System nicht kennt. Erst die Vereinigung der Ausführungsmöglichkeiten von d_1 und d_2 zeigt, daß dem nicht so ist, sondern daß der Benutzer nur solche Ausführungsmöglichkeiten kennt, die auch das System kennt.

[18] vgl. Kap. 2.5.1 *Erkennung von falschen oder suboptimalen Plänen*

4.2.2.2 Nutzbares Planungswissen

Eine Erklärung, wie ein Plan-Concept ausgeführt werden kann, macht für einen Benutzer nur dann Sinn, wenn er anhand dieses Vorschlags handeln kann. Für eine Dekomposition des Plan-Concepts die als Basis für die Erklärung dienen soll heißt dies, daß der Benutzer jeden Schritt der Dekomposition zumindest im Prinzip ausführen können muß. Für nicht-atomare Schritte muß *das System* wiederum eine Erklärung geben können, für die das gleiche Qualitätskriterium gilt. Es reicht nicht, wenn *der Benutzer* weiß wie er die entsprechenden Plan-Concepts ausführen kann, denn dieses Wissen könnte falsch sein. Die *korrekten* Ausführungsmöglichkeiten, die der Benutzer kennt, sind per Definition auch dem System bekannt, wie im vorangegangen Abschnitt gezeigt wurde.

Eine Dekomposition, die in diesem Sinne die Basis für eine sinnvolle Erklärung der Ausführungsmöglichkeiten eines Plan-Concepts bilden kann, wird als *nutzbar für den Benutzer* (bzw. kurz *nutzbar*) bezeichnet und durch folgendes Beurteilungsprädikat beschrieben:

Definition 4.2 ***Nutzbarkeit einer Dekomposition für den Benutzer***

$$\begin{aligned} \mathit{nutzbar}_{\mathrm{SBU}}(d) \quad :\Leftrightarrow \quad & \forall pc \;\; pc \in \mathit{plan\text{-}concepts}(d) \Rightarrow (pc \in \mathrm{SBUC} \wedge \\ & (\mathit{atomar}_{\mathrm{SBUB}}(pc) \vee \exists d' \;\; d' \in \mathit{dekomp}_{\mathrm{SB}}(pc) \wedge \mathit{nutzbar}_{\mathrm{SBU}}(d'))) \end{aligned}$$

Der Präsentationsprozeß nutzt dieses Kriterium bei der Auswahl einer Dekomposition für die Erklärung eines Plan-Concepts. Für das bereits angesprochene Beispiel des Herunterfahrens eines Windows-Computers[19] gibt es zwei Standardmöglichkeiten:

a) *Betätigen der "Start"-Schaltfläche*, dann *Auswahl von "Beenden"*, dann *Auswahl von "Herunterfahren"*, dann *Betätigen der "OK"-Schaltfläche*

b) *Betätigen der Dreifach-Tastenkombination <Strg>, <Alt>, <Entf>*, dann *Betätigen der Tastenkombination <Alt>, <H>*

Gehen wir davon aus, daß das System beide Möglichkeiten kennt und dies jeweils durch eine Dekomposition d_a bzw. d_b in der Partition SB modelliert ist, so ist d_b nach dem ersten Anschein die bessere, weil kürzere Alternative. Kann der Benutzer – bspw. wegen einer Behinderung – nicht drei Tasten gleichzeitig betätigen, ist $\mathit{nutzbar}_{\mathrm{SBU}}(d_b)$ falsch, weil das *Betätigen der Dreifach-Tastenkombination* nicht im Handlungspotential des Benutzers enthalten ist. Der Präsentationsprozeß muß diese Alternative damit als mögliche Erklärung für das Herunterfahren des Computers verwerfen. Gehen wir außerdem davon aus, daß das System kein Wissen über das Betätigen von Schaltflächen modelliert hat, so ist die Möglichkeit a) nur dann nutzbar, wenn der Benutzer die entsprechenden Plan-Concepts *atomar* kennt. Ist dem System keine nutzbare Dekomposition für ein nicht-atomares Plan-Concept bekannt, ist das Plan-Concept insgesamt für den Benutzer nicht nutzbar. Entsprechend kann die Definition des Prädikats *nutzbar* in naheliegender Weise für Plan-Concepts erweitert werden:

Definition 4.3 ***Nutzbarkeit eines Plan-Concepts für den Benutzer***

$$\mathit{nutzbar}_{\mathrm{SBU}}(pc) :\Leftrightarrow \mathit{atomar}_{\mathrm{SBUB}}(pc) \vee \exists d \;\; d \in \mathit{dekomp}_{\mathrm{SB}}(pc) \wedge \mathit{nutzbar}_{\mathrm{SBU}}(d)$$

Für die Abschätzung, ob der Benutzer mit einem Plan seine Ziele erreichen kann, hat die Nutzbarkeit der verwendeten Plan-Concepts damit die gleiche Bedeutung wie die Frage, ob die Plan-Concepts zum Handlungspotential des Benutzers gehören. Der erweiterte Plangenerierungsprozeß, der in Kap. 4.5 *Optimierte Plangenerierung* vorgestellt wird, verwendet deshalb nur solche Plan-Concepts, die im Handlungspotential des Benutzer enthalten *und* nutzbar sind. Für Pläne, die nur aufgrund des

[19] für Computer mit dem Betriebssystem Microsoft-Windows® in einer deutschsprachigen Version ab '95.

Handlungspotentials erzeugt wurden, kann der Präsentationsprozeß scheitern, wenn dem Benutzer für einzelne Plan-Concepts keine ausreichende Erklärung gegeben werden kann.

Dennoch ist es sinnvoll die Nutzbarkeit von Plan-Concepts für einen Benutzer und dessen Handlungspotential im Benutzermodell strikt auseinander zu halten. Der Grund dafür liegt nicht nur darin, die realen Sachverhalte möglichst adäquat zu modellieren, sondern vor allem, daß damit angemessen auf mögliche Veränderungen des Benutzermodells reagiert werden kann. Nehmen wir als Beispiel dazu an, daß der Benutzer einer Benutzergruppe angehört, der zwar erlaubt ist den Computer herunterzufahren, der aber der Zugriff zur Windows Task-Leiste – und damit das Betätigen der "Start"-Schaltfläche – verwehrt ist. Gehen wir wieder von den oben genannten Möglichkeiten zum Herunterfahren eines Windows-Computers aus, ist von den Möglichkeiten die das System kennt nur b) nutzbar. Für einen Benutzer, der nicht drei Tasten gleichzeitig betätigen kann, ist dann das Plan-Concept *Herunterfahren eines Computers* nur dann nutzbar, wenn er davon atomare Kenntnis hat. Das ist unabhängig von seinem Handlungspotential, insbesondere von seinem Recht den Computer herunterzufahren. Stellt sich im Verlauf einer Interaktion mit dem Benutzer heraus, daß er entgegen einer früheren Annahme doch atomare Kenntnis über das Herunterfahren hat, ändert sich die Nutzbarkeit des Plan-Concepts für den Benutzer. Ihm kann dann ein Plan präsentiert werden, der das Herunterfahren des Computers enthält. Für ihn (sinnvolle) Erklärungen über Ausführungsmöglichkeiten dieses Plan-Concepts sind in dem Fall aber weder möglich noch notwendig.

Gehört der Benutzer umgekehrt einer Benutzergruppe an, die nicht das Recht hat den Computer herunterzufahren, kann er das Plan-Concept nicht ausführen, auch wenn er bspw. Möglichkeit b) kennt und ausführen kann oder das Plan-Concepts atomar kennt.

4.2.2.3 Verläßliches Wissen über die Ausführungsmöglichkeiten von Plan-Concepts

Die Nutzbarkeit einer Dekomposition eines Plan-Concepts für einen Benutzer basiert darauf, daß das System ausreichendes Wissen hat, um dem Benutzer die Plan-Concepts der Dekomposition in einer Weise erklären zu können, daß dieser danach handeln kann. Solche Erklärungen sind natürlich nicht notwendig, wenn der Benutzer selbst über dieses Wissen verfügt. Kennt der Benutzer mehrere Ausführungsalternativen, kann er eine beliebige davon als Rezept für sein Handeln auswählen. Wählt er eine Ausführungsalternative, die das System für falsch hält, wird er seine Ziele wahrscheinlich nicht erreichen können. Die Aufgabe des Präsentationsprozesses ist es aber gerade den Plan so zu beschreiben und zu erklären, *daß* der Benutzer seine Ziele damit erreicht. Der Präsentationsprozeß kann sich deshalb nur dann auf das Wissen des Benutzers verlassen, wenn dieses (in Bezug auf einen zu erklärenden Sachverhalt) nicht nur korrekt, sondern auch frei von falschen Ausführungsalternativen ist. Wie bei der Definition einer korrekten Dekomposition kann so garantiert werden, daß der Benutzer den Plan korrekt ausführt, solange er nur anhand der Ausführungsmöglichkeiten handelt, die er nach Ansicht des Systems kennt. Die Entscheidung, ob das Wissen des Benutzers in diesem Sinne ausreichend ist und damit auf weitere Erklärungen des Systems verzichtet werden kann, kann anhand der Beurteilungsprädikate *verläßlich* für ein Plan-Concept und *selbst-nutzbar* für eine Dekomposition eines Plan-Concepts getroffen werden, die wie folgt definiert sind:

Definition 4.4 ***Verläßliches Benutzerwissen über Ausführungsmöglichkeiten eines Plan-Concepts***

$$\begin{aligned} \textit{verläßlich}_{\text{SBU}}(pc) \quad :\Leftrightarrow \quad & \exists d \;\; d \in \textit{dekomp}_{\text{SBUB}}(pc) \wedge \textit{selbst-nutzbar}_{\text{SBU}}(d) \wedge \\ & \forall d \;\; d \in \textit{dekomp}_{\text{SBUB}}(pc) \Rightarrow \textit{korrekte-dekomp}_{\text{SBU}}(pc, d) \end{aligned}$$

Definition 4.5 ***Selbständige Nutzbarkeit einer Dekomposition für den Benutzer***

$$\begin{aligned} \textit{selbst-nutzbar}_{\text{SBU}}(d) \quad :\Leftrightarrow \quad & \forall pc \;\; pc \in \textit{plan-concepts}(d) \Rightarrow \\ & (pc \in \text{SBUC} \wedge (\textit{atomar}_{\text{SBUB}}(pc) \vee \textit{verläßlich}_{\text{SBU}}(pc))) \end{aligned}$$

Das System schreibt einem Benutzer demnach genau dann *verläßliches Wissen* über die Ausführungsmöglichkeiten eines Plan-Concepts zu, wenn dieser ausschließlich korrekte Ausführungsmöglichkeiten kennt und mindestens eine davon selbständig ausführen kann; d.h. keine fremde Hilfe oder Erklärungen benötigt. Für die Präsentation eines Plan-Concepts ist *verläßliches* Wissen über die Ausführungsmöglichkeiten eines Plan-Concepts damit gleichwertig zur *atomaren* Kenntnis eines Plan-Concepts. In beiden Fällen sind keine zusätzlichen Erklärungen erforderlich. Der wesentliche Unterschied besteht darin, daß dem Benutzer die atomare Kenntnis eines Plan-Concepts (bspw. durch Beobachtung und/oder Zuordnung zu einer Benutzergruppe) zugesprochen werden kann und dann explizit im Benutzermodell repräsentiert wird, während die Beurteilung der Verläßlichkeit aus dem Benutzerwissen über die Ausführungsmöglichkeiten der Plan-Concepts abgeleitet wird und damit nur indirekt im Benutzermodell enthalten ist. Da sich das Benutzerwissen im Verlauf einer Interaktion zwischen System und Benutzer – bspw. durch Erklärungen – ändern kann, gilt dies auch für die Beurteilung über die Verläßlichkeit. Die umfassende Kompetenz, die mit der atomaren Kenntnis verbunden ist, kann ein Benutzer dadurch aber nicht erlangen. Ähnlich wie beim Modell der Handlungsmöglichkeiten eines Benutzers können sich lediglich die *Annahmen über* die atomare Kenntnis eines Benutzers verändern – bspw. wenn das System durch Beobachtung zu einer neuen Einschätzung des Benutzers kommt. Eine Veränderung der atomaren Kenntnis selbst ist dagegen durch (geplantes) Handeln von System oder Benutzer nicht möglich.

4.2.3 Zusätzliche Akteure

Wenngleich in dieser Arbeit nicht davon Gebrauch gemacht wird, lassen sich die Definitionen der Beurteilungsprädikate für die Beurteilung des Benutzerwissens durch das Systems auf weitere Akteurkombinationen übertragen, sofern deren Kenntnisse und ggf. Fähigkeiten im Benutzermodell repräsentiert sind. Beurteilt das System anstelle des Benutzers einen anderen Akteur A, so ist lediglich die Bezeichnung U für den Benutzer durch die Bezeichnung für den Akteur A zu ersetzen.

Beurteilt nicht das System, sondern ein beliebiger anderer Akteur O einen Akteur A, so ist zu beachten, daß die Beurteilungen grundsätzlich auf einen Wissenskontext[20] bezogen sind, der Annahmen über O's Wissen repräsentiert. Da alles im System repräsentierte Wissen letztlich die Ansicht des System darstellt, ist dieser Kontext in den Wissenskontext des Systems, möglicherweise aber auch noch in die Wissenskontexte weiterer Akteure eingebettet.

Bezeichne K einen solchen Wissenskontext, können verallgemeinerte Definitionen für die Beurteilungsprädikate gewonnen werden, indem in den speziellen Definitionen der vorangegangen Abschnitte die Kennung für das Systemwissen (SB) durch K und die Kennung für den Benutzer (U) durch die Kennung des beurteilten Akteurs (im Beispiel: A) substituiert werden. Bspw. lautet dann das Prädikat *korrekte-dekomp* für die Beurteilung, ob eine Dekomposition für einen Akteur korrekte Ausführungsmöglichkeiten eines Plan-Concepts beschreibt:

Definition 4.6 ***Für Akteur A korrekte Ausführungsmöglichkeiten eines Plan-Concepts***

$$
\begin{aligned}
&\textit{korrekte-dekomp}_{\mathrm{KA}}(pc,d) :\Leftrightarrow d \in \textit{dekomp}_{\mathrm{KAB}}(pc) \wedge \\
&\quad (\forall s_1,\ldots,s_n,pc_1,\ldots,pc_n \;\; (s_1 : pc_1,\ldots,s_n : pc_n) \in \textit{lin}(d) \Rightarrow \\
&\qquad\qquad \exists d', s'_1,\ldots,s'_n \;\; d' \in \textit{dekomp}_{\mathrm{K}}(pc) \wedge (s'_1 : pc_1,\ldots,s'_n : pc_n) \in \textit{lin}(d')) \wedge \\
&\quad (\forall pc', d'' \;\; (pc' \in \textit{plan-concepts}(d) \wedge d'' \in \textit{dekomp}_{\mathrm{KAB}}(pc')) \Rightarrow \textit{korrekte-dekomp}_{\mathrm{KA}}(pc', d''))
\end{aligned}
$$

Der Wissenskontext K kann in BGP-MS durch eine Partition repräsentiert werden, die nach üblicher Konvention[21] durch eine Folge von <Akteur>-<Modalität>-Paaren bezeichnet wird, wobei die

[20] im Sinne von [Cohen, 1978], vgl. Kap. 3.3.1 *Akteurmodelle*

[21] vgl. Kap. 3.3.1 *Akteurmodelle*

Modalität grundsätzlich B für *Belief* ist und damit einen Wissens- bzw. Annahmen-Kontext bezeichnet. Da die Beurteilungen letztlich Systemannahmen darstellen, beginnt eine solche Folge grundsätzlich mit SB.

Gehen wir bspw. davon aus, daß das System annimmt, daß ein Akteur Z annimmt, daß Akteur O einen weiteren Akteur A beurteilt, wird der entsprechende Wissenskontext in BGP-MS durch eine Partition repräsentiert, die mit SBZBOB bezeichnet wird. Die dann für die Beurteilung maßgeblichen Partitionen des Benutzermodells sind außer SBZBOB als Bezug für korrektes Wissen, die Partitionen SBZBOBAB und SBZBOBAC, die in diesem Kontext die Annahmen über A's Wissen und Handlungspotential repräsentieren. Die Definitionen der Beurteilungsprädikate für einen speziellen Akteur und einen speziellen Kontext können aus den verallgemeinerten Definitionen gewonnen werden, indem A bzw. K durch die Kennungen des speziellen Akteurs bzw. des speziellen Kontextes substituiert werden. Für das Prädikat *korrekte-dekomp* ergibt sich für das Beispiel:

Definition 4.7 ***Für Akteur A korrekte Ausführungsmöglichkeiten eines Plan-Concepts im Kontext SBZBOB***

$$\begin{aligned}
&\textit{korrekte-dekomp}_{\text{SBZBOBA}}(pc,d) :\Leftrightarrow d \in \textit{dekomp}_{\text{SBZBOBAB}}(pc) \wedge \\
&\quad (\forall s_1,\ldots,s_n,pc_1,\ldots,pc_n \;\; (s_1 : pc_1,\ldots,s_n : pc_n) \in \textit{lin}(d) \Rightarrow \\
&\qquad\qquad \exists d',s'_1,\ldots,s'_n \;\; d' \in \textit{dekomp}_{\text{SBZBOB}}(pc) \wedge (s'_1 : pc_1,\ldots,s'_n : pc_n) \in \textit{lin}(d')) \wedge \\
&\quad (\forall pc',d'' \;\; (pc' \in \textit{plan-concepts}(d) \wedge d'' \in \textit{dekomp}_{\text{SBZBOBAB}}(pc')) \Rightarrow \\
&\qquad\qquad \textit{korrekte-dekomp}_{\text{SBZBOBA}}(pc',d''))
\end{aligned}$$

Das Prädikat *nutzbar* hängt auch vom Handlungspotential des Akteurs ab:

Definition 4.8 ***Nutzbarkeit einer Dekomposition für Akteur A im Kontext SBZBOB***

$$\begin{aligned}
&\textit{nutzbar}_{\text{SBZBOBA}}(d) :\Leftrightarrow \\
&\quad \forall pc \;\; pc \in \textit{plan-concepts}(d) \Rightarrow (pc \in \text{SBZBOBAC} \wedge \\
&\qquad (\textit{atomar}_{\text{SBZBOBAB}}(pc) \vee \exists d' \;\; d' \in \textit{dekomp}_{\text{SBZBOB}}(pc) \wedge \textit{nutzbar}_{\text{SBZBOBA}}(d')))
\end{aligned}$$

Aus den verallgemeinerten Definitionen können auch die Prädikate für die Beurteilungen des Systems durch den Benutzer und des Benutzers durch sich selbst gewonnen werden. In beiden Fällen ist der Beurteilungskontext SBUB. Der zu beurteilende Akteur ist im ersten Fall das System, im zweiten Fall der Benutzer selbst.

4.3 Inhaltsbestimmung und Bewertung einer Planpräsentation

In den vorangegangenen Abschnitten wurde gezeigt, welche Kenntnisse ein Benutzer für eine erfolgreiche Ausführung eines Plans benötigt, den er im Prinzip ausführen kann. Außer den Schritten und der Struktur des Plans waren dies in erster Linie Informationen über die Ausführungsmöglichkeiten der einzelnen Planschritte. Hat der Benutzer selbst atomares oder verläßliches Wissen über Ausführungsmöglichkeiten der Planschritte kann und sollte eine Planpräsentation auf eine explizite Nennung dieses bekannten Wissens verzichten. Für alle Planschritte, für die der Benutzer kein solches Wissen hat, muß die Planpräsentation entsprechende Erklärungen enthalten, die aus dem Systemwissen abgeleitet sind. Einem Plan-Concept können mehrere Dekompositionen zugeordnet sein, die die Ausführungsmöglichkeiten des Plan-Concepts repräsentieren. Da diese Alternativen im allgemeinen nicht zu gleich guten Erklärungen für den Benutzer führen werden, müssen sie bewertet werden, um eine möglichst gute Alternative auswählen zu können. Natürlich können Dekompositionen, die für den Benutzer nicht nutzbar sind, von vorn herein von der Betrachtung ausgeschlossen werden.

Besondere Beachtung erfordern Erklärungen von Ausführungsmöglichkeiten eines Plan-Concepts, wenn der Benutzer selbst bereits Annahmen darüber hat, die aber zumindest teilweise vom System für falsch gehalten werden. Nennt das System dem Benutzer eine korrekte Ausführungsmöglichkeit, so erlangt der Benutzer dadurch *kein* verläßliches Wissen über die Ausführungsmöglichkeiten des Plan-Concepts. Dies gilt selbst dann, wenn – wie in kooperativen Dialogsituationen üblich – davon ausgegangen wird, daß sich der Benutzer die genannte Erklärung zu eigen macht, denn seine falschen Annahmen werden dadurch nicht automatisch korrigiert. Dennoch soll für die Planpräsentation im Rahmen dieser Arbeit davon ausgegangen werden, daß der Benutzer dem – i.a. angeforderten – Rat des Systems folgt und seine mglw. falschen Annahmen damit nicht für die Planausführung zum Tragen kommen. In Abschnitt 4.3.3 *Die Bedeutung falscher Annahmen des Benutzers für die Planpräsentation* wird noch näher auf falsche Benutzerannahmen eingegangen. Bei der Auswahl, welche von mehreren Ausführungsmöglichkeiten dem Benutzer genannt werden soll, werden allerdings die bevorzugt, die einen solchen Konflikt nicht enthalten.

Zusammenfassend kann damit der erforderliche Inhalt einer benutzerspezifischen Präsentation eines Plans wie folgt bestimmt werden:

- für jeden Planschritt eine Beschreibung zur Identifizierung
- die Bedingungen an die Reihenfolge, in der die Planschritte ausgeführt werden müssen
- die Causal Links des Plans
- für jedes Plan-Concept, das im Plan verwendet wurde
  - falls der Benutzer das Plan-Concept nicht kennt:
    eine Beschreibung des Plan-Concepts, sofern diese nicht bereits durch die Causal Links präsentiert wurde
  - falls der Benutzer das Plan-Concept nicht atomar kennt und auch keine verläßlichen Kenntnisse von den Ausführungsmöglichkeiten hat:
    eine Beschreibung einer möglichst guten Ausführungsmöglichkeit, die für den Benutzer nutzbar ist, und falls erforderlich weitere Erklärungen zu dieser Ausführungsmöglichkeit

Eine lineare Folge von Planschritten ist oft leichter zu verstehen, als eine partielle Ordnung der Planschritte. Außerdem muß zumindest für eine textuelle Beschreibung ohnehin eine feste Reihenfolge festgelegt werden, in der die Schritte genannt werden. Im weiteren wird deshalb ein Plan grundsätzlich als lineare Folge beschrieben und damit von einer expliziten Nennung bzw. generell einer Behandlung der Reihenfolgebedingungen abgesehen. Diese lineare Darstellung bezieht sich nur auf die Schritte eines Plans. Die Struktur, die sich durch Dekompositionen von Plan-Concepts ergibt, bleibt davon unberührt.

Da die Ausführungsmöglichkeiten selbst (einfache) Pläne sind, kann deren Beschreibung, einschließlich ggf. erforderlicher weiterer Erklärungen der dafür erforderlichen Subschritte, in der gleichen Weise erfolgen wie der Plan selbst. Der Prozeß der Planpräsentation kann deshalb durch einen rekursiven Algorithmus realisiert werden, wie er im folgenden Abschnitt beschrieben wird. Die maximale Rekursionstiefe ist dabei durch die Dekompositionshierarchie der Plan-Concepts begrenzt. Für die Entscheidung zwischen verschiedenen Ausführungsmöglichkeiten berechnet der Algorithmus für jede Ausführungsmöglichkeit eine Bewertung. Diese beruht auf einer Abschätzung des Aufwandes, den der Benutzer für das Verstehen einer Präsentation der jeweiligen Ausführungsmöglichkeit hat. Da Ausführungsmöglichkeiten lediglich Sonderfälle von Plänen darstellen, kann der Algorithmus durch eine geringfügige Erweiterung für die Bewertung der gesamten Planpräsentationen eingesetzt werden. Die folgende Beschreibung geht von vorn herein von dieser Erweiterung aus. Anschließend wird

die Gesamtbewertung einer Planpräsentation noch ausführlicher behandelt. Danach wird noch auf die Rolle falscher Benutzerannahmen für die Planpräsentation und auf einige Arbeiten aus der Psychologie zum Verstehen von Plänen und Anweisungen eingegangen. Abschließend wird am Beispiel von Benutzerpräferenzen und Erfolgswahrscheinlichkeiten für die Ausführung von Plan-Concepts gezeigt, wie weitere Kriterien für eine benutzerspezifische Gesamtbewertung eines Plans mit berücksichtigt werden können.

4.3.1 Der Algorithmus zur Planpräsentation

Abb. 4.2 zeigt den Algorithmus, der sowohl die Information ermittelt, die für die benutzerspezifische Präsentation eines Plans erforderlich ist, als auch eine Bewertung der Präsentation berechnet. Der Algorithmus ist auf die Funktionen *present-plan*, *present-plan-concept* und *present-recipe-for-plan-concept* aufgeteilt, die jeweils die beiden Werte *presentation* – eine Datenstruktur für die Planpräsentation – und *effort* – einen Zahlenwert für deren Bewertung – als Ergebnis liefern. Auf diese Ergebnisse kann jeweils durch Hilfsfunktionen *presentation* bzw. *effort* zugegriffen werden. Die Funktion *present-plan* bestimmt diese Ergebnisse für einen Plan und damit auch eine Ausführungsmöglichkeit eines Plan-Concepts, die als Dekomposition des Plan-Concepts repräsentiert ist. *present-plan-concept* und *present-recipe-for-plan-concept* bestimmen ihre Werte jeweils für ein Plan-Concept. Formal wird dies durch die Typangabe *:plan* bzw. *:plan-concept* der ersten formalen Parameter der Funktionen notiert.

Die Präsentationsbewertung wird hier zunächst als Kriterium für die Auswahl alternativer Ausführungsmöglichkeiten eines Plan-Concepts genutzt (Zeile r4). Auf die Verwendung der Bewertung für den Gesamtplan wird weiter unten noch eingegangen. Der Gesamtbewertung wird für jede Komponente die zur Beschreibung hinzugefügt wird ein Kostenfaktor hinzuaddiert, mit dem hier zunächst der Aufwand des Benutzers für das Verstehen der Beschreibung abgeschätzt wird. Diese Abschätzung lehnt sich an Arbeiten zum Textverstehen an, auf die in Abschnitt 4.3.4 *Arbeiten zum Verstehen von Plänen und Anweisungen aus der Psychologie* noch näher eingegangen wird und basiert i.w. auf der Informationsmenge, die der Benutzer aufnehmen muß. Diese Informationsmenge ist proportional zur Zahl der informationstragenden Komponenten der Beschreibung, da der Algorithmus die gesamte Information, die der Benutzer für das Ausführen des Plans benötigt, explizit darstellt. Dieses einfache Modell der Aufwandabschätzung für das Verstehen einer Beschreibung sei hier lediglich beispielhaft als eine *Möglichkeit* eingesetzt. Ohne grundsätzliche Änderung des Präsentationsprozesses kann es ersetzt werden, sobald ein kognitiv adäquateres Modell vorliegt.

Als Basiseinheit für den Aufwand wird die Konstante k_{pres} definiert. Sie steht für den Aufwand, der durch das Auftreten eines Planschritts in einer Planpräsentation bei einem Benutzer entsteht. Der absolute Wert der Konstanten ist allerdings nur dann von Belang, wenn ein absoluter Wert für den Gesamtaufwand der Präsentation benötigt wird. Ansonsten spielt k_{pres} lediglich die Rolle einer *Maßeinheit* für den Aufwand.

Der Algorithmus verwendet diese Konstante für die Berücksichtigung des Aufwandes für Causal Links (p1), für einen Planschritt (p5), für eine Ausführungsmöglichkeit eines Plan-Concepts (c8) und bei der Bewertung von falschem Benutzerwissen (c7). Jeder Causal Link des Plans wird dabei mit k_{cl} Einheiten berücksichtigt. Der Wert von k_{cl} wird für die Beispiele auf 2 festgelegt, da durch einen Causal Link mehrere Informationen vermittelt werden (neben der Abhängigkeit zwischen den beiden Planschritten, Information über Effekt bzw. Vorbedingung der Planschritte). Durch Funktion *effort-plan-concept* (Zeile p5) kann der Wert, den ein Planschritt zur Gesamtbewertung beiträgt, in Abhängigkeit vom zugehörigen Plan-Concept beeinflußt werden. Diese Möglichkeit wird aber erst weiter unten genutzt; hier liefert die Funktion konstant 1, womit der Beitrag eines Planschritts mit genau einer Basiseinheit in die Bewertung eingeht.

present-plan (p:plan, cs:set-of-plan-concepts)
(p1) $\mathit{effort} := |\mathit{causal\text{-}links}(p)| * k_{cl} * k_{pres}$
(p2) $\mathit{presentation} := \mathit{linearize}(p)$
(p3) **for each** $s \in \mathit{steps}(p)$
(p4) ergänze *presentation* um $\mathit{name}(\mathit{plan\text{-}concept}(s))$
(p5) $\mathit{effort} := \mathit{effort} + \mathit{effort\text{-}plan\text{-}concept}(\mathit{plan\text{-}concept}(s)) * k_{pres}$
(p6) **for each** $pc \in \mathit{plan\text{-}concepts}(p) \backslash cs$
(p7) $\mathit{effort} := \mathit{effort} + \mathit{effort}(\mathit{present\text{-}plan\text{-}concept}(pc, cs))$
(p8) ergänze *presentation* um *presentation* $(\mathit{present\text{-}plan\text{-}concept}\ (pc, cs))$
(p9) $cs := cs \cup \{pc\} \cup \mathit{plan\text{-}concepts}(\mathit{presentation}(\mathit{present\text{-}plan\text{-}concept}(pc, cs)))$

present-plan-concept (pc:plan-concept, cs:set-of-plan-concepts)
(c1) $\mathit{presentation} := \{\}$
(c2) $\mathit{effort} := 0$
(c3) **if** $pc \notin$ SBUB **then**
(c4) $\mathit{presentation} := \mathit{presentation}(\mathit{descript}(pc))$
(c5) $\mathit{effort} := \mathit{effort}(\mathit{descript}(pc))$
(c6) **if not** $(\mathit{atomar}_{\mathrm{SBUB}}(pc)$ **or** $\mathit{verläßlich}_{\mathrm{SBU}}(pc))$ **then**
(c7) **if** $\mathit{dekomp}_{\mathrm{SBUB}}(pc) \neq \{\}$ **then** $\mathit{effort} := \mathit{effort} + k_{corr} * k_{pres}$ **fi**
(c8) $\mathit{effort} := \mathit{effort} + k_{pres} + k_{hier} * \mathit{effort}(\mathit{present\text{-}recipe\text{-}for\text{-}plan\text{-}concept}(pc, cs))$
(c9) ergänze *presentation* um $\mathit{presentation}(\mathit{present\text{-}recipe\text{-}for\text{-}plan\text{-}concept}(pc, cs))$

present-recipe-for-plan-concept (pc:plan-concept, cs:set-of-plan-concepts)
(r1) $ec := \{d \in \mathit{dekomp}_{\mathrm{SB}}(pc) \mid \mathit{nutzbar}_{\mathrm{SBU}}(d)\}$
(r2) **if** $ec = \{\}$ **then** $\mathit{effort} := \infty$ (*Präsentation scheitert*)
(r3) **else**
(r4) **for** $d \in ec \wedge \mathit{effort}(\mathit{present\text{-}plan}(d, cs)) = \min_{(d' \in ec)} \mathit{effort}(\mathit{present\text{-}plan}(d', cs))$
(r5) $\mathit{presentation} := \mathit{presentation}(\mathit{present\text{-}plan}(d, cs))$
(r6) $\mathit{effort} := \mathit{effort}(\mathit{present\text{-}plan}(d, cs))$

Abbildung 4.2: Algorithmus zur Erzeugung und Bewertung einer Planpräsentation

Die Bewertung einer Ausführungsmöglichkeit eines Plan-Concepts (c8) berücksichtigt zwei Effekte. Einerseits ist alleine das Einführen eines neuen Teilplans mit Aufwand verbunden – bspw. für das Umorganisieren des Arbeitsgedächtnisses des Benutzer. Andererseits bewirkt gerade diese Umorganisation, daß der Benutzer die neue Information besser verarbeiten kann, als dies bei einer unstrukturierten stetigen Anhäufung der Information der Fall wäre. Aus diesem Grund wird der berechnete Aufwand für eine Ausführungsmöglichkeit mit dem Faktor k_{hier}, multipliziert, der normalerweise kleiner als 1 ist. In Abschnitt 4.3.4 *Arbeiten zum Verstehen von Plänen und Anweisungen aus der Psychologie* wird darauf noch genauer eingegangen und auch ein Wert für die Konstante vorgeschlagen.

Der Start des Algorithmus erfolgt durch Aufruf der Funktion *present-plan* mit dem zu präsentierenden Plan als ersten und der leeren Menge als zweiten Parameter. Die Funktion initialisiert zunächst die Variablen *effort* und *presentation*, die bei Beendigung der Funktionsausführung die Ergebniswerte der Funktion werden. Der initiale Wert von *effort* wird auf k_{cl} Aufwand-Basiseinheiten je relevantem[22] Causal Link im Plan festgelegt (p1). Die Menge der Causal Links des Plans wird dabei durch

[22] vgl. Abschnitt 4.1 *Planstruktur*

die Funktion *causal-links* bestimmt. Wird sie für eine Dekomposition aufgerufen, ist das Ergebnis die leere Menge und *effort* wird dementsprechend mit 0 initialisiert. Die Variable *presentation* wird mit einer Linearisierung des Plans initialisiert; d.h. mit der Datenstruktur des zu präsentierenden Plans die ggf. um Reihenfolgebedingungen ergänzt wird, sodaß sich eine lineare Ordnung der Planschritte ergibt (p2).

Die Schritte des Plans werden dem Benutzer durch Nennung der Namen der zugehörigen Plan-Concepts mitgeteilt. Dazu iteriert die Funktion über alle Schritte des Plans (p3) und fügt jedem Schritt der Planpräsentation den entsprechenden Namen hinzu (p4). Die Hilfsfunktion *plan-concept(s)* liefert das zum Schritt *s* gehörende Plan-Concept. In der nächsten Zeile erhöht die Funktion die Präsentationsbewertung für jeden Schritt des Plans um den Aufwand, der mit dem zugehörigen Plan-Concept verbunden ist. Der Aufwand wird wie oben beschrieben durch die Funktion *effort-plan-concept* ermittelt. Er wird auch dann für jeden Schritt berücksichtigt, wenn mehrere Planschritte Exemplare des gleichen Plan-Concepts sind, wie bspw. die beiden *launch*-Schritte im Plan aus Abb. 4.1. Während dies für die Verarbeitung der einzelnen Planschritte gerechtfertigt ist, sind weitergehende Erklärungen zu Planschritten, die Exemplare des gleichen Plan-Concepts sind, nur einmal notwendig. Solche Erklärungen generiert die Funktion deshalb durch eine Iteration über die Plan-Concepts des Plans (p6). Davon ausgenommen sind die Plan-Concepts, die in der Menge *cs* enthalten sind, die der Funktion als zweiter Parameter übergeben wurde. Die Menge enthält alle Plan-Concepts, die bereits bearbeitet wurden; d.h. für die die Präsentation bereits weitergehende Erklärungen enthält, oder für die festgestellt wurde, daß solche Erklärungen nicht notwendig sind.

Die Präsentation eines einzelnen Plan-Concepts wird dann durch Aufruf der Funktion *present-plan-concept* realisiert. Der Funktion werden außer dem zu präsentierenden Plan-Concept, die Menge der bereits bearbeiteten Plan-Concepts als Parameter mitgegeben. Die beiden Ergebnisse des Funktionsaufrufs werden der Präsentationsbewertung (p7) bzw. der Planpräsentation (p8) hinzugefügt. Außerdem wird die Menge der bereits bearbeiteten Plan-Concepts aktualisiert, indem ihr das Plan-Concept das erklärt werden sollte, sowie weitere dafür notwendige Plan-Concepts hinzugefügt werden (p9).

Funktion *present-plan-concept* initialisiert zunächst *presentation* mit einer leeren Beschreibung (c1) und *effort* mit 0 (c2). Kennt der Benutzer ein Plan-Concept überhaupt nicht (c3), kann ihm eine Beschreibung des Plan-Concepts gegeben werden. Dabei geht es hier noch nicht um eine Beschreibung *wie* das Plan-Concept ausgeführt wird, sondern lediglich darum dem Benutzer eine Vorstellung des Plan-Concepts zu geben, sofern dessen Namen dafür nicht ausreicht. Bspw. können ihm die wichtigsten Effektkomponenten genannt werden; d.h. der Zweck weswegen das Plan-Concept *normalerweise* ausgeführt wird. Eine solche Beschreibung selbst ist vergleichbar zu allgemeinen Objektbeschreibungen, die sinnvollerweise wiederum benutzerspezifisch erfolgen sollten. Da allgemeine Objektbeschreibungen aber nicht Thema dieser Arbeit sind, wird hier nicht näher auf mögliche Realisierungen eingegangen. Es wird stattdessen davon ausgegangen, daß die Funktion *descript* sowohl eine solche Beschreibung als auch einen Wert für den Aufwand ermittelt und daß der Zugriff auf die Ergebnisse wiederum durch Zugriffsfunktionen *presentation* bzw. *effort* möglich ist. Die Ergebnisse werden in (c4) der Planpräsentation bzw. in (c5) der Präsentationsbewertung hinzugefügt.

Für eine Erklärung *wie* der Benutzer ein Plan-Concept ausführen kann, wird die Funktion *present-recipe-for-plan-concept* eingesetzt. Sie wird für jedes Plan-Concept aufgerufen, das der Benutzer nicht atomar kennt und für das er auch kein verläßliches Wissen über die Ausführungsmöglichkeiten besitzt. Für die Präsentation von Ausführungsmöglichkeiten können wiederum Erklärungen von Plan-Concepts erforderlich sein. Um hierbei mehrfache Beschreibungen des gleichen Plan-Concepts verhindern zu können, wird der Funktion die Menge der bereits bearbeiteten Plan-Concepts als Parameter mitgegeben. Die Ergebnisse der Funktion werden wieder der Präsentationsbewertung (c8) bzw. der Planpräsentation (c9) hinzugefügt. Hat der Benutzer Wissen über die Ausführungsmöglichkeiten

eines Plan-Concepts, das aber weder atomar noch verläßlich – also unvollständig oder fehlerhaft – ist, so kann dieses Wissen zu Fehlern bei der Planausführung führen (vgl. Abschnitt 4.3.3 *Die Bedeutung falscher Annahmen des Benutzers für die Planpräsentation*). Um die Verwendung von Ausführungsmöglichkeiten die solche Plan-Concepts enthalten möglichst zu vermeiden, wird der Aufwand für ein solches Plan-Concept in (c7) erhöht, sodaß sich die Chancen für die Auswahl der Ausführungsmöglichkeit verschlechtern (vgl. wieder Abschnitt 4.3.3).

Funktion *present-recipe-for-plan-concept* ermittelt die Information und eine zugehörige Bewertung für die Präsentation einer für den Benutzer möglichst guten Ausführungsmöglichkeit eines Plan-Concepts. Es wird vorausgesetzt, daß die Funktion nicht für Plan-Concepts aufgerufen wird, die der Benutzer atomar kennt oder für die er verläßliches Wissen über die Ausführungsmöglichkeiten hat, weil dafür keine weiteren Erklärungen erforderlich sind. Die Funktion bestimmt zunächst die Menge der Dekompositionen des Plan-Concepts, die für den Benutzer nutzbar sind, aus dem Systemwissen(r1). Im ungünstigsten Fall kennt das System keine solche Ausführungsmöglichkeit, womit die Präsentation insgesamt scheitern würde (r2). Im Abschnitt 4.2.2.2 *Nutzbares Planungswissen* wurde bereits darauf eingegangen. Durch die verbesserte Plangenierung in Abschnitt 4.5 *Optimierte Plangenerierung* kann dieser Fall vermieden werden.

Kennt das System mindestens eine Dekomposition des Plan-Concepts die für den Benutzer nutzbar ist, wird diejenige mit der geringsten Bewertung ausgewählt (r4). Für die Präsentation und die Bewertung der in Frage kommenden Dekompositionen wird wiederum die Funktion *present-plan* eingesetzt. Durch den zweiten Parameter der (indirekt) rekursiven Funktionsaufrufe erhält die Funktion die notwendige Information, um (im Schritt (p6)) mehrfache Erklärungen von Plan-Concepts zu vermeiden.

4.3.2 Gesamtbewertung einer Planpräsentation

Die Bewertung einer Planpräsentation kann natürlich nicht nur zur Entscheidung zwischen Ausführungsalternativen eines Plan-Concepts eingesetzt werden, sondern auch zur Beurteilung des gesamten Plans. In Kap. 3.1 *Handlungspotential und Wissen von Benutzern* wurde darauf hingewiesen, daß die Vorgehensweise in dieser Arbeit – erst Generierung, dann Präsentation eines Plans – keine Einbahnstraße darstellt, sondern daß ein Plan, der langwierige Erklärungen erfordert, durch den Präsentationsprozeß zurückgewiesen werden kann und dadurch die Generierung eines alternativen Plans notwendig wird. Als Kriterium für “langwierig” eignet sich die Bewertung der Planpräsentation auf Basis der Aufwandschätzung für das Verstehen der Präsentation durch den Benutzer. Übersteigt dieser Aufwand für eine Planpräsentation einen Grenzwert der Zumutbarkeit, kann der Plan zurückgewiesen werden. Allerdings gibt es derzeit keine praktikablen Verfahren, wie ein solcher Grenzwert sinnvoll bestimmt werden könnte. Ein Ansatzpunkt ist die Erkenntnis der psychologischen Forschung zum Textverstehen, nachdem ein Mensch nur eine bestimmte Anzahl von Propositionen in seinem Arbeitsgedächtnis speichern kann (vgl. bspw. [Miller, 1956], [Kintsch und van Dijk, 1978]). Solche Propositionen wären hier bspw. die Causal Links oder eine Aussage, daß ein Planschritt B nach einem Planschritt A auszuführen ist. Allerdings läßt sich auch damit nicht einfach ein Grenzwert bestimmen. Zum einen ist kein Verfahren bekannt, das aus einem Text eindeutig genau diejenigen Propositionen bestimmt, die ein Hörer oder Leser in seinem Arbeitsgedächtnis speichert[23]. Für Planbeschreibungen könnte es bspw. je Causal Link die dreistellige Relation zwischen den beiden beteiligten Planschritten sein. Direkt dazu gehören aber auch die Aussagen über den Effekt des einen und die Vorbedingung des anderen Planschritts. Weitere Propositionen können leicht inferiert werden; bspw. daß das System Effekt und Vorbedingung der Planschritte kennt. Zweitens ist die nutzbare Kapazität des Arbeitsgedächtnisses von Menschen nicht nur individuell verschieden, sondern auch situationsabhängig. Insbesondere

[23] vgl. bspw. [Grabowski, 1991, Kap. 4.1]

spielt dabei die kognitive Belastung eine Rolle. Allerdings zeigten Berthold und Jameson in [Berthold und Jameson, 1999], daß es äußerst schwierig ist, dafür durch Beobachtung eines Benutzers verläßliche Daten zu ermitteln[24]. Außerdem stellt die Kapazität des Arbeitsgedächtnisses keine absolute Grenze für das Verstehen eines Satzes oder einer Planpräsentation dar, denn der Hörer/Leser kann Teile der Gesamtinformation in das Langzeitgedächtnis verlagern. Liegt die Planpräsentation während der Plan*ausführung* vor, verliert die Kapazität des Arbeitsgedächtnisses an Bedeutung, weil evtl. verloren gegangene Informationen jederzeit durch erneutes Lesen wiedergewonnen werden können. Dennoch ist offensichtlich, daß einem Benutzer auch in einem solchen Fall keine beliebig umfangreiche Planpräsentation zugemutet werden kann.

Abgesehen von den Schwierigkeiten für die aktuelle Situation einen sinnvollen Grenzwert festzulegen, den die Bewertung der Planpräsentation nicht überschreiten darf, wäre es ohnehin vorteilhafter, die Bewertung mit anderen Alternativen zur vergleichen, also wie bei der Entscheidung über die beste Ausführungsalternativen eines Plan-Concepts zu verfahren. Da der Prozeß der Planerzeugung aufwendig ist, ist es dabei allerdings nicht sinnvoll alle möglichen minimalen[25] Pläne die die Ziele erreichen zu erzeugen und dann daraus den besten auszuwählen. Als Alternative kann die Bewertung allerdings von vorn herein in die Plangenerierung integriert werden. Dies ist Thema von Kap. 4.5 *Optimierte Plangenerierung*.

4.3.3 Die Bedeutung falscher Annahmen des Benutzers für die Planpräsentation

Weiter oben wurde argumentiert, daß der Benutzer nur dann keine weiteren Erklärungen zu Ausführungsmöglichkeiten eines nicht-atomaren Plan-Concepts benötigt, wenn er über verläßliches Wissen dazu verfügt. Falls dies nicht der Fall ist, wird nicht unterschieden, ob der Benutzer gar kein entsprechendes Wissen hat oder ob sein Wissen nach Ansicht des Systems falsch ist. Dem Benutzer wird in beiden Fällen eine korrekte und für ihn nutzbare Ausführungsmöglichkeit präsentiert. Hat der Benutzer kein eigenes Wissen über die Ausführungsmöglichkeiten, bleibt ihm letztlich keine andere Möglichkeit zur Ausführung eines entsprechenden Planschritts, als dem Vorschlag des Systems zu folgen. Glaubt er dagegen selbst zu wissen, wie das Plan-Concept ausgeführt werden kann, stehen ihm neben dem Vorschlag des Systems noch andere Ausführungsalternativen zur Verfügung, von denen mindestens eine nach Ansicht des Systems falsch ist.

Um eine solche Situation bei einer Planpräsentation möglichst zu vermeiden, verschlechtert der Algorithmus aus Abb. 4.2 in Zeile (c7) die Bewertung einer Präsentation, wenn der Benutzer über entsprechendes Wissen verfügt, das aber nicht verläßlich ist. Durch den Faktor k_{corr} kann dabei der Stellenwert falscher Benutzerannahmen bestimmt werden. Je höher der Wert ist, desto stärker beeinflußt eine falsche Benutzerannahme die Bewertung und desto schlechter werden die Chancen einer Ausführungsmöglichkeit als Beste ausgewählt werden. Der Wert kann also als Maß für die "Vorsichtigkeit" des Systems angesehen werden, weil dadurch bestimmt wird in welchem Maße aufwendigere Planpräsentationen gegenüber solchen bevorzugt werden, die zu einer fehlerhaften Planausführung durch den Benutzer führen könnten.

Nicht zum Thema dieser Arbeit gehört eine Korrektur der falschen Benutzerannahmen. Es sei allerdings angemerkt, daß eine solche Korrektur zwar wünschenswert, aber im Rahmen einer Hilfestellung für den Benutzer nicht unproblematisch ist. Zum einen hat der Benutzer die falschen Annahmen, die er nach Ansicht des Systems hat, nicht notwendigerweise explizit geäußert wie bspw. in [McCoy, 1989]. Evtl. nimmt das System fälschlicherweise an, daß der Benutzer falsche Annahmen hat. Ein Korrektur könnte dann vom Benutzer als unakzeptable Belehrung empfunden werden. Dies

[24] Neuere Forschungen (vgl. [Müller *et al.*, 2001]) haben indes gezeigt, daß bestimmte Merkmale von gesprochener Sprache (bspw. Sprechpausen) Hinweise auf die kognitive Belastung des Sprechers geben können.

[25] Minimal heißt hier, daß der Plan keine Schritte enthält, die weder direkt noch indirekt zum Ziel beitragen.

könnte auch eintreten, wenn eine Planpräsentation viele Richtigstellungen von tatsächlich falschen Benutzerannahmen enthält.

Als weitere Möglichkeit zur Erhöhung der Wahrscheinlichkeit für eine korrekte Planausführung durch den Benutzer, kann die Präsentation um einen Hinweis ergänzt werden, daß der Plan nur in der präsentierten Weise ausgeführt werden sollte, falls das System annimmt, daß der Benutzer falsche Annahmen hat. Ein solcher Hinweis ist mit nur geringem Aufwand verbunden. Im Gegensatz zu einer Richtigstellung von falschem Wissen führt dies auch nicht vom eigentlichen Ziel, der Präsentation des Plans weg und vermeidet die o.g. Akzeptanzprobleme, falls der Benutzer gar kein falsches Wissen haben sollte.

4.3.4 Arbeiten zum Verstehen von Plänen und Anweisungen aus der Psychologie

Das Forschungsgebiet, das sich besonders mit dem Phänomen *Verstehen* befaßt hat, ist die Psychologie (vgl. bspw. [Kintsch, 1998]). Speziell zum Verstehen von Plänen oder Planbeschreibungen gibt es allerdings nur wenige Arbeiten. Diese gehören zum Teilgebiet *Textverstehen*. Zum Verstehen eines Textes im engeren Sinne versteht man dabei (bspw. nach [Hoppe-Graff, 1984]) den Prozeß, den Inhalt des Textes in eine interne Struktur im Gedächtnis des Lesers zu transformieren, sodaß daraus der Inhalt später wieder rekonstruiert werden kann. Der Gesamtprozeß von Transformation, Speicherung und Rücktransformation wird dann *Textverarbeitung* genannt.

Der Inhalt eines Textes kann dabei durch eine Menge von Propositionen dargestellt werden, die während des Verstehensprozesses erkannt und untereinander in Beziehung gesetzt werden. Die Propositionen bestehen dabei aus einem Prädikat mit einem oder mehreren Argumenten und entsprechen damit prädikatenlogischen Aussagen. Der Inhalt des Satzes

T ist ein Drucker

läßt sich bspw. durch die Proposition

(ISA T Drucker)

beschreiben. Die Art der Beschreibung entspricht damit der Beschreibung der Semantik von Texten durch logische Formeln, wie sie in der KI weit verbreitet ist. Die Relevanz von Propositionen für den Verstehensprozeß wurde durch viele Experimente untermauert. Bspw. zeigen Kintsch und Keenan [Kintsch und Keenan, 1973], daß die Lesezeit eines Satzes gleicher Länge mit wachsender Zahl zu verarbeitender Propositionen ansteigt.

Ein Ansatz den Verstehensprozeß zu modellieren ist Kintsch und van Dijks *Modell der zyklischen Textverarbeitung* [Kintsch und van Dijk, 1978]. Das Modell geht von der bereits erwähnten Begrenzung des Arbeitsgedächtnisses eines Menschen aus. Um auch längere Texte verstehen zu können, wird angenommen, daß eine zusammengehörige Teilmenge der Propositionen des Textes zu sogenannten Chunks zusammengefaßt in das (unbegrenzte) Langzeitgedächtnis verschoben wird. Im Arbeitsgedächtnis verbleiben einige wenige Propositionen als Anknüpfungspunkte. Dieser Prozeß wiederholt sich mit fortschreitender Verarbeitung des Textes. Dabei können auch mehrere Anknüpfungspunkte zu einem Chunk zusammengefaßt und in das Langzeitgedächtnis verschoben werden. Insgesamt ergibt sich damit eine baumartige Struktur, die Kohärenzgraph genannt wird. Nach [Miller und Kintsch, 1980] erfolgt die Zusammenfassung zu Chunks meist anhand der Oberflächengliederung eines Textes. Es wird angenommen, daß dieser Prozeß nur wenig kognitive Ressourcen des Lesers beansprucht. Im Falle einer Planpräsentation, wie in den vorangegangenen Abschnitten vorgestellt, kann die Dekomposition eines Planschritts in Teil-Schritte den Prozeß der Zergliederung eines Textes zusätzlich unterstützen. Im Algorithmus aus Abb. 4.2 wurde deshalb bereits vorgeschlagen den Aufwand für die Beschreibung einer Ausführungsmöglichkeit geringer zu bewerten, indem er mit dem Faktor $k_{hier} < 1$

multipliziert wird. Da der Aufwand für die Strukturierung aber ohnehin nicht groß ist, sei für Beispiele in dieser Arbeit von einem Wert von 0,9 für k_{hier} ausgegangen.

Aufbauend auf dem genannten Verstehensprozeß entwickelten Kintsch und Vipond [Kintsch und Vipond, 1979] ein Modell der *Textverständlichkeit*. Als Maß für die Verständlichkeit eines Textes wird dabei das Verhältnis der Zahl reproduzierter Propositionen zur Lesezeit definiert. Das Maß bezieht also nicht nur den Verstehens-, sondern den gesamten Textverarbeitungsprozeß mit ein. Es ist außer von den Eigenschaften des Textes auch von den Eigenschaften des Lesers abhängig – insbesondere der Kapazität seines Arbeits- und Kurzzeitgedächtnisses. Wenngleich dieser Ansatz wegen seiner Präzision und umfassenden Beschreibung des Textverarbeitungsprozesses Anerkennung genießt, bietet er vielerlei Angriffspunkte für Kritik. [Grabowski, 1991, Kap.4] widmet sich dem ausführlich. Ein Punkt ist dabei die bereits erwähnte Tatsache, daß die Zahl der Propositionen, die für die Darstellung des Inhalts eines Textes benötigt wird, keineswegs eindeutig bestimmbar ist (vgl. [Grabowski, 1991, Kap.4.1]).

Für diese Arbeit ist der Ansatz aber vor allem deshalb nicht einfach übertragbar, weil es bei der Präsentation eines Plans nicht darum geht, den Benutzer möglichst schnell in die Lage zu versetzen möglichst viele Propositionen wiedergeben zu können. Vielmehr muß er in die Lage versetzt werden, mithilfe des Plans seine Ziele zu erreichen.

Dies berücksichtigen die Arbeiten von Clark und Clark [Clark und Clark, 1968] sowie von Dixon [Dixon, 1982], indem der Einfluß unterschiedlich strukturierter Anweisungstexte sowohl auf die Zeit für das Lesen der Anweisung als auch auf die Fehlerhäufigkeit beim Ausführen der Anweisung untersucht wird. Da hierbei zwar die Oberflächenstruktur variiert, der Inhalt aber konstant gelassen wurde, sind die Erkenntnisse allerdings erst dann relevant, wenn die Planpräsentation in eine textuelle Darstellung transformiert wird. Für die Festlegung des Inhalts und dabei insbesondere für die Entscheidung zwischen Ausführungsalternativen können sie dagegen nicht genutzt werden.

Eine alternative Sicht vom *Verstehen* einer Anweisung schlagen Mannes und Kollegen in [Mannes und Kintsch, 1991] und [Doane *et al.*, 1992] vor. Sie gehen davon aus, daß der Empfänger der Anweisung bzw. Benutzer wissen muß was *getan* werden muß. Das *Verstehen* der Anweisung besteht nun genau darin, dieses Wissen zu ermitteln. Da eine *Anweisung* hier weniger eine Handlungsanweisung, als eher eine Zielbeschreibung ist, müssen die einzelnen Schritte geplant werden. Der Verstehensprozeß ist damit i.w. ein Planungsprozeß. Wichtig ist dabei, daß – wie allgemein bei Ansätzen, die von *mentalen Modellen*[26] ausgehen – das Wissen des Benutzers berücksichtigt wird; hier insbesondere sein Planungswissen. Die Besonderheit beim hier eingesetzten *Construction-Integration Model* [Kintsch, 1988] ist, daß davon ausgegangen wird, daß in der *Integrations-Phase* des Verstehensprozesses die Begriffe aus dem Langzeitgedächtnis des Benutzers hervorgehoben werden, die im aktuellen Kontext mit der Anweisung zusammenhängen. Der Planungsprozeß verwendet dann nur Planoperatoren, die hervorgehobenem Wissen entsprechen und kann damit wesentlich zielgerichteter arbeiten, als wenn – wie beim klassischen Planen – das gesamte Wissen eingesetzt wird.

Kieras und Bovair gehen in [Kieras und Bovair, 1984] von einer speziellen Variante eines mentalen Modells aus, das sie *Device Model* nennen. Dadurch wird die Vorstellung modelliert, die ein Benutzer von der Funktionsweise eines technischen Gerätes hat. Kieras und Bovair untersuchten, ob es sinnvoll ist einem Benutzer der ein Gerät bedienen soll zunächst die Funktionsweise zu erklären, sodaß er eine solche Vorstellung aufbauen kann. Die Bedienung selbst wurde den Testpersonen in jedem Fall durch Folgen von Bedienungsaktionen des Gerätes erklärt. Als Ergebnis zeigte sich, daß die Kenntnis der Funktionsweise zu einer erhöhten Ausführungsgeschwindigkeit bei verringerter Fehlerhäufigkeit führt. Besonders auffällig war, daß Testpersonen denen die Funktionsweise der Testapparatur erklärt wurde, sehr viel häufiger Bedienungsfolgen einsetzten, die gegenüber den gelernten

[26] vgl. [Johnson-Laird, 1980], [van Dijk und Kintsch, 1983]

effizienter, d.h. kürzer waren. Als Gesamtergebnis der Untersuchung stellen Kieras und Bovair fest, daß es mithilfe der Kenntnis über die Funktionsweise offenbar (leichter) möglich ist, eigene Bedienungsstrategien zu entwickeln und dies auch einzusetzen um Aktionsfolgen wieder zu rekonstruieren, wenn sie teilweise vergessen wurden. Für diese Schlußfolgerungen war insbesondere die Kenntnis der konkreten Funktionszusammenhänge zwischen den Funktionseinheiten (in den Experimenten der Energiefluß) wichtig, während die Kenntnis allgemeiner Funktionsprinzipien dafür weniger hilfreich war. Für die Bedienung sehr einfacher Geräte halten die Autoren es allerdings nicht für sinnvoll, einem Benutzer die Funktionsweise zu erklären, weil diese vom Benutzer leicht selbst erkannt werden kann oder die Kenntnis für eine Bedienung gar nicht notwendig ist.

Die Untersuchung stützt damit den Ansatz dieser Arbeit, daß dem Benutzer bei der Präsentation eines Plans nicht nur die Schritte in einer geeigneten Reihenfolge genannt werden, sondern auch die Funktion, die die Schritte im Plan haben. Auf der anderen Seite ist dies nicht immer notwendig, insbesondere nicht, wenn der Benutzer die Zusammenhänge selbst leicht erkennen kann. Darauf wird noch in Kap. 4.4 *Berücksichtigung von Benutzer-Inferenzen* eingegangen.

4.3.5 Alternative Modelle zur Planbewertung

Die bisher vorgeschlagene Bewertung einer Planpräsentation basiert ausschließlich auf der Abschätzung des Aufwands für das Verstehen der Planpräsentation durch den Benutzer. Darüber hinaus sind weitere Kriterien für die Auswahl denkbar. Naheliegende Beispiele sind die Präferenzen des Benutzers bezüglich der Ausführung von Plan-Concepts und die Wahrscheinlichkeit mit der er durch das Ausführen eines Plan-Concepts auch tatsächlich den erwarteten Effekt erzielt. Zwar beschränkt sich diese Arbeit auf die bisher behandelte Art der Bewertung, dennoch soll hier gezeigt werden, daß der generelle Ansatz nicht darauf beschränkt ist und deshalb ein mögliches Vorgehen für die beiden Beispiele grob skizziert werden.

Eine Bewertung anhand der Präferenzen des Benutzer ist insofern vergleichbar zur Bewertung anhand des Aufwandes für das Verstehen der Präsentation, als jedes einzelne Auftreten eines Planschrittes einen additiven Beitrag zu der Bewertung hinzufügt. Ist dieser Beitrag umgekehrt proportional zu einem Maß für die Präferenz des Benutzers für das Ausführen des zugehörigen Plan-Concepts, kann eine vom Benutzer präferierte Ausführungsalternative eines Plan-Concepts bestimmt werden, indem diejenige mit der geringsten Bewertung ausgewählt wird. Die Auswahl ist damit analog zur Auswahl einer Ausführungsalternative mit dem geringst möglichen Aufwand (r4 in Abb. 4.2). Präferenz-Werte für Plan-Concepts können im Benutzermodell gespeichert werden, wie dies bereits Rich [Rich, 1983] für ein System vorgeschlagen hat, daß einem Benutzer Bücher empfehlen sollte, die für ihn interessant sind. Solche Werte sind für jedes Plan-Concept erforderlich, daß in einem Plan vorkommen kann. Das sind – unabhängig davon ob der Benutzer sie kennt – die Plan-Concepts seines Handlungspotentials. Der Zugriff auf diese Werte erfolgt im Präsentationsalgorithmus aus Abb. 4.2 in Zeile (p5) durch die Funktion *effort-plan-concept*. Die persönliche Einstellung eines Benutzers für oder gegen die Ausführung eines Plan-Concepts hängt oft nicht nur von dem Plan-Concept selbst ab, sondern auch davon *wie* das Plan-Concept ausgeführt wird bzw. werden muß. Bspw. werden Benutzer, die für die Computerbedienung die Verwendung einer Maus bevorzugen, vermutlich solche Plan-Concepts präferieren, für die eine Ausführungsmöglichkeit mithilfe von Maus-Aktionen zur Verfügung steht.

Für Plan-Concepts, die der Benutzer nicht atomar kennt, setzt sich der Präferenz-Wert deshalb aus einer Komponente für das Plan-Concept selbst und einer für die Ausführungsmöglichkeit zusammen. Ein Effekt, daß die Berücksichtigung einer Dekomposition eines Plan-Concepts selbst die Bewertung verändert, wie im Algorithmus in Zeile (c8) durch die Komponente k_{pres} und den Faktor k_{hier}, ist hierbei allerdings nicht angebracht. Ein weiterer Unterschied ergibt sich bei Plan-Concepts, für die der Benutzer verläßliches Wissen hat oder die bereits erklärt wurden. Während das System dafür keine

weiteren Erklärungen abgibt und so auch kein Aufwand für das Verstehen beim Benutzer anfallen kann, müssen die Präferenz-Werte auch von Teil-Schritten berücksichtigt werden, weil der Benutzer die Schritte ja sehr wohl ausführen muß. Da das System dabei aber keine Vorgaben mehr macht und der Benutzer frei entscheiden kann, stehen als (sinnvolle) Ausführungsalternativen nur diejenigen zur Verfügung, für die er verläßliche Kenntnisse hat. Per Definition von verläßlichem Wissen muß es davon mindestens eine geben. Stehen mehrere zur Verfügung kann angenommnen werden, daß der Benutzer sich anhand seiner Vorlieben entscheidet und demnach diejenige mit der geringsten Bewertung auswählt. Es bleibt anzumerken, daß sich der Präferenz-Wert eines Plan-Concepts nicht nur eignet die subjektiven, persönlichen Vorlieben des Benutzers zur modellieren. Es kann damit auch der Aufwand für die Ausführung des Plan-Concepts durch den Benutzer modelliert werden – bspw. der damit verbundene Zeitverbrauch.

Eine Bewertung eines Plans anhand von Erfolgswahrscheinlichkeiten trägt der Tatsache Rechnung, daß die Ausführung eines Planschrittes auch dann nicht immer die modellierten Effekte des zugehörigen Plan-Concepts erreicht, wenn alle Vorbedingungen erfüllt sind. Im Bereich Handlungsplanung werden unter dem Schlagwort *Planen unter Unsicherheit* verstärkte Anstrengungen unternommen, dies auch im Planungsprozeß zu berücksichtigen (vgl. bspw. [Kushmerick *et al.*, 1995], [Weld *et al.*, 1998]). Für die Generierung eines Plans gehen die meisten praktischen Planungssysteme – und auch diese Arbeit – allerdings zur Vereinfachung des Planungsprozesses von den Voraussetzungen des klassischen Planens aus (vgl. Kap. 2.1.3 *Voraussetzungen beim Planen*) und ignorieren Unsicherheiten damit. Die Vorgehensweise ist gerechtfertigt, wenn Pläne nur für solche Anwendungsbereiche generiert werden, die geringe Unsicherheiten aufweisen; d.h. in denen die Voraussetzungen des klassischen Planens weitgehend zutreffen. Damit kann erwartet werden, daß solche Pläne *meistens* ihre Ziele erreichen. Dies ist insofern zufriedenstellend, als es Pläne die *garantiert* ihre Ziele erreichen in der Realität ohnehin nicht gibt. Zu Beginn dieses Kapitels wurde auch argumentiert, daß unerwartete – d.h. vom Planungssystem nicht berücksichtigte – Hindernisse, die der Planausführung im Wege stehen, oft vom Benutzer als intelligenten Akteur beseitigt werden können. Dennoch bleibt die Möglichkeit, daß die Ausführung eines Planschritts scheitert, weil diese – in einer aktuellen Situation – zu hohe Anforderungen an die Konzentration oder Motorik des Benutzers stellt oder weil er trotz der Ratschläge des Systems falsch ausgeführt wird. Das Handlungspotential des Benutzers enthält zwar (bei korrekter Modellierung) keine Plan-Concepts, die dieser *niemals* korrekt ausführen kann; es wäre aber nicht sinnvoll nur solche Plan-Concepts für das Handlungspotential zuzulassen, die der Benutzer *immer fehlerlos* ausführt, da es solche Plan-Concepts strenggenommen in der Realität nicht geben kann. Die Berücksichtigung von jeweils unterschiedlichen, benutzerspezifischen Wahrscheinlichkeiten für die erfolgreiche Ausführung der Plan-Concepts bietet die Möglichkeit, die Wahrscheinlichkeit abzuschätzen, mit der der Benutzer durch das Ausführen eines Plan seine Ziele erreicht. Daran kann sich dann auch eine Entscheidung zwischen mehreren Ausführungsmöglichkeiten eines Plan-Concepts orientieren. Die einzelnen Erfolgswahrscheinlichkeiten können in gleicher Weise wie die Präferenz-Werte im Benutzermodell gespeichert werden. Neben den üblichen Möglichkeiten zur Gewinnung von aktuellen Werten für einen Benutzer kann der Wert auch durch die Beobachtung, wie oft ein Benutzer ein Plan-Concept ausgeführt hat, beeinflußt werden, wie dies in [Peter und Rösner, 1994] vorgeschlagen wird.

Liegt ein Plan als Folge von Planschritten vor, deren Erfolgswahrscheinlichkeiten jeweils voneinander unabhängig sind, so kann die Erfolgswahrscheinlichkeit des gesamten Plans als Produkt der einzelnen Erfolgswahrscheinlichkeiten berechnet werden, da für den Erfolg des Plans *jeder* Planschritt erfolgreich ausgeführt werden muß. Da für nicht-atomare Plan-Concepts davon ausgegangen wird, daß der Benutzer zu deren Realisierung Teil-Planschritte ausführen muß, die selbst einen (Teil-)Plan bilden, ist die Erfolgswahrscheinlichkeit eines solchen Plan-Concepts abhängig von der Erfolgswahrscheinlichkeit des Teil-Plans. Für eine erfolgreiche Ausführung des Plan-Concepts ist die erfolgrei-

che Ausführung des Teil-Plans eine notwendige Voraussetzung. Es bietet sich also ähnlich wie bei den Präferenz-Werten an, die Gesamt-Erfolgswahrscheinlichkeit eines Plan-Concepts in einen Anteil für das Plan-Concept selbst und einen Anteil für den Teil-Plan aufzuspalten, wobei diese allerdings multiplikativ verknüpft werden müssen. Der Anteil für das Plan-Concept selbst berücksichtigt dabei, daß die Ausführung eines Teil-Plans i.a. noch keine hinreichende Bedingung für die erfolgreiche Ausführung des Plan-Concepts ist. Im Beispiel des *Herunterfahrens eines Computers* könnte die Aktion trotz korrekt ausgeführtem Teil-Plan scheitern, weil bspw. zum Zeitpunkt der Ausführung zufällig noch eine Datenübertragung zu einem anderen Computer stattfindet.

Soll die Bewertung einer Ausführungsalternative oder der gesamten Planpräsentation nicht nur anhand eines Kriteriums erfolgen, muß die Funktion *effort-plan-concept* im Präsentationsalgorithmus Werte für alle Kriterien liefern. Das Ergebnis dieser Funktion und auch die Komponente *effort* der Ergebnisse der Funktionen des Präsentationsalgorithmus müssen dementsprechend Vektoren sein. Die Entscheidung über die *beste* Ausführungsalternative kann dann nicht mehr einfach durch die Auswahl der Alternative mit der geringsten Bewertung erfolgen. Stattdessen ist eine Entscheidungsfunktion erforderlich, die die unterschiedlichen Gewichte der verschiedenen Kriterien berücksichtigt. Darüber hinaus können dabei auch weitere Aspekte berücksichtigt werden, bspw. daß ein Wert einer einzelnen Komponente einen Grenzwert nicht über- oder unterschreiten darf.

4.4 Berücksichtigung von Benutzer-Inferenzen

In den vorangegangenen Abschnitten wurden die Verfahren zur Bestimmung der Information beschrieben, die ein Benutzer benötigt, um mit einem Plan (den er im Prinzip ausführen kann) seine Ziele zu erreichen. Dabei wurden zwar die Kenntnisse des Benutzers berücksichtigt, um zu vermeiden, daß ihm Informationen präsentiert werden die er bereits kennt. Unberücksichtigt blieb aber seine Fähigkeit Inferenzen zu ziehen, um benötigte Informationen zu bekommen. Beschreibungen, die die gesamte darzustellende Informationsmenge ohne Berücksichtigung dieser Inferenzfähigkeit explizit enthalten, werden von Hörern/Lesern oft als unnatürlich und umständlich empfunden. Dies zeigt bspw. Mellish und Evans' erster Ansatz eine natürlichsprachliche Beschreibung eines Plans zu generieren [Mellish und Evans, 1989], der vom Planungssystem NONLIN [Tate, 1976] erzeugt wurde. Solche Beschreibungen widersprechen auch Grice's Kooperativitätsprinzip [Grice, 1975], nach dem sich eine sprachliche Äußerung auf die relevanten Aspekte beschränken sollte und insbesondere nichts enthalten sollte, das der Hörer selbst leicht schließen kann.

4.4.1 Verwandte Ansätze zur Berücksichtigung von Benutzer-Inferenzen

Im Forschungsgebiet *Generierung von natürlichsprachlichen Texten* haben sich besonders Zukerman, McConachy und Horacek dieser Problematik angenommen. Zukerman und McConachy schlugen für ein Intelligentes-Tutor-System vor, die Fähigkeit des Benutzers – hier des Schülers – zu berücksichtigen, um langweilige Texte zu vermeiden ([Zukerman und McConachy, 1993b], [Zukerman und McConachy, 1995]). Langeweile wird demnach bei schwachen Schülern bereits durch die Textlänge, bei guten Schülern durch eine große Zahl von Aussagen, die sie selbst leicht schließen können hervorgerufen. Horacek ([Horacek, 1997]) verwendet einen ähnlichen Ansatz für die Verbesserung von Erklärungen eines Expertensystems. Beide Ansätze setzen ein Benutzermodell ein, daß die Annahmen des Systems über Schlußfolgerungsfähigkeiten des Benutzer durch Inferenzregeln repräsentiert.

Für die Beschreibung von Plänen setzt Young [Young, 1999] darauf auf und untersucht, wie sich das Weglassen irrelevanter Komponenten aus einer Planbeschreibung auf die Häufigkeit auswirkt, mit der Benutzer den Plan fehlerhaft ausführen. Irrelevant sind für Young solche Komponenten, die der Benutzer selbst mit begrenztem Aufwand inferieren kann. Die möglichen Inferenzen eines Benutzers

werden dabei nicht durch explizit repräsentierte Inferenzregeln modelliert. Stattdessen sieht Young einen um irrelevante Komponenten reduzierten Plan als unvollständigen Plan in einem Zwischenzustand des Planungsprozesses an. Zur Vervollständigung des Plans kann sich ein Benutzer demnach eines Planungsalgorithmus bedienen, um fehlende Komponenten eines Plans zu ergänzen. Young geht davon aus, daß die Benutzer dafür jeweils für *sie* spezifische Planungsalgorithmen einsetzen. Er führt allerdings nicht aus, welche Effekte er von solchen *benutzerspezifischen Algorithmen* erwartet und wie der Algorithmus, den ein bestimmter Benutzer verwendet, bestimmt werden könnte. Für seine Untersuchungen setzt Young auch nur einen einzigen Planungsalgorithmus[27] ein. Die Abschätzung ob ein Benutzer einen unvollständigen Plan mit begrenztem Aufwand vervollständigen kann, beruht auf der Anzahl der Pläne, die dieser Algorithmus untersucht, bis er eine Lösung findet.

Zur Gewinnung von unvollständigen Plänen, die als Ausgangspunkt für diese Abschätzung dienen können, bedient sich Young zweier Verfahren, dem *Local Brevity* und dem *Plan Path* Verfahren. Im ersten Verfahren werden analog zum Vorgehen von Dale und Reiter [Dale und Reiter, 1995] sukzessive Komponenten aus dem Plan entfernt, die anhand einer Bewertungsfunktion als weniger wichtig eingestuft werden. Die Entfernung der Komponenten endet, wenn jede weitere Verkürzung (um eine einzelne Komponente) dazu führt, daß der Planungsalgorithmus den Restplan bei vorgegebener Aufwandsbegrenzung nicht mehr vervollständigen kann. Die Kriterien der Bewertungsfunktion leitet Young aus empirischen Untersuchungen über die Wichtigkeit von Komponenten von Erzählungen oder Geschichten ab [Trabasso und Sperry, 1985], [Trabasso und van den Broeck, 1985], [van den Broeck, 1988], [Graesser *et al.*, 1980], die er auf Pläne überträgt. Demnach sind bspw. Planschritte um so wichtiger, je mehr Beziehungen sie zu ihrer Umgebung haben; d.h. je mehr Causal Links von ihnen ausgehen oder zu ihnen führen. Darüber hinaus wird beachtet, daß durch das Entfernen von Komponenten die *referenzielle Integrität*[28] der Beschreibung nicht verletzt wird; d.h. bspw. daß ein Planschritt nicht entfernt wird, wenn in dem Plan (noch) Causal Links mit diesem Schritt vorkommen.

Das *Plan Path* Verfahren bedient sich des Lösungsweges, der bei der Generierung des Plans beschritten wurde, indem genau diejenigen Pläne für die Aufwandsabschätzung verwendet werden, die einen Zwischenschritt bei der Konstruktion des vollständigen Plans darstellten. Die dafür notwendige Bedingung, daß als Ausgangsbasis für die Planpräsentation nicht nur der vollständige Plan, sondern auch die Zwischenschritte bekannt sind, ist bei Young erfüllt. Da ein Planschritt bei der Generierung niemals ohne den Causal Link, der seine Existenz im Plan rechtfertigt, in einen Plan eingefügt wird, kann ein solcher Causal Link in einem unvollständigen Plan, der nach dieser Methode gewonnen wurde, niemals fehlen. Dies gilt auch dann, wenn dieser für einen Benutzer offensichtlich ist und eine explizite Nennung demnach redundant wäre.

Es sei angemerkt, daß zumindest für das *Local Brevity* Verfahren nicht garantiert werden kann, daß es nicht Pläne mit weniger Komponenten geben könnte, die der Benutzer bei gegebener Aufwandsbegrenzung dennoch vervollständigen kann[29]. Es ist denkbar, daß eine alternative Reihenfolge bei der Entfernung der Komponenten zu einem solchen Ergebnis führen kann. Da beim *Plan Path* Verfahren der gleiche Algorithmus für die Plangenerierung und die Aufwandsabschätzung zum Einsatz kommt, nimmt die Zahl der noch zu berücksichtigen Pläne mit jedem Schritt der Generierung ab. Es kann also eindeutig festgestellt werden, ab welchem Generierungsschritt der Benutzer den Plan selbst vervollständigen kann. Evtl. kürzere "Seitenwege" kann es nicht geben, weil der Generierungsprozeß diese sonst selbst beschritten hätte.

Dennoch sind die Ergebnisse mit beiden Verfahren sehr vergleichbar. Gegenüber einer vollständigen expliziten Darstellung aller verfügbaren Informationen (die sich an [Mellish und Evans, 1989] anlehnt) enthalten die nach diesen Verfahren verkürzten Pläne nur rund die Hälfte der Komponen-

[27] DPOCL [Young *et al.*, 1994b]; s.a. Kap. 2.4.3 *Decompositional Partial-Order Causal Link Planning*

[28] vgl. [Kintsch, 1988], [Dale und Reiter, 1995]

[29] vgl. die Ergebnisse in [Dale und Reiter, 1995] für die Vereinfachung referenzieller Ausdrücke.

ten. Gleichzeitig sank der Anteil der Pläne, die falsch ausgeführt wurden (ihr Ziel nicht erreichten) von gut 17% auf weniger als 3%. Interessant ist auch Youngs Vergleich mit einer sogenannten *primitiven* Planpräsentation, die sich auf die Nennung der Planschritte in einer geeigneten Reihenfolge beschränkt, ohne darüber hinausgehende Strukturinformationen zu nennen. Die Zahl der genannten Komponenten ist dabei in etwa genauso groß, wie die nach den beschriebenen Verfahren verkürzten Pläne; die Fehlerrate liegt aber mit knapp 13% deutlich darüber. Andererseits ist die Fehlerrate aber immer noch (etwas) geringer als bei der ausführlichen Beschreibung.

Insgesamt wird damit die Vermutung bestätigt, daß es bei der Planpräsentation ein *Zuviel* an Information geben kann, das den Benutzer letztlich überfordert und Fehler bei der Planausführung begünstigt. Ebenfalls bestätigt sich, daß das gezielte Entfernen von Komponenten, die ein Benutzer leicht inferieren kann, einem ungezielten Weglassen aller Strukturinformation hinsichtlich der Wahrscheinlichkeit für eine korrekte Planausführung deutlich überlegen ist.

4.4.2 Relativer Aufwand und Benutzerwissen

Da der Präsentationsalgorithmus aus Abb. 4.2 keine Benutzer-Inferenzen berücksichtigt, dürfte die damit erzeugte Informationsmenge für viele Benutzer zu groß sein. Es bietet sich also an, Youngs Ideen zum Entfernen von leicht inferierbaren Komponenten einer Planpräsentation auch für diese Arbeit zu nutzen. Bei näherer Betrachtung zeigen sich allerdings eine Vielzahl von konzeptuellen Unterschieden zu Youngs Ansatz. Gemeinsam ist die Auffassung, daß es legitim ist anzunehmen, daß ein planender Benutzer ein vergleichbares Modell über Veränderungsmöglichkeiten hat wie ein (klassisches) Planungssystem und demnach auch ähnliche Schlußfolgerungen ziehen muß, um einen Plan zu vervollständigen. In dieser Arbeit wird aber nicht von benutzerspezifischen Planungsalgorithmen ausgegangen. Abgesehen von der Schwierigkeit festzustellen, welchen Planungsalgorithmus ein spezieller Benutzer einsetzt, erscheint eine Abschätzung des Benutzerverhaltens aufgrund unterschiedlicher Planungsalgorithmen wenig plausibel. Die für das Planen notwendigen Schlußfolgerungen und Verfahren gehören zur allgemeinen Kompetenz eines Menschen, die er ständig in den unterschiedlichsten Anwendungsfeldern einsetzt. Gravierende Divergenzen sind dabei also kaum zu erwarten und auch nicht belegt. Unterschiede gibt es dagegen bei der freien Kapazität des Arbeitsgedächtnisses, das der Benutzer für die Plangenerierung verwenden kann und insbesondere bei den Benutzerkenntnissen. Ersteres berücksichtigt Young durch eine benutzerspezifische Funktion die abschätzt, ob die Kapazitätsgrenze des Benutzers noch für weitere Berechnungen ausreicht. Dabei wird die Belastung des Arbeitsgedächtnisses durch die Interpretation der Plan*präsentation*, die ja parallel zur Planvervollständigung stattfindet, nicht berücksichtigt. Ebenfalls unberücksichtigt bleibt der Einfluß unterschiedlicher Benutzerkenntnisse auf die Möglichkeit, einen unvollständigen Plan zu vervollständigen. Dadurch, daß Youngs bei der Simulation der Benutzerinferenzen die gleichen Planoperatoren verwendet wie bei der ursprünglichen Planerzeugung durch das System wird impliziert, daß die Benutzer grundsätzlich die gleichen Kenntnisse bzgl. der Veränderungsmöglichkeiten im Sachbereich haben wie das System. Davon kann aber gerade bei Benutzern eines Hilfesystems nicht ausgegangen werden. Vielmehr muß damit gerechnet werden, daß die Benutzer Kenntnislücken oder sogar falsche Annahmen haben.

Ein weiterer wesentlicher Unterschied zu Youngs Ansatz besteht darin, daß in dieser Arbeit davon ausgegangen wird, daß die Schritte des fertigen Plans zwar für den Planungsprozeß nicht dekomponiert werden können, daß dies dem Benutzer aber zumindest für die nicht-atomaren Schritte sehr wohl möglich ist. In Abschnitt 4.2.1 *Dekomposition eines Plan-Concepts* wurde argumentiert, daß es so möglich ist, dem Benutzer für Planschritte Ausführungsmöglichkeiten zu präsentieren, die als feste Rezepte vorliegen und sinnvollerweise nicht geplant werden. Youngs hierarchischer Planungsalgorithmus verwendet zwar auch Dekompositionen von komplexen Planoperatoren, allerdings

nur während der Plangenerierung selbst. Die Möglichkeit, einen Benutzer zu unterstützen der nicht weiß wie ein Schritt des fertigen Plans ausgeführt wird, besteht damit nicht. Eine solche Notwendigkeit kann bei Youngs Ansatz nur vermieden werden, wenn von vorn herein ein Plan auf einem sehr niedrigen Abstraktionsniveau geplant würde. Abgesehen von den bereits genannten Nachteilen bzgl. Modellierungs- und Generierungsaufwand, wäre eine zugehörige Präsentation für viele Benutzer zu detailliert. Daran könnte auch Youngs Ansatz zur Vereinfachung der Präsentation nicht viel ändern, weil wegen der Aufwandsbegrenzung für die Vervollständigung der Planbeschreibung nur eine begrenzte Zahl von Komponenten aus der Präsentation entfernt werden kann, die nicht vom Wissen des Benutzers abhängt.

Auch beim Verfahren zur Eliminierung von Komponenten aus der Präsentation ergeben sich Unterschiede. Young ermittelt zuerst durch Weglassen von Komponenten aus dem vollständigen Plan Kandidaten für die Präsentation und prüft dann, ob eine Rekonstruktion des vollständigen Plans mit begrenztem Aufwand möglich ist. In dieser Arbeit wird dagegen die Entscheidung über das Entfernen für jede Komponente der Präsentation einzeln getroffen. Dazu werden die Benutzeraufwände gegenübergestellt, die für das Verstehen der Komponente in der Präsentation bzw. für die Rekonstruktion der Komponente, wenn sie aus der Präsentation entfernt wurde, geschätzt wurden. Es ist aber nicht für alle Komponenten sinnvoll, sie überhaupt für eine Eliminierung in Betracht zu ziehen. Außerdem ergeben sich aus der Forderung nach Wahrung der referenziellen Integrität der Beschreibung Bedingungen an das Entfernen von Komponenten. Vor der Behandlung der Abschätzung des Rekonstruktionsaufwandes werden deshalb im folgenden Abschnitt die entsprechenden Bedingungen zusammengestellt.

4.4.2.1 Bedingungen für das Entfernen von Komponenten aus einer Planpräsentation

Eine Planpräsentation enthält eine Dekomposition zu einem Planschritt nur dann, wenn der Prozeß zur Inhaltsbestimmung festgestellt hat, daß der Benutzer diese Information nicht dem eigenen Wissen entnehmen kann. Dementsprechend ist es nicht sinnvoll, sie zur Vereinfachung der Planpräsentation wieder zu entfernen. Auch der Planschritt selbst kann nicht aus der Planpräsentation entfernt werden; selbst dann nicht, wenn der Benutzer den Schritt selbst leicht rekonstruieren könnte. Das Entfernen eines solchen Schrittes würde die referenzielle Integrität der Beschreibung verletzen, da dann ein Schritt erklärt werden müsste, der in der Beschreibung sonst gar nicht vorkommt. Eine erste Bedingung, die an das Entfernen eines Planschritts aus einer Planpräsentation geknüpft ist, lautet dementsprechend:

(bs1) ein Planschritt s darf *nicht* aus einer Planpräsentation entfernt werden, wenn die Präsentation eine Dekomposition von s enthält.

Um die referenzielle Integrität der Beschreibung zu gewährleisten, sind darüber hinaus folgende Bedingungen einzuhalten:

Ein Planschritt s darf *nicht* aus einer Planpräsentation entfernt werden, wenn die Präsentation

(bs2) einen Causal Link (r, l, s) oder (s, l, r) enthält *oder*

(bs3) einen Planschritt r enthält und der *Plan* einen Causal Link (r, l_1, s) enthält, aber keinen Causal Link (r, l_2, t) von r zu einem anderen Planschritt als s, der in der *Planpräsentation* enthalten ist. l_1 und l_2 dürfen verschieden sein, müssen aber nicht.

Dabei wird zwischen *Plan* und Plan*präsentation* unterschieden. (bs2) besagt, daß ein Planschritt erst dann aus der Plan*präsentation* entfernt werden darf, wenn diese auch keinen Causal Link mehr enthält, der zu dem Planschritt hin oder von ihm weg führt. Im Plan selbst bleiben die Causal Links unverändert. Die Bedingung von (bs3) beruht dagegen auf den Causal Links des *Plans*, unabhängig davon, ob sie in der *Präsentation* enthalten sind oder nicht. Die Bedingung soll verhindern, daß ein Planschritt der Planpräsentation entfernt wird, wenn er *in der Präsentation der einzige Grund* für

die Existenz eines anderen Planschritts *r* ist, der ebenfalls in der Präsentation enthalten ist. *Einziger Grund für die Existenz von r in der Präsentation* heißt, daß kein weiterer Planschritt *t* in der *Präsentation* enthalten ist, zu dessen Vorbedingung *r* ebenfalls beiträgt (zu dem es also einen Causal Link im Plan gibt). Dabei kommt es darauf an, ob ein solcher Schritt *t* tatsächlich in der *Planpräsentation* enthalten ist; nicht nur, ob er im Plan enthalten ist. Zur Veranschaulichung sei vom Plan aus Abb. 4.1 S. 174 ausgegangen und angenommen, daß der Schritt *ftp-bin-mode* zum Einschalten des Binär-Modus des FTP-Programms nicht präsentiert werden soll, weil der Benutzer den Schritt leicht rekonstruieren kann. Dann ist der Schritt *ftp-dnload-bin* zum Dateitransfer der – in diesem Sinne – einzige Grund für die Existenz von Schritt *launch*(*ftp-prg*) in der Planpräsentation. Dies ist unabhängig davon, daß *launch*(*ftp-prg*) tatsächlich zu den Vorbedingungen zweier Planschritte beiträgt. Diese beiden Bedingungen hängen also im Gegensatz zu Bedingung (bs1) davon ab, welche Komponenten bereits aus der Präsentation entfernt wurden.

Der Aufwand beim Vervollständigen eines unvollständigen Plans durch das Hinzufügen fehlender Planschritte ergibt sich im wesentlichen durch die Auswahl geeigneter Planoperatoren, die Vorbedingungskomponenten von (bereits im Plan befindlichen) Planschritten oder Komponenten des Ziels erreichen sollen sowie der Überprüfung, ob die Effektkomponenten nicht bereits Erreichtes wieder zerstören. Stehen bei einer Operatorauswahl mehrere Alternativen zur Verfügung, die die gleiche Bedingung erreichen, muß ermittelt werden mit welchem Operator eine Lösung möglich ist. Hier besteht – wie beim ursprünglichen Planungsprozeß – die Möglichkeit viel unnötigen Aufwand zu treiben, wenn sich erst spät herausstellt, daß die Wahl falsch war. Um den Benutzer bei der Rekonstruktion eines unvollständigen Plans von vorn herein vor dieser Situation zu bewahren, werden keine Planschritte aus der Planpräsentation entfernt, die im Plan Bedingungen erreichen, die *nach Wissen des Benutzers* auch durch andere Planoperatoren erreicht werden können. Daraus ergibt sich eine weitere Bedingung, die an das Entfernen eines Planschritts aus einer Planpräsentation geknüpft ist:

(bs4) ein Planschritt *s*, der Exemplar des Plan-Concepts *ps* ist, darf *nicht* aus einer Planpräsentation entfernt werden, wenn der Plan einen Causal Link (*s*, *l*, *r*) enthält und der Benutzer ein von *ps* verschiedenes Plan-Concept *ps'* kennt, das *l* als Effektkomponente hat.

Weitere Möglichkeiten für unnötigen Aufwand bestehen, wenn der Benutzer annimmt, daß ein Planschritt, der rekonstruiert werden muß, mehr Vorbedingungs- oder Effektkomponenten hat als tatsächlich – d.h. nach Ansicht des Systems – bestehen. Der zusätzliche Aufwand besteht darin, daß die zusätzlichen Vorbedingungskomponenten erfüllt werden müssen, bzw. daß überprüft wird, ob die zusätzlichen Effektkomponenten Causal Links bedrohen[30] und ggf. daß diese Bedrohungen beseitigt werden. Da diese Komponenten bei der Plangenerierung nicht berücksichtigt wurden, kann allerdings (ohne Simulation der möglichen Planungsprozesse des Benutzers) nicht einmal abgeschätzt werden, ob hierfür überhaupt eine Lösung existiert. Auch diese Möglichkeiten werden deshalb durch eine weitere Bedingung ausgeschlossen:

(bs5) ein Planschritt *s*, darf *nicht* aus einer Planpräsentation entfernt werden, wenn der Benutzer eine falsche Vorbedingungs- oder eine falsche Effektkomponente von *s* kennt; d.h. eine Komponente, die das System nicht kennt[31].

Eine weitere naheliegende Bedingung ist, daß nur solche Planschritte aus einer Planpräsentation entfernt werden dürfen, die der Benutzer gut genug kennt, um sie wieder in den Plan einfügen zu können. Das heißt insbesondere, daß er alle Vorbedingungs- und Effektkomponenten des Planschritts kennt, die im Plan eine Rolle spielen. Diese Bedingung wird allerdings nicht direkt als Bedingung an das Entfernen von Planschritten gestellt, sondern wirkt indirekt über die folgende Bedingung an das Ent-

[30] vgl. Kap. 2.1.1 *Least Commitment Planning*

[31] Es sei an die Sprechvereinfachung aus Abschnitt 4.2 erinnert, daß Kenntnisse über einen Planschritt von den Kenntnissen über das Plan-Concept von dem er abgeleitet ist geerbt werden.

fernen von Causal Links:

(bc1) ein Causal Link (r, l, s) darf nur dann aus einer Planpräsentation entfernt werden, wenn der Benutzer weiß, daß l sowohl Effektkomponente von r als auch Vorbedingungskomponente von s ist.

Damit wird einerseits sichergestellt, daß der Benutzer die notwendigen Kenntnisse zur Rekonstruktion des Causal Links hat. Da ein Planschritt s nach (bs2) aber nur entfernt werden darf, wenn in der Plan*präsentation* weder Causal Links zu dem Schritt hin noch welche davon weg führen, kann s erst dann entfernt werden, wenn alle Causal Links *des Plans* die s enthalten bereits entfernt worden sind. Demnach muß Bedingung (bc1) für alle diese Causal Links erfüllt sein. Der Benutzer muß demnach auch alle Vorbedingungs- und Effektkomponenten des Planschritts kennen, die im Plan genutzt werden. Es verbleiben die Effektkomponenten, die zwar nicht für einen anderen Planschritt genutzt werden, die aber Einfluß auf die Reihenfolge der Planschritte haben, weil sie einen Causal Link des Plans bedrohen können. Daraus ergibt sich die folgende Bedingung, die der Einfachheit halber auch Effektkomponenten umfaßt, von denen ein Causal Link ausgeht:

(bs6) ein Planschritt s, darf nur dann aus einer Planpräsentation entfernt werden, wenn der Benutzer alle Effektkomponenten von s kennt, die einen Causal Link des Plans bedrohen können.

4.4.2.2 Aufwandsabschätzung für die Rekonstruktion von Komponenten

Sind die Bedingungen für das Entfernen einer Komponente erfüllt, kann die Entscheidung über das Entfernen anhand des damit verbundenen Nutzens getroffen werden. Ein Nutzen ergibt sich immer dann, wenn das Entfernen der Komponente den gesamten Aufwand für das Verstehen der Planpräsentation plus der Rekonstruktion der entfernten Komponenten verringert. Ein solches relatives Kriterium hat gegenüber Youngs absoluter Aufwandsbegrenzung den Vorteil, daß es unabhängig von der Länge eines Plans wirkt. Eine feste Begrenzung des Gesamtaufwandes kann bei langen Plänen dazu führen, daß nur ein *relativ* geringer Anteil der Komponenten entfernt wird. Dabei bleibt unberücksichtigt, daß der Benutzer nicht alle Informationen gleichzeitig im Arbeitsgedächtnis haben muß, sondern nur solche, die mit dem aktuell betrachteten Planschritt in Beziehung stehen (vgl. dazu auch [Doane *et al.*, 1992]). Bei einem kurzen Plan dagegen könnte sogar vollständig auf eine Präsentation verzichtet werden bzw. der inakzeptable Rat

Denken Sie selbst nach.

gegeben werden.

Für die Abschätzung des Aufwands für das Rekonstruieren von Komponenten eines unvollständigen Plans – kurz Rekonstruktionsaufwand – bietet es sich an, analog zur Konstante k_{pres}, die in Abschnitt 4.3 *Inhaltsbestimmung und Bewertung einer Planpräsentation* als Maßeinheit für den Aufwand bei der Präsentation definiert wurde, eine entsprechende Maßeinheit für den Rekonstruktionsaufwand zu definieren. Auch hierfür ist der absolute Wert nicht von Bedeutung. Für den Vergleich solcher Aufwände ist stattdessen das Verhältnis der beiden Einheiten wichtig. Der hier relevante Basisaufwand für die Überprüfungen, ob ein Planoperator eine Bedingung erreichen kann, bzw. ob zwei Literale (hier eine Effektkomponente und das Literal eines Causal Links) im Widerspruch stehen, sei deshalb relativ zu k_{pres} durch $k_{check} * k_{pres}$ definiert. Auf einen Wert für k_{check} wird im nächsten Abschnitt noch eingegangen.

Um einen Causal Link (r, l, s) rekonstruieren zu können muß der Benutzer wissen, daß der Plan die Schritte r und s enthält, daß r die Effektkomponente l und daß s die Vorbedingungskomponente l hat. Nach (bc1) sind die letzten beiden Bedingungen für jeden Causal Link erfüllt, der aus der

Präsentation entfernt wurde. Vorausgesetzt, daß der zu vervollständigende Plan alle Schritte enthält, braucht der Benutzer dann nur noch den nächsten Vorgänger von Schritt *s* im Plan zu suchen, der *l* als Effektkomponente hat. Der Aufwand beschränkt sich also auf die Überprüfung von *r* und der Planschritte zwischen *r* und *s*. Da der Plan linear präsentiert wird, kann die Position eines Planschritts eindeutig bestimmt werden. Liefert die Funktion $pos_P(s)$ diesen Wert für einen Schritt *s* eines Plans *P*, so ist der Aufwand für die Rekonstruktion eines Causal Links (*r*, *l*, *s*) im Plan *P* unter den genannten Voraussetzungen:

(1) $(pos_P(s) - pos_P(r)) * k_{check} * k_{pres}$

Läßt sich *l* nach Ansicht des Benutzers nur durch ein einziges Plan-Concept *pc* erreichen, reduziert sich der Aufwand, weil dann nicht mehr verglichen werden muß, ob der Planschritt *l* als Effektkomponente hat. Stattdessen reicht es zu prüfen, ob der Planschritt ein Exemplar von *pc* ist. Es sei angenommen, daß sich der Vergleichsaufwand dadurch halbiert. Der gesamte Aufwand wird dadurch:

(2) $((pos_P(s) - pos_P(r))/2) * k_{check} * k_{pres}$

Wurden der Präsentation auch Planschritte entfernt, haben außer *r* und *s* nur die Schritte dazwischen einen Einfluß auf die Rekonstruktion eines Causal Links (*r*, *l*, *s*). Analog zum Vorgehen beim Planungsprozeß ist die Rekonstruktion des Causal Links erforderlich, um die Vorbedingung von *s* zu erfüllen. Wir gehen deshalb davon aus, daß *s* nicht aus der Präsentation entfernt oder bereits rekonstruiert wurde. Fehlen Schritte zwischen *r* und *s* wird der tatsächliche Aufwand für die Rekonstruktion geringer; die Formel stellt für diesen Fall eine obere Grenze dar. Fehlt dagegen *r*, so sind mehr Überprüfungsschritte erforderlich. Da sie aber durch das Entfernen bzw. die dadurch notwendige Rekonstruktion des Planschritts *r* verursacht werden, wird dieser Aufwand durch den Aufwand für die Rekonstruktion von Planschritten berücksichtigt.

Zur Abschätzung des Aufwandes für das Rekonstruieren eines Planschritts *r* wird von folgender Vorstellung über die Vorgehensweise eines Benutzers ausgegangen. Damit der Benutzer überhaupt zu dem Schluß kommt, daß er den Schritt einfügen muß, wird er keinen anderen Schritt im Plan gefunden haben, der eine notwendige, aber nicht erfüllte Vorbedingung *l* für einen anderen Schritt *s* herstellt. Im ungünstigsten Fall hat er dazu alle Schritte bis zum Anfang des Plans überprüft. Da der Aufwand für das Überprüfen bis zur Position von *r* im ursprünglichen Plan bereits beim Aufwand für das Rekonstruieren des Causal Links berücksichtigt wurde, verbleibt der Aufwand für das Überprüfen der $pos_P(r)$-1 Schritte von der Position von *r* bis zum Anfang des Plans. Da *l* wegen (bs4) nur durch ein einziges Plan-Concept erreicht werden kann, wird hierfür wiederum jeweils eine halbe Basiseinheit k_{check} * k_{pres} berücksichtigt. Dies ist dennoch eine worst-case Abschätzung, denn möglicherweise enthält der rekonstruierte Plan zum Zeitpunkt der Überprüfung (noch) nicht alle Schritte vor *r*. Die Auswahl des Plan-Concepts selbst ist wegen (bs4) trivial. Der Aufwand sei deshalb wieder mit einer halben Basiseinheit abgeschätzt. Der eigentliche Aufwand für das *Hinzufügen* eines Planschritts zu einem unvollständigen Plan besteht in der Überprüfung, an welcher Position er hinzugefügt werden kann, ohne dadurch bestehende Causal Links zu bedrohen. Beginnt der Benutzer seine Suche an der Position unmittelbar vor *s* und fährt in Richtung Plananfang fort, so muß er spätestens an der Position Erfolg haben, die das Planungssystem für *r* festgelegt hat und die deshalb konfliktfrei sein muß. Werden alle diese Positionen berücksichtigt, kann eine Effektkomponente von *r* alle Causal Links bedrohen, die vor *s* beginnen und nach *r* enden, mit Ausnahme der Causal Links, die von *r* selbst ausgehen. Sei die Anzahl dieser Causal Links, die unterschiedliche Bedingungen haben *num-cl* (*r*, *s*) und sei *num-eff* (*r*) die Zahl der Effektkomponenten von *r* die der Benutzer kennt, so sind im ungünstigsten Fall *num-cl* (*r*, *s*) * (*num-eff* (*r*)-1) Überprüfungen erforderlich. Dies berücksichtigt, daß die Effektkomponente die *l* erreicht keine Überprüfung erfordert, weil *l* ohnehin wahr sein muß, damit die Vorbedingung von *s* erfüllt wird. Dadurch kann das Erreichen von *l* keinen Konflikt

hervorrufen. Der (maximale) Gesamtaufwand für das Rekonstruieren eines Planschritts r, der eine Vorbedingungskomponente für Schritt s erfüllen soll, ist demnach:

$$(3)\ (\textit{num-cl}(r,s) * (\textit{num-eff}(r) - 1) + (pos(r) - 1)/2 + 1/2) * k_{check} * k_{pres}$$

bzw. vereinfacht

$$(4)\ (\textit{num-cl}(r,s) * (\textit{num-eff}(r) - 1) + pos(r)/2) * k_{check} * k_{pres}$$

Da von einem Planschritt mehrere Causal Links ausgehen können und nicht vorausgesehen werden kann, welche offene Bedingung der Benutzer zuerst erreichen will, muß das Maximum über alle Planschritte s, für die es ein Literal l_s und einen Causal Link (r, l_s, s) gibt, verwendet werden. Die Aufwandsabschätzung für das Hinzufügen von Planschritten gilt nur für solche Schritte, die nach den genannten Bedingungen aus einer Präsentation entfernt werden dürfen, da dann kein Aufwand für evtl. Fehlentscheidungen bei der Auswahl eines geeigneten Plan-Concepts anfällt.

Um den Gesamtaufwand für das Verstehen einer Präsentation plus dem Aufwand für die Rekonstruktion evtl. entfernter Komponenten möglichst klein zu halten, können die Komponenten entfernt werden, für die der Aufwand für das Verstehen größer ist, als die für die Rekonstruktion. Dementsprechend sei der (Einzel-) *Nutzen* durch das Entfernen einer Komponente als (geschätzter) Aufwand für das Verstehen der Komponente in der Präsentation abzüglich des (geschätzten) Rekonstruktionsaufwandes definiert. Nach den o.g. Bedingungen (bs2) und (bs3) kann das Entfernen von Planschritten davon abhängen, ob andere Komponenten entfernt wurden. Deshalb kann der Gesamtaufwand auch dann sinken, wenn eine Komponente mit negativem Nutzen entfernt wird und dadurch weitere Komponenten mit positivem Nutzen entfernt werden können. Umgekehrt besteht auch die Möglichkeit, daß das Entfernen einer Komponente das Entfernen weiterer Komponenten verhindert. Dieser Fall ergibt sich aus Bedingung (bs3) für das Entfernen von Planschritten: erreicht ein Planschritt r jeweils eine Vorbedingung für die Schritte $s_1 \ldots s_n$ in der Präsentation, so besagt die Bedingung, daß mindestens einer dieser Schritte *nicht* entfernt werden darf, sofern nicht auch r entfernt wird.

Analog zu Dale und Reiters Ergebnissen bei der Vereinfachung von referenziellen Ausdrücken [Dale und Reiter, 1995] kann also auch hier nicht erwartet werden, daß ein Algorithmus zum Entfernen von Komponenten aus einer Planpräsentation garantieren könnte, daß seine Lösungen optimal sind, sofern nicht *alle* zulässigen Kombinationen des Entfernens berücksichtigt werden. Optimal meint dabei, daß es keine andere Lösung gibt, bei der die Summe aus Aufwand für das Verstehen der Präsentation plus dem Rekonstruktionsaufwand geringer ist. Im anschließenden Abschnitt wird der in dieser Arbeit verwendete Algorithmus näher beschrieben. Er beruht darauf, zunächst sukzessive Komponenten aus der Präsentation zu entfernen. Dabei werden Komponenten bevorzugt, die die Entfernung anderer Komponenten nicht behindern.

Abschließend sei angemerkt, daß die Aufwandsschätzungen für das Rekonstruieren von Causal Links und Planschritten grob vergleichbar zur Abschätzung der Wichtigkeit dieser Komponenten in einer Planbeschreibung bei Youngs *Local Brevity* Verfahren sind. Der Wichtigkeitswert für einen Planschritt berücksichtigt dabei auch die Zahl der Causal Links, die *zu* dem Planschritt führen. Für den Ansatz in dieser Arbeit haben solche Causal Links nur indirekt Einfluß darauf, ob eine Komponente aus einer Planpräsentation entfernt wird, wenn dadurch die referenzielle Integrität der Präsentation verletzt werden könnte. Der Unterschied kann auf die unterschiedlichen Ansätze zurückgeführt werden, die der Bewertung, welche Komponenten aus einer Präsentation entfernt werden sollen, zugrunde liegen. Youngs Abschätzung der Wichtigkeit beruht auf empirischen Untersuchungen, wie gut sich Testpersonen an bestimmte Komponenten einer Erzählung oder Geschichte *erinnern* können. Es wird dabei nicht berücksichtigt, *warum* sich die Testpersonen gut erinnert haben und inwiefern die Ergebnisse auf einen Plan übertragbar sind. Wie Kieras und Bovair in [Kieras und Bovair, 1984] gezeigt haben, sind Testpersonen bei der Bedienung von technischen Geräten in der Lage fehlende

Erinnerung durch Schlußfolgerungen auszugleichen, die sie aufgrund ihrer Kenntnisse von der Funktionalität des Gerätes haben. Entsprechende Komponenten könnten demnach auch aus einer Planpräsentation entfernt werden. Dies steht aber im Gegensatz zu Youngs Ansatz, Komponenten, an die sich die Testpersonen gut erinnern, als besonders wichtig zu betrachten und deshalb auf jeden Fall in der Präsentation zu belassen. Der Ansatz den Aufwand abzuschätzen, den ein Benutzer für die Rekonstruktion von zuvor entfernten Komponenten hat, erscheint in diesem Licht sinnvoller.

4.4.3 Algorithmus für das Entfernen von Komponenten aus einer Planpräsentation

Der Algorithmus hat das Ziel den (geschätzten) Gesamtaufwand für die Präsentation eines Plans – wie im vorangegangen Abschnitt beschrieben – zu reduzieren, indem Komponenten die der Benutzer leicht schließen kann, aus einer vollständigen Planpräsentation entfernt werden. Wie bereits genannt, liegt die Schwierigkeit darin, daß sich durch die Bedingungen (bs2) und (bs3) für das Entfernen von Komponenten Abhängigkeiten ergeben. Ein einfacher Algorithmus, der sukzessive alle Komponenten aus der Präsentation entfernt, deren Aufwand für das Verstehen größer ist als für das Rekonstruieren, kann damit kein optimales Ergebnis liefern. Ein Vergleich aller Komponenten-Kombinationen, die nach den Bedingungen für das Entfernen von Komponenten zulässig wären, würde zwar eine optimale Lösung erlauben, wäre aber mit unakzeptabel hohem Berechnungsaufwand verbunden. Der in Abb. 4.3 vorgestellte Algorithmus wurde deshalb als Kompromiß zwischen Berechnungsaufwand und "Güte" der zu erwartenden Ergebnisse entworfen. Auf mögliche Varianten, die zu Lasten des Berechnungsaufwandes bessere Ergebnisse versprechen, wird noch kurz eingegangen.

Für die Beschreibung des Algorithmus benötigen wir zunächst eine Formalisierung der Abhängigkeiten zwischen den Komponenten eines Plans bzw. einer Planpräsentation. Die Definitionen beziehen sich grundsätzlich auf einen Plan P, sowie D, den aktuellen Zustand der Präsentation, die für P erzeugt wurde. Während P durch den Algorithmus nicht verändert wird, werden aus D im Laufe des Algorithmus Komponenten entfernt. Sowohl P als auch D werden im folgenden als Menge der darin enthaltenen Komponenten verwendet. Da keine Dekompositionen aus der Präsentation entfernt werden, beschränkt sich die Betrachtung auf Causal Links und Planschritte. Zu Beginn der Verarbeitung sind P und D gleich.

Die (Lösch-)Abhängigkeiten seien durch die folgenden Funktionen auf Elementen von P bzw. D definiert, die jeweils die Mengen der davon abhängigen Komponenten liefern:

$$
\begin{aligned}
\mathit{dep}(k) &:= \{s \mid (k=(r,l,s) \wedge k \in \mathrm{P}) \vee (k=(s,l,r) \wedge k \in \mathrm{P})\} \quad \cup \\
&\quad \{s \mid \exists l_1\ (k,l_1,s) \in \mathrm{P} \wedge \nexists l_2,t\ (k,l_2,t) \in \mathrm{P} \wedge s \neq t \wedge t \in \mathrm{D}\} \\
\mathit{trans\text{-}dep}(k) &:= \mathit{dep}(k) \cup \bigcup_{s \in \mathit{trans\text{-}dep}(k)} \mathit{dep}(s) \\
\mathit{poss\text{-}dep}(k) &:= \{s \mid (k=(r,l,s) \wedge k \in \mathrm{P}) \vee (k=(s,l,r) \wedge k \in \mathrm{P})\} \quad \cup \\
&\quad \{s \mid \exists l_1\ (k,l_1,s) \in \mathrm{P} \wedge \nexists l_2,t\ (k,l_2,t) \in \mathrm{P} \wedge s \neq t \wedge t \in \mathrm{NEK}\} \\
\mathit{excl\text{-}dep}(k) &:= \{ks \in \mathit{dep}(k) \mid \forall k'\ (k' \in \mathrm{D} \wedge ks \in \mathit{poss\text{-}dep}(k')) \rightarrow k' = k\} \\
\mathit{excl\text{-}set\text{-}dep}(k,g) &:= \{ks \in \mathit{dep}(k) \mid \forall k'\ (k' \in \mathrm{D} \wedge ks \in \mathit{poss\text{-}dep}(k')) \rightarrow k' \in g \cup \{k\}\}
\end{aligned}
$$

Funktion *dep* liefert die Menge der Komponenten aus P, die direkt von k abhängen. Die erste Teilmenge ergibt sich aus (bs2), die zweite aus (bs3). *trans-dep* liefert zusätzlich die Komponenten, die *indirekt* von k abhängig sind. *trans-dep* ist der transitive Abschluß von *dep*, wenn beide als Relationen auf Komponenten aufgefaßt werden. *poss-dep*(k) liefert alle Komponenten aus P, die *möglicherweise* von k abhängig sind. Zusätzlich zu *dep*(k) sind das die Komponenten, für die nicht ausgeschlossen werden kann, daß sie von k abhängig *werden*, wenn im Verlauf des Algorithmus Komponenten aus D entfernt werden. Die Bedingung in der zweiten Teilmenge der Definition fordert deshalb, daß von k

kein anderer Causal Link zu einer Komponente aus NEK ausgehen darf. NEK ist die Menge der Komponenten, die nicht aus D entfernt werden können. Diese Menge wird in Zeile (12) des Algorithmus mit den Komponenten initialisiert, die nach Bedingungen (bs1), (bs4), (bs5), (bs6) oder (bc1) nicht aus der Planpräsentation entfernt werden dürfen oder direkt oder indirekt von solchen Komponenten abhängig sind. Die Menge kann sich im dritten Schritt des Algorithmus vergrößern, wenn eine Komponente durch Prozedur *del-set* (s. Abb. 4.5) entfernt wird und eine andere dadurch von einer nicht-entfernbaren Komponente abhängig wird.

Die Funktion *excl-dep*(k) liefert die Komponenten aus P, die *exklusiv* von k abhängen; d.h. sie hängen direkt von k ab, aber nicht – auch nicht *möglicherweise* – von anderen Komponenten, die sich (noch) in D befinden. Entsprechendes gilt für *excl-set-dep*(k,g); allerdings dürfen die Komponenten dabei außer von k auch von Elementen der Komponentenmenge g abhängen.

1. Initialisierung

Menge der Komponenten, die nicht aus der Präsentation entfernt werden können

(11) NEI $:= \{k \in \mathrm{P} \mid k$ darf nach (bs1), (bs4), (bs5), (bs6) oder (bc1) *nicht* aus D entfernt werden$\}$

(12) NEK $:=$ NEI $\cup \bigcup_{k \in \mathrm{NEI}}$ *trans-dep* (k)

Agenda (Ag)

(13) **for each** $cl \in$ *causal-links*(P) \ NEK **do** *check-del* (cl)

2. Entfernen von Komponenten gemeinsam mit exklusiv abhängigen Komponenten

(21) $Ag' := \{\}$
(22) **while** $Ag \neq Ag'$ **do**
(23) $\quad Ag' := Ag$
(24) $\quad$ **for each** $k \in Ag$ **do**
(25) $\quad\quad$ **if** *gain*(*adv-set*$(k,\{\})) \geq 0$ **then**
(26) $\quad\quad\quad$ G $:=$ *del-cand*(*adv-set*$(k,\{\})$); $Ag := Ag \setminus \{k\}$; D $:=$ D \ G
(27) $\quad\quad\quad$ $Ag := Ag \cup (\mathrm{D} \cap \bigcup_{ks \in \mathrm{G}}$ *excl-dep*$(ks))$

3. Berücksichtigung aller nicht-abhängigen Komponenten

(30) $Ag :=$ D \ (NEK $\cup \bigcup_{k \in \mathrm{D}}$ *dep*(k))
(31) $Ag' := \{\}$
(32) **while** $Ag \neq Ag'$ **do**
(33) $\quad Ag' := Ag$
(34) $\quad$ **for each** $k \in Ag$ **do**
(35) $\quad\quad$ **if** *gain*(*adv-set*$(k,\{\})) \geq 0$ **then**
(36) $\quad\quad\quad$ *del-set*$(k,$ *del-cand*(*adv-set*$(k,\{\}))$)

Abbildung 4.3: Algorithmus für das Entfernen von Komponenten aus einer Planpräsentation

Nach der bereits genannten Initialisierung der Menge der nicht-entfernbaren Komponenten, wird die Agenda (Ag) aufgebaut. Die Agenda ist eine globale Datenstruktur, die die Menge der Komponenten repräsentiert, die zwar nach allen Bedingungen für das Entfernen von Komponenten gelöscht werden könnten, für die sich aber zum aktuellen Verarbeitungsstand der Aufwand nicht lohnt; d.h. deren Nutzen negativ ist. Beim Aufbau der Agenda wird dabei nur der einzelne Nutzen der Komponenten berücksichtigt. Der Aufbau erfolgt durch die Hilfsfunktion *check-del* (vgl. Abb. 4.4). Die

Funktion wird in Schritt (13) für jeden Causal Link des Plans aufgerufen, der nach (bc1) entfernt werden darf. Die Causal Links sind die einzigen Komponenten, die zu Beginn der Verarbeitung überhaupt entfernt werden können. Planschritte dürfen wegen (bs2) zunächst nicht entfernt werden, weil von jedem Planschritt, der zum Plan beiträgt, ein Causal Link ausgehen muß. *check-del* fügt Causal Links mit negativem Nutzen zur Agenda hinzu (c2) und entfernt die anderen aus der Planpräsentation (c4). Dies ist ohne weiteres möglich, da das *Entfernen* eines Causal Links nicht das Entfernen anderer Komponenten beeinträchtigen kann. Das gilt in gleicher Weise für Komponenten, die exklusiv von einer gerade entfernten Komponente abhängen. Hat die Funktion eine Komponente entfernt, wird sie daher rekursiv für alle exklusiv davon abhängenden Komponenten aufgerufen (c5), um auch diese zu entfernen oder zur Agenda hinzuzufügen. Die Terminierung der Rekursion ist sichergestellt, weil die Abhängigkeits-Relation zyklenfrei ist. Dies wiederum ergibt sich daraus, daß nur Planschritte untereinander abhängig sein können und das nur dann, wenn sich ein Causal Link zwischen ihnen befindet. In einem korrekten Plan bilden die Causal Links aber keine Zyklen.

check-del (k:plan-component)

(c1) **if** $k \notin$ NEK **then**
(c2) **if** *Nutzen*(k) < 0 **then** $Ag := Ag \cup \{k\}$
(c3) **else**
(c4) D := D \ $\{k\}$
(c5) **for each** $ks \in$ D $\cap$ *excl-dep*(k) **do** *check-del* (ks)

Abbildung 4.4: Funktion *check-del*

Nach der Initialisierung wird sukzessive geprüft (24,25), ob es sinnvoll ist, die Elemente der Agenda gemeinsam mit davon abhängigen Komponenten zu entfernen (26). Dies ist immer dann der Fall, wenn der Gesamtnutzen der abhängigen Komponenten so groß ist, daß er den negativen Nutzen der Agenda-Komponente aufwiegt. Die Funktion *adv-set* berechnet sowohl den Gesamtnutzen der abhängigen Komponenten als auch die Menge dieser Komponenten selbst. Die Hilfsfunktionen *gain* und *del-cand* ermöglichen den Zugriff auf die beiden Ergebnisse. Auf die Arbeitsweise der Funktion wird weiter unten eingegangen. An dieser Stelle ist wichtig, daß *adv-set* keine Komponenten berücksichtigt, die von anderen Agenda-Komponenten abhängig sind. Dadurch wird verhindert, daß der Nutzen von abhängigen Komponenten mehreren Komponenten zugeschlagen wird. Außerdem können die Komponenten, die *adv-set* ermittelt hat, entfernt werden, ohne das Entfernen anderer Komponenten zu beeinträchtigen. Komponenten, die von einer der durch *adv-set* ermittelten Komponenten exklusiv abhängen, aber nicht mit entfernt wurden, haben einen negativen Nutzen. Sie werden deshalb in Zeile (27) zur Agenda hinzugenommen.

Das Entfernen von Komponenten kann dazu führen, daß sich neue exklusive Abhängigkeiten ergeben, bspw. wenn eine Komponente a von genau zwei weiteren Komponenten b und c (nicht-exklusiv) abhängt und b entfernt wird. Durch neue exklusive Abhängigkeiten kann *adv-set* mehr Komponenten berücksichtigen und deshalb evtl. einen höheren Gesamtnutzen berechnen. Aus diesem Grund wird dieser zweite Schritt des Algorithmus solange wiederholt, bis sich die Agenda nicht mehr verändert (22). Die Terminierung der Iteration ist sichergestellt, weil jede Komponente des Plans maximal einmal zur Agenda hinzugenommen werden kann und jeder Iterationsschritt (bis auf den letzten) mindestens eine Komponenten aus der Agenda entfernt.

Nach dem zweiten Schritt des Algorithmus sind die Komponenten aus der Präsentation entfernt, für die sich das Entfernen lohnt und nicht das Entfernen anderer Komponenten behindern kann. Der dritte Schritt arbeitet in gleicher Weise wie der zweite, allerdings werden nun bei der Berechnung des Nutzens auch solche Komponenten berücksichtigt, deren Entfernen das Entfernen anderer Kom-

ponenten verhindern kann. In Zeile (30) werden deshalb alle Komponenten aus D auf die Agenda gesetzt, die weder zu den nicht-entfernbaren Komponenten gehören noch von anderen Komponenten aus D abhängig sind. Das Entfernen der Komponenten ist nun aufwendiger und ist deshalb in Prozedur *del-set* in Abb. 4.5 zusammengefaßt.

del-set (*k*:plan-component, G:set-of-plan-components)

(s1) $Ag := Ag \setminus \{k\}$, $\mathrm{D} := \mathrm{D} \setminus \mathrm{G}$
(s2) **for each** $ka \in \{s \in \mathrm{D} \mid k \in \textit{poss-dep}(s)\}$ **do**
(s3) $\quad \mathrm{Ra} := \mathrm{D} \cap \textit{poss-dep}(ka)$
(s4) $\quad$ **if** $|\,\mathrm{Ra}\,| = 1$ **then**
(s5) $\quad\quad Ag := Ag \setminus \mathrm{Ra}$
(s6) $\quad\quad$ **if** $ka \in \mathrm{NEK}$ **then** $\mathrm{NEK} := \mathrm{NEK} \cup \mathrm{Ra} \cup \textit{trans-dep}(\textit{first}(\mathrm{Ra}))$
(s7) $Ag := Ag \cup (\,\mathrm{D} \cap \bigcup_{ks \in \mathrm{G}} \textit{dep}(ks)) \setminus (\bigcup_{kv \in \mathrm{D}} \textit{dep}(kv))$

Abbildung 4.5: Prozedur *del-set*

Der wesentliche Unterschied gegenüber der Vorgehensweise in den Zeilen (26, 27) besteht darin, daß nun berücksichtigt wird, daß sich die Menge der nicht-entfernbaren Komponenten verändern kann. Dieser Fall kann auftreten, wenn eine Agenda-Komponente *k* entfernt wird, die *möglicherweise* von einer anderen Komponente *ka* aus D abhängig war. Diese Variante der Abhängigkeit besteht, wenn von *ka* Causal Links sowohl zu *k* als auch zu anderen Komponenten ausgehen. Besteht von *ka* nach dem Entfernen von *k* nur noch ein einziger Causal Link zu einer Komponente aus D, ist für diese Komponente die Bedingung für die Abhängigkeit von *ka* erfüllt. Die Komponente wird deshalb von der Agenda ausgeschlossen (s5). Ist *ka* eine nicht-entfernbare Komponente, so gilt dies nun auch für die (neu) davon abhängige Komponente sowie alle Komponenten, die davon direkt oder indirekt abhängig sind. Dementsprechend werden diese Komponenten in (s6) zu NEK hinzugefügt. Die Hilfsfunktion *first* liefert das erste (hier einzige) Element einer Menge. Es sei angemerkt, daß *nur* das Entfernen einer Agenda-Komponente dazu führen kann, daß sich die Menge der nicht-entfernbaren Komponenten verändern kann, weil andere Komponenten nur dann entfernt werden, wenn sie exklusiv abhängig sind, also nicht noch *möglicherweise* von einer anderen Komponente. Die letzte Zeile der Prozedur entspricht wiederum dem Vorgehen im zweiten Schritt des Algorithmus (Zeile (27)). Zur Agenda werden hier allerdings nicht nur Komponenten hinzugenommen, die von einer der entfernten Komponenten *exklusiv* abhängig sind. Die Komponenten dürfen von anderen Komponenten aber höchstens *möglicherweise* abhängig sein.

Die in Abb. 4.6 gezeigte Funktion *adv-set* wurde im zweiten und dritten Schritt des Algorithmus eingesetzt, um den Gesamtnutzen einer Komponente, die als erster Funktionsparameter (*k*) angegeben werden muß, einschließlich der davon abhängigen Komponenten zu berechnen. Beginnend mit Zeile (a2) iteriert die Funktion über alle (max. zwei) Komponenten die direkt von *k* abhängen. Um zu verhindern, daß der Nutzen einer abhängigen Komponente mehrfach verwendet wird oder das Entfernen von Komponenten das Entfernen anderer Komponenten behindert, werden auch hier wieder nur die *exklusiv* abhängigen Komponenten berücksichtigt. Allerdings ist es nicht erforderlich, daß die Komponenten exklusiv von *k* abhängen. Es reicht vielmehr aus, wenn sie ausschließlich von der Menge der Komponenten abhängen, die gemeinsam entfernt werden. Diese Menge (*del-cand*) kann der Funktion als zweiter Parameter mitgegeben werden, wenn durch vorangegangene Verarbeitungsschritte bereits Elemente ermittelt wurden. In Zeile (a1) wird dieser Menge *k* hinzugefügt. Danach wird die Menge jeweils um solche Komponenten ergänzt, deren Entfernen einen positiven Nutzen erbringt. Sowohl dieser Nutzen als auch die ergänzte Menge *del-cand* werden wiederum durch rekur-

sive Aufrufe der Funktion berechnet und in (a4, a5) bzw. (a8, a9) den beiden Gesamtergebnissen der Funktion zugeschlagen.

adv-set (*k*:plan-component, *del-cand*:set-of-plan-components)

(a1) *del-cand* := *del-cand* ∪ {*k*}, *gain* := *Nutzen*(*k*)
(a2) **for each** *ks* ∈ (D ∩ *excl-set-dep*(*k*, *del-cand*)) \ *del-cand* **do**
(a3) **if** *Nutzen*(*ks*) ≥ 0 **then**
(a4) *gain* := *gain* + *gain*(*adv-set*(*ks*, *del-cand*))
(a5) *del-cand* := *del-cand*(*adv-set*(*ks*, *del-cand*))
(a6) **for each** *ks* ∈ (D ∩ *excl-set-dep*(*k*, *del-cand*)) \ *del-cand* **do**
(a7) **if** *gain*(*adv-set*(*ks*, *del-cand*)) ≥ 0 **then**
(a9) *gain* := *gain* + *gain*(*adv-set*(*ks*, *del-cand*))
(a8) *del-cand* := *del-cand*(*adv-set*(*ks*, *del-cand*))

Abbildung 4.6: Funktion *adv-set*

Die zweite Iteration ab Zeile (a6) ist weitgehend identisch zur ersten ab Zeile (a2). Durch diese Aufteilung der von *k* abhängigen Komponenten in solche, deren Nutzen einzeln größer oder gleich bzw. kleiner Null ist, wird erreicht, daß zuerst die Komponenten bestimmt werden, deren Entfernen auf jeden Fall nützlich ist. Die Reihenfolge des Entfernens spielt hier deshalb eine Rolle, weil eine Komponente *ks* von mehreren der gemeinsam zu entfernenden Komponenten abhängen kann. *ks* darf aber nur dann entfernt, und der damit verbundene Nutzen verwendet werden, wenn *alle* diese Komponenten entfernt werden. Ob das so ist, kann erst mit der Entscheidung über das Entfernen der letzten dieser Komponenten festgestellt werden. Der Nutzen kann deshalb auch nur bei der Berechnung des Gesamtnutzens für diese letzte Komponente berücksichtigt werden. Gehen wir zum Beispiel davon aus, daß *adv-set*(a,{}) für die Teilstruktur in Abb. 4.7 berechnet werden soll und daß *Nutzen*(b) < 0, *Nutzen*(c) > 0, (*Nutzen*(b) + *Nutzen*(d)) > 0. Wäre die Reihenfolge der Überprüfung der von a abhängigen Komponenten $b \rightarrow c \rightarrow d$, so würde sich ergeben, daß b nicht entfernt werden kann, weil der einzelne Nutzen dafür nicht ausreicht und keine Komponente ausschließlich von $\{b, a\}$ abhängt. Es kann also lediglich Komponente c entfernt werden, obwohl das erlaubte gemeinsame Entfernen von b, c, d insgesamt einen Nutzen bringen würde. Werden – wie in *adv-set* – zunächst die Komponenten bearbeitet, deren Nutzen einzeln ≥ 0 ist, vergrößert sich die Menge der Komponenten, die auf jeden Fall entfernt werden können. Von dieser Menge hängt wiederum ab, ob der Nutzen abhängiger Komponenten berücksichtigt werden darf. Im Beispiel kann b dann im zweiten Lauf entfernt werden, weil inzwischen *del-cand* = $\{a, c\}$ ist und der rekursive Aufruf *adv-set* $(b, \{a, c\})$ den Nutzen von d mit berücksichtigt.

Der Algorithmus behandelt allerdings keine Fälle, die das gemeinsame Entfernen von mehr als einer Komponente mit negativem Nutzen erfordern. Im Beispiel würde keine der Komponenten entfernt, wenn zwar der Gesamtnutzen *Nutzen*(b) + *Nutzen*(c) + *Nutzen*(d) > 0, aber der Einzelnutzen von b und c jeweils negativ wären. Eine systematische Berücksichtigung müsste alle Kombinationen von entfernbaren Komponenten auf ihren Nutzen hin untersuchen. Da der Berechnungsaufwand dafür exponentiell von der Anzahl der entfernbaren Komponenten abhängt, werden solche Mehrfachabhängigkeiten weder in dieser Funktion noch im Haupt-Algorithmus berücksichtigt.

Der Algorithmus verfolgt die Strategie, zunächst (im Initialisierungs- und im zweiten Schritt) Komponenten zu entfernen, deren Entfernen keine negativen Auswirkungen auf das Entfernen anderer Komponenten hat. Die Reihenfolge, in der Komponenten entfernt werden, ist damit irrelevant. In der Initialisierungsphase werden dabei bereits mit geringem Berechnungsaufwand Komponenten entfernt. Dadurch wird die Zahl der Komponenten verringert, die noch entfernt werden können und

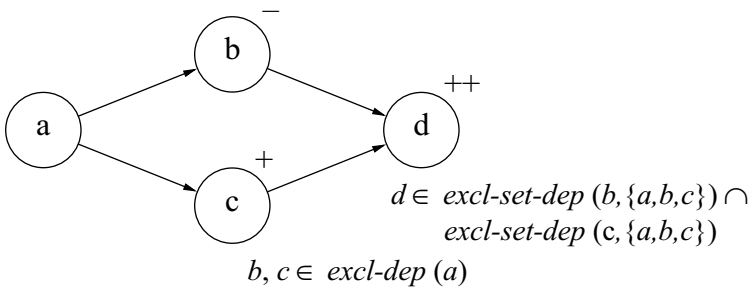

Abbildung 4.7: Beispiel einer Komponente, die von mehreren anderen abhängig ist.

im zweiten Schritt durch ein aufwendigeres Verfahren bearbeitet werden. Auch durch den zweiten Schritt verringert sich wiederum die Zahl der Komponenten, die noch entfernt werden können. Erst das Entfernen von Komponenten im dritten Schritt kann das Entfernen weiterer Schritte verhindern. Ein optimales Gesamtergebnis, d.h. ein geringst möglicher Gesamtaufwand, läßt sich deshalb nur dann erzielen, wenn die Komponenten in der optimalen Reihenfolge entfernt werden. Die Berechnung einer solchen Reihenfolge ist allerdings nicht einfacher als die Berechnung der optimalen Menge zu entfernender Komponenten und wird deshalb hier nicht in Betracht gezogen. Da auch andere einfache Kriterien für Bearbeitungsreihenfolgen – bspw. der Gesamtnutzen der Komponenten – nicht garantieren können, daß die Reihenfolge optimal ist, wird auf die Berechnung solcher Kriterien verzichtet.

Als Ergänzung zu der hier vorgeschlagenen Top-Down-Strategie, die von den Komponenten ausgeht die entfernt werden dürfen, wäre auch der umgekehrte Weg möglich. Dabei könnte von den Komponenten mit positivem Einzel-Nutzen ausgegangen werden, auch wenn sie von anderen Komponenten abhängig sind. Um eine solche Komponente entfernen zu können, müssten alle Komponenten von denen sie direkt oder indirekt abhängig ist mit entfernt werden. Für die Entscheidung, ob dies zu einer Verbesserung der Präsentationsbewertung beiträgt, müsste die Summe der Werte für den einzelnen Nutzen dieser Komponenten gebildet werden. Hier wird allerdings auf diese Möglichkeit verzichtet. Der Grund dafür liegt in der Einschätzung, daß es nur selten vorkommen dürfte, daß Komponenten nicht entfernt werden, weil im Schritt drei eine ungünstige Reihenfolge für die Verarbeitung gewählt wurde. Die Einschätzung wird einerseits durch das im Anschluß besprochene Beispiel anhand des vereinfachten Plan aus Abb. 4.1 gestützt. Im diesem Beispiel werden alle Komponenten, die entfernt werden können, bereits in der Initialisierungsphase entfernt. Für den vollständigen Plan, der in Kap. 5.3.2 *Benutzerspezifische Pläne* gezeigt wird, ist dies im zweiten Schritt der Fall. Die folgende Überlegung zeigt zudem, daß kaum zu erwarten ist, daß ein Plan Abhängigkeitsstrukturen aufweisen wird, bei dem der vorgestellte Algorithmus ein großes Potential zur Verringerung des Gesamtaufwandes ungenutzt läßt. Eine Gruppe von Abhängigkeiten besteht von Causal Links zu Planschritten. Da Causal Links bei der Präsentation hoch bewertet werden und oft leicht zu rekonstruieren sind, haben sie oft einen positiven Nutzen und können deshalb ohne die Berücksichtigung weiterer Komponenten entfernt werden, wenn der Benutzer das erforderliche Wissen für die Rekonstruktion hat. Abhängigkeiten zwischen Planschritten bestehen entlang der Causal Links; d.h. weisen also grundsätzlich in Richtung Planende. Abhängige Planschritte haben deshalb alleine durch ihr späteres Vorkommen im Plan einen erhöhten Rekonstruktionsaufwand. Da der Aufwand für das Verstehen eines Planschritts andererseits gering ist, dürfte der Nutzen für das Entfernen meist nur gering sein. Dies gilt um so mehr, wenn ein Planschritt über mehrere Stufen indirekt von anderen abhängig ist. Selbst wenn der Algorithmus solche Beiträge nicht berücksichtigen sollte, können sie keinen wesentlichen Einfluß auf den Gesamtaufwand für die Präsentation haben.

Es sei noch angemerkt, daß der Algorithmus eine Begrenzung enthält die verhindern soll, daß die Zahl der präsentierten Planschritte unter eine vorgebbare Zahl sinkt. Ohne eine solche Begrenzung könnte es bei sehr einfachen, kurzen Plänen dazu kommen, daß alle Planschritte entfernt werden und dem Benutzer überhaupt kein Plan präsentiert wird. Aus Übersichtsgründen wurde auf eine detaillierte Darstellung verzichtet.

Als Beispiel für die Anwendung des Algorithmus sei auf den Beispielplan aus Abb. 4.1 S. 174 zurückgegriffen. Um den Algorithmus anwenden zu können, benötigen wir zunächst einen Wert für k_{check}. Youngs Untersuchungen haben gezeigt, daß die Rekonstruktion von Komponenten für einen Benutzer oft einfacher ist als das Lesen der entsprechenden Beschreibung. Da für die Rekonstruktion nach dem o.g. Vorgehen oft mehrere Basiseinheiten für das Überprüfen erforderlich sind, sei k_{check} mit 0,3 angenommen.

Wir gehen für das Beispiel von einem Benutzer mit umfassendem Wissen im Planungssachbereich aus; d.h. dem Benutzer wird das gleiche Wissen zugeschrieben wie dem System. Als nicht-entfernbare Komponenten können dann die Schritte *ftp-dnload-bin* und *open-file-tpx* festgestellt werden. *open-file-tpx* ist von *ftp-dnload-bin* abhängig, weil von *ftp-dnload-bin* ausschließlich Causal Links zu *open-file-tpx* ausgehen. *ftp-dnload-bin* darf wg. Bedingung (bs4) nicht entfernt werden, weil eine Vorbedingung von *open-file-tpx* (Datei im lokalen Dateisystem) auch durch *Mount* erreicht werden kann.

Der einzelne Nutzen der Causal Links wird aus dem geschätzten Aufwand für das Verstehen (einheitlich $k_{cl} * k_{pres}$ mit $k_{cl} = 2$) abzüglich des Rekonstruktionsaufwandes nach Formel (2) des vorangegangenen Abschnitts berechnet. Da Erzeuger- und Verbraucher-Schritte höchstens einen Abstand von 3 Positionen haben, ist dieser maximal 3 * 0,3 * k_{pres} = 0,9 * k_{pres}; d.h. der einzelne Nutzen jedes Causal Links ist positiv. Die Causal Links werden also alle in der Initialisierungsphase entfernt. Der Nutzen für die Planschritte berechnet sich aus k_{pres} abzüglich des Rekonstruktionsaufwandes nach Formel (4). Dies sind im einzelnen für die entfernbaren Komponenten

launched(*ftp-prg*) (0 + 1/2) * 0,3 * k_{pres} = 0,15 * k_{pres}
launched(*txt-proc*) (0 + 2/2) * 0,3 * k_{pres} = 0,3 * k_{pres}
ftp-bin-mode (0 + 3/2) * 0,3 * k_{pres} = 0,45 * k_{pres}

Der einzelne Nutzen ist also auch für diese Planschritte positiv. Die Komponenten sind außer von den Causal Links nicht von weiteren Komponenten abhängig. Mit dem Entfernen des Causal Links zwischen *launched*(*ftp-prg*) und *ftp-bin-mode* werden durch einen rekursiven Aufruf von *check-del* auch diese beiden Schritte mit entfernt. In gleicher Weise wird *launched*(*txt-proc*) entfernt, nachdem der Causal Link zwischen diesem Schritt und *open-file-tpx* entfernt wurde.

4.4.4 Diskussion

In diesem Abschnitt wurden Verfahren vorgeschlagen, wie eine Planpräsentation auf den für einen Benutzer relevanten Inhalt beschränkt werden kann. Das heißt, daß aus einer vollständigen Präsentation die Komponenten entfernt werden, die der Benutzer selbst leicht schließen kann und die deshalb für ihn redundant sind. Wenngleich kaum Zweifel daran bestehen, daß redundante Information in einer Präsentation vermieden werden sollte[32], fehlen empirische verifizierte Modelle, aus denen sich präzise ableiten ließe, welche Komponenten einer Planpräsentation unter welchen Umständen *leicht* von einem Benutzer geschlossen werden können, bzw. die sogar eine Abwägung der Aufwände für das Verstehen und eine Rekonstruktion ermöglichen würden. Auch das hier vorgestellte Modell ist nicht

[32] vgl. allerdings Walker, die diese Position in [Walker, 1996] relativiert, indem sie Fälle aufzeigt, in denen redundante Information durchaus nützlich ist.

empirisch verifiziert, weil die notwendigen Untersuchungen weit über den Rahmen dieser Arbeit hinausgehen würden. Es stellt also – zumindest in Bezug auf die Bewertung des Aufwandes – lediglich eine Arbeitshypothese dar. Dies gilt auch für die Abschätzung der Konstanten k_{cl} und k_{check} sowie die Festlegung, daß einige einfache Überprüfungen nur hälftig in den Gesamtaufwand eingehen. Young verwendet bei seinem Ansatz für alle Einzelaspekte explizite Konstanten, deren Wert er ebenfalls annimmt. Dabei erscheint Youngs Vermutung plausibel, daß solche Konstanten nicht universell festliegen, sondern von der konkreten Anwendung abhängen. Insbesondere dürften sie von der Art der Darstellung abhängen. Beispielsweise ist der Aufwand für eine Präsentation eines Causal Links in einer textuellen Darstellung verhältnismäßig hoch, während er in einer grafischen Darstellung (bspw. wie in Abb. 4.1) kaum ins Gewicht fällt. Auch dürfte es eine Rolle spielen, ob die Planpräsentation während der Planausführung – quasi als Gedächtnisstütze – zur Verfügung steht. In einem solchen Fall ist die Länge der Präsentation weniger wichtig, weil sich der Benutzer auf Teile der Präsentation konzentrieren kann und während des Fortschritts alte Information vergessen und neue aufnehmen kann.

Weitgehend unabhängig von den genauen Bewertungen, kann die Wirkung des hier vorgestellten Modells tendenziell abgeschätzt werden:

- Causal Links werden sehr häufig entfernt, wenn der Benutzer das notwendige Wissen zur Rekonstruktion hat, weil ihr Präsentationaufwand mit k_{cl} (=2) Basiseinheiten verhältnismäßig groß ist, sie aber oft leicht rekonstruiert werden können. Dazu passen die Untersuchungen von Dixon [Dixon, 1982] der feststellt, daß Aktionen bei Instruktionstexten eine wichtigere Rolle spielen als die Effekte, die sie erreichen. Es widerspricht auch nicht der zu Anfang des Kapitels herausgestellten Wichtigkeit von Causal Links für das Verstehen des Plans und das Weiterplanen, denn es werden keine Causal Links entfernt, wenn der Benutzer nicht das notwendige Wissen für deren Rekonstruktion hat.

- Für die Rekonstruktion eines Planschrittes ist zumeist ein höherer Aufwand erforderlich. Dies gilt insbesondere, wenn der Planschritt mehrere Effektkomponenten hat, die im Plan zwar nicht gebraucht werden, die aber berücksichtigt werden müssen, weil sie bestehende Causal Links bedrohen könnten. Da der Aufwand für die Präsentation eines Planschrittes vergleichsweise gering ist, lohnt ein Entfernen eines Planschritts nur, wenn dieser sehr wenige Effektkomponenten hat. Außerdem wird der Aufwand für die Rekonstruktion zum Ende des Plans hin höher. Damit lohnt das Entfernen eines Planschritts ab einer bestimmten Position im Plan überhaupt nicht mehr. Dies erscheint zunächst unbefriedigend, ist allerdings in der Tatsache begründet, daß vor der Rekonstruktion eines Planschritts zunächst geklärt werden muß, ob nicht ein anderer, weiter vorn im Plan befindlicher Schritt eine offene Bedingung erfüllen kann. Dabei müssen um so mehr Schritte berücksichtigt werden, je weiter der zu rekonstruierende Schritt vom Plananfang entfernt ist.

- Abgesehen vom wachsenden Aufwand für das Rekonstruieren von Planschritten zum Ende des Plans hin, ist die Wirkung des Modells unabhängig von der Planlänge gegeben. Dies wird erreicht, indem für jede einzelne Komponente abgewogen wird, ob der Aufwand für die Rekonstruktion der Komponente für den Benutzer geringer ist, als der für die Präsentation.

Völlig unabhängig von der Aufwandsabschätzung ist die Berücksichtigung des Benutzerwissens. Die Bedingungen an das Entfernen von Causal Links oder Planschritten stellen unabhängig vom Aufwand für eine Rekonstruktion sicher, daß keine Komponenten aus einer Präsentation entfernt werden, die der Benutzer nicht wieder herstellen kann, weil er nur mangelhaftes oder falsches Wissen hat.

4.5 Optimierte Plangenerierung

Bis hierher wurden die Prozesse zur Generierung eines benutzerspezifischen Plans und zur Erzeugung einer benutzerspezifischen Präsentation dieses Plans separat behandelt. Dies erleichtert die systematische Beschreibung der Eigenschaften beider Prozesse und die Realisierung der Prozesse durch eigenständige Programmodule, die unabhängig voneinander weiterentwickelt werden können. Dennoch sind die Prozesse nicht völlig unabhängig. In Kapitel 4.3 *Inhaltsbestimmung und Bewertung einer Planpräsentation* wurde gezeigt, daß die Möglichkeit besteht, daß die benutzerspezifische Präsentation eines Plans scheitern kann. Ein solcher Plan ist damit für den Benutzer unbrauchbar, obwohl er ihn im Prinzip ausführen könnte. Er muß deshalb zur Generierung einer Alternative an den Planungsprozeß zurückgewiesen werden. Eine solche Rückweisung kann auch sinnvoll sein, wenn eine Präsentation zu langwierig wäre, weil der Benutzer viele Schritte des Plans nicht kennt und deshalb viele Erklärungen benötigt. In Abschnitt 4.3.2 *Gesamtbewertung einer Planpräsentation* wurde aber auf die Schwierigkeit hingewiesen, einen sinnvollen Schwellwert für die Bewertung eines Plans anzugeben und stattdessen ein Vergleich alternativer Pläne vorgeschlagen. Ständen alle Pläne zur Verfügung, die dem Benutzer ermöglichen seine Ziele zu erreichen, könnte der optimale Plan, d.h. der mit der besten Bewertung, ausgewählt und dem Benutzer präsentiert werden. Allerdings dürfte ein solches Vorgehen am viel zu hohen Berechnungsaufwand scheitern, da bereits die Generierung eines einzelnen Plans sehr aufwendig ist.

In diesem Abschnitt werden deshalb Möglichkeiten behandelt, wie von vorn herein ein möglichst "guter" Plan erzeugt werden kann. Damit ist ein Plan gemeint, der nicht nur im Prinzip vom Benutzer ausgeführt werden kann, sondern der auch für ihn nutzbar ist (d.h. für den der Präsentationsprozeß nicht scheitert) und dessen Bewertung einen möglichst guten (d.h. hier kleinen) Wert ergibt. In die Bewertung können dabei neben dem geschätzten Aufwand des Benutzers für das Verstehen der Präsentation auch die in Abschnitt 4.3.5 *Alternative Modelle zur Planbewertung* genannten Kriterien zur Planbewertung einfließen und – zumindest teilweise – die im vorangegangenen Abschnitt behandelten Benutzerinferenzen berücksichtigt werden. Eine Rückweisung aufgrund eines gescheiterten Präsentationsprozesses ist für einen solchen Plan ausgeschlossen. Ein Bewertungs-Schwellwert, dessen Überschreitung zur Rückweisung des Plans führen könnte, erscheint damit nicht mehr sinnvoll, weil der Generierungsprozeß Pläne mit guter Bewertung bevorzugt und nicht erwartet werden kann, daß ein erneuter Generierungsprozeß einen (deutlich) besseren Plan hervorbringt.

4.5.1 Nicht nutzbare Plan-Concepts

Die erste Verbesserung des Planungsprozesses, die garantieren soll, daß ein erzeugter Plan auch präsentiert werden kann, ist leicht zu realisieren. Das Scheitern des Präsentationsalgorithmus aus Abb. 4.2 S. 189 tritt genau dann auf, wenn der Plan ein für den Benutzer nicht nutzbares Plan-Concept enthält. Ein Verzicht auf die Verwendung nicht nutzbarer Plan-Concepts bei der Plangenerierung schießt diesen Fall aus. Damit drängt sich die Frage auf, ob es dann nicht sinnvoll ist, alle nicht nutzbaren Plan-Concepts aus dem Handlungspotential des Benutzers zu entfernen. Dafür spräche, daß damit eine Sonderbehandlung vermieden würde und es für die Erzeugung eines Plans keinen Unterschied macht, ob der Benutzer aufgrund seiner Fähigkeiten und Befugnisse nicht in der Lage ist ein Plan-Concept zu verwenden oder ob er es nicht verwenden kann, weil er nicht weiß *wie* er es ausführen muß und das System ihm dabei auch keine Hilfestellung geben kann. Dagegen spricht allerdings, daß so die Modellierung der veränderlichen Benutzerkenntnisse und seiner (im Modell) unveränderlichen Fähigkeiten und Befugnisse vermischt werden. Im Abschnitt 4.2.2.2 *Nutzbares Planungswissen* wurde im Zusammenhang mit der Definition der Nutzbarkeit eines Plan-Concepts ein Beispiel genannt, bei dem sich die Ansicht über die Benutzerkenntnisse, nicht aber die über seine Befugnisse verän-

dert haben. Wichtig ist dabei, daß eine Veränderung der Ansichten über die Nutzbarkeit eines Plan-Concepts auch die Nutzbarkeit anderer Plan-Concepts nach sich ziehen kann. Dies ändert aber nichts an den Fähigkeiten oder Befugnissen des Benutzers. Bspw. kann unter Windows ein freigegebenes Verzeichnis eines fremden Rechners durch einen Befehl in einem DOS-Fenster gemountet werden. Nimmt das System nun an, daß dem Benutzer das Öffnen solcher Fenster untersagt ist, ist nicht nur das dazu gehörende Plan-Concept nicht nutzbar, sondern auch das Mounten eines Verzeichnisses. Würde dieses Plan-Concept aus dem Handlungspotential entfernt, dürfte bei einer späteren Änderung der Ansicht über die Benutzerbefugnisse über das Öffnen eines DOS-Fensters nicht automatisch auch das Mounten zum Handlungspotential hinzugenommen werden. Der Grund liegt darin, daß zu diesem Zeitpunkt nicht unterschieden werden könnte, ob das Mounten nur deshalb nicht im Handlungspotential enthalten ist, weil angenommen wurde, daß der Benutzer kein DOS-Fenster öffnen durfte, oder ob der Grund vielmehr in einer mangelnden Befugnis für das Mounten liegt.

Da sich die Ansichten über die Kenntnisse und das Handlungspotential des Benutzer zwar durch Beobachtung, nicht aber durch geplantes Handeln ändern können, wird unmittelbar vor der Verwendung eines Plan-Concepts für die Erzeugung des Plans geprüft, ob das Plan-Concept nutzbar ist. Im negativen Falle wird das Plan-Concept dann nicht verwendet. Im positiven Fall kann es im Plan eingesetzt werden, ohne daß die Gefahr besteht, daß der Plan dadurch nicht präsentiert werden könnte.

4.5.2 Steuerung des Planungsprozesses

Viele Planungssysteme ermöglichen die Steuerung bzw. Beeinflussung der Entscheidungen des Planungsprozesses. Die Zielsetzung ist dabei, den Planungsprozeß so zu leiten, daß dieser möglichst schnell (irgend-) eine Lösung des Planungsproblems findet. UCPOP, das Planungssystem, das in dieser Arbeit eingesetzt wird, ermöglicht mehrere Eingriffsmöglichkeiten (vgl. [Barrett *et al.*, 1995]), die auch für eine Bevorzugung von Plänen mit möglichst guter Präsentationsbewertung genutzt werden können.

Beim in UCPOP eingesetzten *Planen durch Planverfeinerung* wird der Planungsprozeß mit einer Aufgabenliste initialisiert, die nur einen *Startplan* enthält[33]. Dieser Plan besteht nur aus einem Start- und einem Zielschritt. Der Startschritt hat eine leere Vorbedingung und eine Beschreibung der Startsituation als Effekt. Der Zielschritt hat eine Beschreibung des Ziels als Vorbedingung und einen leeren Effekt. Der Planungsprozeß bearbeitet seine Aufgabenliste, indem er den ersten Plan(kandidaten) der Liste entnimmt und prüft, ob es ein vollständiger Plan ist. Vollständig heißt (i.w.), daß es zu jeder Vorbedingungskomponente jedes Planschritts einen Causal Link geben muß und kein Causal Link bedroht sein darf (vgl. wieder Kap. 2.1.1). Falls dem nicht so ist, wird der Plan verfeinert, indem ein sogenannter *Mangel* bzw. *Flaw* (bspw. eine offene Vorbedingung oder eine Bedrohung eines Causal Links) ausgewählt wird und der Plan um Maßnahmen erweitert wird, diesen Mangel zu beheben. Dies kann bspw. durch das Hinzufügen neuer Schritte, Reihenfolgebedingungen oder Variablenbindungen erreicht werden. Jede der Maßnahmen führt zu einem neuen Plan, der anhand einer Bewertung so in die Arbeitsliste einsortiert wird, daß der Plan mit der geringsten Bewertung am Anfang der Liste steht. Im nächsten Schritt des Planungsprozesses wird also der Plan bearbeitet, der die geringste Bewertung aufweist. Wird für die Bewertung eine Funktion eingesetzt, die monoton mit der Anzahl der Komponenten des Plans (Planschritte, Causal Links) wächst, so ist sichergestellt[34], daß der erste Plan, für den der Planungsprozeß Vollständigkeit feststellt, die geringstmögliche Bewertung aller Lösungen des Planungsproblems hat. Alle anderen Pläne der Arbeitsliste können bestenfalls gleich bewertet und ebenfalls vollständig sein. Da die Maßnahmen zur Vervollständigung von Plänen die Bewertung nur erhöhen, nicht aber verringern können, ist es nicht möglich, daß ein Plan der Arbeitsliste zu einer

[33] vgl. a. Kap. 2.1.1 *Least Commitment Planning*

[34] Korrektheit und Vollständigkeit des Planungsverfahrens vorausgesetzt

Lösung führt, die besser bewertet ist als der erste vollständige Plan. Gerevini und Schubert diskutieren in [Gerevini und Schubert, 1996] die Auswirkungen von verschiedenen Bewertungsfunktionen in Hinblick auf das Ziel, möglichst schnell eine Lösung des Planungsproblems zu finden.

Um Pläne mit möglichst guten Präsentationsbewertungen zu bevorzugen, kann die effort-Komponente der Funktion *present-plan* des Präsentationsalgorithmus aus Abb. 4.2 als Bewertungsfunktion eingesetzt werden. Für deren Funktionsweise ist es irrelevant, ob der Plan vollständig ist (wie während der Planpräsentation) oder nicht (wie hier). Da eine Bewertungsfunktion während der Plangenerierung für *jeden* Plankandidaten berechnet werden muß – für die Beispiele in dieser Arbeit einige Tausend –, ist der direkte Einsatz der aufwendigen Funktion *present-plan* aber nicht sinnvoll. Als Alternative wird deshalb die Funktion *estimate-pres-effort* eingesetzt, die die Bewertung für einen Plan(kandidaten) p durch folgende Formel abschätzt:

$$(1) \quad \sum_{s \in steps(p)} \big(\textit{effort-plan-concept}(\textit{plan-concept}(s)) + k_{cl} * \textit{unknown-preconds}(\textit{plan-concept}(s))\big) + \sum_{pc \in spc(p)} \textit{expl-effort}(pc)$$

Der Vorteil dieser Funktion besteht darin, daß hier lediglich Werte aufsummiert werden, die bereits *vor* Beginn des Planungsprozesses für jedes nutzbare Plan-Concept bestimmt werden können. Die bereits beim Präsentationsalgorithmus aus Abb. 4.2 eingesetzte Funktion *effort-plan-concept* sowie die Funktionen *unknown-preconds* und *expl-effort* sind Zugriffsfunktionen auf diese vor Beginn des Planungsprozesses berechneten Werte. Ebenfalls zuvor bestimmt sind die Mengen der Teil-Plan-Concepts eines jeden Plan-Concepts, die für die Funktion *spc* benötigt werden (s.u.).

Die Funktion *unknown-preconds* liefert die Anzahl der Vorbedingungskomponenten eines Plan-Concepts, die der Benutzer nicht kennt, deren Wert er aber durch sein Handeln negieren kann. Der Planungsprozeß muß für jede Vorbedingungskomponente einen Causal Link erzeugen. Kann der Benutzer den Wert der zugehörigen Bedingung negieren, ist ein solcher Causal Link für die Präsentation relevant (vgl. Kap. 4.1 *Planstruktur*); d.h. er muß dem Benutzer genannt werden. Ein späteres Entfernen des Causal Links durch die Berücksichtigung von Benutzerinferenzen ist nicht möglich, weil der Benutzer die Vorbedingungskomponente nicht kennt. Für solche Vorbedingungskomponenten wird deshalb der Aufwand für die Causal Links berücksichtigt, die durch sie präsentiert werden müssen. Weitere Causal Links werden nicht berücksichtigt, weil sie häufig nicht explizit in der Planpräsentation genannt werden müssen (vgl. Kap. 4.4.4 *Diskussion*).

Funktion *expl-effort* liefert den Anteil der Präsentationsbewertung für ein Plan-Concept, der zwar (im Falle, daß der Benutzer kein atomares oder verläßliches Wissen hat) auch den Aufwand für die Präsentation der besten Ausführungsmöglichkeit berücksichtigt. Aufwände für evtl. erforderliche Erklärungen der einzelnen Schritte dieser Ausführungsmöglichkeit werden aber nicht berücksichtigt. Dadurch wird auch bei mehrfachem Auftreten eines Plan-Concepts in einer Präsentation vermieden, daß der Aufwand für Erklärungen mehrfach in der Formel berücksichtigt wird, obwohl die Erklärungen nur einmal gegeben werden. Der Anteil der Bewertung für Plan-Concepts wird durch Summierung von *expl-effort* über alle Plan-Concepts der Menge $spc(p)$ berechnet. Die Menge sei für einen Plan p durch

$$spc(p) \quad := \quad \textit{plan-concepts}(p) \cup \bigcup_{pc \in \textit{plan-concepts}(p)} \textit{part-plan-concepts}(pc)$$

definiert und enthält durch die Vereinigung aller Teil-Plan-Concepts auch solche Plan-Concepts, die lediglich für die Präsentation von Ausführungsmöglichkeiten erforderlich sind. Die Menge der Teil-Plan-Concepts, die für die Präsentation eines einzelnen Plan-Concepts *pc* erforderlich ist, wird wie-

derum *vor* Beginn des Planungsprozesses mithilfe von Funktion *present-plan-concept* des Präsentationsalgorithmus aus Abb. 4.2 durch

$$\textit{part-plan-concepts}(pc) \quad := \quad \textit{plan-concepts}(\textit{presentation}(\textit{present-plan-concept}(pc, \{\})))$$

berechnet. Auf diesen Wert kann mit der Funktion *part-plan-concepts* (*pc*) zugegriffen werden.

Der Wert der Funktion *expl-effort* wird ebenfalls *vor* Beginn des Planungsprozesses für jedes nutzbare Plan-Concept bestimmt. Der folgende Algorithmus in Abb. 4.8 berechnet diesen Wert für ein Plan-Concept *pc*:

(e1) *effort* := **if** $pc \in$ SBUB **then** 0 **else** *effort*(*descript*(*pc*))
(e2) **if not** ($atomar_{\text{SBUB}}(pc)$ **or** $verläßlich_{\text{SBU}}(pc)$) **then**
(e3) **if** $dekomp_{\text{SBUB}}$ $(pc) \neq \{\}$ **then** *effort* := *effort* + k_{corr}
(e4) *effort* := *effort* + 1
(e5) $ec := \{d \in dekomp_{\text{SB}}\ (pc) \mid nutzbar_{\text{SBU}}(d)\}$
(e6) **for** $d \in ec \wedge$ *effort*(*present-plan*(d, $\{\}$)) = $\min_{(d' \in ec)}$ *effort*(*present-plan*(d', $\{\}$))
(e7) **for each** $s \in$ *steps* (d)
(e8) *effort* := *effort* + k_{hier} * *effort-plan-concept*(*plan-concept*(s))
(e9) *expl-effort* (*pc*) := *effort*

Abbildung 4.8: Algorithmus zur Berechnung des Erklärungsaufwandes (*expl-effort*) für ein Plan-Concept.

Der Algorithmus arbeitet in gleicher Weise wie die Funktion *present-plan-concept* des Präsentationsalgorithmus. Hier wird allerdings nur der Aufwand berechnet. Dabei wird auch nur die Präsentation *einer* Dekomposition des Plan-Concepts *ohne* evtl. erforderliche Erklärungen der Plan-Concepts dieser Dekomposition berücksichtigt. Die Berechnungsschritte sind deshalb in *einer* Funktion zusammengefaßt, ohne daß dafür weitere rekursive Aufrufe erforderlich wären. Lediglich für die Auswahl der besten Ausführungsalternative in Schritt (e6) wird auf die rekursiv definierte Funktion *present-plan* des Präsentationsalgorithmus zurückgegriffen.

Für alle in diesem Abschnitt beschriebenen Aufwandsabschätzungen wurde auf die explizite Verwendung der Aufwandsmaßeinheit k_{pres} verzichtet, weil die berechneten Werte lediglich untereinander verglichen werden und ihr absoluter Betrag keine Rolle spielt.

Die Funktion *estimate-pres-effort* wächst monoton mit der Zahl der Planschritte, die einem Plankandidaten während des Planungsprozesses hinzugefügt werden. Ihr Einsatz als Bewertungsfunktion für die Plangenerierung bewirkt deshalb, daß der erzeugte Plan den geringstmöglichen Wert für die Abschätzung der Präsentationsbewertung hat. Die Schätzung selbst kann aber nicht exakt den Einfluß berücksichtigen, der sich erst aus dem fertigen Gesamtplan auf die Bewertung ergibt. Dies gilt insbesondere für den Aufwand von Komponenten, die nicht explizit genannt werden müssen, weil sie der Benutzer selbst leicht schließen kann. Es kann also *nicht garantiert* werden, daß es keinen Plan geben kann, dessen Bewertung besser als die des erzeugten Plans ist. Eine solche Garantie ist allerdings auch nicht unbedingt notwendig. Durch die Berücksichtigung der Präsentationsbewertung bei der Generierung soll lediglich verhindert werden, daß Pläne erzeugt werden, die dem Benutzer einen hohen Aufwand für das Verstehen der Präsentation aufbürden, weil viele Komponenten erklärt werden müssen, obwohl es auch Lösungen gibt, die ohne langwierige Erklärungen auskommen. Dieses Ziel wird hier aber dennoch erreicht, weil gerade der Aufwand für solche Erklärungen einen großen Einfluß auf die Bewertung hat und die zugehörigen Planschritte wegen Bedingung (bs1) aus Abschnitt 4.4 *Berücksichtigung von Benutzer-Inferenzen* nicht aus der Präsentation entfernt werden können. Das heißt, daß *bevorzugt* Pläne generiert werden, von denen der Benutzer möglichst viele

der verwendeten Plan-Concepts kennt und die damit keine langwierigen Erklärungen benötigen. Diese Bevorzugung darf nicht damit verwechselt werden, daß dem Benutzer *nur noch* Pläne präsentiert werden, die *keinerlei* Erklärungen für den Benutzer erfordern. Dies würde ein Hilfesystem auch ad absurdum führen. Die Berücksichtigung der Präsentationsbewertung bei der Plangenerierung kann allerdings dazu führen, daß dem Benutzer ein Plan mit etwas mehr Schritten zugemutet wird, wenn er die dabei verwendeten Plan-Concepts gut kennt, anstelle eines kürzeren Plans, der aber (viele) unbekannte Plan-Concepts enthält. Andererseits kann und soll das Bewertungsverfahren keinesfalls Pläne völlig ausschließen, die für den Benutzer einen hohen Erklärungsbedarf haben, wenn dem Benutzer ansonsten gar keine Lösung angeboten werden könnte.

Eine Berücksichtigung weiterer Bewertungskriterien, die an die Verwendung von Plan-Concepts gebunden sind, kann in gleicher Weise erfolgen. Beispiele für solche Kriterien sind die bereits in Abschnitt 4.3.5 *Alternative Modelle zur Planbewertung* behandelten Benutzerpräferenzen oder die Wahrscheinlichkeit einer erfolgreichen Ausführung. Wie dort bereits erwähnt, spielt es bei diesen Kriterien keine Rolle, ob ein Plan-Concept mehrfach für die Planausführung eingesetzt wird – jedes Auftreten muß einzeln berücksichtig werden. Damit wird der Einfluß eines Plan-Concepts auf die Bewertung unabhängig vom Gesamtplan und kann im Voraus berechnet werden. Die Aufwandsabschätzung in Formel (1) kann dadurch auf die Summierung aller Anteile aller Planschritte vereinfacht werden. Auch das Problem, daß nachfolgende Verarbeitungsstufen die Bewertung wieder verändern, tritt hier nicht auf, da der Benutzer die erforderlichen Planschritte unabhängig davon ausführen muß, ob sie in der Planpräsentation enthalten sind oder nicht.

4.6 Die innere Struktur von Plan-Concepts

In Kapitel 3.2 *Repräsentation von Planoperatoren* wurde argumentiert, daß sich die Operatoren für die Plangenerierung als Begriffe (Concepts) auffassen lassen. Es wurde gezeigt, wie sich deren Vorbedingungen und Effekte komponentenweise[35] als Attribute dieser Plan-Concepts in Beschreibungslogik repräsentieren lassen. Dieser Repräsentation kam zugute, daß die Komponenten von Vorbedingungen und Effekten vergleichbare Charakteristika aufweisen wie sonstige Objekteigenschaften. Insbesondere haben speziellere Plan-Concepts i.a. mehr solche Komponenten oder ihr Wertebereich ist eingeschränkt. Für ein Plan-Concept bedeutet ein eingeschränkter Wertebereich von Vorbedingung bzw. Effekt, daß es für speziellere Objekte anwendbar ist bzw. einen spezielleren Effekt erzielt.

Die Vorgehensweise läßt sich allerdings nicht in gleicher Weise auf Dekompositionen übertragen, die in diesem Kapitel für die Repräsentation von Ausführungsmöglichkeiten eines Plan-Concepts vorgeschlagen wurden. Der offensichtlichste Unterschied besteht darin, daß einem Plan-Concept mehrere Dekompositionen zugeordnet werden können. Eine Spezialisierung des Plan-Concepts muß i.a. auch auf eine spezielle Weise ausgeführt werden; d.h. ihm dürfen meist *nicht* alle Dekompositionen vererbt werden. Vielmehr können die Dekompositionen selbst als Spezialisierung des Plan-Concepts, dem sie zugeordnet sind, aufgefaßt werden, denn sie realisieren dieses Plan-Concept auf eine spezielle Weise. Dabei muß eine einzelne Dekomposition *als Ganzes* aufgefaßt werden. Diese Auffassung läßt sich nicht auf die *Schritte* einer Dekomposition übertragen. Sie spielen vielmehr – wie die Bezeichnung *Dekomposition* bereits andeutet – die Rolle von Teilkomponenten.

Die Beziehung zwischen einem Objekt und seinen Teilkomponenten läßt sich in üblichen Beschreibungslogiken, wie sie bspw. in [Heinsohn *et al.*, 1994] und [Baader, 1996] beschrieben werden, nicht adäquat ausdrücken. Da solche *Ganzes-Teil* (oder *Teil-Ganzes*) Relationen einen wichtigen Grundbaustein für die Beschreibung von Objekten darstellten, gibt es viele Ansätze sie in Beschrei-

[35] Vorausgesetzt, Vorbedingung und Effekt eines Planoperators lassen sich jeweils durch eine Konjunktion von Literalen – den Vorbedingungs- bzw. Effektkomponenten – beschreiben.

bungslogiken zu integrieren. Im nächsten Abschnitt wird auf einige der Vorstellungen kurz eingegangen. Danach folgt die Beschreibung, wie diese Ideen für die Modellierung von Dekomposition in BGP-MS genutzt werden können. Abschließend wird die Beurteilung von Dekompositionen mit der Subsumption von Weida und Litmans Temporal Constraint Networks verglichen.

4.6.1 Teil-Ganzes-Relationen in Beschreibungslogik

Für Entwicklung von Beschreibungslogiken stand die Beschreibung von Objekten durch ihre charakteristischen *Eigenschaften* im Vordergrund (vgl. Kap. 3.2.1 *Beschreibungslogiken*). Neben den Eigenschaften sind aber oft auch die Bestandteile aus denen ein Objekt aufgebaut ist charakteristisch. Zwar kann die Beziehung eines Objekts zu seinen Bestandteilen als Eigenschaft des Objekt aufgefaßt werden. In Beschreibungslogik läßt sich das bspw. durch die Definition einer Rolle *hat-Teil* zu den Bestandteilen ausdrücken. Allerdings sind Schlußfolgerungen, die sich aus der besonderen Natur der *Ganzes-Teil* Relation ergeben, durch eine solche beschreibungslogische Repräsentation formal nicht gerechtfertigt. Nur in wenigen beschreibungslogischen Formalismen läßt sich bspw. die Transitivität einer Rolle ausdrücken, wie sie der *Ganzes-Teil* Relation zumeist zugeschrieben wird. So muß ein durch

$$\textit{Text-Dokument-A} \doteq \geq 1\textit{hat-Teil} : \textit{Absatz}$$

beschriebenes Textdokument grundsätzlich als echte Verallgemeinerung von

$$\textit{Text-Dokument-S} \doteq (\geq 1\textit{hat-Teil} : \textit{Absatz}) \sqcap (\geq 1\textit{hat-Teil} : \textit{Satz})$$

klassifiziert werden, sofern *Absatz* nicht *Satz* subsumiert. Demgegenüber sind *Text-Dokument-A* und *Text-Dokument-S* gleich, sofern ein Absatz als

$$\textit{Absatz} \doteq \geq 1\textit{hat-Teil} : \textit{Satz}$$

definiert ist und die Transitivität der *Ganzes-Teil* Relation berücksichtigt wird.

Allerdings ist die Transitivität weder die einzige Eigenschaft der Beziehung(en) zwischen einem Ganzen und seinen Teilen noch gilt sie uneingeschränkt. Das obige Beispiel läßt sich weiterentwickeln, indem die Bestandteile eines Textdokuments über Absätze und Sätze hinaus weiter dekomponiert werden – bspw. in Buchstaben und Satzzeichen, die sich aus Bögen, Strichen und Punkten zusammensetzen. Dennoch kann man kaum sagen, daß sich ein Textdokument aus einer Menge von Bögen, Strichen und Punkten zusammensetzt. Andererseits können einige, aber keinesfalls alle Eigenschaften des Textdokuments auf seine Bestandteile – bis hin zu den einzelnen Bögen, Strichen und Punkten – übertragen werden. Ein Beispiel *dafür* ist die Textfarbe, ein Beispiel *dagegen* die Länge.

Winston, Chaffin und Herrmann unterscheiden in [Winston *et al.*, 1987] sechs Typen von *Teil-Ganzes* Beziehungen anhand der drei Eigenschaften *funktional*, *gleichartig* und *trennbar*. Dabei ist eine Beziehung *funktional*, wenn das Teil für das Ganze eine bestimme Funktion hat, wie bspw. die Pointe für einen Witz. Eine Beziehung ist *gleichartig*, wenn sich Teil und Ganzes von der Substanz her nicht unterscheiden, wie bspw. eine Wurstscheibe und die Wurst. *Trennbar* bedeutet schließlich, daß das Teil vom Ganzen zumindest im Prinzip abgetrennt werden kann, wie bspw. eine Karte vom Kartenstapel, nicht aber eine Oase von der Wüste. Tabelle 4.1 faßt die Beziehungen und ihre Eigenschaften zusammen. Neben einer klaren Abgrenzung der *Teil-Ganzes* Beziehungen von anderen Beziehungen, war eine Zielsetzung der Autoren, die Unstimmigkeiten bei der Transitivität von *Teil-Ganzes* Beziehungen beseitigen zu können. Nach ihrer Auffassung sind *Teil-Ganzes* Beziehungen *eines Typs* grundsätzlich transitiv, während die Komposition unterschiedlicher Typen zu “Problemen mit der Transitivität” führt. Die Hypothese wird durch Beispiele von solchen problematischen Fällen

unterstützt. Formale Kriterien für die Beurteilung fehlen allerdings. Iris, Litowitz und Evens setzen sich in [Iris *et al.*, 1988] damit auseinander und schlagen alternative Beziehungen vor. Weitere Vorschläge finden sich in [Gerstl und Pribbenow, 1996] und [Sattler, 1998]. Winston, Chaffin und

		Eigenschaften		
Relationstyp	Beispiel	funktional	gleichartig	trennbar
Komponente / zusammengesetztes Objekt	Pointe – Witz	+	–	+
Element/Sammlung	Karte – Kartenstapel	–	–	+
Teilmenge/Masse	Wurstscheibe – Wurst	–	+	+
Material/Objekt	Stahl – Fahrrad	–	–	–
Hauptpunkt/Aktivität	Bezahlen – Kaufen	+	–	–
Ort/Gebiet	Oase – Wüste	–	+	–

Tabelle 4.1: *Teil-Ganzes* Beziehungen nach [Winston *et al.*, 1987, S.421]

Herrmanns *Teil-Ganzes* Beziehungen können als Spezialisierung einer allgemeinen (transitiven) *Teil-Ganzes* Beziehung aufgefaßt werden. Für die Beschreibung von Objekten oder Begriffen sind sie allerdings dennoch oft zu allgemein und müssen weiter spezialisiert werden. Bspw. kann der Beziehungstyp *Komponente/zusammengesetztes-Objekt* zu *Antriebsrad-von* spezialisiert werden, um die Funktion eines Teils (ein Rad) für das Ganze (ein Auto) ausdrücken oder dieses Teil von anderen – evtl. gleichartigen – unterscheiden zu können. Außer den Beziehungen zwischen einem Ganzen und seinen Teilen, können auch die Beziehungen der Teile untereinander für die Beschreibung eines Ganzen relevant sein. Bspw. können die verschiedenen Klassen von Vierecken durch die relative Anordnung der Seiten und ihrer Größenverhältnisse beschrieben werden: ein Parallelogramm ist ein Viereck mit jeweils 2 Paaren paralleler, gleich langer Seiten.

Die Vielfalt der möglichen Beziehungen, sowie die unterschiedlichen Auffassungen über grundlegende *Teil-Ganzes* Beziehungen haben zu einer Vielzahl von Ansätzen zur Integration von *Teil-Ganzes* Beziehungen in Beschreibungslogik geführt. Übersichten finden sich bspw. in [Artale *et al.*, 1996a] und [Lambrix, 2000, Kap.4]. Das Spektrum reicht dabei von Ansätzen, die zwar *Ganzes-Teil* Beziehungen von anderen Beziehungen abgrenzen, aber die besonderen Eigenschaften der Beziehungen nicht bei beschreibungslogischen Inferenzen berücksichtigen (bspw. [Speel und Patel-Schneider, 1994]) bis hin zu Untersuchungen, welche Anforderungen diese Eigenschaften an die Ausdrucksstärke einer Beschreibungslogik stellen (bspw. [Sattler, 1998]). Sattler weist dabei in ihrem Schlußwort explizit darauf hin, daß es keinen “besten” Ansatz geben kann, weil mit steigender Ausdrucksfähigkeit auch die Komplexität der beschreibungslogischen Inferenzen (insb. Feststellung von Erfüllbarkeit und Subsumption) steigt und die Inferenzen für einige Fälle sogar unentscheidbar sind. Eine Entscheidung über die “richtige” Modellierung kann damit sinnvollerweise nur getroffen werden, wenn für den konkreten Einsatzbereich die Aufgaben und erforderlichen Inferenzen bekannt sind. Dies müssen – je nach Anwendungsbereich – nicht notwendigerweise beschreibungslogische Inferenzen sein. Lambrix stellt in [Lambrix, 2000] für seinen Anwendungsbereich Logistik die Wichtigkeit von Inferenzen heraus, die speziell auf den Beziehungen zwischen Ganzen und Teilen beruhen, bspw. die automatische Zusammenstellung von Ganzen aus Teilen, die Konstruktion von komplexen Objekten sowie die Feststellung, ob und welche Teile einem Objekt fehlen, um einer Beschreibung zugenügen.

Die Entscheidung über die Modellierung von Dekompositionen und deren Schritte im folgenden Abschnitt beginnt deshalb mit der Bestimmung der Aufgaben und Schlußfolgerungen, für die Dekompositionen in dieser Arbeit verwendet werden.

4.6.2 Repräsentation von Dekompositionen in BGP-MS

Dekompositionen wurden für die ursprüngliche Plangenerierung nicht betrachtet. Erst in Kap. 4.5 *Optimierte Plangenerierung* wurden Dekompositionen auch für die Generierung berücksichtigt, um den Erklärungsbedarf eines Plans für den jeweiligen Benutzer abschätzen zu können. Da diese Abschätzung lediglich eine Vereinfachung gegenüber der Vorgehensweise für die Präsentation des Plans darstellt, reicht es aus, die für die Präsentation relevanten Aspekte zu behandeln.

Im Rahmen der Präsentation eines Plans werden Dekompositionen von Plan-Concepts für zwei Aufgaben eingesetzt. Dekompositionen im Modell des Systemwissens repräsentieren die Möglichkeiten *wie* ein Plan-Concept ausgeführt werden kann. Sie bilden damit die Basis für Erklärungen, die dem Benutzer gegeben werden können, wenn der nicht das notwendige Wissen hat. Für die Repräsentation ergibt sich daraus die Forderung, daß die für die Erklärung notwendigen Informationen daraus ableitbar sind. Dies sind im einzelnen:

- die Zugehörigkeit einer Dekomposition zu einem Plan-Concept
- die Schritte einer Dekomposition
- die Reihenfolgebedingungen der Schritte einer Dekomposition
- die Zugehörigkeit der Schritte einer Dekomposition zu den Plan-Concepts, deren Exemplare sie sind

Für den Präsentationsprozeß müssen die einzelnen Dekompositionen eines Plan-Concepts außerdem unterscheidbar sein, und der Planungsprozeß muß die Plan-Concepts, die für Dekompositionen verwendet werden, von denen unterscheiden können, die für die Plangenerierung verwendet werden dürfen.

Die zweite Aufgabe, für die Dekompositionen eingesetzt werden, ist die Beurteilung des Benutzerwissens über Ausführungsmöglichkeiten von Plan-Concepts, wie in Abschnitt 4.2.2 *Beurteilung von Wissen, wie ein Plan-Concept ausgeführt wird* beschrieben. Die dort beschriebenen Beurteilungsverfahren benötigen außer den oben beschriebenen Informationen lediglich die Information über atomare Kenntnisse des Benutzers. Diese betreffen allerdings nicht Dekompositionen, sondern Plan-Concepts.

Insbesondere die Beurteilung der Korrektheit einer Dekomposition hat Ähnlichkeiten mit der Feststellung der Subsumption zwischen Concepts. Im folgenden Abschnitt wird allerdings gezeigt, warum die Subsumption für die Beurteilung der Korrektheit von Ausführungsmöglichkeiten nicht die gewünschten Ergebnisse liefert und deshalb nicht eingesetzt werden kann.

Die genannten Aufgaben erfordern für die Ganzes-Teil Beziehung, die zwischen einer Dekomposition und ihren Schritten besteht, keine der speziellen Inferenzen, die üblicherweise mit der Teil-Ganzes Beziehung in Verbindung gebracht werden. Diese Beziehung muß sich demnach *für das Wissensrepräsentationssystem* nicht unbedingt von anderen Beziehungen zwischen Objekten unterscheiden und ihre Repräsentation erfordert auch keine Erweiterungen des in dieser Arbeit verwendeten Formalismus SB-ONE. Vielmehr kann die in Kap. 3.2 *Repräsentation von Planoperatoren* gewählte Vorgehensweise, Planungswissen in einen beschreibungslogischen Formalismus einzubetten, auch für die Repräsentation von Dekompositionen eingesetzt werden. Die dort genannten Argumente *für* eine

solche Repräsentation – insbesondere die Vorteile bei der Akquisition der Benutzereigenschaften – gelten in gleicher Weise für die Repräsentation von Dekompositionen.

Weiter oben wurde bereits argumentiert, daß die Beziehung zwischen einer einzelnen Dekomposition und ihren Schritten eine *Ganzes-Teil* Beziehung ist, während die Dekomposition selbst als Spezialisierung eines Plan-Concepts aufgefaßt werden kann. Eine Dekomposition eines Plan-Concepts *pc*, die in Abschnitt 4.2.1 *Dekomposition eines Plan-Concepts* S. 179 als Paar (*steps*, *constraints*) definiert wurde, wird deshalb als Spezialisierung von *pc* repräsentiert. Da für jedes Concept beliebig viele Spezialisierungen definiert werden können, lassen sich einem Plan-Concept beliebig viele Dekompositionen zuordnen. Ein so definiertes *Dekompositions-Concept* erbt sowohl die Zugehörigkeit zur Klasse der Plan-Concepts als auch die speziellen Eigenschaften von *pc* – insbesondere deren Vorbedingungs- und Effektkomponenten. Um Dekompositionen für den Plangenerierungsprozeß von anderen Plan-Concepts unterscheiden zu können, wird jede Dekomposition grundsätzlich auch als Spezialisierung eines besonderen Plan-Concepts *decomposition* definiert. *decomposition* selbst ist eine primitive Spezialisierung des allgemeinsten Plan-Concepts *plan-concept* ohne zusätzliche Attribute.

Ist es – wie in unserem Falle – nicht erforderlich, die Besonderheiten zu berücksichtigen, die mit einer *Ganzes-Teil* Beziehung verbunden werden, lassen sich solche Beziehungen in Beschreibungslogik durch Rollen zwischen dem Ganzen und seinen Teilen repräsentieren. Der Präsentationsprozeß muß die einzelnen Schritte einer Dekomposition auseinander halten können. Einem *Dekompositions-Concept* wird deshalb für jeden Schritt s_i ein *Attribut* mit Namen *step_si* zugeordnet. Für jeden Schritt einer Dekomposition ist festgelegt, von welchem Plan-Concept es abgeleitet ist. Das läßt sich in Beschreibungslogik durch den Typ bzw. die Werterestriktion des Attributs ausdrücken. Hat eine Dekomposition also die Schritte $\{s_1{:}pc_1, \ldots, s_n{:}pc_n\}$ werden für das *Dekompositions-Concept n* Attribute *step_s1*, ..., *step_sn* mit den Werterestriktionen $pc_1, \ldots, pc_n$ definiert. Mit den Namen der Attribute werden keine Reihenfolgebedingungen impliziert. Diese sind vielmehr durch die Ordnungsrelation *before* auf den Planschritten gegeben. Analog zum Vorgehen bei der Repräsentation von zweistelligen Relationen zwischen Vorbedingungs- oder Effektkomponenten eines Plan-Concepts, wird die Relation *before* durch *Role-Value-Maps* (*RVMs*) repräsentiert[36]. Für jedes Tupel $(s_i, s_j) \in before(d)$ wird für das *Dekompositions-Concept d* eine RVM $step_sj \subseteq step_si \circ before$ definiert.

Der Entwickler der Benutzermodellierungswissensbasis (BMWB) muß Dekompositionen allerdings nicht als beschreibungslogische Ausdrücke oder in der Syntax von SB-ONE angeben. Vielmehr kann er Dekompositionen als LISP-*Structure bgp-ms-decomp-struc* beschreiben. Für die Umsetzung in die SB-ONE-Darstellung – einschließlich der Ergänzung des Super-Concepts *decomposition* und des Eintrags in eine BGP-MS-View – steht die Funktion *bgp-ms-def-decomposition-from-struclist* zur Verfügung, die in einem Aufruf eine *Liste* von Dekompositions-Structures verarbeiten kann[37]. Die Angaben für eine *bgp-ms-decomp-struc* umfassen neben dem Plan-Concept, das dekomponiert wird, die Schritte der Dekomposition mit ihren Typen sowie die Reihenfolgebedingungen. Außerdem *kann* einer Dekomposition ein Name zugeordnet werden. Abb. 4.9 zeigt die Syntax die dafür eingehalten werden muß in EBNF[38].

Der Typ eines Planschritts – d.h. das Plan-Concept dessen Exemplar der Planschritt ist – wird dabei durch den Namen des Plan-Concepts angegeben. Das Beispiel aus Abb. 4.10 zeigt eine Dekomposition des Plan-Concepts *ftp-download-file-bin*, der der Name *d3-ftp-bin* zugeordnet wird. Sie besteht aus den beiden Schritten s_1 – Auswahl des Namens der Datei, die transferiert werden soll – und s_2 – Betätigen der Tastenkombination <Alt>, <T>. Die Reihenfolgebedingung nach Schlüssel-

[36] vgl. Kap. 3.2.2 *Planoperatoren in Beschreibungslogik*, insb. Tabelle 3.3 und Abb. 3.5.

[37] vgl. die Beispiele in Kap. 5.2.3 *Initialer Aufbau einer Benutzermodellierungswissensbasis*.

[38] vgl. a. Abb. 3.3 S. 111 in Kap. 3.2.2 *Planoperatoren in Beschreibungslogik*, die auch die Definition von *predicate* enthält sowie Anh. A.1 für die Beschreibung von EBNF aus [Jensen und Wirth, 1985].

decompstruc-def	=	[“:decomposition” “”’ *decomposition-name*] “:of” “”’ *plan-concept-name*
		“:steps” “’(” {“(” *step-name plan-concept-name* {*step-constraint*} “)”}$^+$ “)”
		[“:precedence” “’(” {“(” *step-name step-name* “)”}$^+$ “)”]
		[“:comment” *any-string*] .
step-constraint	=	“(” “=” *plan-concept-var-name step-var-name* “)” .
name		Name nach LISP-Syntax
any-string		String nach LISP-Syntax

{X} steht für 0 oder beliebig viele Vorkommen von X
{X}$^+$ steht für ein oder beliebig viele Vorkommen von X

Abbildung 4.9: EBNF-Syntax für die Definition von Dekompositionen von Plan-Concepts.

wort “:precedence” gibt an, daß s_1 vor s_2 ausgeführt werden muß. Diese Dekomposition wird noch in Kap. 5.3.2.4 *Pläne für Nick* benötigt, wenn es darum geht, ob und ggf. welche Erklärungen der Beispielbenutzer *Nick* für einen benutzerspezifischen Plan benötigt.

```
:decomposition 'd3-ftp-bin  :of  'ftp-download-file-bin
:steps '((s1 select-file-menu-item (= ?src-name-val ?file-name-val))
         (s2 push-2key-alt-t))
:precedence    '((s1 s2))
:comment       "haeufige Fehlannahme: Alt-T anstelle von Alt-O."
```

Abbildung 4.10: Beschreibung einer Dekomposition des Plan-Concepts *ftp-download-file-bin*.

Zusätzlich können für einen Planschritt Bindungsbedingungen zwischen den Parametern des dekomponierten Plan-Concepts und des Schritts angegeben werden. Sie können dazu genutzt werden, für den Benutzer eine Beschreibung der an den Planschritten beteiligten Objekte zu erzeugen. In dem Beispiel zeigt die Bindungsbedingung an, daß der (Vorbedingungs-)Parameter *?src-name-val* von *ftp-download-file-bin* mit dem (Vorbedingungs-)Parameter *?file-name-val* des ersten Dekompositionsschritts übereinstimmen muß – daß also für das Herunterladen einer Datei gerade der Datei-Menü-Eintrag ausgewählt werden muß, der dem Namen der Datei entspricht. Allerdings sind die zugehörigen Objekt-Beschreibungen nicht Thema dieser Arbeit und im Beispielmodell lediglich enthalten, um die prinzipielle Möglichkeit aufzuzeigen.

Für die beschreibungslogische bzw. SB-ONE Repräsentation von Bindungsbedingungen werden wieder Role-Value-Maps eingesetzt. Dabei wird eine Bindungsbedingung

(= *plan-concept-var-name step-var-name*)

eines Schritts s_i in die RVM

pre_<plan-concept-var-name> = *step_si* ∘ *pre_<step-var-name>*

umgesetzt. *pre_<plan-concept-var-name>* bzw. *pre_<step-var-name>* sind dabei die Vorbedingungsattribute des zu dekomponierenden Plan-Concepts bzw. des Plan-Concepts des Schritts s_i, die zu den Variablen *plan-concept-var-name* bzw. *step-var-name* gehören (vgl. Kap. 3.2.2 *Planoperatoren in Beschreibungslogik* S. 110).

Für die noch ausstehende Modellierung der *atomaren* Kenntnis eines Plan-Concepts durch das Prädikat *atomar* für Plan-Concepts, bieten sich zwei Möglichkeiten an:

1. Ein atomares Plan-Concept wird durch Zuordnung eines speziellen Super-Concepts *atomic* ausgezeichnet. (*atomic* selbst wird dann wie *decomposition* als primitive Spezialisierung von *plan-concept* ohne zusätzliche Attribute definiert).
2. Für ein atomares Plan-Concept wird eine "besondere" Dekomposition definiert, die selbst keine Schritte hat und vom Concept *atomic* subsumiert wird.

Die erste Möglichkeit erscheint naheliegender, hat aber den Nachteil, daß durch Vererbung auch *alle* Spezialisierungen des Plan-Concepts von *atomic* subsumiert würden. Da ein Akteur, der ein Plan-Concept atomar kennt, nicht auch *alle* Spezialisierungen davon atomar kennen muß, ist die Subsumption $pc \Rightarrow atomic$ im Modell der Benutzerannahmen kein hinreichendes Kriterium dafür, ob der Benutzer das Plan-Concept *pc* atomar kennt oder nicht. Die zweite Möglichkeit hat dieses Manko nicht. Allerdings muß hier für jedes atomare Plan-Concept ein Dekompositions-Concept repräsentiert werden. Dies kann zu einer signifikanten Vergrößerung der Wissensbasis führen, ohne das damit viel Information repräsentiert würde.

Der Nachteil der ersten Möglichkeit betrifft nur Plan-Concepts, für die Spezialisierungen modelliert sind. Möglichkeit 1. kann also ohne weiteres für Plan-Concepts ohne Spezialisierungen genutzt werden. In dieser Arbeit werden deshalb beide Repräsentations-Möglichkeiten unterstützt. Der Funktion *bgp-ms-declare-concepts-atomic*, die eine Liste von Plan-Concepts als *atomar* auszeichnet, kann ein Parameter mitgegeben werden der bestimmt, welche Möglichkeit gewählt werden soll. Wird der Parameter nicht angegeben, werden Plan-Concepts ohne Spezialisierungen nach der ersten Möglichkeit markiert, ansonsten wird nach der zweiten Möglichkeit verfahren. Die Funktion *knpp-test-atomic*, die das Prädikat *atomar* für ein Plan-Concept in einer View bestimmt, berücksichtigt beide Möglichkeiten.

Für die in Kap. 4.2.1 *Dekomposition eines Plan-Concepts* S. 178 eingeführte "*unvollständig*" Markierung von Plan-Concepts, die nur für die Schritte von Dekompositionen verwendet werden, sind die gleichen Überlegungen wie für die Markierung von Plan-Concepts als *atomar* anwendbar. Allerdings ist eine solche Markierung nur für Plan-Concepts notwendig, für die keine Spezialisierungen (außer Dekompositions-Concepts) definiert sind, da nur solche für die Plan-Generierung in Betracht kommen. Die einfachere Möglichkeit, die Markierung durch Zuordnung zu einem speziellen Super-Concepts zu realisieren, reicht deshalb aus. Damit entfällt die Notwendigkeit einer besonderen Markierungsfunktion. Als Super-Concept für diese Markierung wird *incomplete* als primitive Spezialisierung von *plan-concept* ohne zusätzliche Attribute definiert. Durch die Funktion *knpp-test-incomplete* kann für ein Plan-Concept festgestellt werden, ob die Markierung besteht oder nicht.

4.6.3 Beurteilung von Ausführungsmöglichkeiten im Vergleich zur Planerkennung

Die Definition der Korrektheit einer Dekomposition in Abschnitt 4.2.2 *Beurteilung von Wissen, wie ein Plan-Concept ausgeführt wird* besagt, daß eine Dekomposition *d* eines Plan-Concepts *pc* (neben weiteren Bedingungen) nur dann als korrekt beurteilt wird, wenn jede Linearisierung dieser Dekomposition in der Menge aller Linearisierung aller Dekompositionen, die das Plan-Concept nach Ansicht des Systems hat, enthalten ist. Sind die Linearisierungen Objekte des Interpretationsbereichs, so lassen sich folgende beschreibungslogische Concepts bilden:

$$\begin{aligned} C_1^I &:= lin(d) \\ C_2^I &:= \bigcup_{d' \in dekomp_{\mathrm{SB}}(pc)} lin(d') \end{aligned}$$

Die Bedingung für die Korrektheit einer Dekomposition kann dann auf die Subsumption $C_1 \Rightarrow C_2$ zurückgeführt werden. Lassen sich für C_1 und C_2 geeignete beschreibungslogische Repräsentationen

finden, kann die Überprüfung der Bedingung dem Wissensrepräsentationssystem überlassen werden. Weida und Litman haben in [Weida und Litman, 1992] eine beschreibungslogische Repräsentation von sogenannten *Temporal Constraint Networks* (TCNs) für die Planerkennung vorgeschlagen. Die Netzwerke bestehen aus Knoten, die (Aktions-) Concepts zugeordnet sind und die durch Allens Zeitrelationen [Allen, 1984] in Beziehung gesetzt werden können. In Kap. 2.2.3 *Planerkennung mit Beschreibungslogik* wurde die fehlende Berücksichtigung von Vorbedingungen und Effekten kritisiert, weil eine solche Abstraktion von den *wesentlichen* Eigenschaften eines Plans für eine Interaktion mit einem Benutzer ungeeignet ist und eine darauf basierende Klassifikation zu falschen Ergebnissen führen kann. Da aber für die Schritte von Dekompositionen ebenfalls auf die Modellierung von Vorbedingungen und Effekten verzichtet wurde, drängt sich die Frage auf, ob Weida und Litmans Idee nicht wenigstens für die Beurteilung von Dekompositionen genutzt werden kann.

Werden Dekompositionen isoliert betrachtet, d.h. ohne das Plan-Concept das sie realisieren, beschränkt sich ihre Beschreibung auf die Planschritte und Reihenfolgebedingungen. Sie lassen sich damit als (vereinfachte) TCNs auffassen, bei denen die Netzwerkknoten den Schritten entsprechen. Eine Entsprechung für *Mengen* von Dekompositionen läßt sich aber nicht in jedem Fall so darstellen. Hierzu wäre die Disjunktion von TCNs erforderlich.

Das wesentliche Problem liegt allerdings wiederum in der fehlenden Berücksichtigung von Vorbedingungen und Effekten bei der Definition der Subsumption in [Weida und Litman, 1992, S.284]. Danach kann ein TCN n_1 ein TCN n_2, das mehr und/oder speziellere Knoten/Schritte hat, subsumieren, obwohl n_2 durch die zusätzlichen oder spezielleren Schritte evtl. nicht alle Effektkomponenten von n_1 hat. Weida und Litmans Beispielnetzwerk *Heat-Noodles*[39], das aus *Nudeln-machen* und danach *Erhitzen* besteht, würde demnach auch ein Netzwerk subsumieren, das einen weiteren Schritt *abkühlen-lassen* nach dem Erhitzen enthält. Den Effekt, daß die Nudeln heiß sind, hat dieses erweiterte Netzwerk allerdings nicht und es kann deshalb auch nicht in einem Plan verwendet werden, in dem heiße Nudeln gebraucht werden. Auf die Probleme durch ungeeignete Spezialisierungen von Plan-Concepts[40] wurde bereits in Kap. 2.2.2 *Kautz' Theorie der Planerkennung* im Zusammenhang mit der Repräsentation von Planbibliotheken für die Planerkennung hingewiesen. Allerdings spielen die daraus resultierenden undurchführbaren Pläne in einer Planbibliothek keine Rolle, wenn davon ausgegangen wird, daß der Benutzer nur korrekte Pläne ausführt und somit niemals falsche Pläne beobachtet werden können.

Für ein Hilfesystem ist eine solche Annahme allerdings nicht sinnvoll, insbesondere nicht wenn festgestellt werden soll, *ob* das Wissen des Benutzers (nach Annahme des Systems) korrekt ist. Da für die Realisierung eines Plan-Concepts nur solche Dekompositionen eingesetzt werden dürfen, die alle Effektkomponenten des Plan-Concepts erreichen, können auch nur solche Dekomposition als korrekt eingestuft werden. Werden Dekompositionen wie im vorangegangen Abschnitt repräsentiert, erben sie Vorbedingungen und Effekte von den Plan-Concepts die sie realisieren. Damit wird zwar ausgedrückt, daß nur solche Folgen von Planschritten zu einer Dekomposition gehören, die den Effekt des zugehörigen Plan-Concepts erzielen. Es ist aber durch beschreibungslogische Inferenzen nicht möglich festzustellen, *ob eine bestimmte* Folge von Planschritten auch tatsächlich den Effekt erzielt. Dazu wäre ein Planungs- bzw. Planprüfungsprozeß erforderlich, der hier nicht eingesetzt werden kann, weil Vorbedingungen und Effekte der Schritte einer Dekomposition nicht repräsentiert sind. In Abschnitt 4.2.1 *Dekomposition eines Plan-Concepts* wurde darauf hingewiesen, daß der Entwickler des Sachbereichsmodells sicherstellen muß, daß das Ausführen einer Dekomposition die im Systemwissen modelliert ist auch den zugehörigen Effekt erzielt. Dies muß zwar für jede erlaubte Ausführungsreihenfolge der Dekompositions-Schritte gelten, wegen möglicher störender Effekte aber nicht für *beliebige* Spezialisierungen im Sinne von Weida und Litman. D.h. das Erreichen des Effekts der

[39] vgl. [Weida und Litman, 1992, Fig.3] und Abb. 2.9 in Kap. 2.2.3 *Planerkennung mit Beschreibungslogik*

[40] Die *Event Types* dort können als Plan-Concepts aufgefaßt werden.

Dekomposition ist nicht für *jede* erlaubte Spezialisierung der Dekompositions-Schritte sichergestellt und insbesondere nicht, wenn zusätzliche Schritte ausgeführt werden. Dementsprechend wird eine Dekomposition *d* eines Plan-Concepts nur dann als korrekt eingestuft, wenn ihre Schritte (bis auf Umbenennung) mit denen einer Dekomposition des Plan-Concepts im Modell des Systemwissens übereinstimmen. Es reicht dazu nicht aus, wenn *d* lediglich von einer Dekomposition des Plan-Concepts im Systemwissen subsumiert[41] wird.

Es sei noch einmal darauf hingewiesen, daß auch die in dieser Arbeit zugrunde gelegten Bedingungen an die Korrektheit einer Dekomposition nicht *garantieren* können, daß der Benutzer einen Planschritt tatsächlich korrekt ausführt. Allerdings stellen die Bedingungen sicher, daß nur solche Dekomposition im Modell des Benutzerwissens als korrekt eingestuft werden, für die das System annimmt, daß es für den Benutzer *möglich* ist danach zu handeln und daß der Benutzer (direkt oder indirekt) umfassende Kenntnisse über die Schritte der Dekomposition hat, sodaß er sie geeignet zur Realisierung des zugehörigen Plan-Concepts einsetzen kann.

4.7 Zusammenfassung

Ziel dieses Kapitels war die Entwicklung von Verfahren, die den Inhalt für die Präsentation eines Plans bestimmen, die den jeweiligen Benutzer eines Hilfe- oder Unterstützungssystems in die Lage versetzen durch Ausführen des Plans seine Ziele zu erreichen. Ausgangspunkt ist ein Plan, der – wie im vorangegangenen Kapitel beschrieben – unter Berücksichtigung des Handlungspotentials des Benutzers erzeugt wurde. Der Benutzer kann den Plan also *im Prinzip* ausführen; möglicherweise fehlen ihm aber für dessen erfolgreiche Ausführung noch Kenntnisse.

Das Verfahren zur Inhaltsbestimmung basiert dabei auf folgenden grundlegenden Annahmen:

1. Der Benutzer muß den Plan oft noch weiter verarbeiten; das kann heißen
   (a) er muß den Plan noch weiter entwickeln, sodaß die einzelnen Schritte für ihn direkt ausführbar sind oder
   (b) er muß den Plan modifizieren, weil unerwartete Hindernisse bei der Ausführung auftreten
2. Der Benutzer muß wissen *wie* er die einzelnen Schritte ausführen kann.

Der erste Punkt wird dadurch unterstützt, daß die Präsentation außer den Schritten des Plans auch Informationen über die Struktur des Plans – dargestellt durch die (relevanten) Causal Links – enthält. Damit kann der Benutzer für jeden Planschritt erkennen, welche Rolle dieser im Plan spielt und insbesondere welche Bedingungen bei einer Weiter- oder Umplanung nicht zerstört werden dürfen.

Beim zweiten Punkt wurde davon ausgegangen, daß es sich bei den Planschritten bzw. den Plan-Concepts von denen sie abgeleitet wurden, um relevante – und damit auch häufig bekannte – Begriffe des Anwendungssachbereichs handelt, mit denen ein bestimmtes Ziel erreicht werden kann. Da dieses Ziel fest mit dem Plan-Concept verbunden ist, kann ein Akteur vorgefertigte Lösungen (*Rezepte*) kennen, wie ein solches Ziel zu erreichen ist. Kennt der Benutzer zu einem der Planschritte keine Rezepte die er auch *nutzen* kann, wird die Planpräsentation – sofern möglich – um eine für ihn nutzbare Lösungsmöglichkeit ergänzt. Die Nutzbarkeit einer Lösung setzt voraus, daß der Benutzer *weiß wie* die einzelnen Schritte der Lösung ausgeführt werden können *und* daß er sie auch *ausführen kann*. Die Inhaltsbestimmung einer benutzerspezifische Planpräsentation muß also nicht nur das Wissen, sondern – wie bei der Generierung des Plans – auch das Handlungspotential des Benutzers berücksichtigen.

[41] im Sinne von Weida und Litman; d.h. wiederum ohne Berücksichtigung von Vorbedingungen und Effekten.

Fehlt dem Benutzer für viele Schritte des Plans das notwendige Wissen, kann die Präsentation eines Plans für den Benutzer unzumutbar lang werden. Um dies so weit wie möglich zu verhindern, wurde der in Kapitel 3 vorgestellte Plangenerierungsprozeß modifiziert, sodaß bevorzugt solche Pläne erzeugt werden, für die der Benutzer möglichst wenige zusätzliche Erklärungen benötigt. Als weitere Kriterien, die den Generierungsprozeß leiten könnten, wurden Benutzerpräferenzen und Erfolgswahrscheinlichkeiten bzgl. der Ausführung von Plan-Concepts vorgeschlagen.

Kennt sich der Benutzer mit den Plan-Concepts des Anwendungsbereichs dagegen gut aus, muß ihm in der Präsentation des Plans nicht jedes Detail explizit genannt werden, wenn es sich aus der Präsentation und dem Wissen des Benutzers ableiten läßt. In Anlehnung an Ideen aus dem Bereich *Generierung natürlichsprachlicher Texte* wurde gezeigt, unter welchen Umständen der Benutzer welche Komponenten leicht ableiten kann, die dann aus der Präsentation entfernt werden können. Das Verfahren wägt dabei zwischen dem Aufwand, den der Benutzer für das Verstehen einer expliziten Darstellung einer Komponente hat, und dem Aufwand für das Schlußfolgern der Komponente, wenn sie nicht explizit genannt wird, ab. Während der Aufwand für die Rekonstruktion von Planschritten häufig hoch ist und sie dementsprechend selten aus einer Planpräsentation entfernt werden, kann die Strukturinformation verhältnismäßig leicht von einem Benutzer mit guten Kenntnissen rekonstruiert werden. Durch Bedingungen, die an das Entfernen von Komponenten aus der Planpräsentation geknüpft sind, wird verhindert, daß Benutzer mit geringeren Kenntnissen einen präsentierten Plan als unvollständig empfinden und ihre Ziele nicht erreichen können.

Benutzerkenntnissen, die nach Ansicht des Systems unvollständig oder falsch sind, wurde im vorgestellten Ansatz lediglich durch die Präsentation von (nach Ansicht des Systems) korrektem Wissen begegnet. Zwar kann angenommen werden, daß ein Benutzer der einen Rat sucht diesen normalerweise auch annimmt; es kann aber – insbesondere bei Fehlannahmen des Benutzers – die Situation auftreten, daß der Benutzer nicht überzeugt ist, daß der Rat tatsächlich richtig ist. In einem solchen Fall wäre ein Klärungsdialog angebracht, in dem Fehlannahmen des Benutzers erklärt werden, der präsentierte Plan gerechtfertigt wird oder eine Alternativlösung präsentiert wird. In dieser Arbeit wird ein solcher Klärungsdialog ausgeklammert und stattdessen auf die Literatur zu diesem Thema verwiesen; bspw. [Chu-Carroll und Carberry, 1998] bzw. generell die in Kap. 2.3 *Gemeinsames Planen (mit einem Benutzer)* vorgestellten Arbeiten zum gemeinsamen Planen. Allerdings ist das Ziel der Planpräsentation, eine solche Situation durch eine angemessene Informationsmenge möglichst zu vermeiden, denn es kann davon ausgegangen werden, daß ein Benutzer eines Hilfesystems eher rasche Hilfe für seine konkrete Aufgabe sucht, als eine Diskussion mit dem System. Gleichwohl bietet die Repräsentation des Planungswissens von Benutzern einen Ansatzpunkt für weitere Forschungen, indem *aus der Sicht eines Benutzers* geplant wird, um die Gründe für Fehlschlüsse des Benutzer zu identifizieren. Eine solche Plangenerierung kann sich der Plan-Concepts aus dem Modell des Benutzerwissens (View SBUB) oder der Benutzerannahmen über seine eigenes Handlungspotential (View SBUBUC) bedienen.

Kapitel 5

Der Demonstrations-Prototyp

In diesem Kapitel soll gezeigt werden, wie die bisher entwickelten Ideen in einem Gesamtsystem zusammengeführt werden können. Es wird zunächst der Aufbau und die Arbeitsweise eines prototypischen Hilfesystems gezeigt, das – entsprechend der Zielsetzung dieser Arbeit – einem Benutzer anhand eines Benutzermodells eine spezifische Handlungsanweisung geben kann, mit der dieser seine Ziele erreichen kann[1]. Dies schließt auch eine Beurteilung des Laufzeitverhaltens der dafür notwendigen Algorithmen und die Begründung für die Wahl des verwendeten Planungsverfahrens mit ein. In den weiteren Abschnitten wird für eine beispielhafte Anwendung gezeigt, welche Schritte der Modellierung notwendig sind und welche Auswirkungen die verschiedenen Annahmen über die Handlungsmöglichkeiten der Beispielbenutzer auf die erzeugten Pläne hat.

Der Demonstrations-Prototyp wurde auf einer Sun UltraSparc 1 (167Mhz) Workstation unter SunOS 5.8 implementiert. Technische Einzelheiten dazu finden sich im Anhang C.1 *Systemumgebung und -voraussetzungen*. Alle Angaben zu den Beispielen beziehen sich auf dieses System.

5.1 Architektur des Gesamtsystems

Abbildung 5.1 zeigt die wichtigsten Komponenten eines Gesamtsystems zur benutzerspezifischen Generierung und Präsentation von Plänen. Zentrale Komponente ist ein um die Funktionalität zur Verarbeitung von Planungswissen erweitertes BGP-MS[2] [Kobsa und Pohl, 1995]. Darüber hinaus sind Komponenten zur Interpretation einer Benutzeräußerung, zur Plangenerierung und zur Planpräsentation erforderlich. Die grau gezeichneten Komponenten wurden in dieser Arbeit nicht behandelt. Die Abbildung zeigt den Informationsfluß zwischen den Komponenten, der in den nächsten beiden Abschnitten anhand der Modellierungsphase und der Erzeugung einer benutzerspezifischen Handlungsanweisung (in der Arbeitsphase) noch näher erläutert wird. Nicht gezeigt ist dagegen die Steuerung der beteiligten Prozesse. Während in der Modellierungsphase nur einzelne Eintragungen in die Benutzermodellierungswissensbasis (BMWB) erfolgen, sind für die Erzeugung einer benutzerspezifischen Handlungsanweisung mehrere Schritte erforderlich, die jeweils die Ergebnisse der vorangegangenen Schritte weiterverarbeiten. Vor der eigentlichen Verarbeitung werden die Plan-Concepts und Dekompositionen der BMWB bewertet und beurteilt[3]. Die Ergebnisse werden für die Erzeugung

[1] Wobei vorausgesetzt wird, daß das System ausreichend gute Kenntnisse über das Wissen und die Handlungsmöglichkeiten des Benutzers hat.

[2] vgl. auch Kap. 3.3 *Struktur eines Benutzermodells mit Handlungspotential und Wissen*. Die Abbildung zeigt lediglich die Komponenten von BGP-MS, die eine wichtige Rolle für diese Arbeit spielen.

[3] vgl. Kap. 4.2.2 *Beurteilung von Wissen, wie ein Plan-Concept ausgeführt wird* und Kap. 4.5.2 *Steuerung des Planungsprozesses*

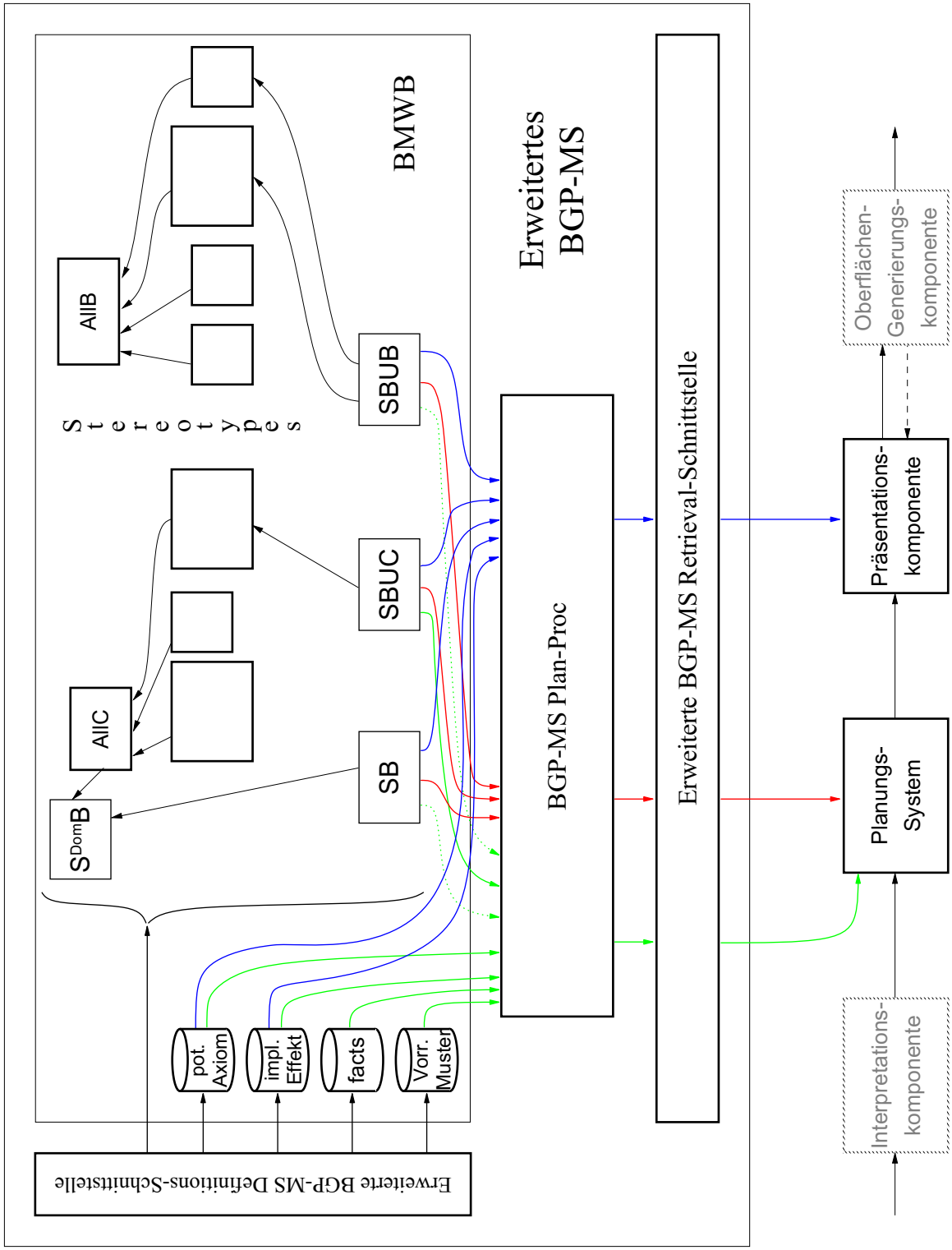

Abbildung 5.1: Architektur des Demonstrations-Prototyps

des benutzerspezifischen Planungssachbereichs[4] und für die Beeinflussung des Planungsprozesses[5]

[4]vgl. Kap. 3.4.2 *Der Planungssachbereich*

[5]in Hinblick auf die Erzeugung von Plänen mit guter Bewertung (vgl. Kap. 4.5.2 *Steuerung des Planungsprozesses*)

benötigt. Die Verarbeitung selbst beginnt mit der Feststellung der Benutzerziele durch die Interpretationskomponente. Das Planungssystem generiert dann einen benutzerspezifischen Plan; d.h. einen Plan, mit dem der jeweilige Benutzer seine zuvor ermittelten Ziele erreichen kann. Außer den Zielen benötigt das Planungssystem dazu einen benutzerspezifischen Planungssachbereich, der anhand des Benutzermodells erzeugt wird. Mit der Ergebnisdatenstruktur des Planungsprozesses – dem Plan mit Strukturinformation – wird die Präsentationskomponente aufgefordert eine für den Benutzer geeignete Präsentation zu erzeugen. Der Rahmen dieser Arbeit ist auf die Festlegung des benutzerspezifischen Inhalts beschränkt – die Erzeugung einer konkreten Darstellungsform wurde dementsprechend nicht behandelt.

Für den Demonstrations-Prototypen wurden die in den nächsten Abschnitten beschriebenen BGP-MS-Erweiterungen und die (in Abb. 5.1 nicht gezeigte) Ansteuerung von BGP-MS und des Planungssystems UCPOP implementiert, um damit benutzerspezifische Pläne zu erzeugen. Auf Beispiele wird noch in Abschnitt 5.3 *Anwendung des Prototyps* eingegangen. In den weiteren Abschnitten dieses Unterkapitels wird noch die Auswahl des Planungssystems UCPOP für diese Arbeit begründet und die Komplexität der wichtigsten Algorithmen angegeben.

5.1.1 Verarbeitung in der Modellierungsphase

In der Modellierungsphase wird die Benutzermodellierungswissensbasis initial aufgebaut. Dazu gehört sowohl die Definition der Struktur als auch der Eintrag der Inhalte in die verschiedenen Views. Ausgangspunkt für diese Arbeit ist eine BGP-MS Version die den AsTRa-Formalismus unterstützt. D.h. eine Version, die im Rahmen von Pohls Dissertation [Pohl, 1997] entstanden ist und gegenüber [Kobsa und Pohl, 1995] u.a. um die Möglichkeit erweitert wurde, Modalitäten frei definieren zu können und auch bei Stereotypes zu berücksichtigen.

Die Struktur eines Benutzermodells, das neben dem Wissen auch das Handlungspotential eines Benutzers enthält, wurde in Kap. 3.3 *Struktur eines Benutzermodells mit Handlungspotential und Wissen* beschrieben. Die Schnittstelle der hier verwendeten BGP-MS Version enthält Funktionen[6] für die Definition von Partitionen und deren Beziehungen untereinander. Neben den Stereotype-Partitionen (jeweils für Wissen und Handlungspotential) wurden in dieser Arbeit vor allem die Partitionen SB (Systemwissen), $S^{Dom}B$ (objektbezogenes Systemwissen), SBUB (Benutzerwissen), SBUC (Handlungspotential des Benutzers) benötigt (vgl. insb. Kap. 3.3.2 *Repräsentation eines Benutzermodells mit Handlungspotential und Wissen im AsTRa-Formalismus*).

In den Kapiteln 3.2 *Repräsentation von Planoperatoren* wurde gezeigt, daß sich nicht nur das begriffliche Wissen über Objekte, sondern auch das Planungswissen im Kern durch SB-ONE bzw. allgemein durch Beschreibungslogik ausreichend präzise beschreiben läßt. Die BGP-MS Schnittstelle stellt Funktionen zur Verfügung, mit denen SB-ONE Elemente (Concepts, Rollen, Role-Value-Maps, ...) in die Views der BMWB eingetragen werden können. Eine Beschreibung des Planungswissens durch einzelne SB-ONE Elemente wäre allerdings unhandlich und würde deutlich von üblichen Darstellungen – wie bspw. durch die PDDL[7] definiert – abweichen. Um die Beschreibung des Planungswissens zu erleichtern, wurde die BGP-MS Schnittstelle erweitert. Anstelle von mehreren Einzeleinträgen sind damit Einträge bzw. Abfragen der auf Basis der für die Modellierung des Planungswissens relevanten Einheiten (bspw. Plan-Concepts, Dekompositionen) in einer an PDDL bzw. UCPOP angelehnten Syntax möglich. Die wichtigsten Funktionen für Einträge sind:

- *bgp-ms-def-plan-concepts-from-struclist* – trägt Plan-Concepts auf Basis der Syntax aus Abb. 3.6 S. 114 in eine oder mehrere Views ein

[6] BGP-MS sieht außerdem einen *Message*-Mechanismus für externe Aufträge vor.
[7] vgl. Kap. 2.1 *Handlungsplanung* und 3.2.2 *Planoperatoren in Beschreibungslogik*

- *bgp-ms-def-decomposition-from-struclist* – trägt Dekompositionen von Plan-Concepts in eine oder mehrere Views (gem. Syntax aus Abb. 4.9 S. 222) ein

- *bgp-ms-declare-concepts-atomic* – markiert Plan-Concepts als *atomic* (vgl. Kap. 4.6.2 *Repräsentation von Dekompositionen in BGP-MS*)

Als Nebenprodukt entstand die Funktion *bgp-ms-defconcepts-from-struclist*, die den Eintrag einer vollständigen Concept-Beschreibung u.a. durch die Angabe von Super-Concepts, Rollen, Attributen und Role-Value-Maps in eine oder mehrere Views ermöglicht. Beispiele für die Verwendung dieser Funktionen finden sich in den Abschnitten 5.2 *Eine Beispiel-Benutzermodellierungswissensbasis*, insb. 5.2.1 und 5.2.3 sowie 5.3.2 *Benutzerspezifische Pläne*. Alle Syntax-Beschreibungen finden sich auch im Anhang A.2 *Zusammenfassung syntaktischer Definitionen in EBNF*.

Aus logischer Sicht würden die bisher genannten Eintragungen für den initialen Aufbau der BMWB ausreichen. In Kap. 3.4.2.4 *Ableitung von Abhängigkeiten aus dem Benutzermodell* wurde aber gezeigt, daß es nicht sinnvoll und bei den meisten Planungssystemen auch nicht möglich ist, *alle* Schlußfolgerungen die gemäß der BMWB zulässig wären beim Planungsprozeß auch tatsächlich zu ziehen – um bspw. festzustellen, ob eine Vorbedingung eines Planoperators erfüllt ist oder nicht. Es wurde vorgeschlagen, dem Entwickler des Sachbereichsmodells die Entscheidung zu übertragen *welche* Schlußfolgerungen relevant sein *könnten* und damit – sofern sie sich durch die aktuelle View der BMWB verifizieren lassen – beim Planungsprozeß berücksichtigt werden sollen. Es wurde dafür die Bezeichnung *(relevante) potentielle Abhängigkeiten* gewählt. UCPOP sieht für die Definition von Abhängigkeiten – wie viele andere Planungssysteme – *Axiome* vor. Schlußfolgerungen, die das Planungssystem nicht ziehen kann, wie bspw. voneinander abhängige Effekte eines Planoperators, müssen dagegen vorweggenommen werden. Dazu wurde in Kap. 3.4.2.4.4 *Ausschließende Eigenschaften bzw. Typzugehörigkeiten* die Definition von sogenannten *impliziten Effekten* vorgeschlagen. In Kap. 3.4.2.7 *Erzeugung des aktiven Teils eines Planungssachbereichs* wurde darauf hingewiesen, daß Laufzeit- und Speicherplatzverbrauch vieler Planungssysteme und insbesondere auch von UCPOP von der Reihenfolge abhängig ist, in der die Vorbedingungen von Planoperatoren angegeben werden. Dem Entwickler des Sachbereichsmodells wird deshalb die Möglichkeit zur Verfügung gestellt, durch Angabe sogenannter *Vorrang-Muster* die Erzeugung der Planoperatoren geeignet zu beeinflussen.

Die erweiterte BGP-MS Schnittstelle sieht die Funktion *bgp-ms-define-pd-dependency* vor, um

- ein Axiom gem. Syntax aus Anhang A.2.4 S. 303

- einen impliziten Effekt gem. Syntax aus Abb. 3.23 S. 151

- ein Vorrang-Muster gem. Syntax aus Abb. 3.24 S. 158

in jeweils eigene Datenbestände der BMWB – namentlich der *Menge der potentiellen*[8] *Axiome*, der *Menge der potentiellen impliziten Effekte* sowie der *Menge der Vorrang-Muster* – einzutragen. Die Syntax der Einträge legt fest, daß das erste Schlüsselwort des Eintrags angibt, ob es sich um ein Axiom, einen impliziten Effekt oder ein Vorrang-Muster handelt.

Um den Modellierungsaufwand gering zu halten, werden die Einträge als *Möglichkeiten* angesehen, die nur dann zur Wirkung kommen, wenn sie durch die für die jeweilige Aufgabe relevante View der BMWB verifiziert werden können. Die Einträge selbst sind dadurch unabhängig von den Views

[8] *potentiell* soll – analog zur Bezeichnung *potentielle Abhängigkeiten* – darauf hinweisen, daß i.a. nicht alle diese Axiome bzw. impliziten Effekte in einem Planungssachbereich zum Einsatz kommen. Ohne weitere Erwähnung sei davon ausgegangen, daß diese Mengen grundsätzlich nur solche Elemente enthalten, die vom Entwickler des Sachbereichsmodells als *relevant* eingestuft wurden.

der BMWB. Sie brauchen – unabhängig von der Zahl der Views – nur einmal angegeben werden und müssen bei einer Änderung des Benutzermodells – bspw. aufgrund der Beobachtung des Benutzers – *nicht* mit geändert werden.

Auch die sogenannten *Facts*, die in UCPOP eine Schnittstelle zwischen Planungs- und LISP-System darstellen, werden view-unabhängig nur einmal in der BMWB – in der *Menge der Facts* – gespeichert. Zum Eintrag kann wieder die Funktion *bgp-ms-define-pd-dependency* genutzt werden, wobei die Syntax aus Anhang A.2.5 S. 304 eingehalten werden muß. In dieser Arbeit wird die Möglichkeit ausschließlich eingesetzt, um im Planungsprozeß bei Bedarf neue Konstanten (für neue Objekte) erzeugen zu können (vgl. Kap. 3.4.2.5 *Erzeugen von Objekten*). Dieser technische Aspekt hat keine Entsprechung im Benutzermodell und wird im Planungsprozeß immer berücksichtigt.

5.1.2 Verarbeitung bei der Erzeugung benutzerspezifischer Handlungsanweisungen

Aus einer Benutzeranfrage muß durch die Interpretationskomponente das Ziel des Benutzers extrahiert und daraus ein Planungsproblem[9] für das Planungssystem erzeugt werden. Die Interpretation einer Benutzeräußerung war nicht Aufgabe dieser Arbeit; wir gehen deshalb davon aus, daß das Planungsziel bereits in einer geeigneten Form vorliegt.

Damit das Planungssystem daraus einen benutzerspezifischen Plan erzeugen kann, benötigt es noch die Spezifikation des Planungssachbereichs; d.h. i.w. die Beschreibung der Planoperatoren, Axiome und Facts. Für die Erzeugung eines benutzerspezifischen Plans wird der Planungssachbereich, wie in Kap. 3.4.2 *Der Planungssachbereich* beschrieben, aus der BMWB abgeleitet. Die zugehörigen Informationsflüsse sind in Abb. 5.1 grün eingezeichnet. Axiome und Facts werden direkt aus der Menge der potentiellen Axiome bzw. der Facts in den Planungssachbereich übernommen – von den Axiomen allerdings nur diejenigen, die sich anhand der für die Aufgabe relevanten View verifizieren lassen. Dies ist für die Erzeugung eines benutzerspezifischen Plans die View SBUC, die die Systemannahmen über das Handlungspotential des Benutzers repräsentiert. Aus dieser View werden auch die Planoperatoren abgeleitet. Neben einer Umformung der beschreibungslogischen Repräsentation in die Notation des Planungssystems, müssen implizite Effekte (aus der Menge der potentiellen impliziten Effekte) hinzugefügt werden, sofern sie sich verifizieren lassen, und die Vorbedingungskomponenten der Operatoren müssen in eine (durch die Menge der Vorrang-Muster bestimmte) geeignete Reihenfolge gebracht werden.

Die insgesamt dafür notwendige Funktionalität ist in der neuen BGP-MS Komponente *Plan-Proc*(essing) zusammengefaßt. Mit dazu gehört auch die Beurteilung von Plan-Concepts, um – wie in Kap. 4.5 *Optimierte Plangenerierung* vorgeschlagen – *nicht nutzbare Plan-Concepts*[10] von der Verarbeitung ausschließen zu können. Dafür muß die Komponente auch die Views für das Wissen des Systems (SB) und für die Annahmen des Systems über das Benutzerwissen (SBUB) berücksichtigen. Die entsprechenden Informationsflüsse sind in Abb. 5.1 grün gestrichelt.

Für den Zugriff auf den zu einer View gehörenden Planungssachbereich wurde die Retrieval-Schnittstelle von BGP-MS um die Funktion *knpp-view-2-planning-domain* erweitert, die im Abschnitt 5.3.2 *Benutzerspezifische Pläne* beim Aufruf des Planungssystems genutzt wird.

Die Generierung eines Plans – als Lösung des Planungsproblems (der Zielsetzung) – anhand des erzeugten Planungssachbereichs ist Aufgabe des Planungssystems. Das Planungssystem wird in dieser Arbeit weitgehend als “Black-Box” Komponente betrieben, um die Festlegung auf ein spezielles System zu vermeiden. Damit soll eine Nutzung der Fortschritte in der Forschung zur Handlungsplanung durch einen Austausch dieser Komponente erleichtert werden. Als Möglichkeit der Einflußnahme auf den Planungsprozeß wurde in Kap. 4.5.2 *Steuerung des Planungsprozesses* lediglich die

[9] vgl. Kap. 3.4.1 *Das Planungsproblem*

[10] vgl. Definition 4.3 *Nutzbares Planungswissen* S. 183

Bewertung von Plankandidaten nach den für die Präsentation relevanten Kriterien vorgeschlagen. In diese Bewertung gehen neben dem Systemwissen (SB) die Annahmen des Systems über das Wissen (SBUB) und das Handlungspotential (SBUC) des Benutzers ein. Die entsprechenden Informationsflüsse sind in Abb. 5.1 rot eingezeichnet.

Das Ergebnis des Planungsprozesses – der benutzerspezifische Plan – ist Ausgangspunkt für die Präsentationskomponente. Deren Aufgabe ist das Ermitteln und Bereitstellen der Information, die der jeweilige Benutzer für die Ausführung des Plans benötigt. Neben den Schritten des Plans gehören dazu auch Informationen über die Planstruktur und ggf. Erklärungen, wie die Schritte (vom jeweiligen Benutzer) ausgeführt werden können. In Kap. 4 *Präsentation eines Plans* wurde beschrieben, wie diese Information unter Berücksichtigung des Benutzermodells ermittelt werden kann. Die zugehörigen Informationsflüsse sind in Abb. 5.1 blau eingezeichnet. Wissensquellen sind dabei die Systemannahmen über das Wissen (SBUB) und das Handlungspotential (SBUC) des Benutzers. Dazu kommt das eigene Wissen des Systems (SB), das für Erklärungen an den Benutzer und für die Beurteilung des Benutzerwissens benötigt wird. Da Planbeschreibungen sinnvollerweise die Begrifflichkeit der Handlungsplanung und nicht der Beschreibungslogik verwenden, nutzt die Präsentationskomponente auch die Komponente *Plan-Proc*, die ihre Funktionalität wiederum über die erweiterte BGP-MS Retrieval-Schnittstelle zur Verfügung stellt. Allerdings muß hierfür kein kompletter Planungssachbereich erzeugt werden. Vielmehr ist der Zugriff auf einzelne Plan-Concepts sowie deren Bestandteile (bspw. Vorbedingungs- und Effektkomponenten, Dekompositionen) und Eigenschaften (bspw. atomare Kenntnis des Benutzers) notwendig[11]. Dabei berücksichtigt *Plan-Proc* auch wieder die Menge der potentiellen impliziten Effekte, um dem Benutzer bspw. relevante Effekte eines Plan-Concepts auch dann nennen zu können, wenn sie nicht explizit repräsentiert sind. Für eine vereinfachte Präsentation der Causal Links eines Plans, wie sie am Ende von Kap. 4.1 *Planstruktur* vorgeschlagen wurde, müssen außerdem die für die jeweilige View geltenden Axiome berücksichtigt werden.

Die konkrete Darstellung der Informationen war nicht Aufgabe dieser Arbeit. Auf entsprechende Möglichkeiten wurde zu Beginn von Kap. 4 hingewiesen.

5.1.3 Plangenerierung mit UCPOP

Zielsetzung dieser Arbeit war die Integration von Planungswissen und Planungsverfahren in den Rahmen logikbasierter Benutzermodellierung, nicht aber die Weiterentwicklung der Planungsverfahren selbst. Um die Auswirkungen eines Benutzermodells auf die Generierung benutzerspezifischer Pläne zeigen zu können, wurde deshalb ein *verfügbares* Planungssystem ausgewählt. Der grobe Rahmen wurde dafür in Kap. 2.7 *Abschließende Bemerkungen und Zusammenfassung* durch die Entscheidung für die klassische Handlungsplanung gesetzt. Da die Benutzeradaption der Plangenerierung durch die Erzeugung eines geeigneten Planungssachbereichs realisiert wurde, muß das Planungssystem selbst lediglich auf Basis eines Planungssachbereichs einen korrekten und (möglichst) vollständigen Plan generieren können und eine (möglichst einfache) Schnittstelle zur Definition des Planungssachbereichs bieten. Beide Aufgaben müssen bei heutiger Rechnertechnik und für den gedachten Einsatzbereich[12] so schnell bewältigt werden, daß sich für ein interaktives System keine inakzeptablen Antwortzeiten ergeben. Für die Präsentation des Plans werden dessen Schritte und Causal Links benötigt (vgl. Kap. 4.1 *Planstruktur*). Das Planungssystem muß deshalb nicht nur in der Lage sein eine Folge von Planschritten zu ermitteln, sondern muß die vollständige Ergebnisdatenstruktur zur Verfügung stellen. Da die Möglichkeit Axiome im Planungssachbereich definieren zu können die Beschreibung von Planoperatoren merklich vereinfacht, wurde diese Möglichkeit als weiteres Auswahlkriterium

[11] vgl. insb. Kap. 4.2.1 *Dekomposition eines Plan-Concepts*

[12] Im Rahmen von Hilfesystemen ist mit einigen zig Planoperatoren und Plänen mit max 10-15 Schritten zu rechnen.

herangezogen. Nicht zuletzt sollte sich das System mit vertretbarem Aufwand in die BGP-MS-Umgebung integrieren oder ankoppeln lassen.

Die Wahl fiel auf UCPOP [Penberthy und Weld, 1992] von der Univ. Washington, das ein erweitertes Partial-Order-Planning Verfahren[13] implementiert. Dieses System steht im (Common-Lisp) Quellcode zu Forschungszwecken zur Verfügung und kann praktisch unverändert[14] zusammen mit BGP-MS in einem Lisp-Prozeß betrieben werden. Der UCPOP Planungsalgorithmus ist nachweislich korrekt und vollständig, womit die Voraussetzungen dafür erfüllt sind, daß eine korrekte Lösung für eine Zielsetzung gefunden wird, wenn es eine gibt[15]. UCPOP implementiert vielfältige Erweiterungen des POP-Basisalgorithmus SNLP [McAllester und Rosenblitt, 1991]. U.a. werden Axiome und Facts (wie in Kap. 3.4.2 *Der Planungssachbereich* beschrieben) beim Planungsprozeß berücksichtigt und für Vorbedingungen und Effekte von Planoperatoren ist eine (gegenüber der STRIPS-Repräsentation) sehr ausdrucksstarke Beschreibung bis hin zu bedingten Effekten zulässig. Von den ausdrucksstarken Beschreibungen der Planoperatoren wurde in dieser Arbeit allerdings kein Gebrauch gemacht. Trotz der vielfältigen Möglichkeiten von UCPOP zeigten die Experimente, daß sich die Antwortzeiten in Grenzen halten. Für die längsten Beispielpläne (s.a. Kap. 5.3.2 *Benutzerspezifische Pläne*) waren deutlich weniger als 10 sec CPU-Zeit auf einer (zum Zeitpunkt der Abgabe der Arbeit) veralteten Sun UltraSparc 1 (167Mhz) Workstation notwendig. Es ist zu erwarten, daß diese Zeiten durch Fortschritte der Rechner-Technik auf ein (für ein interaktives System) erträgliches Maß reduziert werden können. Die Angabe des Planungssachbereichs erfolgte ohne merkliche Verzögerung.

Im Rahmen der klassischen Handlungsplanung standen vor allem die folgenden Verfahren als Alternative zu POP-Verfahren zur Diskussion

- hierarchische Verfahren (mit Operatorabstraktion)
- deduktive Verfahren
- Graphplan-basierte Verfahren

Der Vorteil hierarchischer Verfahren[16] besteht vor allem darin, im Normalfall schneller eine Lösung eines Planungsproblems zu finden und/oder komplexere Probleme lösen zu können. Dieser Vorteil gegenüber nichthierarchischen Verfahren ergibt sich daraus, daß die Lösungssuche durch die Expansion abstrakter Operatoren einschränkt wird. Im Idealfall betrifft die Einschränkung nur solche Pfade, die nicht oder nur zu einer ungünstigen Lösung führen. Die möglichen Expansionen sind durch die Dekompositionen vorgegeben, die wiederum durch den Entwickler des Sachbereichsmodells definiert werden. Dadurch bringt der Entwickler Wissen über "übliche" gute Lösungen in das Sachbereichsmodell ein. Dies ist aber nur dann sinnvoll, wenn der Entwickler die Möglichkeiten im Sachbereich zu handeln (möglichst) genau kennt. Gerade dies widerspricht allerdings dem Ansatz dieser Arbeit, der ja davon ausgeht, daß die Handlungsmöglichkeiten der verschiedenen Benutzer unterschiedlich ist und sich die Annahmen des Systems über die Handlungsmöglichkeiten des einzelnen Benutzers während eines Dialogs ändern können. Es sei angemerkt, daß manche hierarchische Planungssysteme (bspw. UMCP [Erol *et al.*, 1994b]) in der Lage sind auch nichthierarchisch zu planen, wenn durch

[13] vgl. Kap. 2.1.1 *Least Commitment Planning*

[14] Es war lediglich eine kleine syntaktische Änderung in einem Modul notwendig, um einen Fehler im CMU-Lisp-Compiler zu umschiffen.

[15] Es sei allerdings noch einmal darauf hingewiesen, daß sich die Korrektheit nur auf das Modell bezieht, das notwendigerweise unvollständig ist (vgl. Kap. 2.1.3 *Voraussetzungen beim Planen*). Auch die Vollständigkeit ist kritisch zu sehen, weil Vollständigkeitsbeweise in der Handlungsplanung i.a. darauf beruhen, daß im Planungssachbereich nur endlich viele Zustände möglich sind, die – wenn überhaupt – mit endlich langen Plänen erreichbar sind. In der Praxis ist die daraus resultierende Zahl verschiedener Pläne meist so groß, daß im Planungsprozeß nicht tatsächlich alle Möglichkeiten untersucht werden können.

[16] vgl. Kap. 2.1.2.2 *Operatorabstraktion*

die Expansion abstrakter Operatoren keine Lösung erreicht werden kann. Damit entfällt allerdings auch der Vorteil des hierarchischen Verfahrens. Dem Ziel einer guten Benutzeradaption steht zudem entgegen, daß zunächst Lösungen mithilfe von Operatorexpansion gesucht werden, die i.a. nur zu Lösungen führen, die nach Kenntnis des Entwicklers *üblicherweise* gut sind. Gute Lösungen für den speziellen Benutzer können so unentdeckt bleiben.

Die ersten Planungsverfahren überhaupt können als *deduktive Verfahren* angesehen werden. In Kap. 2.1 *Handlungsplanung* wurden die Nachteile dieser ersten Planungsansätze aufgezeigt, die schließlich dazu geführt haben, daß deduktive Verfahren in der Handlungsplanung (zumindest für den praktischen Einsatz) kaum mehr Beachtung finden. Durch den Einsatz taktischer Theorembeweiser (wie bspw. im PHI-Projekt[17]) kann der Beweisprozeß zwar dahin geleitet werden hauptsächlich Schlußfolgerungen zu ziehen, die für den Planungsprozeß nützlich sind. Ein Nachweis, daß sich damit praktische Problem lösen lassen oder daß sich sogar ein Vorteil gegenüber den anderen Planungsverfahren ergeben würde, konnte aber bislang nicht geführt werden (vgl. a. [Biundo, 1993]). Verfechter dieses Verfahrens heben die nachweisliche Korrektheit des Verfahrens hervor. Im Rahmen der für Planungssysteme zulässigen Modellierung kann die Korrektheit allerdings auch für die anderen Verfahren nachgewiesen werden. Da bei deduktiven Planungsverfahren die volle Prädikatenlogik erster Stufe als Modellierungsmittel zur Verfügung steht, kann die Realität präziser modelliert werden, als dies bei den anderen Verfahren möglich ist. Insbesondere Abhängigkeiten im Sachbereich können so besser berücksichtigt werden. Wird diese Möglichkeit allerdings ausgenutzt, wird im Planungsprozeß jede einzelne Fragestellung der Art “ist die Vorbedingung eines Planschritts erfüllt oder nicht” zu einem semi-entscheidbaren prädikatenlogischen Problem.

Graphplan-basierte Planungsverfahren zeichnen sich vor allem durch eine sehr hohe Geschwindigkeit aus. Ein weiterer Vorteil ist die Unabhängigkeit der Lösungssuche von der Reihenfolge, in der einzelne Vorbedingungskomponenten der Operatoren angegeben sind (vgl. [Blum und Furst, 1995]). In Bezug auf die Entscheidung über den Einsatz eines Planungssystems in dieser Arbeit war der entscheidende Nachteil, daß stabile Planungssysteme zum Zeitpunkt der Entscheidung noch nicht zur Verfügung standen. Da dieser Nachteil inzwischen behoben ist, ist es bei der Weiterentwicklung des Prototyps auf jeden Fall sinnvoll zu erproben, ob ein solches Planungssystem gegen UCPOP ausgetauscht werden kann. Ein weiterer Nachteil graphplan-basierter Verfahren besteht darin, daß ein Hinzufügen neuer Konstanten – wie sie in UCPOP für das Erzeugen neuer Objekte genutzt werden kann – sehr viel schwieriger zu realisieren ist, als beim POP-Verfahren und dementsprechend (bislang) auch bei keinem System verwirklicht ist. Als Ausweg bietet es sich an, vor Beginn des Planungsprozesses eine “ausreichende” Zahl von Konstanten des Typs *non-existent*[18] zu definieren, die im Planungsprozeß für neue Objekte verwendet werden können. Für Anwendungs- oder Problembereiche, für die eine sinnvolle Zahl solcher Konstanten nicht angegeben werden kann, kann UCPOP auch als alternatives zweites Planungssystem bereitgehalten werden.

5.1.4 Eigenschaften der Algorithmen

Im Verlauf der Arbeit wurde für die verschiedenen Algorithmen gezeigt, daß sie terminieren – sofern dies nicht offensichtlich war. In diesem Abschnitt wird nun noch die Laufzeit-Komplexität der wichtigsten Algorithmen, die für die Verarbeitung einer Benutzeranfrage erforderlich sind, betrachtet. Nicht betrachtet werden dagegen die Verarbeitungsschritte für den Aufbau der BMWB in der Entwicklungsphase. Die dafür erforderlichen Eintragungen gemäß der erweiterten BGP-MS Schnittstelle (s.o.) werden in BGP-MS Funktionsaufrufe nach bisheriger Schnittstelle umgeformt und bzw. oder führen zu Eintragungen in die neuen Datenbestände, d.h. in die Mengen der *potentiellen Axio-*

[17] vgl. [Bauer *et al.*, 1993] und Kap. 2.5.3 *Erzeugen von Plänen*

[18] Dieser Typ wurde in Kap. 3.4.2.5 *Erzeugen von Objekten* für “noch nicht erzeugte Objekte” eingeführt.

me, der *potentiellen impliziten Effekte*, der *Vorrang-Muster* oder der *Facts*. Gegenüber der bisherigen Verarbeitung durch BGP-MS fällt damit an zusätzlichem Aufwand der Aufwand für die Umformung in BGP-MS Funktionsaufrufe und für die Eintragungen in die neuen Datenbestände an. Für beide Fälle hängt dieser Aufwand linear von der Zahl der Komponenten der Beschreibungen ab. Ebenfalls nicht betrachtet wird der Aufwand für die Verwaltung von Veränderungen des Modells, die durch neue Erkenntnisse über den Benutzer veranlaßt werden. Hierfür wird die bisherige Funktionalität von BGP-MS (basierend auf Pohls Dissertation [Pohl, 1997]) ohne Änderung eingesetzt. Die o.g. neuen Datenbestände verändern sich im Laufe einer Interaktion mit einem Benutzer nicht.

5.1.4.1 Initiale Bewertung und Beurteilung von Plan-Concepts

Für den Ausschluß nicht nutzbarer Plan-Concepts (vgl. Kap. 4.5) muß vor der Erzeugung des Planungssachbereichs eine Beurteilung der Plan-Concepts hinsichtlich ihrer Nutzbarkeit (vgl. Def. 4.3 S. 183) vorgenommen werden. Für die Bewertung von Plankandidaten während des Planungsprozesses wurde in Kap. 4.5.2 S. 215 die Funktion *estimate-pres-effort* vorgeschlagen, die Werte verwendet, die vor Beginn des Planungsprozesses für jedes Plan-Concept berechnet werden müssen. Dies sind *die Zahl der dem Benutzer nicht bekannten Vorbedingungen*, *die Menge der (ggf. indirekten) Teil-Concepts* (indirekt durch Funktion *spc*) sowie *der Anteil der Präsentationsbewertung des Plan-Concepts selbst* ohne evtl. erforderliche Erklärungen von Teil-Concepts. Für den Zugriff wurden die Funktionen *unknown-preconds*, *part-plan-concepts* und *expl-effort* definiert. Die Berechnung der letzten beiden beruht (u.a.) auf der Einschätzung, welche Plan-Concepts für den Benutzer *nutzbar* sind und über welche Plan-Concepts er *verläßliches* Wissen hat. Letzteres beruht wiederum auf einer Einschätzung, ob der Benutzer korrekte Dekompositionen von Plan-Concepts kennt und ob er Dekompositionen selbständig nutzen kann (vgl. die Definitionen 4.1 S. 181 sowie 4.4 und 4.5 S. 184). Die initiale Beurteilung und Bewertung von Plan-Concepts erfolgt deshalb sinnvollerweise in folgender Reihenfolge:

1. Bestimmung, welche Plan-Concepts für den Benutzer nutzbar sind und welche nicht.
2. Bestimmung, welche der Dekomposition, die der Benutzer kennt, auch korrekt sind.
3. Bestimmung, zu welchen Plan-Concepts der Benutzer *verläßliches* Wissen über die Ausführungsmöglichkeiten hat und welche Dekompositionen für ihn *selbständig nutzbares* Wissen darstellen.
4. Bestimmung der Zahl der dem Benutzer nicht bekannten Vorbedingungen für jedes Plan-Concept.
5. Berechnung der Menge *part-plan-concepts* für jedes Plan-Concept.
6. Berechnung der Funktion *expl-effort* für jedes Plan-Concept.

Die weiteren Betrachtungen setzen voraus, daß die Dekompositionshierarchie zyklenfrei ist[19].

Die Nutzbarkeit eines nicht-atomaren Plan-Concepts aus dem Handlungspotential des Benutzers hängt davon ab, ob das System eine Dekomposition kennt, die für den Benutzer nutzbar ist. Die Nutzbarkeit einer Dekomposition hängt wiederum davon ab, ob sie nur nutzbare Plan-Concepts enthält. Da Zyklenfreiheit der Dekompositionshierarchie vorausgesetzt ist, muß jedes nicht-atomare Plan-Concept $pc \in$ SBUC nur genau einmal beurteilt werden, wozu im ungünstigsten Fall alle Dekompositionen $d \in dekomp_{SB}(pc)$ betrachtet werden müssen. Insgesamt können dies höchstens alle Dekompositionen aus SB sein, da jedes Plan-Concept $pc \in$ SBUC auch Element von SB ist. Für die Beurteilung einer einzelnen Dekomposition muß höchstens jeder ihrer Schritte berücksichtigt werden. Eine obere Grenze für die Zahl der zu berücksichtigenden Komponenten ist demnach die Gesamtzahl der

[19] vgl. Kap. 4.2.2 *Beurteilung von Wissen, wie ein Plan-Concept ausgeführt wird*

Plan-Concepts aus SBUC plus der Dekomposition aus SB plus der Schritte, die in Dekompositionen von SB verwendet werden.

Die Korrektheit einer Dekomposition ist in Def. 4.1 S. 181 rekursiv definiert. Bei Zyklenfreiheit der Dekompositionshierarchie kann die Beurteilung aller Dekompositionen durch einen rekursiven Algorithmus realisiert werden, der jede Dekomposition genau einmal bearbeitet. Für die Beurteilung einer Dekomposition *d* eines Plan-Concepts *pc* ist gefordert, daß die Menge der Linearisierungen der Schritte von *d* in der Menge der Linearisierungen aller Dekompositionen von *pc* in SB enthalten sein muß. Der Mengeninklusionstest "$A \subseteq B$" läßt sich (bei geeigneter Vorsortierung) durch maximal $|B|$ Vergleiche realisieren. Die Zahl der Linearisierungen einer Dekomposition mit *s* Schritten ist im ungünstigsten Fall *Fakultät(s)*. Nach Definition 4.1 müssen allerdings nur gleich lange Dekompositionen betrachtet werden. Deshalb kann es höchstens zu einem Vergleich von zwei Dekompositionen kommen, deren Länge gleich dem kleineren Wert für die Maximallänge einer Dekomposition aus SB und SBUC ist. Sei dieser Wert *minmaxl*, sind für die Beurteilung der Korrekteit aller d_{SBUB} Dekomposition, die der Benutzer kennt höchstens

$$d_{SBUB} * Fakultät(minmaxl).$$

Vergleiche erforderlich.

Allerdings ist diese obere Schranke für die Praxis wenig aussagekräftig, weil es kaum vorkommen dürfte, daß ein Plan-Concept in viele Schritte zerlegt werden kann, die dann in beinahe[20] beliebiger Reihenfolge ausgeführt werden dürfen. Um sicher zu verhindern, daß es zur Laufzeit – also bei der Beantwortung einer Benutzerfrage – zu sehr vielen Vergleichen kommt, muß bei der Modellierung sichergestellt werden, daß die Dekompositionen eines Plan-Concepts aus SB mit gleichen Schritten nur begrenzt viele Linearisierungen haben. Da in dieser Arbeit davon ausgegangen wird, daß die direkten Annahmen des Systems (im Gegensatz zu Annahmen über den Benutzer) unveränderlich sind, ist eine solche Höchstgrenze dann auch die Maximalzahl der Vergleiche, die für die Korrektheitsbestimmung einer einzelnen Dekomposition notwendig sind.

Alternativ kann die Zahl der notwendigen Vergleiche eingeschränkt werden, wenn das Systemwissen für eine gegebene Schrittmenge *sm* höchstens *eine* Dekomposition eines Plan-Concepts enthalten darf. Für die Bestimmung der Korrektheit einer Dekomposition aus SBUB, die die gleichen Schritte hat, reicht es dann aus, die zugehörigen Ordnungsrelationen zu vergleichen, die aus höchstens $|sm|^2$ Elementen bestehen können.

Im dritten Schritt liegen die Beurteilung der Korrektheit für jede Dekomposition aus SBUB bereits vor. Dann ist die Beurteilung, ob der Benutzer *verläßliches* Wissen über die Ausführungsmöglichkeiten eines Plan-Concepts hat und ob er *selbständig nutzbares* Wissen über eine Dekomposition hat, völlig analog zur Beurteilung von *nutzbarem* Wissen. Der Unterschied besteht lediglich darin, daß hier das Wissen des Benutzers und nicht das des Systems relevant ist. Die Zahl der Beurteilungschritte berechnet sich deshalb in gleicher Weise wie oben, wobei allerdings die Zahl der Dekompositionen und Schritte im Benutzerwissen verwendet werden muss.

Die Bestimmung der Zahl der dem Benutzer *unbekannten Vorbedingungen* der Plan-Concepts kann durch einfachen Vergleich der Plan-Concepts in SB und SBUB durchgeführt werden. Er hängt damit linear von der Zahl der Plan-Concepts in SBUB ab.

Für die Bestimmung der Teil-Plan-Concepts eines Plan-Concepts sieht Kap. 4.5.2 *Steuerung des Planungsprozesses* die Verwendung der Funktion *present-plan-concept* aus dem Präsentationsalgorithmus vor. Die Abschätzung des Aufwandes sei deshalb bis zur Behandlung des Präsentationsalgorithmus (s.u.) zurückgestellt.

[20] Falls die Schritte in *jeder* Reihenfolge ausgeführt werden dürfen, reicht ein Test, ob die beiden zu vergleichenden Dekomposition aus gleichen Schritten bestehen.

Es bleibt noch die Berechnung der Funktion *expl-effort*. Der Algorithmus aus Abb. 4.8 S. 216 arbeitet in gleicher Weise wie die Funktion *present-plan-concept* aus dem Präsentationsalgorithmus. Die aufwendigste Berechnung ist die Bestimmung der für den jeweiligen Benutzer besten Ausführungsalternative; d.h. die Auswahl der Dekomposition mit der günstigsten Bewertung. Diese Auswahl erfolgte aber bereits für die Bestimmung der Teil-Plan-Concepts eines Plan-Concepts. Da auf diese Entscheidung zurückgegriffen werden kann, fällt hier kein erneuter Aufwand an. Im ungünstigsten Fall, also wenn der Benutzer eine Erklärung für das Plan-Concepts benötigt, bleibt damit lediglich die Aufsummierung der Anteile, die die Teil-Plan-Schritte (ohne weitere Erklärung) zur Bewertung beitragen (Schritte e7-e8). Der Aufwand für die Berechnung der Funktionswerte für alle Plan-Concepts aus SBUC hängt also linear von der Zahl dieser Plan-Concepts * Zahl der Schritte der längsten Dekomposition in SB ab und liegt damit unter dem Aufwand für die Bestimmung der Nutzbarkeit.

5.1.4.2 Erzeugung eines Planungssachbereichs

In Kap. 3.4.2 *Der Planungssachbereich* wurde die Erzeugung eines Planungssachbereichs im Detail beschrieben. Die folgenden Betrachtungen zur Komplexität der einzelnen Schritte bezieht sich immer auf eine *aktuelle View*, anhand der der Planungssachbereich erzeugt wird.

5.1.4.2.1 Axiome Axiome, die für den Planungssachbereich erzeugt werden, entstammen der *Menge der potentiellen Axiome* (s.o.). Jedes Element dieser Menge wird dahingehend überprüft, ob es in der jeweils aktuellen View gültig ist. Nur im positiven Fall wird es als Axiom in den Planungssachbereich übernommen. Da Beschreibungen potentieller Axiome nach der Syntax aus Anhang A.2.4 *Beschreibung von Axiomen für den Planungssachbereich* auch der Syntax von UCPOP genügen, ist keine syntaktische Umformung notwendig.

In Kap. 3.4.2.4 *Ableitung von Abhängigkeiten aus dem Benutzermodell* wurden die Formen von Abhängigkeiten behandelt, die sich aus der BMWB ableiten und im Planungssachbereich durch Axiome ausdrücken lassen:

1. $\forall x\, S(x) \rightarrow A(x)$
2. $\forall x\, (\exists y\, T_1(y) \wedge R(y,x)) \rightarrow T_2(x)$
3. $\forall x\, (\exists y\, R(x,y) \wedge AT(y) \wedge B(x)) \rightarrow S(x)$

Die Überprüfung der Gültigkeit aller drei Formen erfordert jeweils die Berechnung einer Subsumptionsbeziehung. In Kap. 3.4.2.6 *Vereinfachte beschreibungslogische Inferenzen* wurde vorgeschlagen, die i.a. sehr aufwendigen Berechnungen durch Tests zu substituieren, die auf Abfragen der SB-ONE Wissensbasis beruhen. Diese Tests kommen für die Erzeugung des Planungssachbereichs zum Einsatz und sollen nun bzgl. ihrer Laufzeit-Komplexität beurteilt werden.

Für die erste Form muß festgestellt werden, ob S von A subsumiert wird. Die entsprechende Abfrage an die SB-ONE Wissensbasis lautet, ob S ein direktes oder indirektes Sub-Concept von A ist. Dies erfordert Vergleiche von allen Super-Concepts von S, die auf einem Pfad zur Wurzel in der Abstraktionshierarchie liegen. Hat die Abstraktionshierarchie eine Baumstruktur sind also maximal Tiefe(Abstraktionshierarchie) Vergleiche erforderlich; ansonsten müssen im ungünstigsten (und eher untypischen) Fall alle Domain-Concepts betrachtet werden.

Für die zweite Form muß festgestellt werden, ob Concept T_1 eine Rolle R hat, deren Werterestriktion VR von T_2 subsumiert wird. Rollen und Werterestriktionen sind direkt in einer SB-ONE Wissensbasis repräsentiert und können demnach mit konstantem Aufwand abgefragt werden. Allerdings fällt noch der Aufwand für die Überprüfung der Subsumption $VR \Rightarrow T_2$ an, wozu wieder maximal Tiefe(Abstraktionshierarchie) bzw. | Abstraktionshierarchie | Vergleiche erforderlich sind.

Für die dritte Form, deren Gültigkeit durch den Subsumptionstest $\exists R : AT \sqcap B \Rightarrow S$ verifiziert werden kann, wurde in Kap. 3.4.2.6 ein Test folgender Bedingungen vorgeschlagen:

a) S ist definiertes Sub-Concept von B

b) S unterscheidet sich von B höchstens durch die Werterestriktion VR der Rolle R

c) R ist ein Attribut (bzw. eine [1 1] anzahlbeschränkte Rolle) von B

d) $AT \Rightarrow VR$

S muß nicht als direktes Sub-Concept von B definiert sein. Die Ableitung ist auch dann gültig, wenn $S \doteq Z_1 \sqcap a_1$, $Z_1 \doteq Z_2 \sqcap a_2$, ... $Z_n \doteq B \sqcap a_{n+1}$ Wichtig ist dabei insbesondere, daß b) gilt; d.h. die beschreibungslogischen Ausdrücke a_i dürfen lediglich Werterestriktionen von R darstellen.

Der Test kann durch einen Algorithmus realisiert werden, der die Bedingungen zunächst für das Super-Concept von S prüft und falls dieses weder B noch die Wurzel der Abstraktionshierarchie ist, durch einen rekursiven Aufruf das Super-Concept bearbeitet. Im ungünstigsten Falle sind maximal Tiefe(Abstraktionshierarchie) bzw. | Abstraktionshierarchie | Aufrufe erforderlich. Für jeden Aufruf ist zudem ein Subsumptionstest (Bedingung d)) erforderlich. Allerdings sind für die Subsumptionstests *aller* Aufrufe höchstens die Concepts auf einem Weg von AT zur Wurzel der Abstraktionshierarchie erforderlich.

Für alle zulässigen Formen potentieller Axiome hängt der Überprüfungsaufwand damit höchstens linear von der Größe des Sachbereichsmodells ab (genauer: von der Anzahl der Domain-Concepts in der entsprechenden View). Da der Aufwand für jedes potentielle Axiom anfällt, ergibt sich der Gesamtaufwand durch Multiplikation mit der Zahl der potentiellen Axiome.

5.1.4.2.2 Facts Facts werden unverändert aus der BMWB in den Planungssachbereich übernommen. Der Aufwand ist damit linear von der Länge der Beschreibung abhängig.

5.1.4.2.3 Planoperatoren Für die Erzeugung eines Planoperators gemäß UCPOP-Syntax aus einem Plan-Concept wurde in Kap. 3.4.2.7 *Erzeugung des aktiven Teils eines Planungssachbereichs* ein vierstufiges Verfahren beschrieben. Der erste Schritt erzeugt die Parameter, Vorbedingung und Effekt direkt aus der beschreibungslogischen Repräsentation. Der Aufwand hängt also von der Länge dieser Beschreibung ab. Da mehrere Vorkommen der gleichen Variablen nur zu einem Parameter führen dürfen, ist diese Abhängigkeit quadratisch[21].

Im zweiten Schritt wird geprüft, welche Elemente aus der *Menge der potentiellen impliziten Effekte* in der aktuellen View gültig sind und dem jeweiligen Operator hinzugefügt werden müssen. Implizite Effekte, wie sie in Kap. 3.4.2.4.4 *Ausschließende Eigenschaften bzw. Typzugehörigkeiten* beschrieben wurden, entsprechen einer Formel der Form:

$$\forall x\ T_1(x) \rightarrow (\neg T_2(x) \wedge \ldots \wedge \neg T_n(x))$$

Solche Formeln sind gültig, wenn in der aktuellen View T_1 von $\neg T_2 \sqcap \ldots \sqcap \neg T_n$ subsumiert wird bzw. wenn $T_1 \parallel T_2$ und ... und $T_1 \parallel T_n$. Wie schon für die Überprüfung von potentiellen Axiomen, wird dafür ein vereinfachter Test aus Kap. 3.4.2.6 *Vereinfachte beschreibungslogische Inferenzen* eingesetzt. Dabei wird für jedes der Paare T_1, T_i geprüft, ob es eine *SB-ONE Disjunkt-Markierung* von T_1 oder einem seiner Super-Concepts zu T_i oder einem seiner Super-Concepts gibt. Dies kommt einem Test auf Disjunktheit von zwei (ungeordneten) Mengen gleich, für den es $O(n * log(n))$ Algorithmen gibt.

[21] Es gibt auch $O(n * log(n))$ Algorithmen für diese Aufgabe (wobei n die Anzahl der Variablen ist). Da diese Algorithmen aber aufwendiger sind und in der Praxis eher kleine Werte von n vorkommen, wäre dadurch kein merklicher Vorteil zu erwarten.

In diesem Falle wäre n das Maximum der Anzahlen der Super-Concepts von T_1 und T_i für die eine Disjunkt-Markierung eingetragen ist plus 1. Für die Arbeit wurde allerdings ein einfacher $O(n^2)$ Algorithmus eingesetzt, da kaum zu erwarten ist, daß diese Zahl sehr groß wird. Im Beispielmodell (vgl. Kap. 5.2.3 *Initialer Aufbau einer Benutzermodellierungswissensbasis* S. 262) haben 13 Concepts eine Disjunkt-Markierung.

Die Gültigkeit der Formeln hängt von der aktuellen View ab, aber nicht von dem Plan-Concept, das gerade verarbeitet wird. Deshalb wird die Gültigkeit aller potentiellen impliziten Effekte unmittelbar bevor die Verarbeitung der einzelnen Plan-Concepts beginnt überprüft. Der Aufwand dafür hängt dann höchstens mit $e * d^2$ von d, der Zahl der Concepts mit Disjunkt-Markierung und e, der Zahl der (verschiedenen[22]) *elementaren* potentiellen impliziten Effekte ab. Unter letzterem werden alle Implikationen der Form $\forall x\ T_1(x) \rightarrow \neg T_2(x)$ verstanden, die sich aus den potentiellen impliziten Effekten ableiten lassen.

Für die Verarbeitung eines einzelnen Plan-Concepts bleibt dann noch die Aufgabe, für jedes einstellige Effekt-Prädikat zu prüfen, ob das Prädikat Prämisse eines der als gültig eingestuften elementaren potentiellen impliziten Effekte ist. Im positiven Falle wird das Konklusions-Prädikat zum Effekt hinzugefügt[23]. Der Berechnungsaufwand ergibt sich aus der Verarbeitung der einstelligen Effekt-Prädikate und dem Suchen in der Menge der potentiellen impliziten Effekte. Letzteres läßt sich bei geeigneter Datenstruktur (bspw. Hash-Tabelle) auf eine kleine Zahl von Vergleichen begrenzen. Insgesamt hängt der Aufwand für das Hinzufügen der impliziten Effekte also linear von der Länge der Beschreibung des Plan-Concepts ab.

Der dritte Verarbeitungsschritt ordnet die einzelnen n Vorbedingungskomponenten des Planoperators anhand eines Vorrang-Musters, sofern eines dafür definiert wurde. Die Verarbeitung iteriert über alle m Elemente des Vorrang-Musters. Ist ein solches Element auch eine Vorbedingungskomponente des Planoperators, wird es einer Liste von geordneten Vorbedingungskomponenten angehängt und gleichzeitig die Vorbedingungskomponente selbst gelöscht. Nach der Iteration wird die Liste von geordneten Vorbedingungskomponenten vor die verbliebenen Vorbedingungskomponenten des Planoperators eingefügt. Insgesamt sind also $n * m$ Schritte erforderlich. Gehen wir davon aus, daß m normalerweise höchstens um einen (kleinen) konstanten Faktor größer als n ist, hängt der Verarbeitungsaufwand höchstens quadratisch von der Zahl der Vorbedingungskomponenten oder allgemeiner von der Länge der Beschreibung des Operators ab[24].

Der letzte Verarbeitungsschritt besteht lediglich in einer syntaktischen Aufbereitung, deren Aufwand linear von der Länge der Beschreibung des Planoperators abhängt.

Insgesamt hängt der Aufwand für die Erzeugung eines Planoperators aus einem Plan-Concept damit höchstens quadratisch von der Länge der Beschreibung des Plan-Concepts bzw. des abgeleiteten Planoperators ab. Der Aufwand fällt für jedes nutzbare Plan-Concept an, für das außer Dekompositionen[25] keine Spezialisierung definiert ist. Der Gesamtaufwand muß also noch mit der Zahl dieser Plan-Concepts multipliziert werden.

5.1.4.2.4 Zusammenfassung: Erzeugung eines Planungssachbereichs Der Aufwand für die Erzeugung eines Planungssachbereichs aus einer View der BMWB hängt von der Größe des Sachbereichmodells ab, das durch diese View repräsentiert wird. Bestimmende Größen sind p, die Zahl der

[22] im Sinne logischer Äquivalenz

[23] Elemente aus Menge der potentiellen impliziten Effekte können damit auch teilweise berücksichtigt werden, wenn sie als vollständige Formel in der aktuellen View nicht gültig sind.

[24] Auch hier kann wieder ein $O(n * log(n))$ Algorithmus eingesetzt werden, was aber wegen der zu erwartenden geringen Zahl von Komponenten keinen merklichen Vorteil verspricht.

[25] vgl. die Einführung der *Dekompositions-Concepts* in Kap. 4.6.2 *Repräsentation von Dekompositionen in BGP-MS*.

nutzbaren Basis-Plan-Concepts[26], die Zahl der Domain-Concepts c, die Zahl der Domain-Concepts mit Disjunkt-Markierung dj, die Zahl der potentiellen Axiome a, die Zahl der elementaren potentiellen impliziten Effekte e und die maximale Länge einer Beschreibung eines Plan-Concepts l. Für die Erzeugung der Axiome ist der Aufwand dann jeweils höchstens proportional zu $a * c$, für die Feststellung der Gültigkeit impliziter Effekte proportional zu $e * dj^2$ und für die Erzeugung der Planoperatoren $p * l^2$. Dabei wird davon ausgegangen, daß die Beurteilung, ob ein Plan-Concept *nutzbar* ist, bereits vor der Erzeugung eines Planungssachbereichs durch die *initiale Bewertung und Beurteilung* (s.o.) vorliegt.

5.1.4.3 Planerzeugung

Der Zeitbedarf des UCPOP-Algorithmus hängt exponentiell von der Planlänge ab (vgl. [Penberthy und Weld, 1992, Kap. 4.2]). Dieses ungünstige Zeitverhalten ist aber nicht auf den UCPOP-Algorithmus beschränkt. Bylander zeigt in [Bylander, 1994], daß polynominale Algorithmen für die Plangenerierung nur möglich sind, wenn die Operatorbeschreibungen so stark eingeschränkt werden, daß sie für viele Sachbereiche nicht mehr anwendbar sind. Dementsprechend kann ein Ergebnis des Planungsprozesses in einem, für ein interaktives System erträglichen Zeitrahmen lediglich dann erwartet werden, wenn es eine Lösung mit wenigen Schritten gibt. Wie bereits erwähnt, haben die längsten Beispielpläne dieser Arbeit zehn Schritte und führen zu einer für ein Experimental-System gerade noch erträglichen Laufzeit. Lösungen für schwierigere Probleme, d.h. solche, die merklich mehr Schritte erfordern, sind aber nur denkbar, wenn wesentlich schnellere (bspw. Graphplan-basierte) Planungsverfahren eingesetzt werden. Es ist allerdings zu bedenken, daß sehr lange Pläne, die dementsprechend auch eine lange Beschreibung und evtl. noch Erklärungen erfordern, den Benutzer leicht überfordern können und damit dem Ziel der Benutzerunterstützung zuwider laufen.

5.1.4.4 Planpräsentation

5.1.4.4.1 Der Präsentationsalgorithmus Der Algorithmus zur Planpräsentation aus Abb. 4.2 S. 189 besteht i.w. aus der Linearisierung des Plans (Schritt p2) und einer Iteration über alle Planschritte (p3) und alle Plan-Concepts (p6), die im Plan verwendet wurden. Die Berechnungskomplexität der Linearisierung eines partiell geordneten Plans mit n Schritten ist $O(n * log(n))$[27]; die der ersten Iteration $O(n)$, da hierfür lediglich Werte aufsummiert bzw. der Präsentation hinzugefügt werden, die bereits als Wert bzw. Datenstruktur direkt abrufbar sind. Vorausgesetzt ist dabei, daß die Ergebnisse der Bewertung des Planungswissens des Benutzer bereits vorliegen.

Die zweite Iteration sammelt die Information, die für die Präsentation der Plan-Concepts des Plans notwendig ist. Maximal können n Plan-Concepts für den Plan verwendet werden – für jeden Planschritt ein anderes. Die Information selbst ist abhängig von den Kenntnissen und Handlungsmöglichkeiten des Benutzers und besteht aus einem statischen Anteil sowie der Erklärung einer Ausführungsmöglichkeit des Plan-Concepts. Letztere hat selbst wieder die Struktur eines Plans und wird durch einen rekursiven Aufruf des Präsentationsalgorithmus bestimmt. Kennt das System mehrere Ausführungsmöglichkeiten die für den Benutzer nutzbar sind, wird diejenige mit der besten Bewertung ausgewählt. Dafür müssen allerdings zunächst *alle* Bewertungen bestimmt werden; d.h. für jede Ausführungsmöglichkeit ein rekursiver Aufruf des Präsentationsalgorithmus mit einem (Teil-) Plan, der maximal die Länge $maxs_{SB}$ hat. Da auch für die Planschritte der Dekompositionen wiederum Erklärungen zu Ausführungsmöglichkeiten notwendig sein können, muß die Zahl der Aufrufe für jede weitere Stufe der Dekompositionshierarchie mit höchstens $(maxs_{SB} * maxd_{SB})$ multipliziert werden,

[26] Darunter seien Plan-Concepts verstanden, für die außer Dekompositionen keine Spezialisierungen definiert sind.
[27] vgl. [Wirth, 1999, Kap.4.3.3] für einen einfachen $O(n^2)$-Algorithmus.

wenn $maxd_{SB}$ die Maximalzahl der Dekompositionen eines Plan-Concepts in SB ist. Die Gesamtzahl der Aufrufe des Präsentationsalgorithmus ist damit höchstens

$$1 + n * maxd_{SB} * (1 + \sum_{i=1...t-1} (maxs_{SB} * maxd_{SB})^i)$$

wenn t die maximale Tiefe der Dekompositionshierarchie angibt. Es kann erwartet werden, daß diese maximale Tiefe sehr gering – d.h. 1 oder 2 – ist, weil alle *relevanten* Gegebenheiten des Sachbereichs bereits durch die Schritte des erzeugten Plans berücksichtigt sind (vgl. Kap. 4.2.1 *Dekomposition eines Plan-Concepts*). Eine weitere obere Grenze der Zahl der Aufrufe ergibt sich dadurch, daß der Präsentationsalgorithmus für jedes Plan-Concept höchstens *eine* Erklärung zuläßt (p6). Damit kann er auch höchstens einmal für jede Dekomposition des Systemwissens aufgerufen werden.

Es bliebe noch anzumerken, daß für die Mengenoperationen in (p6) und (p9) eine konstante Zahl von Schritten erforderlich ist und die Gesamtschrittzahl für (p9) linear von der Zahl der zu *cs* hinzugefügten Elemente abhängt. Da die Operationen auf die feste Menge der Plan-Concepts in SBUC bezogen sind, können die Mengenoperationen durch einen einfachen Test bzw. eine Markierung dieser Plan-Concepts realisiert werden.

Der größte Anteil an der Zahl der Berechnungsschritte für einen *einzelnen* Aufruf des Präsentationsalgorithmus ist also für die Linearisierung des Plans erforderlich. Beim ersten Aufruf ist die Planlänge n, alle weiteren Aufrufe erfolgen mit Dekompositionen aus SB; d.h. die Planlänge ist maximal $maxs_{SB}$. Die Gesamtzahl der Berechnungsschritte kann demnach auf höchstens

$$k * (n * log_2(n) + n * maxd_{SB} * (1 + maxs_{SB} * maxd_{SB}) * maxs_{SB} * log_2(maxs_{SB}))$$

abgeschätzt werden, wenn von einer Dekompositionshierarchie mit Tiefe 2 ausgegangen wird. Werden für n 10, für $maxd_{SB}$ 5 und für $maxs_{SB}$ 8 eingesetzt, ergibt die Formel 49240 * k Schritte. Der konstante Faktor k steht für die Maximalzahl der einzelnen Berechnungsschritte, die für die einzelnen Operationen des Algorithmus erforderlich sind. Er hängt von der konkreten Realisierung des Algorithmus ab. Es sei angemerkt, daß die Formel für die Gesamtzahl der Berechnungsschritte eine obere Grenze darstellt, die nur dann erreicht werden kann, wenn tatsächlich alle Plan-Concepts eines Plans in zwei Stufen dekomponiert werden können und der Benutzer von keinem der Plan-Concepts verläßliches oder atomares Wissen hat.

Es bliebe noch die Abschätzung des Aufwandes für die initiale Bestimmung der Teil-Plan-Concepts für jedes Plan-Concept des Handlungspotentials eines Benutzers (s. Abschnitt 5.1.4.1 *Initiale Bewertung und Beurteilung von Plan-Concepts*). Diese Bestimmung verwendet die Funktion *present-plan-concept* aus dem Präsentationsalgorithmus. Übernehmen wir die bisherige Argumentation und Annahmen, sind höchstens

$$k * (maxd_{SB} * (1 + maxs_{SB} * maxd_{SB}) * maxs_{SB} * log_2(maxs_{SB}))$$

Berechnungsschritte für jedes Plan-Concept erforderlich.

Die Komplexitätsbetrachtungen zum Präsentationsalgorithmus deuten darauf hin, daß die Repräsentation der Dekompositionen, die für Erklärungen an den Benutzer genutzt werden, verbessert werden kann. Diese Dekompositionen in SB sind partiell geordnete Folgen von Planschritten, die für Erklärungen zunächst linearisiert werden. Da die Linearisierungen nicht benutzerspezifisch sind, können sie auch unmittelbar bei der Modellierung des Systemwissens erfolgen und zusätzlich abgespeichert werden. Dabei kann auch das Wissen des Entwickler des Sachbereichsmodells über "*gute*" Linearisierungen einfließen. Es sei angemerkt, daß solche total geordneten Dekompositionen lediglich eine Zusatzinformation darstellen und die bisherigen Dekompositionen nicht ersetzten können, weil diese für die Beurteilung der Korrektheit des Benutzerwissens benötigt werden.

5.1.4.4.2 Der Algorithmus für das Entfernen von Komponenten aus einer Planpräsentation

Der Algorithmus aus Abb. 4.3 S. 206 hat die Aufgabe möglichst viele Komponenten aus einer Planpräsentation zu entfernen, für die sich ein positiver Nutzen ergibt. Er muß auf der anderen Seite die Löschabhängigkeiten zwischen Komponenten berücksichtigen. Diese Abhängigkeiten wurden zu Beginn von Kap. 4.4.3 *Algorithmus für das Entfernen von Komponenten aus einer Planpräsentation* durch Funktionen definiert, die Mengen von abhängigen Komponenten liefern. Die Mengen werden nicht als explizite Datenstruktur repräsentiert, sondern implizit durch die Struktur oder durch Markierungen der Plan-Komponenten. Die Komponenten, die von einem Causal Link abhängig sind, sind bspw. unmittelbar durch die beiden beteiligten Planschritte gegeben. Die Komponenten, die aus der Planpräsentation entfernt wurden bzw. die nicht entfernbar sind, werden entsprechend markiert. Die für die Beschreibung des Algorithmus verwendeten Tests, ob ein Element in einer Menge enthalten ist bzw. generell die Mengenoperationen dienen lediglich der besseren Lesbarkeit. Sie lassen sich durch Tests von Markierungen und booleschen Verknüpfungen realisieren, die höchstens eine konstante Zahl von Verarbeitungsschritten benötigen und nicht von der Größe der Mengen abhängen. Ausnahme ist lediglich die Berechnung der indirekten Abhängigkeiten *trans-dep*, für die im ungünstigsten Fall alle Schritte eines Plans betrachtet werden müssen.

Da die Möglichkeit Causal Links aus einer Planpräsentation zu entfernen niemals vom Vorhandensein anderer Plankomponenten abhängig ist, muß der Algorithmus für manche Aufgaben nur die Planschritte, nicht aber Causal Links betrachten. Entsprechend hängt die Zahl der Verarbeitungsschritte für solche Teilaufgaben von p, der Zahl der Planschritte, und nicht von n, der Gesamtzahl der Komponenten der Präsentation ab.

Der Algorithmus beginnt mit der Prüfung, welche Komponenten der Präsentation nicht entfernt werden können. Jede Komponente muß dafür auf eine Reihe von Kriterien überprüft werden – die Zahl der Verarbeitungsschritte hängt also linear von n ab.

Schritt (12) verwendet die indirekte Abhängigkeit von Komponenten *trans-dep*. Es werden zunächst für jede Komponente die (direkt) *abhängigen* und die *möglicherweise abhängigen* Komponenten bestimmt. Die Berücksichtigung der indirekten Abhängigkeit kann dann leicht durch eine Funktion erfolgen, die alle von einer nicht entfernbaren Komponente abhängigen Planschritte, die nicht selbst schon zur Menge der nicht entfernbaren Komponenten (NEK) gehören, in einem rekursiven Aufruf bearbeitet. Die Bearbeitung selbst besteht dann lediglich noch in der Markierung des Planschritts als zu NEK gehörig. Ein Planschritt kann von mehreren, aber nur begrenzt vielen[28] anderen Komponenten abhängig sein. Demnach können mehrere Exemplare der Funktion einen Planschritt für die Weiterverarbeitung in Betracht ziehen. Zu einer Weiterverarbeitung kommt es allerdings höchstens einmal, wenn der Schritt noch nicht als zu NEK gehörig markiert war. Die Gesamtzahl der Verarbeitungsschritte für (12) kann also nicht stärker als linear mit n wachsen.

Schritt (13) iteriert über die (höchstens $n-p$) Causal Links, für die nicht bereits bekannt ist, daß sie nicht entfernt werden können. Die Verarbeitung erfolgt durch Prozedur *check-del*, die eine Komponente entweder zur Agenda hinzufügt oder aus der Präsentation entfernt. In letzterem Fall wird die Prozedur für alle (höchstens 2) *exklusiv* abhängigen Komponenten aufgerufen. Rekursive Aufrufe erfolgen lediglich für *exklusiv* abhängige Komponenten, und jede Komponente kann nur von genau einer anderen exklusiv abhängig sein. Damit ist sichergestellt, daß die Iteration maximal $(n-p)$ direkte Aufrufe von *check-del* und weitere maximal p rekursive Aufrufe bewirkt. Da die einzelnen Schritte der Funktion konstanten Zeitbedarf haben, hat Schritt (13) wieder linearen Zeitbedarf.

Im zweiten Abschnitt des Algorithmus wird die Agenda abgearbeitet. In Kap. 4.4.3 wurde die Terminierung des Algorithmus damit begründet, daß jede Komponente höchstens einmal auf die Agenda kommen kann und jede Iteration (22) mindestens eine Komponente aus Agenda (und Präsentation)

[28] Die Zahl ist auf die Zahl der Causal Links die zu ihm führen und damit auf die Maximalzahl der Vorbedingungskomponenten begrenzt – hängt also insbesondere *nicht* von n oder p ab.

entfernt. Die while-Schleife kann also höchstens n-mal abgearbeitet werden, womit der Rumpf der for-Schleife (24) höchstens $n^2/2$ wiederholt wird. Dies ist auch die Gesamtkomplexität dieses zweiten Abschnitts. Um dies zu zeigen, sei zunächst von einer Modifikation ausgegangen. Dabei werden die Komponenten, die in der ursprünglichen Version in Schritt (27) neu zur Agenda hinzukommen zunächst in der Menge Ag'' zwischengespeichert und erst nach Ende der for-Schleife (24) – bspw. in einem neuen Schritt (28) – zu Ag hinzugenommen. Ag'' wird vor der Iteration – bspw. als Ergänzung von Schritt (23) – mit der leeren Menge initialisiert. Durch diese Modifikation werden nicht weniger Schritte erforderlich, allerdings wird die Verarbeitung von Komponenten, die neu zur Agenda hinzukommen, auf den jeweils folgenden Iterationsschritt der while-Schleife verschoben. Damit kann sichergestellt werden, daß in einem Iterationsschritt der while-Schleife die Zahl der Aufrufe der Funktion *adv-set* – einschließlich aller rekursiven Aufrufe – $2n$ nicht übersteigt. Der Grund liegt darin, daß a) rekursive Aufrufe von *adv-set* niemals eine Agenda-Komponente als ersten Parameter haben können, daß b) die Konstruktion der Funktion sicherstellt, daß – ausgehend von einem Agenda-Element – rekursive Aufrufe nur zweimal je Komponente möglich sind und daß c) durch die Verwendung der exklusiven Abhängigkeiten, die Mengen von Lösch-Kandidaten (*del-cand*) für alle Agenda-Elemente disjunkt sind. Die Zeit für Verarbeitung innerhalb der Funktion *adv-set* kann auf einen konstanten Wert begrenzt werden, wenn die Menge der Lösch-Kandidaten wieder durch Markierung der Komponenten repräsentiert wird[29]. Dies gilt auch für die (für Schritte (a2) und (a6) erforderliche) Feststellung, ob ein Planschritt von der Menge der Lösch-Kandidaten exklusiv abhängig ist. Unabhängig von der Zahl der Lösch-Kandidaten kann die Zahl der Überprüfungen bei geeigneter Repräsentation der Abhängigkeiten auf zwei Überprüfungen der Abhängigkeiten von Causal Links sowie auf die Zahl der *Schritte*, von denen ein Planschritt abhängig sein kann[30], begrenzt werden.

Weitere Verarbeitungsschritte in der Iteration (24) sind das Anbringen der Lösch-Markierung in (26) und ggf. das Hinzufügen von Komponenten zur Agenda in (27), bzw. in der modifizierten Version zu Ag'' in (28). In beiden Fällen hängt der Aufwand linear von der Zahl der gelöschten Komponenten ab. Da diese Zahl auf die Anzahl der Aufrufe der Funktion *adv-set* begrenzt ist, hängt die Obergrenze für die Verarbeitungsschritte in einer Iteration der while-Schleife wieder linear von n ab.

Der dritte Abschritt des Algorithmus hat i.w. den gleichen Ablauf wie der zweite. Als zusätzliche Aufgabe kommt hinzu, daß für ein Element k der Agenda, das in (36) aus dem Plan entfernt wird, geprüft werden muß, ob dadurch bisher *mögliche* Abhängigkeiten zu *tatsächlichen* Abhängigkeiten gewandelt werden und sich dadurch die Menge der nicht entfernbaren Komponenten (NEK) vergrößert. Dies wird durch die Funktion *del-set* realisiert. Zu überprüfen sind die Komponenten, von denen k bisher *möglicherweise* (aber nicht sicher) abhängig war (s2). Es kann sich deshalb nur um mögliche Abhängigkeiten von anderen Planschritten handeln. Da die Zahl solcher Abhängigkeiten auf die Maximalzahl der Vorbedingungskomponenten eines Operators beschränkt ist, ist auch die Zahl der Iterationen (s2) auf diese konstante Zahl begrenzt. Für die Ergänzung der Menge der nicht entfernbaren Komponenten in Schritt (s6) wird die *indirekte* Abhängigkeit zugrundegelegt, wodurch im ungünstigsten Fall p Komponenten berücksichtigt werden müssen. Allerdings kann das Hinzufügen durch eine rekursive Prozedur realisiert werden die abbricht, wenn eine abhängige Komponente bereits in NEK enthalten ist. Damit kann garantiert werden, daß das Hinzufügen für den gesamten Algorithmus insgesamt nur p mal auftreten kann. In Schritt (s7) wird analog zu Schritt (27) die Agenda ergänzt. Hierbei sei angemerkt, daß der Ausschluß von Komponenten, die von anderen aus D abhängig sind, nur für die Komponenten relevant ist, die von Elementen aus G abhingen. Die Prüfung kann mit konstantem Aufwand je Komponente durchgeführt werden.

[29] Dabei wird ausgenutzt, daß die Menge der Lösch-Kandidaten nur einmal für alle rekursiven Aufrufe der Funktion repräsentiert werden muß. Die Menge selbst wird außer durch Markierungen durch eine (ungeordnete) Liste repräsentiert, die später noch zum Einsatz kommt.

[30] also auf die Maximalzahl der Vorbedingungskomponenten die ein Operator im Modell hat.

Die etwas aufwendigere Verarbeitung des dritten Abschnitts führt also nicht dazu, daß die quadratische Abhängigkeit der Laufzeit von der Zahl der zu präsentierenden Komponenten überschritten wird. Dies ist damit auch die Gesamtkomplexität des Algorithmus zum Entfernen von Komponenten aus einer Planpräsentation.

5.1.4.5 Zusammenfassung

Die Aufwände sowohl für die initiale Bewertung und Beurteilung von Plan-Concepts als auch für die Erzeugung des Planungssachbereichs hängen von der Größe des Benutzermodells ab – genauer: von den Modellen von Wissen und Handlungsmöglichkeiten des Benutzers sowie vom (direkten) Systemwissens. Hinzu kommen die Mengen der potentiellen Axiome und impliziten Effekte, mit denen der Entwickler des Sachbereichsmodells festlegt, welche (der gemäß Benutzermodell zulässigen) Inferenzen für den Planungssachbereich berücksichtigt werden sollen.

Für das Laufzeitverhalten der initialen Bewertung und Beurteilung ist die Zahl der Plan-Concepts, Dekompositionen und Schritte von Dekomposition maßgeblich. Die bestimmenden Faktoren für die Erzeugung des Planungssachbereichs sind neben der Anzahl der Plan-Concepts in SBUC auch deren Komplexität; d.h. i.w. die Zahl der Vorbedingungs- und Effektkomponenten. Dazu kommt die Zahl der Domain-Concepts im Systemwissen (und damit auch in SBUC) sowie die Zahl der potentiellen Axiome und der (elementaren) potentiellen impliziten Effekte. Die Abhängigkeiten sind meist linear oder höchstens quadratisch. Eine Ausnahme bildet die Bestimmung der Korrektheit von Dekomposition, deren Berechnungskomplexität beschränkt werden kann, indem die Zahl möglicher Ausführungsvarianten von Plan-Concepts in SB auf einen sinnvollen Wert begrenzt wird.

Die Laufzeit der anderen Algorithmen hängt vom Plan selbst, d.h. der Anzahl der Planschritte bzw. Plankomponenten insgesamt ab. Die Abhängigkeit ist für die Algorithmen zur Präsentation höchstens quadratisch (von der Zahl der Plankomponenten (Schritte und Causal Links)). Mit exponentieller Abhängigkeit von der Planlänge hat der Planungsalgorithmus den größten Anteil am Laufzeitbedarf. Für ein interaktives System sind damit Pläne nur bis zu einer begrenzten Länge handhabbar. Die Experimente mit UCPOP haben gezeigt, daß die Laufzeit für die Erzeugung von Plänen mit zehn Schritten auch bei einer relativ langsamen Maschine noch erträglich sind. Es kann erwartet werden, daß in Zukunft auch merklich längere Pläne erzeugt werden können. Diese Erwartung stützt sich nicht nur auf die steigende Leistungsfähigkeit der Rechnerhardware, sondern vor allem auch auf die – im Vergleich zum POP Verfahren – wesentlich schnelleren Graphplan-basierten Planungsverfahren. Es ist aber ohnehin nicht wünschenswert für ein Hilfesystem beliebig lange Pläne zu erzeugen, da ansonsten die Gefahr besteht, den Benutzer mit der Beschreibung eines solchen Plans zu überfordern. Dies gilt besonders, wenn der Benutzer Erklärungen für mehrere Planschritte benötigt.

5.2 Eine Beispiel-Benutzermodellierungswissensbasis

Dieser Abschnitt zeigt, wie eine Benutzermodellierungswissensbasis (BMWB) aufgebaut werden kann, die sich für ein adaptives Hilfesystem – wie es in den vorangegangenen Kapiteln beschriebenen wurde – eignet. Konkretes Anwendungsbeispiel ist ein (gedachtes) *anwendungsübergreifendes Hilfesystem für computerunterstützte Büroarbeitsplätze* das in der Lage ist, für einen Benutzer eine spezifische Handlungsanweisung zu erzeugen, mit dem dieser seine Ziele erreichen kann. Die Beispielanwendung soll demonstrieren, welchen Einfluß die verschiedenen Komponenten des Benutzermodells auf die Erzeugung einer solchen Handlungsanweisung haben. Das Beispielziel "*Der Benutzer möchte einen Ausdruck einer Textverarbeitungsdatei haben, die sich auf einem fremden Computer befindet.*" ist dafür gut geeignet, weil für verschiedene Benutzer – wegen unterschiedlicher Zugriffsrechte – nicht die gleichen Lösungen geeignet sind, bzw. sie das Ziel evtl. auch gar nicht

erreichen können. Außerdem liegt das Ziel bereits jenseits der Möglichkeiten konventioneller Hilfesysteme, weil für die Aufgabe mehrere Programme bzw. Betriebssystemfunktionen eingesetzt werden müssen. Bevor im letzten Teil dieses Abschnitts die Einzelheiten des Aufbaus der Beispiel-BMWB gezeigt werden, sei zunächst auf die prinzipielle Vorgehensweise und auf den Beispielsachbereich eingegangen. Die Beispiel-BMWB ist Ausgangsbasis für die Generierung benutzeradaptierter Pläne im nachfolgenden Abschnitt 5.3 *Anwendung des Prototyps*.

5.2.1 Vorgehensweise

Für den Aufbau der BMWB sei zunächst vom üblichen Vorgehen beim Einsatz eines Benutzermodellierungssystems ausgegangen. Bspw. muß die (für das jeweilige System relevante) Begrifflichkeit festgestellt werden und es muß festgelegt werden, wie präzise bzw. mit welcher Granularität sie modelliert werden soll. Dabei muß auch berücksichtigt werden, daß die zu erwartenden Benutzer unterschiedliche Auffassungen darüber haben können. Nicht zuletzt muß festgelegt werden, auf welche Weise Einträge ins Benutzermodell kommen und in diesem Zusammenhang auch, in welche Gruppen die zu erwartenden Benutzer anhand welcher typischen Eigenschaften eingeteilt werden können. Die wesentlichen zusätzlichen Aufgaben, die durch die Verarbeitung von Planungswissen entstehen, bestehen aus der Beschreibung der *Veränderungsmöglichkeiten* im Sachbereich, der Ermittlung von Benutzergruppen mit homogenen *Handlungsmöglichkeiten* und der Aufbereitung von *Abhängigkeiten* im Sachbereich für die Verarbeitung durch das Planungssystem.

Die Begrifflichkeit des Beispielsachbereichs wurde im Verlauf der Arbeit bereits in Auszügen angesprochen; auf Einzelheiten wird weiter unten noch eingegangen. Da das objektbezogene Wissen für die neuen Aspekte dieser Arbeit keine zentrale Rolle spielt, gehen die Beispiele davon aus, daß dieses Wissen für das System und alle Benutzer gleich ist[31]. Für die Modellierung bedeutet dies, daß alle Domain-Concepts sowohl im Systemwissen SB als auch in den Systemannahmen über das Benutzerwissen SBUB enthalten sind. Der nachfolgende Abschnitt über die besonderen Aspekte des Beispielsachbereichs zeigt allerdings, daß die *Modellierung* des Planungs-Wissens durchaus Einfluß auf die Modellierung des objektbezogenen Wissens haben kann.

Für diese Arbeit sind die Unterschiede der Benutzer bezüglich ihrer Handlungsmöglichkeiten und ihrer Auffassungen über Veränderungsmöglichkeiten im Sachbereich zentral. Beides wird durch Plan-Concepts beschrieben, auf die weiter unten noch eingegangen wird. Für die Demonstration von unterschiedlichen Handlungsanweisungen, die sich durch die Adaption an verschiedene Benutzer ergeben kann, folgen wir den Vorschlägen aus Kap. 3.3.3 *Stereotypes* und gehen von den fünf Beispielbenutzern *Oscar*, *Trixie*, *Nick*, *Theo* und *Sue* aus, die sich sowohl bzgl. ihrer Handlungsmöglichkeiten als auch bzgl. ihrer Auffassungen über Veränderungsmöglichkeiten unterscheiden.

Ebenso folgen wir den Vorschlägen zu Benutzergruppen. Bezogen auf ihr Wissen wurden die Benutzergruppen *UNIX-Benutzer*, *Windows-Experten*, *Netzwerk-Anfänger* und *Netzwerk-Experten* vorgeschlagen; bezogen auf ihre Handlungsmöglichkeiten (die i.w. durch Zugriffsrechte bestimmt waren) die Benutzergruppen *Angestellte*, *Gäste* und *studentische Hilfskräfte*. Um das Beispiel zu *nutzbaren* Dekompositionen aus Kap. 4.2.2.2 *Nutzbares Planungswissen* zu demonstrieren, wird noch die Gruppe der *nichtbehinderten* Benutzer eingeführt, die im Beispiel Tastenkombinationen mit drei Tasten – insb. die Tastenkombination *<Strg>-<Alt>-<Entf>* – betätigen kann. Außerdem wird das Herunterfahren eines Computers durch das Plan-Concept *shutdown-pc* modelliert. Dieses Plan-Concept sollen alle Benutzer kennen. Die Befugnis zu seiner Ausführung haben aber nur *Angestellte* und der Beispielbenutzer *Nick* (vgl. Abschnitt 5.3.2.4 *Pläne für Nick*).

Der Aufbau der BMWB kann in vier Teile unterteilt werden:

[31] Auf eine Ausnahme wird noch in Abschnitt 5.3.2.4 *Pläne für Nick* eingegangen.

1. Die Definition der Struktur der BMWB; d.h. der Aufbau der BGP-MS-Partitionenhierarchie einschl. der Stereotype-Partitionen
2. Der Eintrag der Domain-Concepts in die verschiedenen Partitionen
3. Der Eintrag der Plan-Concepts in die verschiedenen Partitionen
4. Die Definition der (potentiellen) Abhängigkeiten im Planungssachbereich

Für den Aufbau der Partitionenhierarchie können direkt die Funktionen des BGP-MS Systems verwendet werden. Für den Eintrag der Domain-Concepts wäre dies allerdings weniger geeignet. BGP-MS sieht lediglich Funktionen vor, mit denen einzelne Teile einer Concept-Beschreibung definiert werden können – bspw. die Concept-Definition selbst, sowie der Eintrag von einzelnen Super-Concepts, Rollen oder Role-Value-Maps. Damit sind für die vollständige Definition eines Concepts mehrere Funktionsaufrufe notwendig. Hinzu kommt, daß es sinnvoll sein kann eine bestimmte Concept-Beschreibung zu verschiedenen Zeitpunkten der Modellierung in eine View des Benutzermodells eintragen zu können. Bspw. wenn einem Benutzer im Laufe des Dialogs mit dem System neue Kenntnisse oder Handlungsmöglichkeiten zugebilligt werden. Um Concept-Definition zusammenhängend beschreiben zu können, wurde deshalb die LISP-Structure *bgp-ms-concept-struc* definiert, die eine Concept-Beschreibung mit allen Komponenten, einschließlich des Typ des Eintrags (*defined* / *primitive* / *singleton*), erlaubt und auch zwischen allgemeinen Rollen und Attributen unterscheidet[32]. Solche Structures können dann durch die Funktion *bgp-ms-defconcepts-from-struclist* in eine oder mehrere Views der BMWB eingetragen werden. Die Funktion realisiert den Eintrag über i.a. mehrere Aufrufe der elementaren BGP-MS-Funktionen. Um die Beschreibungen mehrfach verwenden zu können, werden sie Konstanten zugewiesen, die beim Eintrag angegeben werden können. Abb. 5.2 enthält bspw. Beschreibungen der Domain-Concepts *fs-file* und *text-file*, die dann für einen Aufruf von *bgp-ms-defconcepts-from-struclist* eingesetzt werden.

Definition von Konstanten für Concept-Beschreibungen:

```
(defconstant                                        domc-fs-file
   (make-bgp-ms-concept-struc
      :name       'fs-file
      :supers     '(named-fs-object)
      :attributes '((fso-contents information)
                    (file-type file-type-c))))

(defconstant                                        domc-text-file
   (make-bgp-ms-concept-struc
      :name       'text-file
      :type       :defined
      :supers     '(fs-file)
      :attributes '((file-type file-type-text))))
```

Verwendung der Konstanten beim Eintrag in die BMWB:

```
(bgp-ms-defconcepts-from-struclist (list domc-fs-file domc-text-file)
                                   :partitions   '(SDomB AllB))
```

Abbildung 5.2: Definition und Verwendung von Konstanten für Concept-Beschreibungen

[32] vgl. dazu Kap. 3.2.1 *Beschreibungslogiken*. Die Syntax, die für eine solche Beschreibung eingehalten werden muß, findet sich in Abb. A.1 S. 302 im Anhang A.2 *Zusammenfassung syntaktischer Definitionen in EBNF*.

```
(defconstant                                        op-open-file-tpx
  (make-bgp-ms-plan-struc
    :name         'open-file-tpx
    :precondition '(and (tpx-file ?f) (local-named-fs-object ?f)
                        (tpx ?app)    (launched ?app))
    :effect       '(active ?app ?f)))

(defconstant                                        op-ftp-download-file-bin
  (make-bgp-ms-plan-struc
    :name          'ftp-download-file-bin
    :supers        '(ftp-download-file)
    :precondition  '(and (ftp-binary-mode ?prog) (bin-file ?src))))
```

Abbildung 5.3: Definition von Konstanten für Plan-Concept-Beschreibungen

Prinzipiell ist dieses Vorgehen auch für Plan-Concepts geeignet, da sie sich formal nicht von Domain-Concepts unterscheiden. Um dem Modellierer die Umsetzung von Plan-Operator-Beschreibungen in die beschreibungslogische Repräsentation zu ersparen, können Plan-Concepts als LISP-Structures vom Typ *bgp-ms-plan-struc* gemäß der Syntax aus Abb. 3.6 S. 114 definiert werden. Abb. 5.3 zeigt die Definition der Plan-Concepts *open-file-tpx* und *ftp-download-file-bin* als *bgp-ms-plan-struc*tures, die Konstanten zugewiesen werden.

Die Structures können durch die Funktion *bgp-ms-def-plan-concepts-from-struclist* in gleicher Weise in die BMWB eingetragen werden, wie *bgp-ms-concept-struc*tures durch *bgp-ms-defconcepts-from-struclist*. Entsprechend sind für Dekomposition, die in dieser Arbeit als "besondere" Plan-Concepts modelliert sind, Structures vom Typ *bgp-ms-decomp-struc* vorgesehen, die durch die Funktion *bgp-ms-def-decomposition-from-struclist* in die BMWB eingetragen werden können[33]. Zur einfachen Unterscheidung, haben Konstantennamen für Domain-Concepts den Präfix "domc-", die für Plan-Concepts den Präfix "op-" und die für Dekompositionen den Präfix "dec-".

Im Gegensatz zu den Concepts werden die potentiellen Abhängigkeiten des Planungssachbereichs nur einmal beim initialen Aufbau der BMWB definiert, weil sie benutzerunabhängig sind. (Welche der potentiellen Abhängigkeiten für die Generierung eines Plans eingesetzt werden, entscheidet sich allerdings jeweils unmittelbar vor dem Aufruf des Planungssystems anhand des dann aktuellen Inhalts der entsprechenden View – für benutzerspezifische Pläne SBUC.) Die Definition von Konstanten, wie für die Concepts, bringt hier also keinen Vorteil. Gleiches gilt auch für die *Vorrang-Muster*, die ebenfalls benutzerunabhängig sind (vgl. Kap. 3.4.2.7 *Erzeugung des aktiven Teils eines Planungssachbereichs*).

Für den Aufbau der Beispiel-BMWB wurden die Definitionen der Konstanten für Domain- und Plan-Concepts sowie die Definitionen der potentiellen Abhängigkeiten des Planungssachbereichs jeweils in eigene Dateien eingetragen, die nach dem Start des Prototyps in die LISP-Umgebung geladen werden können. Da die Vorrang-Muster in engem Zusammenhang zu den Plan-Concepts stehen, finden sich ihre Definitionen gemeinsam mit den Konstanten für die Plan-Concepts in einer Datei. Die Dateien sind auch im elektronischen Anhang dieser Arbeit enthalten und zwar im Verzeichnis *KN-PP-Demo* unter den Namen *office-world-domain-concepts.lisp*, *office-world-plan-concepts.lisp* und *office-world-pd-dependencies.lisp*.

[33]Die einzuhaltende Syntax wird in Abb. 4.9 S. 222 in Kap. 4.6.2 *Repräsentation von Dekompositionen in BGP-MS* beschrieben. Dort findet sich auch ein Beispiel für die Definition einer Dekomposition.

5.2.2 Der Beispielsachbereich

Als Beispiel-Anwendungssachbereich wurde eine heute übliche Büroumgebung mit vernetzten Computerarbeitsplätzen ausgewählt. Ein Grund der für diesen Sachbereich spricht sind die üblicherweise sehr unterschiedlichen Kenntnisse und Handlungsmöglichkeiten der Benutzer, die eine *benutzerangepaßte* Hilfestellung wünschenswert erscheinen lassen. Außerdem liegen bereits einfache Problemstellungen wie das eingangs genannte Beispielziel jenseits der Fähigkeiten bisheriger Hilfesysteme. Typische Objekte in einem solchen Anwendungsgebiet sind – neben den Benutzern selbst – vernetzte Computer mit Anwendungsprogrammen und (oft unterschiedlichen) Betriebssystemen zu denen auch die Verwaltung von Dateien gehört. Dazu kommen Dokumente, Computer-Peripheriegeräte wie Drucker, Datenträger uvm. Die Genauigkeit der Modellierung wird wesentlich von den *Handlungen* bestimmt, die im Sachbereich eine Rolle spielen. Weiter unten wird dies am Beispiel der Modellierung des Dateisystems gezeigt. Wichtige Handlungen für die Beispiele sind das Ausdrucken einer Datei sowie der FTP-Dateitransfer und das Mounten von Verzeichnissen, mit denen sich Dateien von fremden Computern in das lokale Dateisystem bringen lassen. Dazu kommen Nebenfunktionen wie das Starten eines Programms, das Öffnen einer Datei, der Wechsel von Verzeichnissen bei FTP uva. Das Planungssystem benötigt für die Erzeugung eines Plans eine Beschreibung der möglichen Handlungen in Form von Planoperatoren. Abb. 5.4 enthält eine kurze (verbale) Beschreibung der wichtigsten Handlungen für die Beispiele. Im Anhang B.2 findet sich eine formale Beschreibung der Handlungen als Plan-Concept-Beschreibungen, so wie sie für die BMWB benötigt wird. Dort finden sich auch die Beschreibungen der Dekompositionen die für Beispiele benötigt werden. Die Abbildungen 5.5 und 5.6 zeigen jeweils eine grafische Darstellung der Dekompositionen der Plan-Concepts *shutdown-pc* bzw. *ftp-download-file-bin*. Erstere modelliert die beiden Alternativen einen (Windows-)PC herunterzufahren (vgl. Kap. 4.2.2.2 *Nutzbares Planungswissen*). Für *ftp-download-file-bin* sind drei alternative Ausführungsmöglichkeiten modelliert:

1. *d1-ftp-bin*: *Auswahl des Dateinamens im Datei-Menü* und dann
   Betätigen der Tastenkombination <Alt>-<O> (*O* für *Open*)
2. *d2-ftp-bin*: *Auswahl des Dateinamens im Datei-Menü*, dann
   Betätigen der Schaltfläche Download und dann *Betätigen der Schaltfläche Transfer*
3. *d3-ftp-bin*: wie *d1-ftp-bin*, allerdings wird die *Tastenkombination <Alt>-<T> betätigt*
   (*T* für *Transfer*)

Die dritte Ausführungsmöglichkeit stellt eine häufig anzutreffende Fehlannahme dar, die bei der Abschätzung des Erklärungsbedarfs für Beispielbenutzer *Nick* (Kap. 5.3.2.4 *Pläne für Nick*) eine Rolle spielen wird. Beim Aufbau der initialen Benutzermodellierungswissensbasis im Abschnitt 5.2.3 werden deshalb nur die Dekomposition *d1-ftp-bin* und *d2-ftp-bin*, die in Abbildung 5.6 durchgehend grün umrahmet sind, in das *System*wissen eingetragen. Hinzu kommen Dekompositionen für *ftp-download-file-text* (analog zu *ftp-download-file-bin*) und für *open-file-tpx* (*Betätigen der Schaltfläche Open*, *Auswahl des Dateinamens im Datei-Menü* und dann *Betätigen der OK-Schaltfläche*).

Das objektbezogene Wissen ist in dieser Arbeit durch *Domain-Concepts* repräsentiert, die in einer Begriffshierarchie angeordnet sind (vgl. Kap. 3.2.2 *Planoperatoren in Beschreibungslogik*). Abb. 5.7 zeigt einen Ausschnitt einer solchen Hierarchie, mit den – für die Beispiele – wichtigsten Domain-Concepts in grafischer SB-ONE Darstellung (vgl. [Profitlich, 1990]). Die Begriffshierarchie modelliert das objektbezogene Wissen des Systems und – da vereinbart ist, daß sich System und Benutzer bzgl. dieses Wissens nicht unterscheiden – aller Benutzer. Der Darstellung fehlen einige Details, die aber für die Beispiele ohne Belang sind. Für die vollständige Beschreibung sei wiederum auf den Anhang – hier B.1 – verwiesen.

open-file-tpx öffnet eine lokale Datei des Typs *tpx-file* mit einem Textverarbeitungssystem vom Typ X; d.h. die Datei wird dadurch aktives Dokument des Textverarbeitungssystems.

mount ein exportiertes Verzeichnis eines fremden Computers bekommt einen (zuvor freien) Namen des lokalen Dateisystems zugewiesen und wird dadurch zum lokalen Verzeichnis. Die Varianten *mount-win* bzw. *mount-ux* arbeiten nur, wenn der lokale Computer ein Windows- bzw. UNIX Betriebssystem hat.

app-print druckt das aktive Dokument eines Anwendungsprogramms mit einem Drucker. Der Drucker hat (besitzt) danach ein neues Papierdokument mit dem Inhalt des Dokuments.

user-get der aktuelle Benutzer übernimmt den Besitz an einem Objekt, das zuvor einen anderen Besitzer hatte.

launch ein Anwendungsprogramm wird gestartet und ist dann aktiv.

ftp-download-file erzeugt eine Kopie einer Datei, die sich im (für das FTP-Programm) aktiven Verzeichnis eines fremden Computers befindet, im aktiven lokalen Verzeichnis.

ftp-download-file-bin ftp-download-file nur für Binär-Dateien – setzt den Binär-Modus des FTP-Programms voraus.

ftp-download-file-text ftp-download-file nur für Text-Dateien – setzt den Text-Modus des FTP-Programms voraus.

remote-copy erzeugt im lokalen Dateisystem eine Kopie einer beliebigen Datei im Netzwerk.

download-file abstraktes Plan-Concept mit dem sich eine Datei *irgendwie* in das lokale Dateisystem bringen läßt.

ftp-open-connection Verbindet ein aktives FTP-Programm mit einem fremden Computer, macht die Wurzelverzeichnisse der beiden beteiligten Computer zu den aktiven Fremd- bzw. Lokal-Verzeichnissen und bringt das FTP-Programm in den Text-Modus.

ftp-chmode-binary Bringt ein aktives FTP-Programm in den Binär-Modus.

ftp-chmode-text Bringt ein aktives FTP-Programm in den Text-Modus.

ftp-cd ändert das aktive Fremd-Verzeichnis eines FTP-Programms.

ftp-lcd ändert das aktive Lokal-Verzeichnis eines FTP-Programms.

shutdown-pc fährt einen Computer herunter.

Abbildung 5.4: Die wichtigsten Plan-Concepts für die Beispiele

Im Kapitel 3.4.2.2 *Abhängigkeiten im Sachbereich* wurde gezeigt, daß es Sinn machen kann, dem Planungssystem die Schlußfolgerung auf einige der Objekteigenschaften zu überlassen. Grundlage für die Schlußfolgerungen des Planungssystems sind die modellierten Abhängigkeiten zwischen Objektklassen, die dem Planungssystem in Form von Axiomen bekannt gemacht werden müssen. Anhand der Modellierung des Dateisystems und der Erzeugung von Objekten wird im Anschluß gezeigt, welche konkreten Schritte notwendig sind. Dabei wird auch deutlich, daß die drei Teile der Sachbereichsbeschreibung (aktiver und passiver Teil sowie Repräsentation von Abhängigkeiten für das

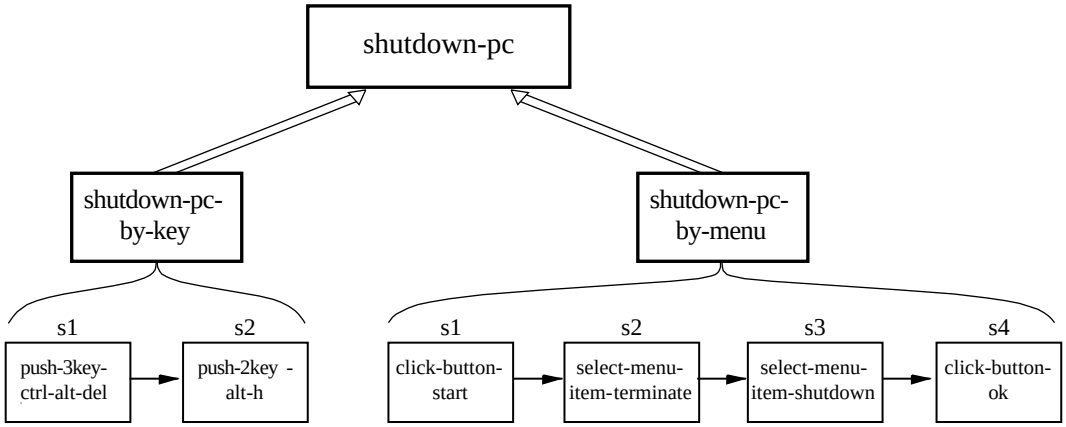

Abbildung 5.5: Dekompositionen von *shutdown-pc*

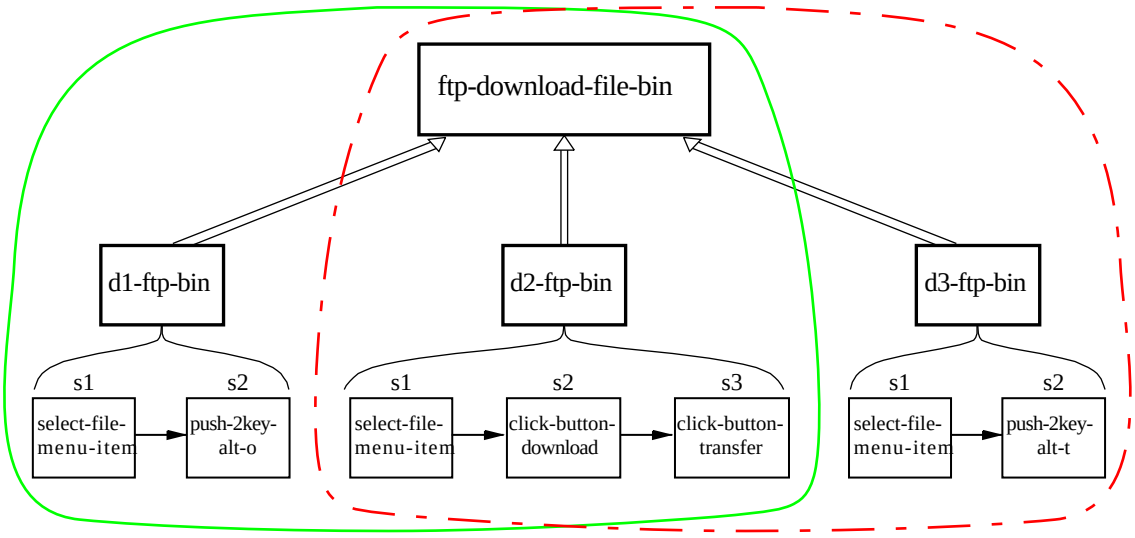

Abbildung 5.6: Dekompositionen von *ftp-download-file-bin*

Planungssystem) eng zusammenhängen. Damit bspw. für das Planungssystem inferierbar ist, ob eine bestimmte Datei eine *Binärdatei* ist (wie beim FTP-Transfer erforderlich), muß das Sachbereichsmodell die Information enthalten welche Dateitypen zum Oberbegriff *Binärdatei* gehören, damit daraus dann ein entsprechendes Axiom abgeleitet werden kann. In Kap. 3.4.2.4 wurde außerdem darauf hingewiesen, daß der Entwickler des Sachbereichsmodells die Menge der *potentiellen Abhängigkeiten*, die für den Planungssachbereich berücksichtigt werden sollen, sinnvoll beschränken muß – im Falle von Axiomen auf eine adäquate Menge von *potentiellen Axiomen*. Zu den *potentiellen Abhängigkeiten* zählen nicht nur solche, die der Entwickler für richtig hält, sondern auch solche, die von den (bzw. einigen der) zu erwartenden Benutzern für richtig gehalten werden *könnten*. Sie sind insb. dann von Bedeutung, wenn anhand der (mglw. falschen) Annahmen eines Benutzer geplant wird. Abb. 5.8 zeigt die Menge der potentiellen Axiome, die für die Beispiele dieser Arbeit eingesetzt werden in prädikatenlogischer Notation[34]. Darunter ist mit *ax-wrong-tpx-text-file* auch ein Axiom, das normalerweise

[34] Die angegebenen Namen sind für die UCPOP-Notation erforderlich und werden auch im Rest dieses Kapitels verwendet. Vgl. a. [Barrett *et al.*, 1995] und die Beschreibung der potentiellen Axiome im Anhang B.3.

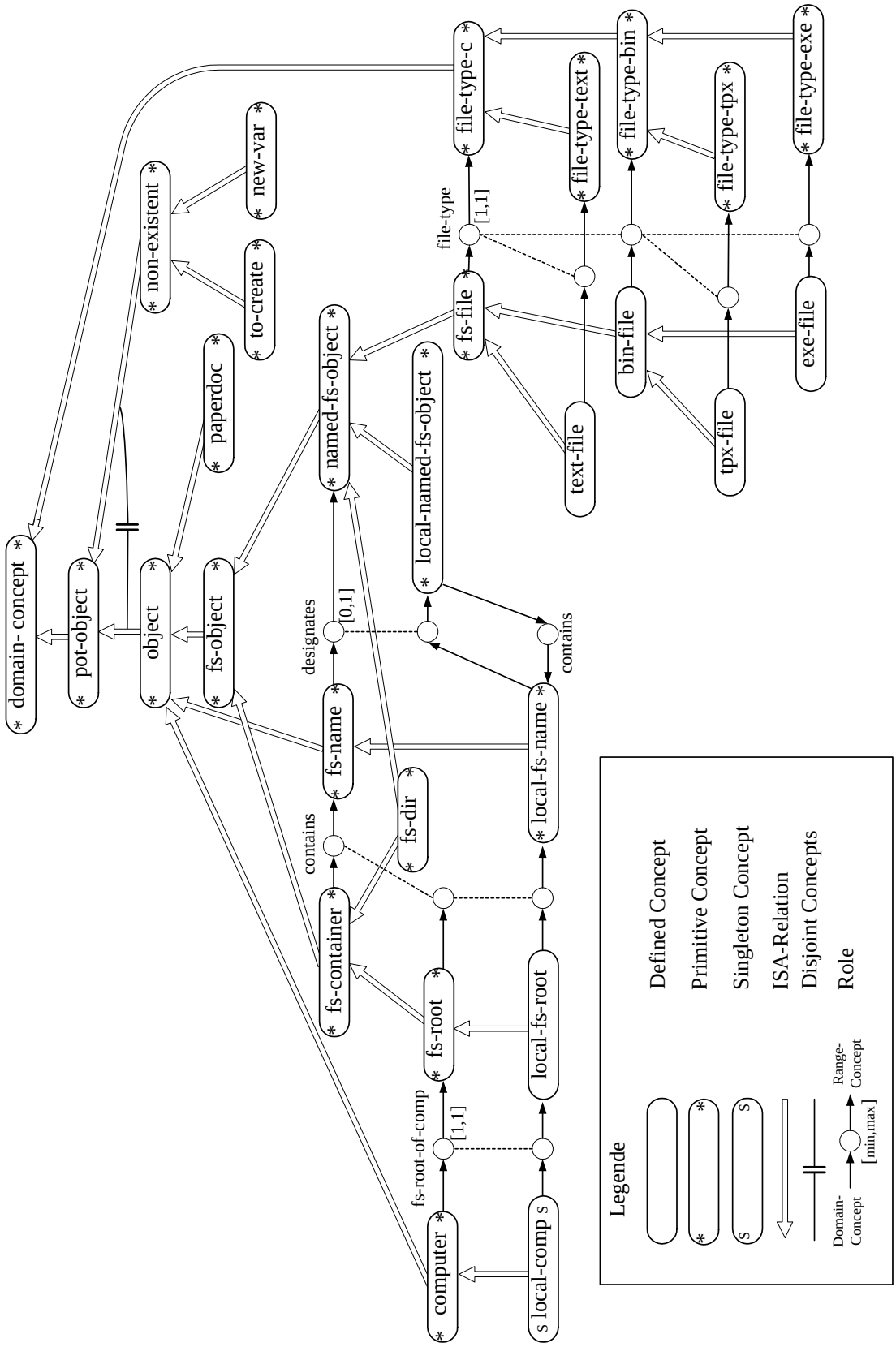

Abbildung 5.7: Grafische SB-ONE-Repräsentation der wichtigsten Domain-Concepts der Beispiele

$$
\begin{array}{l}
\textit{ax-text-file}: \\
\quad \forall x \quad (\exists y\ \textit{file-type-text}(y) \wedge \textit{file-type}(x,y) \wedge \textit{fs-file}(x)) \rightarrow \textit{text-file}(x) \\
\textit{ax-tpx-file}: \\
\quad \forall x \quad (\exists y\ \textit{file-type-tpx}(y) \wedge \textit{file-type}(x,y) \wedge \textit{fs-file}(x)) \rightarrow \textit{tpx-file}(x) \\
\textit{ax-exe-file}: \\
\quad \forall x \quad (\exists y\ \textit{file-type-executable}(y) \wedge \textit{file-type}(x,y) \wedge \textit{fs-file}(x)) \rightarrow \textit{exe-file}(x) \\
\textit{ax-bin-file1}: \\
\quad \forall x \quad \textit{exe-file}(x) \rightarrow \textit{bin-file}(x) \\
\textit{ax-bin-file2}: \\
\quad \forall x \quad \textit{tpx-file}(x) \rightarrow \textit{bin-file}(x) \\
\textit{ax-wrong-tpx-text-file}: \\
\quad \forall x \quad \textit{tpx-file}(x) \rightarrow \textit{text-file}(x) \\
\textit{ax-local-fs-root}: \\
\quad \forall x \quad (\exists y\ \textit{local-comp}(y) \wedge \textit{fs-root-of-comp}(y,x)) \rightarrow \textit{local-fs-root}(x) \\
\textit{ax-local-fs-name1}: \\
\quad \forall x \quad (\exists y\ \textit{contains}(y,x) \wedge \textit{local-fs-root}(y)) \rightarrow \textit{local-fs-name}(x) \\
\textit{ax-local-fs-name2}: \\
\quad \forall x \quad (\exists y\ \textit{contains}(y,x) \wedge \textit{local-named-fs-object}(y)) \rightarrow \textit{local-fs-name}(x) \\
\textit{ax-local-named-fs-object}: \\
\quad \forall x \quad (\exists y\ \textit{designates}(y,x) \wedge \textit{local-fs-name}(y)) \rightarrow \textit{local-named-fs-object}(x) \\
\textit{ax-local-fs-object-of-comp}: \\
\quad \forall x,y \quad \textit{fs-root-of-comp}(x,y) \vee \\
\qquad\qquad (\exists n,d\ \textit{designates}(n,y) \wedge \textit{contains}(d,n) \wedge \\
\qquad\qquad\qquad \textit{local-fs-object-of-comp}(x,d)) \rightarrow \textit{local-fs-object-of-comp}(x,y) \\
\textit{ax-find-to-create-object}: \\
\quad \forall x \quad \textit{to-create}(x) \rightarrow \textit{non-existent}(x) \\
\textit{ax-create-new-var}: \\
\quad \forall x \quad \textit{new-var}(x) \rightarrow \textit{non-existent}(x)
\end{array}
$$

Abbildung 5.8: Die potentiellen Axiome des Beispielsachbereichs

als falsch angesehen wird, aber gleichzeitig eine beliebte Fehlannahme darstellt, nämlich die Annahme, daß Dateien für ein Textverarbeitungssystem Text-Dateien sind[35]. Sein Einsatz wird im Zuge der Erzeugung benutzerspezifischer Pläne – speziell Abschnitt 5.3.2.4 *Pläne für Nick* – gezeigt.

Mit Ausnahme der Negation gehören ausschließende Typzugehörigkeiten zu den Abhängigkeiten im Planungssachbereich, die nicht direkt vom Planungssystem berücksichtigt werden können. In Kap. 3.4.2.4.4 *Ausschließende Eigenschaften bzw. Typzugehörigkeiten* wurden dafür *implizite Effekte* eingeführt, die bei der Erzeugung des Planungssachbereichs explizit gemacht werden müssen. Auch hierfür muß der Entwickler des Sachbereichsmodells eine sinnvolle Menge als *potentielle* Abhängig-

[35] Dies ist auf die Textverarbeitungssysteme der Beispiele bezogen – die Annahme kann für andere (hier nicht modellierte) Textverarbeitungssysteme durchaus wahr sein kann.

keiten definieren, über deren Verwendung anhand des Benutzermodells (View SBUC für benutzerspezifische Pläne) entschieden wird. In dieser Arbeit werden ausschließende Typzugehörigkeiten nur im Zusammenhang mit der Erzeugung von Objekten eingesetzt, auf die weiter unten noch eingegangen wird.

5.2.2.1 Das Dateisystem

Die Modellierung des Dateisystems orientiert sich an einem hierarchischen Dateisystem, wie es in heute üblichen Betriebssystemen vorkommt. Die wichtigsten Dateisystemobjekte (DS-Objekte) sind Dateien die Information tragen und einen Dateityp haben sowie Container, die Namen von DS-Objekten enthalten können. Container sind entweder Verzeichnisse oder Wurzelobjekte eines Dateisystems. Jedem Computer ist genau ein solches Wurzelobjekt zugeordnet.

Die Modellierung soll folgende Dateisystemoperationen ermöglichen:

- Einrichten von Dateisystem-*Links*[36] (Verweise auf bzw. Verknüpfungen mit anderen DS-Objekten)
- Kopieren einer Datei (Anlegen eines neuen Datei-Objekts mit gleichem Typ und Inhalt wie das Original)

Außerdem soll zwischen lokalen DS-Objekten – solchen, die sich von dem (ausgezeichneten) lokalen Computer direkt erreichen lassen – und nicht-lokalen DS-Objekten unterschieden werden, weil hierfür unterschiedliche Zugriffsbedingungen bestehen.

5.2.2.1.1 Dateisystem-Links Die Möglichkeit, Dateisystem-Links (DS-Links) einzurichten, modelliert zum einen die Fähigkeit aktueller Betriebssysteme. Darüber hinaus werden DS-Links im Beispiel dazu genutzt, Verzeichnisse eines fremden Computers zu mounten – also unter einem neuen Namen im lokalen Dateisystem verfügbar zu machen. Das Einrichten eines Links wird durch das Zuordnen eines Dateisystem-Namens (DS-Namens) zu einem DS-Objekt modelliert. DS-Objekte und DS-Namen werden in der BMWB durch die Domain-Concepts *fs-object* und *fs-name* modelliert. Die Zuordnung wird für Dateien (*fs-file*) und Verzeichnisse (*fs-dir*) – die beide von *named-fs-object* subsumiert werden – durch die Rolle *designates* (*designates* $\Rightarrow$ *fs-name* $\times$ *named-fs-object*) repräsentiert. Der Wurzel eines Dateisystems (*fs-root*) ist (normalerweise) kein Name zugeordnet.

Dementsprechend können Container (*fs-container*) Dateien oder Verzeichnisse nicht direkt enthalten, sondern enthalten stattdessen nur die zugehörigen Namen. Werden – wie im Beispiel des Mountens von Dateien – einem DS-Objekt mehrere Namen zugeordnet, kann auf diese Weise modelliert werden, daß dieses DS-Objekt zu mehreren Containern gehört. Bspw. bezeichnen in Abb. 5.10 sowohl der Name *FN-PC-drive-N* als auch *FN-KN-home* das Verzeichnis *KN-home*. Damit wird modelliert, daß *KN-home* sowohl zu *PC-root* als auch zu *KN-root* gehört.

5.2.2.1.2 Kopieren von Dateien Das Kopieren einer Datei wird durch die Erzeugung eines neuen Dateiobjekts mit den gleichen Eigenschaften wie das Original modelliert. Das heißt, daß das neue Objekt die gleichen Attributwerte[37] wie das Original haben muß. Ausnahme ist lediglich der Dateiname, über den ja auch die Lokation der Datei im Dateisystem festgelegt wird. Aus diesem Grund muß dafür auch ein neues Namensobjekt erzeugt werden, das die Datei bezeichnet – und zwar unabhängig davon, ob der textuelle Wert des Dateinamens gleich bleiben soll oder nicht. Außerdem muß

[36] Der englische Begriff *Link* wird anstelle von *Verknüpfung* verwendet, um Verwechslungen mit Verknüpfungen der Windows-Betriebssysteme zu vermeiden.

[37] Attributwert ist hier im Datenbanksinne als *Eigenschaftswert der ein Objekt beschreibt* zu verstehen.

noch sichergestellt werden, daß das neue DS-Objekt zu den gleichen (Datei-) Typen gehört, wie das Original.

Der Typ einer Datei bestimmt bspw., ob sie von einem bestimmten Textverarbeitungssystem bearbeitet werden kann, ob sie direkt ausdruckbar ist und in welchem Modus sie via FTP transferiert werden kann. Im Kapitel 3.4.2.2 *Abhängigkeiten im Sachbereich* wurde die Problematik der Modellierung von Typzugehörigkeit bei der Repräsentation des Planungssachbereichs behandelt und vorgeschlagen, daß das Planungssystem die Typzugehörigkeit aus Objektmerkmalen inferieren soll. Das Planungssystem benötigt dazu *Axiome*, die bei der Erzeugung des Planungssachbereichs aus dem Benutzermodell abgeleitet werden. Das Objektmerkmal, aus dem sich die Zugehörigkeit zu einem Dateityp schließen läßt, ist in der beschreibungslogischen Repräsentation der BMWB durch das Attribut *file-type* modelliert.

Im folgenden Beispiel soll gezeigt werden, wie die Inferenzen eingesetzt werden. Dabei soll eine Datei für die Textverarbeitung X (Dateityp *tpx-file*) zunächst via FTP in das lokale Dateisystem kopiert und dann vom Textverarbeitungssystem geöffnet werden[38]. Dazu ist neben dem Schluß vom Objektmerkmal auf den Dateityp auch der Schluß auf die Zugehörigkeit zu einem abstrakteren Dateityp notwendig.

Für den Dateitransfer stehen die Operatoren *ftp-download-file-bin* für Binär-Dateien (Dateityp *bin-file*) und *ftp-download-file-text* für Text-Dateien (Dateityp *text-file*) zur Verfügung. Die beschreibungslogische Darstellung einer Datei für die Textverarbeitung X (Dateityp *tpx-file*) lautet (vgl. Anhang B.1 *Domain-Concepts*):

(1) $\mathit{tpx\text{-}file} \doteq \mathit{bin\text{-}file} \sqcap \mathit{file\text{-}type} : \mathit{file\text{-}type\text{-}tpx}$

Jedes Objekt vom Typ *tpx-file* steht also zu genau einem Objekt vom Typ *file-type-tpx* in Relation *file-type*. Die Schlußfolgerung, daß eine Datei (Dateityp *fs-file*) zum speziellen Dateityp *tpx-file* gehört, wird durch das Axiom *ax-tpx-file*:

(2) $\forall x\ (\exists y\ \mathit{file\text{-}type\text{-}tpx}(y) \wedge \mathit{file\text{-}type}(x,y) \wedge \mathit{fs\text{-}file}(x)) \rightarrow \mathit{tpx\text{-}file}(x)$

möglich. Nach Kap. 3.4.2.4 *Ableitung von Abhängigkeiten aus dem Benutzermodell* läßt sich ein solches Axiom aus dem Benutzermodell ableiten, wenn in der aktuellen View (bspw. SBUC für die Erzeugung benutzerspezifische Pläne) die Subsumptionsbeziehung

(3) $\mathit{file\text{-}type} : \mathit{file\text{-}type\text{-}tpx} \sqcap \mathit{fs\text{-}file} \Rightarrow \mathit{tpx\text{-}file}$

besteht. Kap. 3.4.2.6 *Vereinfachte beschreibungslogische Inferenzen* gibt (mögliche) Bedingungen an, wann eine Subsumptionsbeziehung dieser Art besteht. Angewendet auf das Beispiel besteht die Beziehung, wenn (1) gilt und außerdem *file-type* ein Attribut von *fs-file* ist und sich *bin-file* von *fs-file* höchstens durch eine speziellere Wertebeschränkung des Attributs *file-type* unterscheidet. Dies ist wegen der Definitionen von *fs-file* und *bin-file* sowie *file-type-tpx* und *file-type-bin* der Fall:

(4) $\mathit{fs\text{-}file} \sqsubseteq \mathit{named\text{-}fs\text{-}object} \sqcap \mathit{file\text{-}type} : \mathit{file\text{-}type\text{-}c} \sqcap \mathit{fso\text{-}contents} : \mathit{information}$

(5) $\mathit{bin\text{-}file} \doteq \mathit{fs\text{-}file} \sqcap \mathit{file\text{-}type} : \mathit{file\text{-}type\text{-}bin}$

(6) $\mathit{file\text{-}type\text{-}tpx} \sqsubseteq \mathit{file\text{-}type\text{-}bin} \sqsubseteq \mathit{file\text{-}type\text{-}c}$

Für die Entscheidung, ob und ggf. welcher FTP-Operator eingesetzt werden kann, reicht dies allerdings noch nicht aus. Hierzu ist noch der Schluß vom speziellen Dateityp *tpx-file* auf die Verallgemeinerung *bin-file* notwendig. Dieser Schluß ist mithilfe des Axioms *ax-bin-file2*:

[38] Dies ist Teil einer der möglichen Lösungen für das eingangs erwähnte Planungsziel, eine Datei zu bekommen, die sich auf einem fremden Computer befindet. Vgl. bspw. Plan (N-C-ne) für Nick in Abschnitt 5.3.2.4 *Pläne für Nick*.

(7) $\forall x\ \textit{tpx-file}(x) \rightarrow \textit{bin-file}(x)$

möglich. Das Axiom läßt sich (wieder nach Kap. 3.4.2.4) ableiten, wenn die Subsumptionsbeziehung

(8) $\textit{tpx-file} \Rightarrow \textit{bin-file}$

besteht. Dies ist der Fall, weil *tpx-file* in (1) als Subconcept von *bin-file* definiert ist. Als FTP-Operator kann demnach *ftp-download-file-bin* eingesetzt werden. Eine analoge Ableitung des für *ftp-download-file-text* erforderlichen Dateityps *text-file* ist dagegen nicht möglich.

Damit für die Dateikopie die gleichen Typzugehörigkeiten abgeleitet werden können wie für das Original, muß sie das gleiche Merkmal für die Typbestimmung – hier der Typ des Wertes des Attributs *file-type* – besitzen wie das Original. Der FTP-Operator stellt dies sicher, indem die Kopie zum gleichen Objekt in Relation *file-type* gesetzt wird wie das Original. Dazu enthält die Vorbedingung des Operators *ftp-download-file-bin* den Ausdruck (vgl. Anhang B.2 *Plan-Concepts*):

$$\textit{bin-file}(\textit{quelle}) \wedge \textit{file-type}(\textit{quelle}, \underline{\textit{quelltyp}})$$

und der Effekt den Ausdruck:

$$\textit{fs-file}(\textit{kopie}) \wedge \textit{file-type}(\textit{kopie}, \underline{\textit{quelltyp}}).$$

Dabei sind *quelle*, *quelltyp* und *kopie* Variablen für die Quelldatei, das Typobjekt, das zur Quelldatei in Beziehung *file-type* steht, und die kopierte Datei. Werden Dateitypen durch Axiome wie (2) und (7) ermittelt[39], so lassen sich für eine Dateikopie die gleichen Typen ableiten wie für das Original. Im diesem Beispiel also *tpx-file* und *bin-file*. Die Dateikopie hat so den erforderlichen Dateityp *tpx-file*, um durch die Textverarbeitung X geöffnet werden zu können (vgl. Operator *open-file-tpx*).

Wichtig ist dabei, daß die Definition des Kopier-Operators unabhängig davon ist, welche spezielleren Dateitypen es gibt. Bspw. können mit *ftp-download-file-bin* alle Varianten von Binär-Dateien unabhängig von ihrem speziellen Typ heruntergeladen werden.

5.2.2.1.3 Lokale Dateien Mit wenigen Ausnahmen (wie die FTP-Funktionalität) erlauben Betriebssysteme Dateioperationen nur auf Dateien im lokalen Dateisystem eines Computers. (Deshalb kann im Beispiel die Datei auf dem fremden Computer nicht direkt mit *open-file-tpx* geöffnet werden.) Dabei kommt es normalerweise nicht auf den Ort des Datenträgers an, sondern lediglich, ob eine Datei über einen Namen des lokalen Dateisystems erreichbar ist. Entsprechendes gilt für Verzeichnisse. Lokale Dateien oder Verzeichnisse müssen also immer (mindestens) einen lokalen Namen haben. Die beschreibungslogische Repräsentation faßt lokale Dateien und lokale Verzeichnisse zu lokalen benamten DS-Objekten (*local-named-fs-object*) zusammen. Lokale Namen (*local-fs-name*) sind wiederum DS-Namen, die in einem lokalen Verzeichnis oder in der Wurzel des lokalen Dateisystems (*local-fs-root*) enthalten sind.

Im Beispiel wird die Beziehung zwischen DS-Namen und benamten DS-Objekten durch die Relation *designates* modelliert. Verzeichnisse oder eine Dateisystemwurzel (die zum Oberbegriff *fs-container* zusammengefaßt sind) können zu DS-Namen in Relation *contains* stehen. Die Wurzel des lokalen Dateisystems ist das (eindeutig bestimmte) DS-Objekt, das zum ausgezeichneten lokalen Computer (*local-comp*) in Beziehung *fs-root-of-comp* steht (vgl. a. Abb. 5.7):

$$\begin{aligned} \textit{designates} &\subseteq \textit{fs-name} \times \textit{named-fs-object} \\ \textit{contains} &\subseteq \textit{fs-container} \times \textit{fs-name} \\ \textit{fs-root-of-comp} &\subseteq \textit{computer} \times \textit{fs-root} \end{aligned}$$

[39] Analog zu diesen beiden Axiomen sind im Beispielsachbereich potentielle Axiome für das Ableiten der Dateitypen *exe-file* und *text-file* enthalten.

Ob ein DS-Objekt lokal ist, kann sich durch DS-Operationen wie (un-)mounting oder löschen verändern. Betrifft eine solche Veränderung Verzeichnisse, so ändert sich auch der Status aller Unterverzeichnisse und der zugehörigen Dateien. Dieser Umstand kann nicht direkt durch den Effekt eines Planungsoperators erreicht werden[40]. Wird der o.g. Zusammenhang zwischen der Lokalität von Dateien, Verzeichnissen und Namen ausgenutzt, kann aus den Beziehungen *contains* und *designates* abgeleitet werden, ob eine Datei lokal ist. Operatoren wie bspw. *mount* brauchen dann lediglich eine Beziehung zwischen einem DS-Namen und einem DS-Objekt zu verändern, um die Lokalität einer im Prinzip unbeschränkten Menge von Objekten zu verändern. Die Abbildungen 5.9 und 5.10 zeigen als Beispiel (ausschnittweise) die Dateisysteme der Computer *PC* und *Konstanz* bevor und nachdem das Verzeichnis *KN-home* (bspw.) durch Operator *mount* an den DS-Namen *Fn-PC-drive-N* gebunden wurde. Die in Abbildungen 5.9 gezeigte Situation stellt einen Teil der Ausgangssituationen *init-UM99-dir-is-exported* und *init-UM99-dir-not-exported* dar, die für die Planerzeugung in Abschnitt 5.3 *Anwendung des Prototyps* verwendet werden. Die Pläne, die den Operator *mount* einsetzen, um die Datei *F-KN-xy-doc* in das lokale Dateisystem zu bekommen (bspw. Plan O-C-e), verändern die Situation wie in Abb. 5.10 gezeigt. Die grau unterlegten Bereiche sind in beiden Situationen der Bereich der *lokalen* DS-Objekte (wobei Computer *PC* der lokale Computer ist). *Lokale benamte* DS-Objekte sind in den Abbildungen gestrichelt umrahmt.

Durch folgende Axiome läßt sich ableiten, ob ein DS-Objekt lokal ist, d.h. ob das Prädikat *local-named-fs-object* für das Objekt gilt:
um die Wurzel des *lokalen* Dateisystems abzuleiten:

$$\textit{ax-local-fs-root}: \quad \forall x \quad (\exists y\ \textit{local-comp}(y) \wedge \textit{fs-root-of-comp}(y,x)) \rightarrow \textit{local-fs-root}(x)$$

um *lokale* (benamte) DS-Objekte abzuleiten:

$$\textit{ax-local-named-fs-object}: \quad \forall x \quad (\exists y\ \textit{designates}(y,x) \wedge \textit{local-fs-name}(y)) \rightarrow \textit{local-named-fs-object}(x)$$

um *lokale* DS-Namen abzuleiten:

$$\textit{ax-local-fs-name1}: \quad \forall x \quad (\exists y\ \textit{contains}(y,x) \wedge \textit{local-fs-root}(y)) \rightarrow \textit{local-fs-name}(x)$$
$$\textit{ax-local-fs-name2}: \quad \forall x \quad (\exists y\ \textit{contains}(y,x) \wedge \textit{local-named-fs-object}(y)) \rightarrow \textit{local-fs-name}(x)$$

Alle vier Axiome sind nach dem Schema

$$\forall x \quad (\exists y\ R(y,x) \wedge T_1(y)) \rightarrow T_2(x)$$

aufgebaut und lassen sich deshalb in gleicher Weise aus der BMWB ableiten. Nach Kapitel 3.4.2.4 *Ableitung von Abhängigkeiten aus dem Benutzermodell* kann ein solches Axiom aus der beschreibungslogischen Repräsentation abgeleitet werden, wenn die Subsumptionsbeziehung

$$T_1 \quad \Rightarrow \quad \forall R : T_2$$

besteht. Eine solche Beziehung besteht, wenn Concept T_1 ein Attribut oder eine allquantifizierte Rolle R mit Werterestriktion T_2 hat. Für die genannten Axiome ist dies für alle Beispiele der Fall. Die zugehörigen Definitionen lauten (vgl. wieder Anhang B.1 *Domain-Concepts*):

$$\begin{aligned}
\textit{local-comp} &\sqsubseteq \textit{computer} \sqcap \textit{fs-root-of-comp} : \textit{local-fs-root} \\
\textit{local-fs-name} &\sqsubseteq \textit{fs-name} \sqcap \forall \textit{designates} : \textit{local-named-fs-object} \\
\textit{local-fs-root} &\doteq \textit{fs-root} \sqcap \forall \textit{contains} : \textit{local-fs-name} \\
\textit{local-named-fs-object} &\sqsubseteq \textit{named-fs-object} \sqcap \forall \textit{contains} : \textit{local-fs-name}
\end{aligned}$$

[40] Dies ist unabhängig davon, daß einige Planungssysteme (wie auch UCPOP) allquantifizierte Effekte verarbeiten können, denn damit lassen sich nur flache Strukturen behandeln, wie in Pednaults [Pednault, 1988] Beispiel zum Transport *aller* Objekte, die sich in einer Aktentasche befinden.

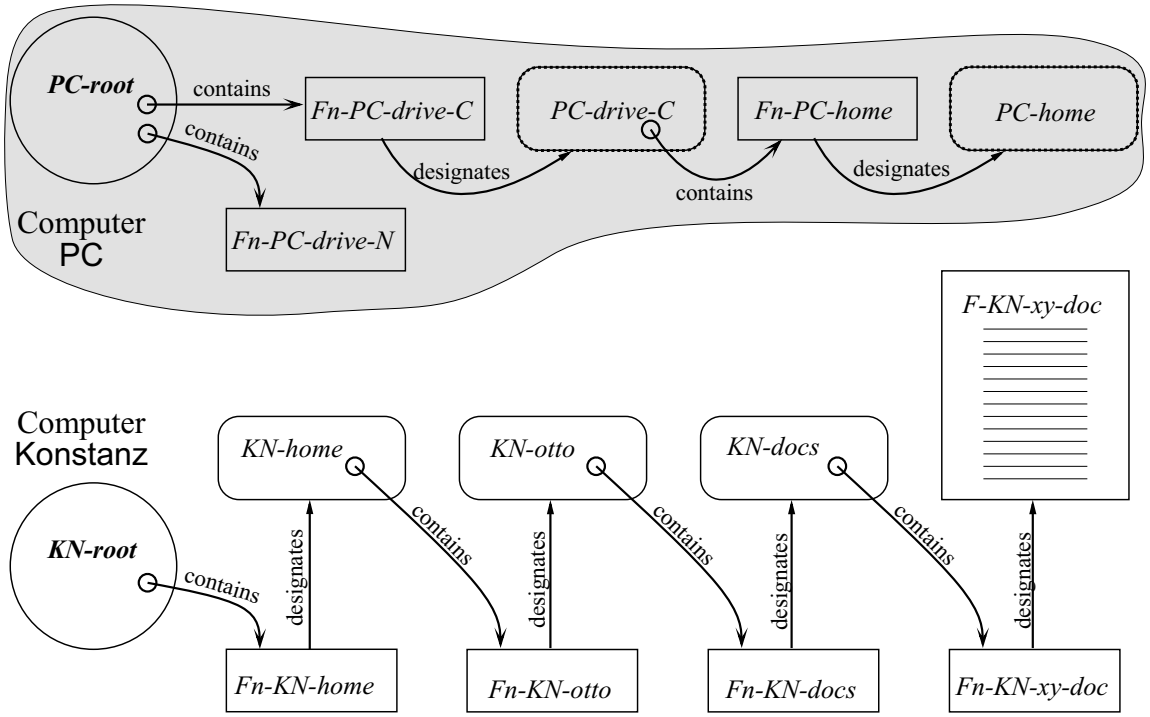

Abbildung 5.9: Ausschnitt der Dateisysteme der Computer *PC* und *Konstanz*

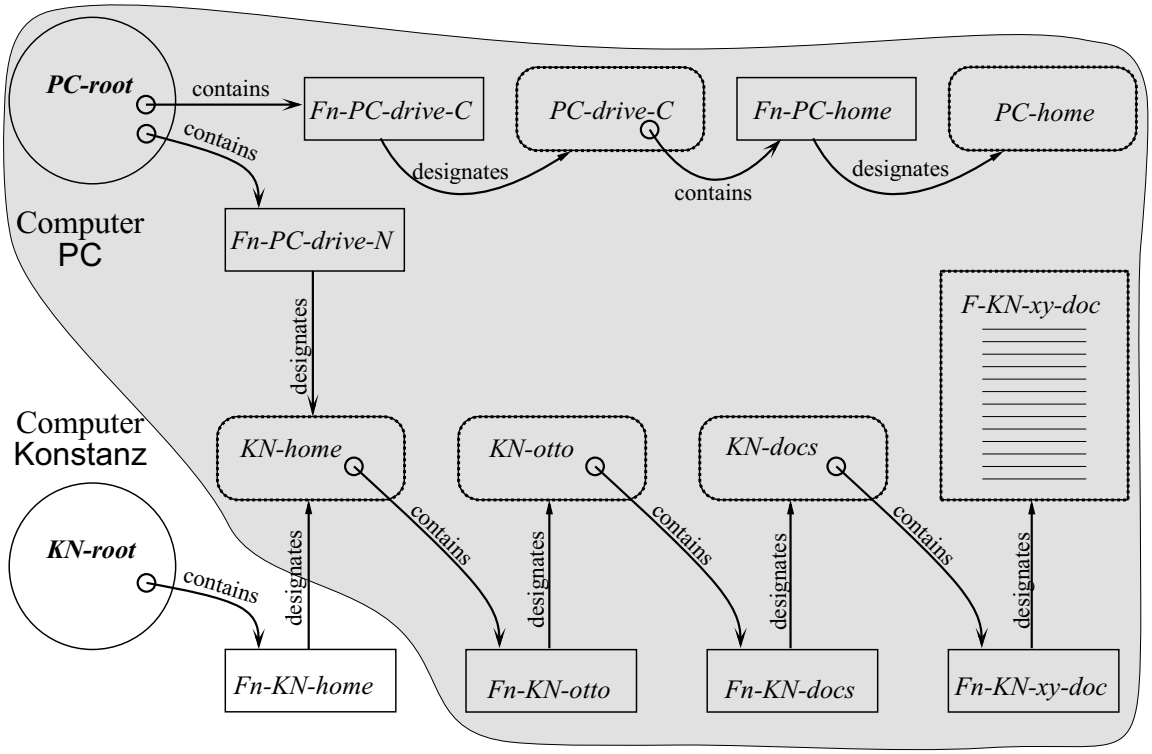

Abbildung 5.10: Ausschnitt der Dateisysteme von *PC* und *Konstanz* nach *mount*

5.2.2.2 Erzeugen von Objekten

Das Erzeugen von Objekten tritt in der Beispielanwendung beim Kopieren von Dateien und beim Drucken auf. Im ersten Fall muß ein neues DS-Objekt und ein neuer DS-Name erzeugt werden (s.o.), im zweiten Fall ein neues Papierdokument. In Kap. 3.4.2.5 *Erzeugen von Objekten* wurde gesagt, daß die Erzeugung von Objekten bei der Plangenerierung dadurch modelliert wird, daß der Effekt des erzeugenden Planoperators einer Konstanten einen Typ für *existierende Objekte* zuordnet. Im Beispielsachbereich sind das alle Subtypen bzw. Sub-Concepts von *object*. Eine solche Konstante darf *vor* Ausführen des Operators natürlich keinem anderen Objekt zugeordnet sein, weil sie ansonsten mehrere Objekte bezeichnen würde. Um dies sicherzustellen, bieten sich zwei Lösungen an. Zum einen kann in der Startsituation vor der Planerzeugung eine "ausreichende" Anzahl von Konstanten mit einem speziellem Typ für "noch zu erzeugende Objekte" definiert werden. Die Planoperatoren die Objekte erzeugen benötigen dann als Vorbedingung für jedes zu erzeugende Objekt genau eine Konstante dieses Typs. Der Effekt des Operators weist diesen Konstanten dann die neuen Objekt-Typen zu und verneint gleichzeitig den speziellen Typ für "noch zu erzeugende Objekte". Bei der zweiten Lösung erzeugt das Planungssystem von sich aus *bei Bedarf* neue Konstanten. In Kap. 3.4.2.5 wurde gezeigt, daß beide Lösungen ihre Berechtigung haben. Das Beispielziel, daß ein neues Papierdokument *erzeugt* werden soll (und nicht etwa einer anderen Person ein Ausdruck der Datei weggenommen werden soll), läßt sich bspw. nur mit der ersten Lösung ausdrücken. I.a. ist es allerdings vorteilhaft dem Planungsprozeß zu überlassen, wann und wieviele Objekte er für die Lösung eines Planungsproblems erzeugt. Im Beispielsachbereich findet sich deshalb sowohl der Typ *to-create* für "noch zu erzeugende Objekte" als auch der Typ *new-var*, mit dem die Erzeugung einer neuen Konstante verbunden ist. Da die Planoperatoren aber unabhängig davon definiert werden sollen, sind die beiden Typen zum Oberbegriff *non-existent* zusammengefaßt, der dann für die Vorbedingung von Planoperatoren die Objekte erzeugen eingesetzt wird. Die Schlußfolgerung auf *non-existent* wird durch die Axiome:

$$\begin{aligned} \textit{ax-find-to-create-object:} \quad & \forall x \quad \textit{to-create}(x) \rightarrow \textit{non-existent}(x) \\ \textit{ax-create-new-var:} \quad & \forall x \quad \textit{new-var}(x) \rightarrow \textit{non-existent}(x) \end{aligned}$$

möglich. Die Axiome können in allen Beispielen abgeleitet werden, weil *to-create* und *new-var* jeweils als Sub-Concepts von *non-existent* modelliert sind (vgl. Abb. 5.7).

Da *non-existent* disjunkt zu allen Objekt-Typen ist, ergibt sich aus logischer Sicht, daß eine Konstante, die ein gerade erzeugtes Objekt bezeichnet, nicht gleichzeitig vom Typ *non-existent* sein kann. Das Planungssystem kann diesen Schluß aber nicht ziehen. Da die Konklusion von Axiomen keine Negation enthalten darf[41], können dem Planungssystem bspw. die Abhängigkeiten

$$\begin{aligned} &(1) \quad \forall x\, \textit{paperdoc}(x) \quad \rightarrow \quad \neg \textit{non-existent}(x) \qquad \text{oder} \\ &(2) \quad \forall x\, \textit{paperdoc}(x) \quad \rightarrow \quad \neg \textit{to-create}(x) \end{aligned}$$

nicht als Axiom mitgegeben werden.

In Kap. 3.4.2.4.4 *Ausschließende Eigenschaften bzw. Typzugehörigkeiten* wurde deshalb die Angabe von *impliziten Effekten* vorgeschlagen. Solche impliziten Effekte kann der Entwickler des Sachbereichmodells analog zu den potentiellen Axiomen benutzerunabhängig definieren. Auch sie werden nur dann bei der Erzeugung eines Planungssachbereichs aus einer View der BMWB verwendet, wenn sie sich logisch aus dieser View ableiten lassen. Im Gegensatz zu den Axiomen bewirkt ein impliziter Effekt, daß die Schlußfolgerungen, die das Planungssystem nicht ziehen kann, vorweggenommen werden, indem der Effekt von Planoperatoren ergänzt wird. *Implizite* Effekte von Planoperatoren

[41] vgl. Kap. 3.4.2.3 *Nichtatomare Formeln im Planungssachbereich*

werden dadurch für das Planungssystem *explizit*, ohne daß das Benutzermodell diese redundante Information ebenfalls explizit enthalten müsste.

Zu beachten ist noch, daß (1) nicht als impliziter Effekt definiert werden darf, weil dies zu einer Effektkomponente $\neg$*non-existent*(x) eines Planoperators führen würde. Im Sinne von UCPOP ist *non-existent* ein *abgeleitetes Prädikat*[42], das nicht im Effekt eines Planoperators vorkommen darf. Um zu erreichen, daß eine *vorhandene* Konstante nicht mehr dem Typ *non-existent* angehört, reicht es allerdings *to-create* zu negieren, weil *non-existent* für eine *vorhandene* Konstante nur dann wahr sein kann, wenn dafür auch *to-create* wahr ist.

Implizite Effekte werden in dieser Arbeit ausschließlich für die Erzeugung von Objekten eingesetzt. Insgesamt sind im Beispielsachbereich die folgenden impliziten Effekte als potentielle Abhängigkeiten definiert:

$$\begin{aligned} \forall x\ \mathit{paperdoc}(x) &\rightarrow \neg \mathit{to\text{-}create}(x) \\ \forall x\ \mathit{fs\text{-}file}(x) &\rightarrow \neg \mathit{to\text{-}create}(x) \\ \forall x\ \mathit{fs\text{-}name}(x) &\rightarrow \neg \mathit{to\text{-}create}(x) \end{aligned}$$

5.2.3 Initialer Aufbau einer Benutzermodellierungswissensbasis

Für den initialen Aufbau der Benutzermodellierungswissensbasis (BMWB) für das Beispielmodell wird von einem geladenen LISP- und BGP-MS-System und einem leeren BGP-MS *environment* ausgegangen[43]. Auf dieses *environment* wird im weiteren ohne neuerliche Erwähnung Bezug genommen. Die Beschreibung hier beschränkt sich auf die in dieser Arbeit behandelten Besonderheiten gegenüber den bisherigen Möglichkeiten[44] von BGP-MS:

- den Aufbau der Partitionenhierarchie mit getrennten Partitionen für Annahmen und Handlungspotential
- den initialen Eintrag der (Plan-)Concept-Beschreibungen
- die Definition der Abhängigkeiten im Planungssachbereich.

Sämtliche Funktionsaufrufe für den Aufbau des Beispielmodells sowie ein Protokoll des Ablaufs, sind in den Dateien *demo-init-bmwb.lisp*[45] bzw. *demo-init-bmwb.log* enthalten. Sie finden sich wie alle anderen in diesem Abschnitt genannten Dateien im Verzeichnis *KN-PP-Demo* des elektronischen Anhangs dieser Arbeit.

Vor dem eigentlichen Aufbau der BMWB werden zunächst die Konstanten, die die Information über die Domain- und Plan-Concepts enthalten, in die LISP-Umgebung geladen:[46]

```
(load "bgp-ms-data:office-world-domain-concepts")
(load "bgp-ms-data:office-world-plan-concepts")
```

[42] vgl. wieder Kap. 3.4.2.3

[43] Technische Einzelheiten dazu werden im Anhang C.1 *Systemumgebung und -voraussetzungen* beschrieben.

[44] vgl. dazu bspw. [Kobsa und Pohl, 1995].

[45] Die im weiteren beschriebenen Funktionsaufrufe sind in der Datei jeweils als Argument der Funktion *pev* enthalten. Die Funktion wird für die Protokollierung eingesetzt: sie gibt ihr Argument mit vorangestelltem "--->" auf die Standardausgabe aus bevor es evaluiert wird. Dadurch lassen sich Ein- und Ausgaben im Protokoll leicht unterscheiden. Die Funktion *com* gibt einen Kommentar auf die Standardausgabe aus.

[46] Die angegebenen Argumente bestehen aus einem LISP-Suchpfad und einem Dateinamen ohne die Erweiterung "*.lisp*". Die Suchpfade, die in dieser Arbeit verwendet werden finden sich in Tab. C.1 S. 322 im Anhang C.1.

Die Vorteile der Verwendung solcher Konstanten gegenüber direkten Einträgen in die BMWB wurden bereits im Abschnitt 5.2.1 angesprochen. Hinzu kommt, daß die Beschreibung des Modellaufbaus trotz der großen Menge an Einzelinformationen übersichtlich bleibt. Zusammen mit den Konstanten für Plan-Concepts werden auch bereits die benutzerunabhängigen Vorrang-Muster in die BMWB geladen. Die ebenfalls benutzerunabhängigen potentiellen Abhängigkeiten im Planungssachbereich – also potentielle Axiome, potentielle implizite Effekte und Facts – werden durch:

```
(load "bgp-ms-data:office-world-pd-dependencies")
```

in die BMWB eingetragen.

Wie eingangs (Abschnitt 5.2.1) angesprochen, folgen wir für den Aufbau der Partitionenhierarchie der Beschreibung aus Kap. 3.3 *Struktur eines Benutzermodells mit Handlungspotential und Wissen* und darin insb. Kap. 3.3.3 mit der Ergänzung eines Stereotypes für nichtbehinderte Benutzer. Dort wurde vorgeschlagen, von den einfachen Annahmen des Systems (in Partition SB) die *objektbezogenen* Annahmen abzutrennen und in einer eigenen Partition ($S^{Dom}B$) zu repräsentieren, um sie dann an die Partitionen vererben zu können, die Handlungsmöglichkeiten von Benutzern modellieren. Dazu wurde $S^{Dom}B$ als Super-Partition zu SB und *All*C definiert[47]. *All*C ist der Stereotype, der die Annahmen des Systems über die Handlungsmöglichkeiten die *jeder* Benutzer hat enthält und dem (indirekt) jeder Benutzer angehört. Weitere Stereotypes für Handlungspotentiale sind *Staff*C, *Guest*C, *Stud*C und *NonHandicap*C, die die Benutzergruppen *Angestellte*, *Gäste*, *studentische Hilfskräfte* und *nichtbehinderte Benutzer* modellieren. Die zugehörigen Partitionen sind direkte Sub-Partitionen von *All*C. Anhand ihres typischen Wissen wurden die Benutzergruppen *UNIX-Benutzer*, *Windows-Experten*, *Netzwerk-Anfänger* und *Netzwerk-Experten* gebildet. Die zugehörigen Partitionen *Ux*B, *WinExp*B, *NetNov*B und *NetExp*B sind direkte Sub-Partitionen von *All*B. Analog zu *All*C repräsentiert *All*B das Wissen, das jedem Benutzer zugesprochen wird. Hinzu kommen die Benutzer-Partitionen SBUB und SBUC, die die Systemannahmen über Wissen und Handlungsmöglichkeiten des jeweiligen Benutzers repräsentieren und aus technischen Gründen die Partitionen *AllSt*C und *AllSt*B. Diese beiden letzteren sind jeweils direkte Sub-Partitionen von *All*C bzw. *All*B und enthalten ihren Inhalt ausschließlich durch Vererbung von diesen Super-Partitionen. Sie haben lediglich den Zweck den Inhalt von *All*C bzw. *All*B in einer Blatt-Partition für die Stereotype-Triggerung[48] zur Verfügung zu stellen. Die beiden Benutzer-Partitionen werden beim initialen Aufbau des Modells zwar definiert, aber erst zur Laufzeit eines adaptiven Systems in die Partitionenhierarchie eingefügt und mit Inhalten gefüllt (vgl. Abschnitt 5.3 *Anwendung des Prototyps*).

Für die Definition von Partitionen sieht BGP-MS die Funktion *define-bgp-ms-partition* vor, die auch die Deklaration einer Partitionen als *primitive* und/oder *Stereotype-Partition*[49] ermöglicht. Der Aufbau der Partitionenhierarchie beginnt mit den Wurzel-Partitionen $S^{Dom}B$, *All*B, SBUB und SBUC:

```
(define-bgp-ms-partition 'SDomB :primitive-flag 't)
(define-bgp-ms-partition 'AllB  :stereotype 't :primitive-flag 't)
(define-bgp-ms-partition 'SBUB)
(define-bgp-ms-partition 'SBUC)
```

Für die Stereotype-Partition *All*B werden noch *Stereotype-Trigger* eingetragen. Stereotype-Trigger sind Regeln, die bestimmen unter welchen Umständen der Stereotype derjenigen Benutzer-Partition zugeordnet wird, die die gleiche Modalität besitzt wie der Stereotype – hier also Benutzer-Partition SBUB weil *All*B ein B(elief)-Steretype ist. Allerdings wird *All*B keine Blatt-Partition bleiben und

[47] vgl. die Abbildungen 3.17 S. 131 und 3.18 S. 132.

[48] s. Abschnitt 5.3.2.

[49] Die alternative Möglichkeit, Steretypes durch die Funktion *define-stereotype* zu definieren, steht zumindest für den Stereotype *All*C nicht zur Verfügung, da seine Super-Partition $S^{Dom}B$ nicht die gleiche Modalität hat. Aus Gründen einer einheitlichen Darstellung wird auch für die anderen Stereotypes auf diese Möglichkeit verzichtet.

deshalb niemals direkt einer Benutzer-Partition zugeordnet. Die (De-)Aktivierungs-Trigger für *All*B lauten deshalb:

```
(set-stereotype-trigger 'AllB :activation-rule '(never))
(set-stereotype-trigger 'AllB :retraction-rule '(never))
```

Für die weiteren Stereotypes sei die Behandlung der (De-)Aktivierungs-Trigger auf Abschnitt 5.3.2 *Benutzerspezifische Pläne* zurückgestellt, weil sie dort im Zusammenhang mit der Verwendung für die Beispiele beschieben werden können. Außerdem sei im weiteren aus Platzgründen für Stereotype-Definitionen auf die Angabe der Parameter *stereotype* und *primitive* verzichtet und lediglich durch "..." daran erinnert, daß beiden für Stereotypes grundsätzlich der Wert 't zugewiesen wird.

Vor der Definition von Sub-Partitionen zu S^{Dom}B und *All*B werden die Inhalte in diese Partitionen eingetragen. Dieses Vorgehen – Definition der Partitionen einer Ebene und unmittelbar danach der Eintrag der Inhalte – wird auch für die weiteren Definitionen beibehalten. Die Ebene einer Partition wird dabei durch die Länge des längsten Pfades (der Relation Super-Partition) von der Partition zur Wurzel der Partitionenhierarchie bestimmt. Im Beispiel gehören demnach die Wurzel-Partitionen S^{Dom}B, *All*B, SBUB und SBUC zur Ebene 0, SB, *All*C, *NonHandicap*C, *Ux*B, *WinExp*B, *NetNov*B, *NetExp*B und *AllSt*B zur Ebene 1 und *Staff*C, *Guest*C, *Stud*C und *AllSt*C zur Ebene 2. Der Grund für das schichtweise Vorgehen des Aufbaus der BMWB liegt in der Art, wie BGP-MS Inhalte in Partitionen einträgt. Ein direkter Eintrag von Concept-Beschreibungen in *Partitionen* ist in BGP-MS nicht vorgesehen. Stattdessen sieht BGP-MS den Eintrag solcher Beschreibungen in *Views* – Mengen von Pfaden von einer Blatt-Partition zu allen erreichbaren Wurzeln – vor. Um eine Concept-Beschreibung in eine View einzutragen, muß die View durch ihre Blatt-Partition angegeben werden. Um bspw. für das Beispielmodell festzulegen, daß das Concept *fs-file* in allen Handlungspotentialen von Benutzern enthalten sein soll, muß dieses Concept in sämtlichen Stereotype-Views für Handlungspotentiale (*Staff*C, *Guest*C, *Stud*C und *AllSt*C) definiert sein. Dies kann durch den Funktionsaufruf:

```
(bgp-ms-define-general-concept :name  'fs-file
                               :views '(StaffC GuestC StudC AllStC))
```

erreicht werden. Der Partitionsmechanismus von BGP-MS entscheidet dann selbst darüber, in welche Partition(en) der Eintrag erfolgt. Sind die beteiligten Partitionen nicht als *primitiv* deklariert worden, erfolgt der Eintrag in Partition *All*C, weil *All*C nur genau die vier genannten Sub-Partitionen hat. Ein Eintrag in *All*C reicht also aus, um das Wissen genau in den gewünschten Views zur Verfügung zu haben. Für Stereotypes ist ein solches "Hochziehen" von Informationen i.a. aber nicht sinnvoll. Gleiches gilt insbesondere für Partition S^{Dom}B, die nur *Domain*-Concepts enthalten soll und nicht auch die *Plan*-Concepts, die in *All*C und SB gemeinsam enthalten sind. Die oben definierten Partitionen wurden deshalb als *primitiv* markiert. Gleiches gilt auch für alle weiteren Partitionen des Beispielmodells (s.u.).

Dadurch entfällt allerdings auch die Möglichkeit zum indirekten Eintrag von Inhalten in Nicht-Blatt-Partitionen. Das schichtweise Vorgehen ermöglicht aber – zumindest beim Aufbau des Modells – direkte Einträge in beliebige Partitionen, weil die Einträge erfolgen, wenn die Partitionen noch keine Sub-Partitionen haben und somit Blatt-Partitionen einer View sind.

Für den Eintrag der Concept-Beschreibungen werden die neuen Funktionen *bgp-ms-def-concepts-from-struclist* bzw. *bgp-ms-def-plan-concepts-from-struclist* eingesetzt. Die Funktionen erwarten als ersten Parameter eine Liste von LISP-Structures, die allgemeine (SB-ONE-) Concept-Beschreibungen bzw. Plan-Concept-Beschreibungen jeweils nach den zugehörigen Syntax-Definitionen aus Anhang A.2 *Zusammenfassung syntaktischer Definitionen in EBNF* enthalten. Als zweiter Parameter wird eine Liste der Namen der Partitionen, in die die Beschreibungen eingetragen werden sollen, erwartet. Die Beschreibungen selbst sind in den Dateien *office-world-domain-concepts.lisp* und *office-*

world-plan-concepts.lisp enthalten. Sie wurden durch das Laden der Dateien (s.o.) an Konstanten gebunden, die hier für den Eintrag der Concept-Beschreibungen eingesetzt werden (vgl. dazu auch die beiden vorangegangen Abschnitte 5.2.1 und 5.2.2). Der Eintrag einer Concept-Beschreibung durch *bgp-ms-def-concepts-from-struclist* führt i.a. nicht nur zur Definition des Concepts, sondern auch zum Eintrag von Rollen, ISA-Links und/oder Role-Value-Maps, die sich aus der Beschreibung ergeben. Funktion *bgp-ms-def-plan-concepts-from-struclist* übernimmt zusätzlich die Transformation einer Operatorbeschreibung nach Kap. 3.2.2 *Planoperatoren in Beschreibungslogik* in die beschreibungslogische Darstellung; d.h. hier in die SB-ONE-Darstellung.

Da das objektbezogene Wissen (nach Abschnitt 5.2.1) in den Beispielen für das System und alle Benutzer gleich ist, bietet es sich an, eine Liste aller Domain-Concepts für den Eintrag dieses Wissens in die BMWB zu verwenden und damit das explizite Aufschreiben der recht langen Liste zu vermeiden. Durch das Laden der Datei *office-world-domain-concepts.lisp* wurde der Variablen *all-dom-concept-syms* eine Liste mit den Namen aller Konstanten für Domain-Concept-Beschreibungen zugewiesen. Der Eintrag aller definierten Domain-Concept-Beschreibungen in die Partitionen $S^{Dom}B$ und *All*B ist dann durch den Aufruf:

```
(bgp-ms-defconcepts-from-struclist
   (mapcar #'eval all-dom-concept-syms) :partitions '(SDomB AllB))
```

möglich. Wenn die BMWB vollständig aufgebaut ist, wird $S^{Dom}B$ in der View SB (für das Systemwissen) und in der Benutzer-View SBUC enthalten sein, während *All*B in der Benutzer-View SBUB enthalten sein wird.

Für die korrekte Behandlung von impliziten Effekten und komplementären Typzugehörigkeiten (vgl. die Abschnitte 3.4.2.4.4 und 3.4.2.4.5) muß noch angegeben werden, welche Concepts gegenseitig ausschließende bzw. komplementäre Begriffe beschreiben. Die neue BGP-MS-Funktion *bgp-ms-declare-concepts-disjoint* trägt für die angegebenen Concepts SB-ONE *Disjunkt-Markierungen* in die angegebenen Views ein; die Funktion *bgp-ms-declare-concepts-complementary Disjunkt-* und *Überdeckungs-Markierungen* (vgl. Fußnote 51 S. 150):

```
(bgp-ms-declare-concepts-disjoint
   '(pot-object plan-concept information)          :views '(SDomB AllB))
(bgp-ms-declare-concepts-disjoint
   '(non-existent object)                          :views '(SDomB AllB))
(bgp-ms-declare-concepts-disjoint '(fs-dir fs-root) :views '(SDomB AllB))

(bgp-ms-declare-concepts-complementary
   '(fs-name-for-mount non_fs-name-for-mount)      :views '(SDomB AllB))
(bgp-ms-declare-concepts-complementary
   '(online non_online)                            :views '(SDomB AllB))
(bgp-ms-declare-concepts-complementary
   '(ftp-binary-mode non_ftp-binary-mode)          :views '(SDomB AllB))
```

Da $S^{Dom}B$ keine Plan-Concepts enthalten soll, werden Plan-Concepts zunächst nur in *All*B eingetragen und damit die Plan-Concept festgelegt, die (normalerweise) jeder Benutzer kennt:

```
(bgp-ms-def-plan-concepts-from-struclist (list
   op-app-print op-user-get op-launch op-open-file-tpx op-shutdown-pc
   op-decomposition op-atomic op-incomplete
   op-click-button op-click-button-ok op-click-button-start
   op-click-button-open op-click-button-download op-click-button-transfer
   op-push-2key op-push-2key-alt-h op-push-2key-alt-o op-push-2key-alt-t
   op-push-3key op-push-3key-ctrl-alt-del op-select-file-menu-item
   op-select-menu-item op-select-menu-item-terminate
   op-select-menu-item-shutdown)                   :partitions '(AllB))
```

Die ersten fünf Konstanten enthalten Beschreibungen von Plan-Concepts, die auch für die Plangenerierung verwendet werden können. *op-decomposition*, *op-atomic* und *op-incomplete* enthalten die Beschreibung von Hilfs-Concepts zur Repräsentation von Dekompositionen (vgl. Abschnitt 4.6.2 *Repräsentation von Dekompositionen in BGP-MS*). Alle anderen beschreiben sogenannte *unvollständige* Plan-Concepts, die nur in Dekompositionen vorkommen (vgl. die Abschnitte 4.2.1 *Dekomposition eines Plan-Concepts*). Die meisten dieser grundlegenden Plan-Concepts kennen alle Benutzer atomar:

```
(bgp-ms-declare-concepts-atomic '(
   app-print user-get launch
   click-button-ok click-button-start click-button-open
   click-button-download click-button-transfer
   push-2key-alt-h push-2key-alt-o push-2key-alt-t
   push-3key-ctrl-alt-del select-file-menu-item
   select-menu-item-terminate select-menu-item-shutdown)
                                                    :partitions '(AllB))
```

Die nächste Ebene der Partitionenhierarche besteht aus Partitionen für das Systemwissen und für Wissens-Stereotypes. Hinzu kommt mit *All*C der Stereotype, der das Handlungspotential repräsentiert das allen Benutzern zugetraut wird, und mit *NonHandicap*C ein Sterotype für die besonderen Handlungmöglichkeiten von nichtbehinderten Benutzern:

```
(define-bgp-ms-partition 'SB         :supers '(SDomB) :primitive-flag 't)
(define-bgp-ms-partition 'AllC         :supers '(SDomB) ...)
(define-bgp-ms-partition 'NonHandicapC :supers '(SDomB) ...)
(define-bgp-ms-partition 'AllStB       :supers '(AllB) ...)
(define-bgp-ms-partition 'NetNovB      :supers '(AllB) ...)
(define-bgp-ms-partition 'NetExpB      :supers '(AllB) ...)
(define-bgp-ms-partition 'UxB          :supers '(AllB) ...)
(define-bgp-ms-partition 'WinExpB      :supers '(AllB) ...)
```

Da das System sämtliche Plan-Concepts kennt (die der Entwickler des Sachbereichsmodells für korrekt hält), sei für die längliche Beschreibung des Eintrags der Plan-Concepts in das Systemwissens, auf Datei *demo-init-bmwb.lisp* des elektronischen Anhangs verwiesen. Zum Systemwissen gehört auch das Wissen über Dekompositionen von Plan-Concepts. Durch die neue BGP-MS-Funktion *bgp-ms-def-decomposition-from-struclist* werden LISP-Structures für Dekompositionen in gleicher Weise in die BMWB eingetragen, wie die Plan-Concepts durch *bgp-ms-def-plan-concepts-from-struclist*. Das Beispielmodell beschränkt sich auf die im Abschnitt 5.2.2 *Der Beispielsachbereich* beschriebenen Dekompositionen für die Plan-Concepts *ftp-download-file-bin*, *ftp-download-file-text*, *open-file-tpx* und *shutdown-pc*. Von den in Abbildung 5.6 S. 250 gezeigten drei Dekompositionen für den FTP-Transfer soll das System allerdings nur die ersten beiden kennen – die dritte wird also als falsch betrachtet:

```
(bgp-ms-def-decomposition-from-struclist (list
   dec-ftp-bin1 dec-ftp-bin2 dec-ftp-txt1 dec-ftp-txt2
   dec-open-file-tpx dec-shutdown-pc-by-key dec-shutdown-pc-by-menu)
                                                 :partitions '(SB))
```

Jedem Benutzern wird zugetraut die folgenden Plan-Concepts ausführen zu können:

```
(bgp-ms-def-plan-concepts-from-struclist (list
   op-app-print op-user-get op-launch op-open-file-tpx
   op-decomposition op-atomic op-incomplete
   op-click-button op-click-button-ok op-click-button-start
   op-click-button-open op-click-button-download op-click-button-transfer
```

```
   op-push-2key op-push-2key-alt-h op-push-2key-alt-o op-push-2key-alt-t
   op-select-file-menu-item op-select-menu-item
   op-select-menu-item-terminate op-select-menu-item-shutdown)
                                                    :partitions '(AllC))
```

Davon können nur die ersten vier für die Planerzeugung eingesetzt werden. Die restlichen sind – wie bei *All*B – *unvollständige* Plan-Concepts, die nur in Dekompositionen vorkommen, oder Hilfs-Concepts zur Repräsentation von Dekompositionen.

Der Stereotype für die besonderen Handlungsmöglichkeiten der nichtbehinderten Benutzer enthält im Beispiel lediglich das Plan-Concept für das Betätigen der Tastenkombination *<Strg>-<Alt>-<Entf>* und die zugehörigen Super-Concepts.

```
(bgp-ms-def-plan-concepts-from-struclist (list
    op-incomplete op-push-3key op-push-3key-ctrl-alt-del)
                                            :partitions '(NonHandicapC))
```

Die noch offenen Wissens-Stereotypes erhalten folgende Einträge. *Netzwerk-Anfängern* werden Kenntnisse für das Öffnen und Schließen einer FTP-Verbindung sowie für das Wechseln des Verzeichnisses auf dem fremden Computer und die abstrakte Variante des FTP-Transfers zugebilligt. Diese Kenntnisse sind – mit Ausnahme der Kenntnisse über den FTP-Transfer – auch atomar:

```
(bgp-ms-def-plan-concepts-from-struclist (list
   op-com-copy op-ftp-download-file  op-ftp-open-connection
   op-ftp-close-connection op-ftp-cd)             :partitions '(NetNovB))
(bgp-ms-declare-concepts-atomic '(
   ftp-open-connection ftp-close-connection ftp-cd):partitions '(NetNovB))
```

Das Plan-Concept *com-copy* repräsentiert eine Verallgemeinerung aller Kopier-Operationen und beschreibt deshalb die gemeinsamen Eigenschaften all dieser Operationen. Für die Plangenerierung wird es auch dann nicht eingesetzt, wenn anhand des *Wissens* eines *Netzwerk-Anfängers* geplant wird, weil dieser mit *ftp-download-file* auch ein spezielleres Plan-Concept kennt (vgl. Kap. 3.4.2.7 *Erzeugung des aktiven Teils eines Planungssachbereichs*). Für *ftp-download-file* selbst ist aber keine Spezialisierung mehr definiert. Dieses Plan-Concept kann deshalb für eine Plangenerierung anhand des *Wissens* eines *Netzwerk-Anfängers* eingesetzt werden, obwohl es vom – für den Transfer wichtigen – Dateityp abstrahiert. Da *Netzwerk-Experten* normalerweise wissen, daß der Dateityp für den FTP-Transfer relevant ist, sind für sie auch die Spezialisierungen *ftp-download-file-bin* und *ftp-download-file-text* sowie Plan-Concepts für die Umschaltung des Transfer-Modus definiert. Außerdem wissen *Netzwerk-Experten*, daß auch das *lokale* Verzeichnis eines FTP-Programms gewechselt werden kann:

```
(bgp-ms-def-plan-concepts-from-struclist (list op-com-copy
   op-ftp-download-file op-ftp-download-file-bin op-ftp-download-file-text
   op-ftp-open-connection op-ftp-close-connection
   op-ftp-chmode-binary op-ftp-chmode-text op-ftp-cd op-ftp-lcd)
                                                :partitions '(NetExpB))
(bgp-ms-declare-concepts-atomic '(
   ftp-open-connection ftp-close-connection ftp-chmode-binary
   ftp-chmode-text ftp-cd ftp-lcd)              :partitions '(NetExpB))
```

Hinzu kommt die Kenntnis von je einer Dekomposition für *ftp-download-file-bin* und *ftp-download-file-text*:

```
(bgp-ms-def-decomposition-from-struclist (list
   dec-ftp-bin2 dec-ftp-txt2)                   :partitions '(NetExpB))
```

UNIX-Benutzern wird die Kenntnis des Mountens unter UNIX sowie eine Ausführungsmöglichkeit für das Öffnen einer Datei im Textverarbeitungssystem X zugetraut:

```
(bgp-ms-def-plan-concepts-from-struclist (list op-mount op-mount-ux)
                                                  :partitions '(UxB))
(bgp-ms-def-decomposition-from-struclist (list dec-open-file-tpx)
                                                  :partitions '(UxB))
```

Letzteres gilt auch für *Windows-Experten*. Darüberhinaus wird ihnen die atomare Kenntnis des Mountens unter Windows und die Kenntnis über das Herunterfahren eines Windows-PCs in zwei Ausführungsvarianten zugetraut:

```
(bgp-ms-def-plan-concepts-from-struclist (list
   op-mount op-mount-win)                         :partitions '(WinExpB))
(bgp-ms-def-decomposition-from-struclist (list
   dec-open-file-tpx dec-shutdown-pc-by-key dec-shutdown-pc-by-menu)
                                                  :partitions '(WinExpB))
(bgp-ms-declare-concepts-atomic '(mount-win)      :partitions '(WinExpB))
```

Es bleibt die Definition von Ebene 2 der Partitionenhierarchie, die ausschließlich Stereotypes für Handlungspotentiale enthält:

```
(define-bgp-ms-partition 'AllStC :supers '(AllC) ...)
(define-bgp-ms-partition 'StaffC :supers '(AllC) ...)
(define-bgp-ms-partition 'GuestC :supers '(AllC) ...)
(define-bgp-ms-partition 'StudC  :supers '(AllC) ...)
```

Für *AllSt*C sind keine Einträge notwendig, weil der Inhalt von *All*C geerbt wird. Für *Staff*C werden dagegen alle Plan-Concepts eingetragen die auf Befugnissen beruhen, weil für *Angestellte* keine Beschränkungen gelten sollen. Im Vergleich dazu dürfen *studentische Hilfskräfte* keine Verzeichnisse mounten und *Gäste* kein FTP benutzen. Das Herunterfahren eines PCs ist (normalerweise) den *Angestellten* vorbehalten:

```
(bgp-ms-def-plan-concepts-from-struclist (list
   op-com-copy op-ftp-download-file
   op-ftp-download-file-bin op-ftp-download-file-text
   op-ftp-open-connection op-ftp-close-connection
   op-ftp-chmode-binary op-ftp-chmode-text
   op-ftp-cd op-ftp-lcd)                     :partitions '(StaffC StudC))
(bgp-ms-def-plan-concepts-from-struclist (list op-mount op-mount-win)
                                             :partitions '(StaffC GuestC))
(bgp-ms-def-plan-concepts-from-struclist (list op-shutdown-pc)
                                             :partitions '(StaffC))
```

Der initiale Aufbau der BMWB ist damit abgeschlossen und BGP-MS kann in den *Run-Time-Modus* geschaltet werden:

```
(set-run-time-use 't)
```

5.3 Anwendung des Prototyps

Dieser Abschnitt zeigt die Planungs-Ergebnisse, die der Demonstrations-Prototyp für die bereits genannten fünf Beispielbenutzer *Oscar*, *Trixie*, *Nick*, *Theo* und *Sue* generiert. Aus dem vorangegangenen Abschnitt wird sowohl die Benutzermodellierungswissensbasis, die dort aufgebaut wurde, als auch das Beispielziel (*Der Benutzer möchte einen Ausdruck einer Textverarbeitungsdatei haben, die sich auf einem fremden Computer befindet.*) übernommen. Für die Durchführung der Beispiele muß

der gesamte Prototyp einschließlich des Planungssystems gestartet sein – technische Details finden sich in Anhang C.1 *Systemumgebung und -voraussetzungen*.

Alle Ausgangssituationen für die Plangenerierung enthalten einen Drucker und ein Textverarbeitungssystem sowie die zu druckende Datei auf dem fremden Computer. Als Varianten werden die beiden folgenden Fälle behandelt:

1. ein Verzeichnis des fremden Rechners, das die auszudruckende Datei (indirekt) enthält, wird exportiert
2. kein Verzeichnis des fremden Rechners wird exportiert – ein Zugriff auf dessen Dateisystem ist deshalb nur mit FTP möglich

Auf einige Details der Beschreibung der Ausgangssituation wird weiter unten noch eingegangen.

Für jeden der Beispielbenutzer wird zum Vergleich auch ein Plan generiert, der nicht von seinem Handlungspotential, sondern von seinem Wissen ausgeht. Ein solcher Plan kann dazu genutzt werden, eine fehlerhafte Planung des Benutzers zu analysieren (vgl. Abschnitt 6.2.4.4 *Fehlerdiagnose* im "Ausblick"-Kapitel dieser Arbeit) oder einfach abzuschätzen, zu welchem Plan der Benutzer vermutlich gekommen wäre, wenn er kein Hilfesystem zur Verfügung gehabt hätte. Speziell für den Beispielbenutzer *Nick* wird auch ein Plan erzeugt, der auf falschen Annahmen über die objektbezogenen Begriffe im Sachbereich beruht, um die Auswirkung auf die Plangenerierung zu zeigen, wie dies in Kap. 3.4.2.4 *Ableitung von Abhängigkeiten aus dem Benutzermodell* besprochen wurde. Ein weiterer Plan für *Nick* zeigt die Unterscheidung von nutzbaren und nicht nutzbaren Dekompositionen.

Für jeden der erzeugten Pläne wird eine grobe Abschätzung abgegeben, welche Plan-Concepts dem Benutzer erklärt werden müssen. Sie beruht, im Gegensatz zur aufwendigen Analyse im Präsentations-Kapitel, lediglich aus der Überprüfung, welche Plan-Concepts des Plans der Benutzer – ggf. atomar – kennt und – falls er Dekompositionen für nicht-atomare Plan-Concepts kennt – ob diese mit dem Systemwissen übereinstimmen und für ihn *nutzbar* (im Sinne von Definition 4.2 S. 183) sind.

5.3.1 Beispiel-Planungsprobleme

Nach Kap. 3.4 *Planungsproblem und Planungssachbereich* stellt ein Planungsproblem die Aufgabenbeschreibung für ein Planungssystem im engeren Sinne dar. Es besteht aus der Startsituation, die die Gegebenheiten beschreibt, wie sie vor Ausführung des Plans gelten, und einer Beschreibung des Ziels, also der Veränderungen gegenüber der Startsituation, die mit Hilfe des gesuchten Plans erreicht werden sollen.

Das o.g. Beispielziel läßt sich i.w. durch die prädikatenlogische Formel:

$$\exists info \quad \textit{fso-contents}(\textit{F-KN-xy-doc}, info) \wedge \textit{paperdoc}(Dx) \wedge \textit{pd-contents}(Dx, info) \wedge \textit{has}(USER, Dx)$$

ausdrücken, wobei *F-KN-xy-doc*, *Dx* und *USER* Konstanten sind. Die Prädikate sollen dabei folgende Bedeutung haben:

fso-contents (x,y)	Dateisystemobjekt x hat Inhalt y
paperdoc (x)	x ist ein Papierdokument
pd-contents (x,y)	Papierdokument x hat Inhalt y
has (x,y)	x besitzt y

Analog zu den Konstanten für die Beschreibung von Plan- und Domain-Concepts wird die Formel (in UCPOP-Syntax) einer Konstanten (*g-UM99*) zugewiesen:

```
(defconstant                                                g-UM99
  '(exists (fso-contents F-KN-xy-doc ?info)
           (and (paperdoc Dx) (pd-contents Dx ?info) (has SC-USER Dx))))
```

Die Konstante wird später beim Aufruf des Planungssystems verwendet (s.u.). Die Definition befindet sich zusammen mit weiteren Beispielzielen, die hier nicht diskutiert werden, in der Datei *office-world-goals.lisp* im Verzeichnis *KN-PP-Demo* des elektronischen Anhangs.

Eine prädikatenlogische Formel vermag allerdings nur eine *Situation* zu beschreiben – keine *Veränderung*. Im Beispiel wird gefordert, daß eine Datei ausgedruckt wird; d.h. daß ein Papierdokument *erzeugt* wird. Wie in Kap. 3.4.2.5 *Erzeugen von Objekten* beschrieben, wird die Erzeugung eines Objekts vom Typ *Papierdokument* in dieser Arbeit (u.a.) dadurch modelliert, daß für eine Konstante X für die *vor* Planausführung *non-existent*(X) wahr war, *nach* Planausführung *paperdoc*(X) wahr ist. Zur vollständigen Beschreibung des Beispiel-Planungsproblems gehört demnach auch, daß für die Startsituation festgelegt wird, daß *non-existent*(Dx) wahr ist. Die Beschreibungen der beiden Startsituationen des Beispiels enthalten deshalb die Angabe *to-create*(Dx), wobei *to-create*(x) *non-existent*(x) impliziert.

Die Beschreibung einer Startsituation für ein Planungsproblem besteht (normalerweise[50]) aus der Angabe der atomaren Formeln, die vor Ausführung des Plans wahr sind. Dabei wird ein Sachverhalt der realen Welt i.a. durch mehrere atomare Formeln beschrieben. Ein Beispiel dafür ist die Beschreibung des Dateisystems eines Computers (vgl. Kap. 5.2.2.1 *Das Dateisystem*). Für die Anwendung des Prototyps sind solche Beschreibungen zusammengefaßt und wiederum Konstanten zugewiesen. Die Beschreibung von Ausgangssituation 2 (s.o.) lautet dann (in UCPOP-Syntax):

```
(append
   ABox-Comp-KN ABox-fs-KN          ABox-Comp-PC   ABox-FS-PC
   ABox-loc-env ABox-Common-Stuff   ABox-file-xy-doc-on-KN
   '((to-create Dx)))
```

Dabei faßt *ABox-fs-KN* die Formeln zur Beschreibung des Dateisystems des fremden Computers zusammen und *ABox-Comp-KN* diejenigen für die restliche Beschreibung dieses Computers (u.a. Betriebssystem, Name). *ABox-fs-PC* und *ABox-Comp-PC* sind die entsprechenden Konstanten für die Beschreibung des lokalen Computers. *ABox-loc-env* beschreibt die lokale Umgebung (Textverarbeitungssystem, Drucker, FTP-Programm und die Festlegung, welcher Computer der lokale Computer ist). *ABox-Common-Stuff* faßt einige Konstanten zur Beschreibung der Betriebssysteme und Dateitypen zusammen. Durch die Formeln in *ABox-file-xy-doc-on-KN* wird die Datei beschrieben, die ausgedruckt werden soll – u.a. Name, Inhalt und Lokation im Dateisystem. Schließlich besagt die Formel (*to-createDx*) – wie oben beschrieben –, daß das durch *Dx* bezeichnete Objekt noch nicht erzeugt wurde. Die gesamte Liste der Formeln wird wieder einer Konstanten (*init-UM99-dir-not-exported*) zugewiesen und später für den Aufruf des Planungssystems verwendet.

Die zweite Ausgangssituation unterscheidet sich von der ersten genau dadurch, daß der fremde Computer das Verzeichnis */home* (das die auszudruckende Datei indirekt enthält) exportiert. Auch hierfür wird eine Konstante definiert (*init-UM99-dir-is-exported*), die sich aus *init-UM99-dir-not-exported* plus der Formel *(exports KN-home)* ergibt.

Die gesamte Beschreibung dieser und einiger weiterer Test-Ausgangssituationen befindet sich in der Datei *office-world-initials.lisp* im Verzeichnis *KN-PP-Demo* des elektronischen Anhangs.

5.3.2 Benutzerspezifische Pläne

Die Erzeugung der Pläne hat für jeden der fünf Beispielbenutzer folgenden schematischen Aufbau. Zuerst wird das Benutzermodell im engeren Sinne, das aus den Annahmen des Systems über die Annahmen und das Handlungspotential des Benutzers besteht, aufgebaut. Technisch heißt dies, daß die Views SBUB und SBUC mit Inhalt gefüllt werden (s.u.). Vor der Erzeugung der Pläne müssen

[50] vgl. Kap. 3.4.1 *Das Planungsproblem* und Kap. 2.1 *Handlungsplanung*

zunächst diejenigen Plan-Concepts bestimmt werden, die für die Plangenerierung verwendet werden. Davon sind nach Kap. 3.4.2.7 *Erzeugung des aktiven Teils eines Planungssachbereichs* und 4.2.1 *Dekomposition eines Plan-Concepts* diejenigen Plan-Concepts einer View ausgeschlossen, die

- Sub-Concepts haben (wobei Dekompositionen unberücksichtigt bleiben)
- *unvollständig* sind (im Sinne v. Kap. 4.2.1)
- Dekompositionen repräsentieren

sowie die (Hilfs-)Plan-Concepts, die zur Kennzeichnung von Dekompositionen oder unvollständiger oder atomarer Plan-Concepts eingeführt wurden.

Die Bestimmung dieser Plan-Concepts ist Teil der Erzeugung des benutzerspezifischen Planungssachbereichs für das Planungssystem. Sie kann aber auch unabhängig davon durch die Funktion *knpp-get-all-operable-plan-concepts* erfolgen. Von dieser zusätzlichen Möglichkeit wird in den Beispielen Gebrauch gemacht, um einen Eindruck davon zu gewinnen, welche Operatoren das Planungssystem für die Konstruktion des Plans einsetzen kann. Der Aufruf des Planungssystems folgt dann für jede der beiden o.g. Ausgangssituationen mit einer Beschreibung dieser Situation, einer Beschreibung des Beispielziels und mit dem erzeugten Planungssachbereich. Dabei wird zunächst jeweils ein Plan anhand des Handlungspotentials des Benutzers erzeugt – also ein Plan, den der Benutzer im Prinzip ausführen und damit das Beispielziel erreichen kann. Danach ein Plan anhand seines Wissens – also ein Plan, den der Benutzer selbst erzeugen kann und von dem *er* annehmen muß, daß sich damit das Beispielziel erreichen läßt. Jeder Plan wird nach seiner Erzeugung hinsichtlich seines Erklärungsbedarfs für den jeweiligen Benutzer beurteilt.

Die vollständigen Protokolle dieser Anwendungen des Prototyps finden sich aus Platzgründen nur in den folgenden Dateien im Verzeichnis *KN-PP-Demo* des elektronischen Anhangs dieser Arbeit:

- *planning-for-example-users.lisp* die erforderlichen Funktionsaufrufe für die Benutzer *Oscar*, *Trixie* und *Nick*
- *plans-for-example-users.log* das Protokoll der dazugehörenden Ausgaben
- *planning-for-theo.lisp* die erforderlichen Funktionsaufrufe für Benutzer *Theo*
- *plans-for-theo.log* das Protokoll der dazugehörenden Ausgaben
- *planning-for-sue.lisp* die erforderlichen Funktionsaufrufe für Benutzerin *Sue*
- *plans-for-sue.log* das Protokoll der dazugehörenden Ausgaben

Die nachfolgende Darstellung und Diskussion der Beispielläufe beschränkt sich auf die wichtigsten Ergebnisse. Teilweise wurden die Ausgaben des LISP-Systems zur besseren Lesbarkeit in Kleinschreibweise gewandelt. Die erzeugten Pläne tragen Bezeichner, mit denen sie leicht in den Protokollen identifiziert werden können.

Der Beispiellauf für die ersten drei Benutzer wurde zusammengefaßt. Er kann damit auch als Beispiel für (recht radikale) Meinungsänderungen des Systems über den aktuellen Benutzer aufgefaßt werden.

5.3.2.1 Wissensakquisitation für das Benutzermodell

Vor den Beispielläufen sei noch kurz beschrieben, wie das Benutzermodell mit Inhalt gefüllt wird. In Kap. 3.2 *Repräsentation von Planoperatoren* wurde argumentiert, daß durch die Modellierung von

Planungswissen als begriffliches Wissen auf die Techniken zur Akquisition von begrifflichem Wissen für das Benutzermodell zurückgegriffen werden kann und deshalb in dieser Arbeit keiner besonderen Behandlung bedarf. Dementsprechend wurde für die Festlegung der Annahmen über den Benutzer im Demonstrations-Prototyp ein sehr einfacher Mechanismus implementiert. Ein Großteil der Annahmen resultieren aus der Zuordnung der Beispielbenutzer zu den Stereotypes, die im vorangegangenen Abschnitt definiert wurden. Die Zuordnung der Stereotypes folgt den Vorschlägen aus Kap. 3.3.3 *Stereotypes* mit der Ergänzung um den Stereotype *NonHandicap*C (s.o.), dem außer *Nick* alle Beispielbenutzer angehören. Beim Aufbau der BMWB wurden für jeden Stereotype Trigger definiert, die den Stereotype genau dann für den aktuellen Benutzer aktivieren, wenn der aktuelle Benutzer nach den Vorschlägen aus Kap. 3.3.3 zur zugehörigen Benutzergruppe gehört. Beispielsweise soll der Stereotype *Windows-Experte* (*WinExp*B), zu dem nur *Nick* gehört, genau dann aktiviert werden, wenn das System der Ansicht ist, daß es sich beim aktuellen Benutzer um *Nick* handelt. Die Ansicht selbst wird durch das Vorhandensein eines Concepts *bgp-ms-user-<Name-des-Benutzers>*, für *Nick* also *bgp-ms-user-Nick*, im Systemwissen modelliert. Die Aktivierungsregel für *WinExp*B lautet deshalb:

```
(set-stereotype-trigger 'WinExpB :activation-rule
   '(bgp-ms-ask '(B S (:concept bgp-ms-user-Nick))))
```

Außerdem muß der Stereotype deaktiviert werden, wenn das System zu der Ansicht kommt, daß es sich beim aktuellen Benutzer *nicht* um *Nick* handelt:

```
(set-stereotype-trigger 'WinExpB :retraction-rule
   '(not (bgp-ms-ask '(B S (:concept bgp-ms-user-Nick)))))
```

Gehören mehrere der Beispielbenutzer zu einer Benutzergruppe, werden die (De-)Aktivierungsregeln logisch verküpft. Bspw. gehören *Nick* und *Theo* zu den *studentische Hilfskräften*, womit für den Stereotype *Stud*C die folgenden Regeln eingetragen werden:

```
(set-stereotype-trigger 'StudC :activation-rule
   '(or (bgp-ms-ask '(B S (:concept bgp-ms-user-Nick)))
        (bgp-ms-ask '(B S (:concept bgp-ms-user-Theo)))))
(set-stereotype-trigger 'StudC :retraction-rule
   '(not (or (bgp-ms-ask '(B S (:concept bgp-ms-user-Nick)))
             (bgp-ms-ask '(B S (:concept bgp-ms-user-Theo))))))
```

Die Zuordnung eines Benutzers zu den vorgesehenen Stereotypes kann dann durch den Eintrag des zugehörigen "Benutzer-Concepts" in die View SB, dem Löschen etwaiger bisheriger "Benutzer-Concepts" und dem Auslösen der BGP-MS Stereotype-Evaluation bewirkt werden. Die Hilfsfunktion *knppa-define-user* faßt die notwendigen Schritte zusammen; bspw. für Benutzer *Nick*:

```
(knppa-define-user 'bgp-ms-user-Nick :activate 't)
```

Nach der Zuordnung der Stereotypes müssen dann noch die über die Stereotypes hinausgehenden Informationen, die über den jeweiligen Benutzer bekannt sind, in das Benutzermodell eingetragen werden. Dazu kommen die gleichen Funktionen zum Einsatz, die schon für den initialen Aufbau der Benutzermodellierungswissensbasis in Abschnitt 5.2.3 verwendet wurden.

Letztlich werden damit alle Einträge in das Benutzermodell durch das Regiebuch der Beispiele bestimmt. Für eine Anwendung des Prototyps in einem konkreten Hilfesystems müssen die Aktivierungsregeln der Stereotypes gegen solche ersetzt werden, die für den Anwendungsbereich adäquat sind. Gleiches gilt für den Eintrag der zusätzlichen Informationen, für die angemessene Bedingungen festgestellt werden müssen, aufgrund welcher Beobachtung(en) einem Benutzer welche zusätzlichen Kenntnisse und/oder Handlungsmöglichkeiten zugesprochen werden.

5.3.2.2 Pläne für Oscar

Oscar ist ein durchschnittlicher Büroangestellter ohne besondere Kenntnisse oder sonstige Eigenschaften. Ihm wird deshalb der Stereotype *AllSt*B für seine Annahmen sowie *Staff*C und *NonHandicap*C für sein Handlungspotential zugeordnet. Die Aktivierung der Stereotypes wird – wie oben beschriebenen – dadurch ausgelöst, daß dem System bekannt gemacht wird, daß nun *Oscar* der aktuelle Benutzer ist:

```
(knppa-define-user 'bgp-ms-user-Oscar :activate 't)
```

Das Ergebnis der Stereotype-Aktivierung kann mit der Funktion *prt-partition* überprüft werden:

```
(prt-partition 'sbub) ->  Partition SBUB has Supers (ALLSTB)
(prt-partition 'sbuc) ->  Partition SBUC has Supers (NONHANDICAPC STAFFC)
```

Als *Angestellter* hat *Oscar* keinerlei Einschränkungen seiner Befugnisse. Sein Handlungspotential enthält daher fast alle Plan-Concepts, die das System kennt. Der Aufruf

```
(knpp-get-all-plan-concepts 'sbuc)
```

gibt alle Plan-Concepts der View SBUC aus:

```
(shutdown-pc mount-win mount ftp-lcd ftp-cd ftp-chmode-text
 ftp-chmode-binary ftp-close-connection ftp-open-connection
 ftp-download-file-text ftp-download-file-bin ftp-download-file
 com-copy push-3key-ctrl-alt-del push-3key select-menu-item-shutdown
 select-menu-item-terminate select-menu-item select-file-menu-item
 push-2key-alt-t push-2key-alt-o push-2key-alt-h push-2key
 click-button-transfer click-button-download click-button-open
 click-button-start click-button-ok click-button incomplete atomic
 decomposition open-file-tpx launch user-get app-print)
```

Durch die Funktion *knpp-get-all-operable-plan-concepts* können diejenigen Plan-Concepts einer View bestimmt werden, die für den Planungsprozeß geeignet sind:

```
(knpp-get-all-operable-plan-concepts 'sbuc)
(app-print user-get launch open-file-tpx ftp-download-file-bin
 ftp-download-file-text ftp-open-connection ftp-close-connection
 ftp-chmode-binary ftp-chmode-text ftp-cd ftp-lcd mount-win shutdown-pc)
```

Für den Aufruf des Planungsprozesses ist die Funktion *knpp-generate-plan* vorgesehen, die als Parameter

- eine LISP-Structure *bgp-ms-planning-domain-struc*, die den Planungssachbereich durch Planoperatoren und UCPOP-Axiome beschreibt[51],
- einen String für eine Beschreibung der Startsituation und
- einen String für eine Beschreibung des Planungsziels

erwartet. Verweisen die Konstanten *init-UM99-dir-is-exported* bzw. *g-UM99* (wie im vorangegangen Abschnitt beschrieben) auf Beschreibungen von Startsituation und Ziel, erzeugt der Aufruf:

```
(setf *user-plan*
      (ucpop:knpp-generate-plan (knpp-view-2-planning-domain 'sbuc)
         :initials (prin1-to-string init-UM99-dir-is-exported)
         :goal     (prin1-to-string g-UM99)))
```

[51] Die Stucture *kann* auch UCPOP-Facts enthalten. Das wird in dieser Arbeit allerdings nicht genutzt, da Facts nicht benutzerspezifisch sind und dem Planungssystem bereits bei der Initialisierung mitgeteilt werden.

einen Plan anhand des Handlungspotentials des aktuellen Benutzers *Oscar*, wobei von der ersten Startsituation ausgegangen wird, in der der fremde Computer ein Verzeichnis exportiert, das die auszudruckende Datei indirekt enthält. Die Funktion *knpp-view-2-planning-domain* übernimmt dabei – wie in Kap. 3.4.2.7 *Erzeugung des aktiven Teils eines Planungssachbereichs* beschrieben – die Umsetzung der beschreibungslogischen Repräsentation von Plan- und Domain-Concepts in UCPOP-Planoperatoren und Axiome. Das Ergebnis des Planungsprozesses ist eine LISP-Structure vom Typ *Plan*, die der Variablen **user-plan** zur weiteren Verarbeitung zugewiesen wird. Zusätzlich gibt UCPOP ein Protokoll aus, das u.a. die Beschreibung der Startsituation und des erzeugten Plans enthält. Wegen der Größe dieser Beschreibungen (diese Startsituation besteht bspw. aus 124 einzelnen Prädikaten) finden sie sich lediglich in den o.g. Protokollen im elektronischen Anhang. Außerdem enthält Anhang C.2 die UCPOP-Ausgabe für diesen Plan (O-C-e) und den nachfolgenden (O-C-ne). Hier seien lediglich die einzelnen Schritte mit ihren (nun gebundenen) Parametern, aber ohne Causal Links angegeben:

```
Step 1 : (LAUNCH MYTP)
Step 2 : (MOUNT-WIN COMP-PC PC-ROOT SC-OS-TYPE-WIN KN-HOME FN-PC-DRIVE-P)
Step 3 : (OPEN-FILE-TPX F-KN-XY-DOC MYTP)
Step 4 : (APP-PRINT FCO-KN-XY-DOC LOC-PR F-KN-XY-DOC DX MYTP)
Step 5 : (USER-GET LOC-PR DX SC-USER)
```

Durch die Schritt-Nummer wird die Ausführungsreihenfolge angegeben. Abb. 5.11a enthält eine graphische Darstellung des Plans, wobei Planschitte durch Rechtecke dargestellt sind, die durch Pfeile für die (*relevanten*[52]) Causal Links verbunden sind. Der mit *goal* bezeichnete Schritt ist als Pseudo-Schritt[53] zu verstehen, der nicht ausgeführt werden muß, sondern lediglich die Bedingungen für das Erreichen des Ziels anzeigt.

Für eine (einfache) Abschätzung des erforderlichen Erklärungsaufwandes für den jeweiligen Benutzer ist die Funktion *knpp-print-plan-comment* vorgesehen, die als Parameter eine UCPOP-Plan-Structure erwartet:

```
(knpp-print-plan-comment *user-plan*)
```

Für den gerade erzeugten Plan ergibt sich:

```
Plan-Concepts des Plans:
     (USER-GET LAUNCH MOUNT-WIN OPEN-FILE-TPX APP-PRINT)
davon kennt Benutzer atomar:
     (USER-GET LAUNCH APP-PRINT)
Benutzer kennt Dekompositionen fuer:
     kein Plan-Concept des Plans
Erklaert werden muessen deshalb:
     (OPEN-FILE-TPX MOUNT-WIN)
```

Oscar hat also die notwendigen Handlungsmöglichkeiten, um das Ziel zu erreichen. Allerdings kennt er keine Dekompositionen für die Plan-Concepts, die im Plan verwendet wurden. Da er auch keine atomare Kenntnis der Plan-Concepts *open-file-tpx* und *mount-win* hat, muß ihm für diese beiden Plan-Concepts erklärt werden, wie sie ausgeführt werden können.

Für die zweite Ausgangssituation (*init-UM99-dir-not-exported*), in der kein geeignetes Verzeichnis exportiert wird und die auszudruckende Datei damit nicht durch Mounten in das lokale Dateisystem gebracht werden kann, ergibt sich bei ansonsten gleichen Bedingungen folgender Plan (O-C-ne), der in Abb. 5.11b auch graphisch dargestellt ist:

[52] vgl. S. 175 in Kap. 4.1 *Planstruktur*
[53] vgl. Kap. 2.1.1 *Least Commitment Planning*

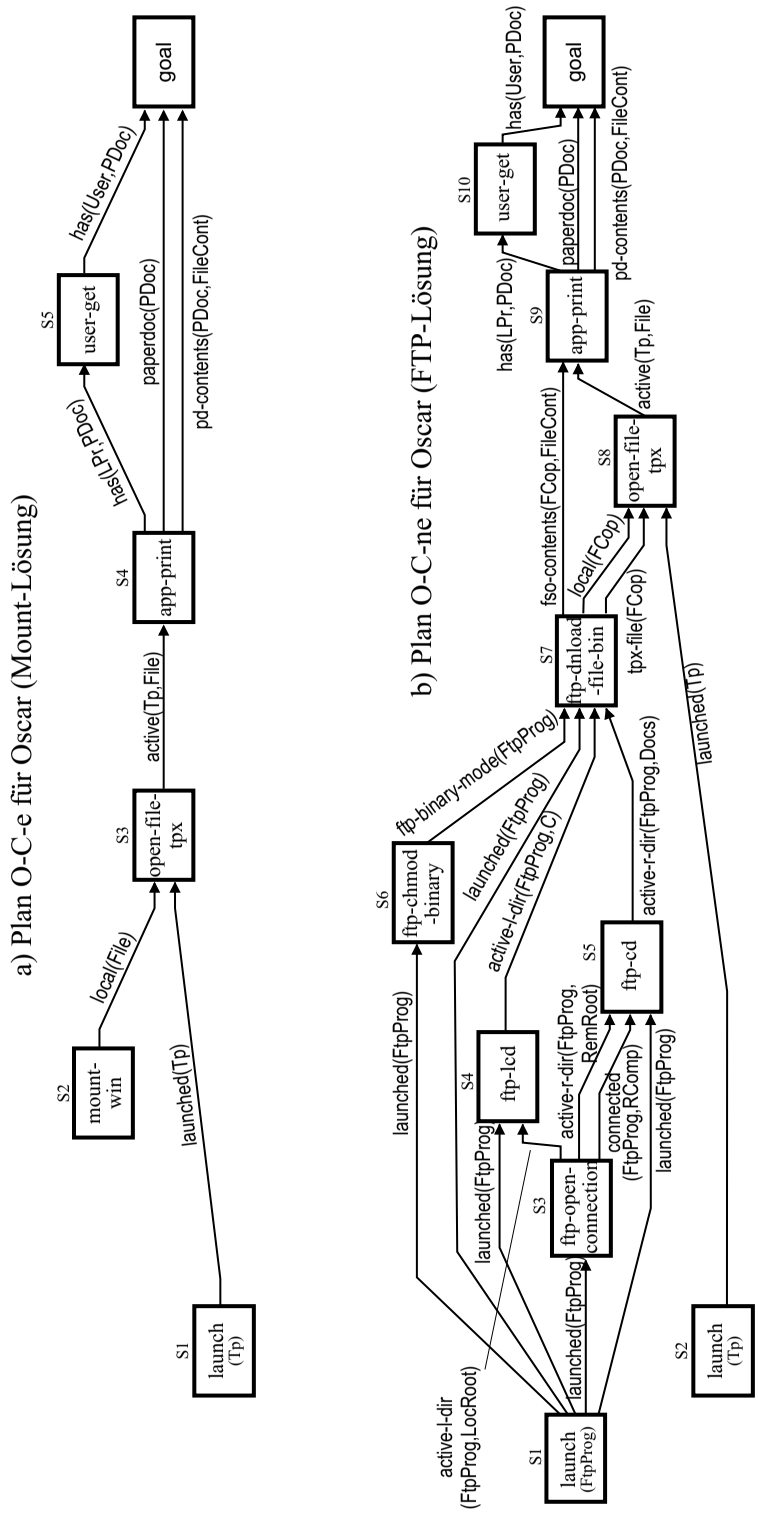

Abbildung 5.11: Pläne für das Ziel *Einen Ausdruck einer Datei bekommen, die sich auf einem fremden Computer befindet.* a) für Ausgangssituation 1, b) für Ausgangssituation 2.

```
Step 1 : (LAUNCH MYTP)
Step 2 : (LAUNCH MYFTP)
Step 3 : (FTP-OPEN-CONNECTION MYFTP COMP-KN PC-ROOT KN-ROOT)
Step 4 : (FTP-LCD MYFTP PC-DRIVE-C PC-ROOT)
Step 5 : (FTP-CD COMP-KN MYFTP KN-DOCS KN-ROOT)
Step 6 : (FTP-CHMODE-BINARY MYFTP)
Step 7 : (FTP-DOWNLOAD-FILE-BIN MYFTP F-KN-XY-DOC PC-DRIVE-C FN-KN-XY-DOC
          SC-FILE-TYPE-TPX obj-426 KN-DOCS FCO-KN-XY-DOC obj-427
          STR-XY-DOC)
Step 8 : (OPEN-FILE-TPX obj-426 MYTP)
Step 9 : (APP-PRINT FCO-KN-XY-DOC LOC-PR obj-426 DX MYTP)
Step 10: (USER-GET LOC-PR DX SC-USER)
```

Oscar kann also auch in diesem Fall das Ziel erreichen, die Beurteilung

```
Plan-Concepts des Plans:
     (USER-GET LAUNCH OPEN-FILE-TPX FTP-LCD FTP-CD FTP-OPEN-CONNECTION
      FTP-CHMODE-BINARY FTP-DOWNLOAD-FILE-BIN APP-PRINT)
davon kennt Benutzer atomar:
     (USER-GET LAUNCH APP-PRINT)
Benutzer kennt Dekompositionen fuer:
     kein Plan-Concept des Plans
Erklaert werden muessen deshalb:
     (FTP-DOWNLOAD-FILE-BIN FTP-CHMODE-BINARY FTP-OPEN-CONNECTION FTP-CD
      FTP-LCD OPEN-FILE-TPX)
```

zeigt aber, daß *Oscar* für diesen Plan einen sehr hohen Erklärungsbedarf hat.

Zum Vergleich wird noch überprüft, ob *Oscar* selbst in der Lage wäre, einen Plan für das Beispielziel zu erstellen. Dazu wird das Planungssystem wiederum mit den beiden Ausgangssituationen aufgerufen. Der Planungssachbereich wird aber nun aufgrund der (Systemannahmen über die) *Annahmen* des Benutzers erzeugt. Die Plangenerierung hat damit die folgenden Plan-Concepts bzw. daraus abgeleitenten Plan-Operatoren zur Verfügung:

```
(open-file-tpx app-print user-get launch shutdown-pc)
```

Damit ist das Ziel von keiner der Ausgangssituationen zu erreichen. *Oscar* ist also (nach Ansicht des Systems) nicht in der Lage, selbst einen Plan zu erzeugen der das Ziel erreicht.

5.3.2.3 Pläne für Trixie

Trixie ist eine typische Unix-Benutzerin mit Netzwerkkenntnissen und wird daher den Stereotypes *Ux*B und *NetExp*B zugeordnet. Da sie keine reguläre Mitarbeiterin sondern Gast ist, wird sie außerdem dem Fähigkeits-Stereotype *Guest*C zugeordnet. Hinzu kommen die Handlungsmöglichkeiten aus Stereotype *NonHandicap*C, weil sie als nichtbehinderte Benutzerin eingestuft wird:

```
Partition SBUB has   Supers (UXB NETEXPB)
Partition SBUC has   Supers (GUESTC NONHANDICAPC)
```

Für die Plangenerierung werden dementsprechend die Plan-Concepts

```
(app-print user-get launch open-file-tpx und mount-win)
```

eingesetzt.

Für die erste Ausgangssituation, in der ein geeignetes Verzeichnis des fremden Computers exportiert wird, reichen diese Handlungsmöglichkeiten aus. Für *Trixie* wird deshalb mit T-C-e der gleiche Plan wie für *Oscar* (O-C-e) mit den Schritten:

launch → mount-win → open-file-tpx → app-print → user-get

erzeugt. Allerdings kennt *Trixie* im Gegensatz zu *Oscar* eine Dekomposition für *open-file-tpx* und benötigt deshalb nur eine Beschreibung für *mount-win*:

```
Plan-Concepts des Plans:
     (USER-GET LAUNCH MOUNT-WIN OPEN-FILE-TPX APP-PRINT)
davon kennt Benutzer atomar:
     (USER-GET LAUNCH APP-PRINT)
Benutzer kennt Dekompositionen fuer:
     OPEN-FILE-TPX: (D-OPEN-FILE-TPX)
Erklaert werden muessen deshalb:
     (MOUNT-WIN)
```

Wird dagegen davon ausgegangen, daß kein geeignetes Verzeichnis exportiert wird, reichen *Trixies* Befugnisse nicht aus, um das Ziel zu erreichen. Der Plangenerierungsprozeß stellt fest, daß es keine Lösung für dieses Planungsproblem gibt (T-C-ne).

Um zu überprüfen, ob *Trixie* selbst einen Plan für das Ziel erstellen kann, werden ihre *Kenntnisse* für die Erzeugung des Planungssachbereichs zugrunde gelegt. Die Plangenerierung hat damit die folgenden Plan-Concepts bzw. daraus abgeleiteten Plan-Operatoren zur Verfügung:

```
(open-file-tpx app-print user-get launch ftp-open-connection
 ftp-close-connection ftp-chmode-binary ftp-chmode-text ftp-cd ftp-lcd
 ftp-download-file-bin ftp-download-file-text mount-ux shutdown-pc)
```

Darin fehlt das Plan-Concept für das Mounten unter Windows. Der Unterschied der beiden Ausgangssituationen wird dadurch irrelevant. Es wird dementsprechend in beiden der gleiche Plan erzeugt (T-B-e und T-B-ne), der die auszudruckende Datei via FTP-Transfer in das lokale Dateisystem bringt:

```
Step 1 : (LAUNCH MYTP)
Step 2 : (LAUNCH MYFTP)
Step 3 : (FTP-OPEN-CONNECTION COMP-KN PC-ROOT MYFTP KN-ROOT)
Step 4 : (FTP-LCD PC-DRIVE-C PC-ROOT MYFTP)
Step 5 : (FTP-CD KN-DOCS COMP-KN KN-ROOT MYFTP)
Step 6 : (FTP-CHMODE-BINARY MYFTP)
Step 7 : (FTP-DOWNLOAD-FILE-BIN F-KN-XY-DOC obj-436 obj-435 MYFTP
          FCO-KN-XY-DOC KN-DOCS FN-KN-XY-DOC STR-XY-DOC
          SC-FILE-TYPE-TPX PC-DRIVE-C)
Step 8 : (OPEN-FILE-TPX MYTP obj-435)
Step 9 : (APP-PRINT obj-435 LOC-PR FCO-KN-XY-DOC MYTP DX)
Step 10: (USER-GET DX LOC-PR SC-USER)
```

Trixie benötigt zu diesem Plan keine Erklärungen:

```
Plan-Concepts des Plans:
     (USER-GET LAUNCH OPEN-FILE-TPX FTP-LCD FTP-CD FTP-OPEN-CONNECTION
      FTP-CHMODE-BINARY FTP-DOWNLOAD-FILE-BIN APP-PRINT)
davon kennt Benutzer atomar:
     (USER-GET LAUNCH FTP-LCD FTP-CD FTP-OPEN-CONNECTION FTP-CHMODE-BINARY
      APP-PRINT)
Benutzer kennt Dekompositionen fuer:
     FTP-DOWNLOAD-FILE-BIN: (D2-FTP-BIN)
     OPEN-FILE-TPX: (D-OPEN-FILE-TPX)
Keine Erklaerungen notwendig
```

Das Beispiel *Trixie* zeigt die Möglichkeit auf, daß ein Benutzer einen Plan erstellen kann, der zwar prinzipiell geeignet ist das Ziel zu erreichen und für den er bzw. sie auch keine Erklärungen benötigt, der aber dennoch für ihn / sie ungeeignet ist, weil ihm / ihr die notwendigen Befugnisse fehlen.

5.3.2.4 Pläne für Nick

Nick ist eine studentische Hilfskraft. Bzgl. seiner Handlungsmöglichkeiten/Befugnisse wird er deshalb dem Stereotype *Stud*C zugeordnet, allerdings – wegen einer Behinderung – nicht wie alle anderen Beispielbenutzer auch dem Stereotype *NonHandicap*C. Aufgrund seiner guten Kenntnisse wird er als Windows- und Netzwerk-Experte eingestuft.

```
Partition SBUB has   Supers (WINEXPB NETEXPB)
Partition SBUC has   Supers (STUDC)
```

Weil *Nick* bereits lange angestellt ist und zuverlässig arbeitet[54], wird ihm soviel Vertrauen entgegengebracht, daß er zusätzliche Befugnisse bekommt: zum Mounten interner, freigegebener Verzeichnisse und zum Herunterfahren eines Computers. Diese zusätzlichen Befugnisse werden in gleicher Weise in das Benutzermodell – hier konkret in die View SBUC – eingetragen, wie beim initialen Aufbau der Benutzermodellierungswissensbasis:

```
(bgp-ms-def-plan-concepts-from-struclist
   (list op-mount op-mount-win op-shutdown-pc) :partitions '(SBUC))
```

Gegenüber einem Netzwerk-Experten kennt *Nick* außerdem jeweils eine weitere Dekomposition für die Plan-Concepts *ftp-download-file-bin* und *ftp-download-file-text*. Auch deren Eintrag erfolgt wie beim initialen Aufbau der BMWB:

```
(bgp-ms-def-decomposition-from-struclist
   (list dec-ftp-bin3 dec-ftp-txt3) :partitions '(SBUB))
```

Für den Planungsprozeß werden damit folgende Plan-Concepts bzw. Plan-Operatoren eingesetzt:

```
(app-print user-get launch open-file-tpx ftp-download-file-bin
 ftp-download-file-text ftp-open-connection ftp-close-connection
 ftp-chmode-binary ftp-chmode-text ftp-cd ftp-lcd mount-win shutdown-pc)
```

Für die erste Ausgangssituation (Verzeichnis wird exportiert) wird damit für *Nick* der Plan N-C-e erzeugt, der die gleichen Schritte hat, wie die entsprechenden Plänen für *Oscar* (O-C-e) und *Trixie* (T-C-e):

launch → mount-win → open-file-tpx → app-print → user-get

Weil sich *Nick* sehr gut auskennt, lautet die Beurteilung des Plans nun aber:

```
Plan-Concepts des Plans:
     (USER-GET LAUNCH MOUNT-WIN OPEN-FILE-TPX APP-PRINT)
davon kennt Benutzer atomar:
     (USER-GET LAUNCH MOUNT-WIN APP-PRINT)
Benutzer kennt Dekompositionen fuer:
     OPEN-FILE-TPX: (D-OPEN-FILE-TPX)
Keine Erklaerungen notwendig
```

Für die zweite Ausgangssituation (Verzeichnis wird *nicht* exportiert) wird der Plan N-C-ne erzeugt, der wiederum die gleichen 10 Schritte wie der entsprechende Plan für *Oscar* (O-C-ne) hat:

launch → launch → ftp-open-connection → ftp-lcd → ftp-cd → ftp-chmode-binary → ftp-download-file-bin → open-file-tpx → app-print → user-get

Allerdings sind für *Nick* andere Erklärungen notwendig:

[54] vgl.a. Kap. 3.3.3 *Stereotypes*

```
Plan-Concepts des Plans:
     (USER-GET LAUNCH OPEN-FILE-TPX FTP-LCD FTP-CD FTP-OPEN-CONNECTION
      FTP-CHMODE-BINARY FTP-DOWNLOAD-FILE-BIN APP-PRINT)
davon kennt Benutzer atomar:
     (USER-GET LAUNCH FTP-LCD FTP-CD FTP-OPEN-CONNECTION FTP-CHMODE-BINARY
      APP-PRINT)
Benutzer kennt Dekompositionen fuer:
     FTP-DOWNLOAD-FILE-BIN: (D2-FTP-BIN D3-FTP-BIN) davon nicht in SB
                            enthalten: (D3-FTP-BIN)
     OPEN-FILE-TPX: (D-OPEN-FILE-TPX)
Erklaert werden muessen deshalb:
     (FTP-DOWNLOAD-FILE-BIN)
```

Nick kennt alle notwendigen Plan-Concepts – die meisten sogar atomar. Für die restlichen kennt *Nick* Dekompositionen. Allerdings ist eine der Dekompositionen von *ftp-download-file-bin* die *Nick* kennt nicht auch dem System bekannt; d.h. das System stuft sie als *falsch* ein. *Nicks* Wissen über das Plan-Concept ist also nicht *verläßlich* (vgl. Kap. 4.2.2.3 *Verläßliches Wissen über die Ausführungsmöglichkeiten von Plan-Concepts*). Nach Kap. 4.3 *Inhaltsbestimmung und Bewertung einer Planpräsentation* muß dem Benutzer in diesem Fall das Plan-Concept erklärt werden[55].

Nicks Handlungspotential und Wissen unterscheiden sich nicht bzgl. der Plan-Concepts, die für die Plangenerierung eingesetzt werden. Dementsprechend sind die Pläne N-B-e und N-B-ne, die auf Basis von *Nicks Wissen* erzeugt werden, identisch zu den entprechdenden Plänen aufgrund von *Nicks* Handlungspotential (N-C-e bzw. N-C-ne).

Speziell für *Nick* wird noch ein Plan für das Herunterfahren des Computers *Comp-PC* erzeugt. Das Ziel lautet also:

```
(down Comp-PC)
```

Der Plan (N-C-PCdn-e) selbst ist trivial, weil das Ziel mit einem einzigen Plan-Concept/Operator erreicht werden kann, das sich auch in *Nicks* Handlungspotential befindet:

```
Step 1 : (SHUTDOWN-PC COMP-PC)
```

Die unten stehende Beurteilung des Plans zeigt die Auswirkung von *Nicks* Behinderung. *Nick* kennt zwar die beiden Ausführungsalternativen zum Herunterfahren eines PCs, er kann aber lediglich die Variante des Herunterfahrens via Menü verwenden. Der erste Schritt der Dekomposition *shutdown-pc-by-key* ist vom Typ *push-3key-ctrl-alt-del*[56] – ein Plan-Concept, das nicht zu *Nicks* Handlungspotential gehört. Diese Dekomposition ist deshalb für *Nick nicht nutzbar*.

```
Plan-Concepts des Plans:
     (SHUTDOWN-PC)
davon kennt Benutzer atomar:
     NIL
Benutzer kennt Dekompositionen fuer:
     SHUTDOWN-PC: (SHUTDOWN-PC-BY-KEY SHUTDOWN-PC-BY-MENU)
                  davon nicht nutzbar: (SHUTDOWN-PC-BY-KEY)
Keine Erklaerungen notwendig
```

Anhand des Beispielbenutzers *Nick*, des Beispielziels und der Startsituation, in der kein Verzeichnis des fremden Rechners exportiert wird, soll auch noch die Auswirkung von falschen Annahmen über die objektbezogenen Begriffe im Sachbereich auf die Erzeugung eines Plans gezeigt werden,

[55] Für eine ausführliche Diskussion der Konsequenzen aus falschen Benutzerannahmen sei auf Kap. 4.3.3 verwiesen.
[56] vgl. Abb. 5.5 S. 250 und Anhang B.2 *Plan-Concepts*.

die in Kap. 3.4.2.4 *Ableitung von Abhängigkeiten aus dem Benutzermodell* besprochen wurde. Dazu werden *Nicks* Annahmen geändert, indem zunächst das Concept für die Kennzeichnung von Textverarbeitungs-Dateien *file-type-tpx* nicht mehr als Spezialisierung einer Kennzeichnung für Binär-Dateien, sondern als Spezialisierung einer Kennzeichnung für Text-Dateien eintragen wird. Danach wird der Typ der Textverarbeitungs-Dateien *tpx-file* selbst in gleicher Weise durch SB-ONE-Funktionen geändert:

```
(add-father (id 'file-type-tpx) (id 'file-type-text) :views '(sbub))
(rem-father (id 'file-type-tpx) (id 'file-type-bin)  :views '(sbub))
(add-father (id 'tpx-file) (id 'text-file) :views '(sbub))
(rem-father (id 'tpx-file) (id 'bin-file)  :views '(sbub))
```

Nick glaubt nun, dass Textverarbeitungs-Dateien Text-Dateien seien, was sich durch die Hilfsfunktion *knppa-prt-concept* überpüfen läßt:

```
(knppa-prt-concept 'tpx-file 'sbub)
DEFINED Concept TPX-FILE (GC31) is-a ((TEXT-FILE (GC29)))
  ACTIVE (GR233): "THING" (GC0)  [(1 INF), OPT]
  ACTIVE-L-DIR (GR2112): "THING" (GC0)  [(1 INF), OPT]
  ACTIVE-R-DIR (GR2047): "THING" (GC0)  [(1 INF), OPT]
  CONTAINS (GR1767): "THING" (GC0)  [(1 INF), OPT]
  DESIGNATES (GR1708): "THING" (GC0)  [(1 INF), OPT]
  FILE-TYPE (GR20): FILE-TYPE-TPX (GC17)  [(1 1), NEC]
  FS-ROOT-OF-COMP (GR1896): "THING" (GC0)  [(1 INF), OPT]
  FSN-VAL (GR1649): "THING" (GC0)  [(1 INF), OPT]
  FSO-CONTENTS (GR19): INFORMATION (GC3)  [(1 1), NEC]
  FSO-CONTENTS (GR103): "THING" (GC0)  [(1 INF), OPT]
  FSO-NAMED (GR18): FS-NAME (GC21)  [(1 INF), OPT]
  HAS (GR162): "THING" (GC0)  [(1 INF), OPT]
  LOCAL-FS-OBJECT-OF-COMP (GR1988): "THING" (GC0)  [(1 INF), OPT]
*** No RVMs ***
```

Obwohl keine *Plan*-Concepts verändert oder gar entfernt wurde, resultiert die Plangenerierung auf Basis von *Nicks*-Annahmen nun im Plan (N-BW-ne):

```
Step 1 : (LAUNCH MYTP)
Step 2 : (LAUNCH MYFTP)
Step 3 : (FTP-OPEN-CONNECTION PC-ROOT KN-ROOT COMP-KN MYFTP)
Step 4 : (FTP-LCD PC-ROOT MYFTP PC-DRIVE-C)
Step 5 : (FTP-CD COMP-KN KN-ROOT MYFTP KN-DOCS)
Step 6 : (FTP-DOWNLOAD-FILE-TEXT obj-462 FN-KN-XY-DOC F-KN-XY-DOC
          SC-FILE-TYPE-TPX STR-XY-DOC PC-DRIVE-C MYFTP FCO-KN-XY-DOC
          obj-463 KN-DOCS)
Step 7 : (OPEN-FILE-TPX obj-462 MYTP)
Step 8 : (APP-PRINT FCO-KN-XY-DOC DX MYTP obj-462 LOC-PR)
Step 9 : (USER-GET DX SC-USER LOC-PR)
```

Der Plan unterscheidet sich von dem entsprechenden Plan (N-B-ne), der auf der Basis von *Nicks bisherigem* Wissen erzeugt wurde, dadurch daß die Datei nun im Text-Modus transferiert wird und demnach auch kein Einschalten des Binär-Modus notwendig ist. Die Ursache für diesen falschen Plan liegt darin, daß *Nicks* Fehlannahme bewirkt, daß anstelle des korrekten potentiellen Axioms

$$\forall x\ \textit{tpx-file}(x) \rightarrow \textit{bin-file}(x)$$

das falsche

$$\forall x\ \textit{tpx-file}(x) \rightarrow \textit{text-file}(x)$$

für die Plangenerierung eingesetzt wird, was sich durch die Funktion *knpp-check-axiom* überprüfen läßt:

```
(mapcar #'(lambda (x) (knpp-check-axiom x 'sbub :print 'reject))
        (knpp-get-all-pot-axioms))
```

Der Wert *'reject* für Parameter *print* bewirkt, daß potentielle Axiome, die sich anhand der angegebenen View ableiten lassen, nur mit Namen, die anderen hingegen vollständig angegeben werden.

```
Axiom nicht akzeptiert: (:AXIOM AX-BIN-FILE2
                         :CONTEXT (TPX-FILE ?V1) :IMPLIES (BIN-FILE ?V1))
(AX-CREATE-NEW-VAR AX-FIND-TO-CREATE-OBJECT AX-LOCAL-FS-OBJECT-OF-COMP
 AX-LOCAL-NAMED-FS-OBJECT AX-LOCAL-FS-NAME1 AX-LOCAL-FS-NAME2
 AX-LOCAL-FS-ROOT  AX-WRONG-TPX-TEXT-FILE NIL AX-BIN-FILE1 AX-EXE-FILE
 AX-TPX-FILE AX-TEXT-FILE)
```

Zum Vergleich liefert die Überprüfung der potentiellen Axiome anhand des Systemwissens SB die erwartete Rückweisung des falschen Axioms, während alle anderen Axiome ableitbar sind.

```
Axiom nicht akzeptiert: (:AXIOM AX-WRONG-TPX-TEXT-FILE
                         :CONTEXT (TPX-FILE ?V1) :IMPLIES (TEXT-FILE ?V1))
(AX-CREATE-NEW-VAR AX-FIND-TO-CREATE-OBJECT AX-LOCAL-FS-OBJECT-OF-COMP
 AX-LOCAL-NAMED-FS-OBJECT AX-LOCAL-FS-NAME1 AX-LOCAL-FS-NAME2
 AX-LOCAL-FS-ROOT NIL AX-BIN-FILE2 AX-BIN-FILE1 AX-EXE-FILE AX-TPX-FILE
 AX-TEXT-FILE)
```

Da das objektbezogene Systemwissen nach Konstruktion des Benutzermodells in alle Partitionen die ein Handlungspotential repräsentieren vererbt wird, werden für die Plangenerierung auf Basis eines Handlungspotentials grundsätzlich die (nach Systemansicht) korrekten Axiome verwendet. Eine Plangenerierung auf Basis der Annahmen eines Benutzers kann dagegen falsche Axiome berücksichtigen, was für die Analyse von falschen Benutzerplänen genutzt werden kann (vgl. wiederum Kap. 3.4.2.4).

5.3.2.5 Pläne für Theo

Theo ist wie *Nick* studentische Hilfskraft, weswegen er dem Stereotype *Stud*C zugeordnet wird. Hinzu kommt *NonHandicap*C für nichtbehinderte Benutzer. Bgzl. seines Wissens wird *Theo* als Netzwerk-Anfänger (*NetNov*B) eingestuft:

```
Partition SBUB has   Supers (NETNOVB)
Partition SBUC has   Supers (NONHANDICAPC STUDC)
```

Außerdem nimmt *Theo* an, daß es ein Plan-Concept *remote-copy* gibt, mit dem sich auch Dateien fremder Computer beliebig kopieren lassen. Der entsprechende Eintrag erfolgt wie bei *Nick* durch die Funktion *bgp-ms-def-plan-concepts-from-struclist*:

```
(bgp-ms-def-plan-concepts-from-struclist (list op-remote-copy)
                                         :partitions '(SBUB))
```

Theos Handlungspotential enthält damit folgende Plan-Concepts, die für den Planungsprozeß eingesetzt werden können:

```
(app-print user-get launch open-file-tpx ftp-download-file-bin
 ftp-download-file-text ftp-open-connection ftp-close-connection
 ftp-chmode-binary ftp-chmode-text ftp-cd ftp-lcd)
```

Gegenüber *Nicks* Handlungspotential fehlt *Theo* die Befugnis zum Mounten von Verzeichnissen. Er kann deshalb auch in der ersten Ausgangssituation (Verzeichnis wird exportiert) nicht den kurzen Plan (N-C-e) ausführen, der für *Nick* erzeugt wurde. Der Plan Th-C-e, der für *Theo* in der ersten Ausgangssituation erzeugt wird, unterscheidet sich deshalb nicht vom Plan Th-C-ne für die zweite Ausgangssituation (Verzeichnis wird nicht exportiert). *Theo* muß die Datei in beiden Situationen via FTP ins lokale Dateiverzeichnis bringen:

```
Step 1 : (LAUNCH MYTP)
Step 2 : (LAUNCH MYFTP)
Step 3 : (FTP-OPEN-CONNECTION KN-ROOT COMP-KN MYFTP PC-ROOT)
Step 4 : (FTP-LCD PC-ROOT MYFTP PC-DRIVE-C)
Step 5 : (FTP-CD COMP-KN KN-ROOT MYFTP KN-DOCS)
Step 6 : (FTP-CHMODE-BINARY MYFTP)
Step 7 : (FTP-DOWNLOAD-FILE-BIN FN-KN-XY-DOC obj-426 F-KN-XY-DOC
          PC-DRIVE-C obj-427 STR-XY-DOC MYFTP FCO-KN-XY-DOC
          SC-FILE-TYPE-TPX KN-DOCS)
Step 8 : (OPEN-FILE-TPX obj-426 MYTP)
Step 9 : (APP-PRINT MYTP obj-426 DX FCO-KN-XY-DOC LOC-PR)
Step 10: (USER-GET LOC-PR DX SC-USER)
```

Da *Theo* lediglich das Wissen eines Netzwerk-Anfängers hat, müssen ihm viele Plan-Concepts erklärt werden:

```
Plan-Concepts des Plans:
     (USER-GET LAUNCH OPEN-FILE-TPX FTP-LCD FTP-CD FTP-OPEN-CONNECTION
      FTP-CHMODE-BINARY FTP-DOWNLOAD-FILE-BIN APP-PRINT)
davon kennt Benutzer atomar:
     (USER-GET LAUNCH FTP-CD FTP-OPEN-CONNECTION APP-PRINT)
Benutzer kennt Dekompositionen fuer:
     kein nicht-atomares Plan-Concept des Plans
Erklaert werden muessen deshalb:
     (FTP-DOWNLOAD-FILE-BIN FTP-CHMODE-BINARY FTP-LCD OPEN-FILE-TPX)
```

In *Theos Vorstellung* stellt sich das Problem dagegen wesentlich einfacher dar. Er geht von weniger Plan-Concepts aus, die für das Erstellen eines Plan eingesetzt werden können:

```
(open-file-tpx app-print user-get launch ftp-open-connection
 ftp-close-connection ftp-cd ftp-download-file remote-copy shutdown-pc)
```

Darin ist auch das Plan-Concept *remote-copy* enthalten, mit dem sich (nach *Theos* Ansicht) Dateien beliebig über Computergrenzen hinweg kopieren lassen. Dementsprechend kann das Beispielziel (hier zunächst für die erste Ausgangssituation) durch den Plan (Th-B-e):

```
Step 1 : (LAUNCH MYTP)
Step 2 : (REMOTE-COPY FN-KN-XY-DOC obj-438 F-KN-XY-DOC PC-DRIVE-C obj-439
          STR-XY-DOC FCO-KN-XY-DOC SC-FILE-TYPE-TPX)
Step 3 : (OPEN-FILE-TPX obj-438 MYTP)
Step 4 : (APP-PRINT MYTP obj-438 DX FCO-KN-XY-DOC LOC-PR)
Step 5 : (USER-GET LOC-PR DX SC-USER)
```

erreicht werden. Der Plan gibt *Theos Ansicht* wieder, wie das Ziel zu erreichen ist. Er ist aber (nach Systemansicht) falsch, weil mit *remote-copy* ein Plan-Concept verwendet wird, das dem System nicht bekannt ist. Die unten stehende Beurteilung des Plans zeigt zudem, daß *Theo* noch Erklärungen zu dem Plan benötigen würde, weil er die Plan-Concepts *remote-copy* und *open-file-tpx* nicht atomar kennt und auch nicht weiß, wie sie ausgeführt werden können.

```
Plan-Concepts des Plans:
     (USER-GET LAUNCH OPEN-FILE-TPX REMOTE-COPY APP-PRINT)
davon kennt Benutzer atomar:
     (USER-GET LAUNCH APP-PRINT)
Benutzer kennt Dekompositionen fuer:
     kein nicht-atomares Plan-Concept des Plans
Erklaert werden muessen deshalb:
     (REMOTE-COPY OPEN-FILE-TPX)
```

Durch den Einsatz von *remote-copy* ist es für die Planerzeugung irrelevant, ob ein Verzeichnis des fremden Computers exportiert wird oder nicht. Für die zweite Ausgangssituation wird deshalb der gleiche Plan erzeugt wie für die erste.

5.3.2.6 Pläne für Sue

Sue ist eine Person, über die das System nur mangelhafte Kenntnisse hat. Dementsprechend wird sie lediglich den Stereotypes zugeordnet, zu denen jeder Benutzer gehört, solange nicht spezifischere Kenntnisse über den Benutzer vorliegen.

```
Partition SBUB has Supers (ALLSTB)
Partition SBUC has Supers (NONHANDICAPC ALLSTC)
```

Außerdem schreibt das System *Sue* die Fähigkeit zu, das Plan-Concept *download-file* ausführen zu können. Das Plan-Concept ist lediglich eine sehr allgemeine Beschreibung von Handlungen, mit denen eine Datei von einem fremden Recher ins lokale Dateisystem gelangt. Dementsprechend kennt das System keine Dekomposition von *download-file* und kann keine Erklärungen zu diesem Plan-Concept geben. Sein Einsatz bei der Plangenerierung macht also nur dann Sinn, wenn der Benutzer selbst atomare Kenntnis davon besitzt. *Sue* wird dieses Wissen vom System zugebilligt. Das Plan-Concept muß also sowohl in View SBUB als auch in SBUC eingetragen und außerdem als *atomar* markiert werden:

```
(bgp-ms-def-plan-concepts-from-struclist (list op-download-file)
                                         :partitions '(SBUB))
(bgp-ms-declare-concepts-atomic '(download-file) :partitions '(SBUB))
(bgp-ms-def-plan-concepts-from-struclist (list op-download-file)
                                         :partitions '(SBUC))
```

Für die Plangenerierung stehen damit die folgenden Plan-Concepts aus *Sues* Handlungspotential zur Verfügung:

```
(app-print user-get launch open-file-tpx download-file)
```

Da *download-file* für *Sue* die einzige Möglichkeit ist, eine fremde Datei in das lokale Dateisystem zu bekommen, und für dieses Plan-Concept irrelevant ist, ob auf dem fremden Computer ein Verzeichnis exportiert wird, werden für beide Ausgangssituationen die gleichen Pläne (S-C-e bzw. S-C-ne) erzeugt:

```
Step 1 : (LAUNCH MYTP)
Step 2 : (DOWNLOAD-FILE F-KN-XY-DOC obj-427 obj-426 PC-DRIVE-C
          FCO-KN-XY-DOC SC-FILE-TYPE-TPX)
Step 3 : (OPEN-FILE-TPX obj-426 MYTP)
Step 4 : (APP-PRINT MYTP FCO-KN-XY-DOC obj-426 DX LOC-PR)
Step 5 : (USER-GET LOC-PR DX SC-USER)
```

Erklärungen benötigt *Sue* lediglich für *open-file-tpx*, das sie weder atomar kennt noch weiß, wie das Plan-Concept ausgeführt werden kann:

```
Plan-Concepts des Plans:
     (USER-GET LAUNCH OPEN-FILE-TPX DOWNLOAD-FILE APP-PRINT)
davon kennt Benutzer atomar:
     (USER-GET LAUNCH DOWNLOAD-FILE APP-PRINT)
Benutzer kennt Dekompositionen fuer:
     kein nicht-atomares Plan-Concept des Plans
Erklaert werden muessen deshalb:
     (OPEN-FILE-TPX)
```

Bezogen auf die Plan-Concepts, die für die Plangenerierung verwendet werden können, besteht der Unterschied zwischen *Sues* Handlungspotential und ihren Annahmen lediglich darin, daß sie weiß, wie man einen PC herunterfahren kann:

```
(open-file-tpx app-print user-get launch download-file shutdown-pc)
```

Da diese Kenntnis für das Beispielziel ohne Belang ist, sind die Pläne, die anhand von *Sues* Wissen erzeugt wurden (S-B-e und S-B-ne), identisch zu den Plänen S-C-e bzw. S-C-ne anhand ihres Handlungspotentials.

Kapitel 6

Zusammenfassung und Ausblick

6.1 Zusammenfassung

Die Zielsetzung dieser Arbeit war die Integration von Planungswissen in den Rahmen logikbasierter Benutzermodellierung, um Pläne benutzerangepaßt erzeugen und präsentieren zu können, sodaß ein (Computer-)Benutzer damit seine Ziele erreichen kann. Der Nutzen der Adaption von Hinweisen und Ratschlägen an den jeweiligen Benutzer wird im Vergleich zu *nicht*-benutzeradaptiven Hilfesystemen deutlich. Da solche Systeme zumeist von "unerfahrenen, gelegentlichen" Benutzern ausgehen, sind die entsprechenden Ratschläge für erfahrenere Benutzer oft zu oberflächlich und damit inakzeptabel. Durch das Vordringen der Computertechnik in sehr viele Lebensbereiche sind heutzutage Benutzer, die in *allen* Gebieten der Computernutzung völlig unerfahren sind, eher selten anzutreffen. Da es andererseits für einen einzelnen Menschen kaum mehr möglich ist, sich in *allen* diesen Gebieten gut auszukennen, gewinnt die Anpassung von Ratschlägen an den jeweiligen Benutzer zunehmend an Bedeutung.

Im Verlauf der Arbeit wurde allerdings gezeigt, daß eine Anpassung eines Ratschlags an die *Kenntnisse* eines Benutzers nicht ausreicht, um sicherzustellen, daß dieser damit seine Ziele erreichen kann. Vielmehr müssen auch seine spezifischen *Möglichkeiten zu handeln* berücksichtigt werden, um ihn vor der Situation zu bewahren, daß er zwar *weiß wie* seine Ziele zu erreichen sind, aber die dazu notwendigen Handlungen nicht ausführen kann. Da die Kenntnisse und die Handlungsmöglichkeiten eines Benutzers keineswegs identisch sein müssen, war es für die Zielsetzung der Arbeit erforderlich, die beiden Aspekte im Benutzermodell strikt zu trennen. Für die Repräsentation beider Aspekte wurde aber dennoch eine einheitliche, beschreibungslogische Darstellung entwickelt, die sich in den Rahmen bisheriger begrifflicher Modellierung von Benutzer*kenntnissen* einfügt. Dadurch wurde erstmalig eine Darstellung von Planungswissen und Handlungsmöglichkeiten erreicht, die es ermöglicht mit dem Benutzer darüber in gleicher Weise zu kommunizieren, wie über objektbezogenes Wissen im Anwendungssachbereich, und die gleichzeitig ausdrucksstark genug ist, um das notwendige Wissen für die Generierung von Plänen darstellen zu können. Begriffliche Repräsentationen von Planungswissen, wie sie bspw. in den Planbibliotheken terminologischer Planerkennungssysteme[1] verwendet werden, erreichen die notwendige Ausdrucksstärke nicht. Durch die begriffliche Repräsentation von Planungswissen und Handlungsmöglichkeiten lassen sich außerdem die Ergebnisse früherer Arbeiten zur Akquisition von Wissen über den Benutzer direkt übertragen, wenn die Trennung der beiden Aspekte des Benutzermodells berücksichtigt wird. Exemplarisch wurde dies für Stereotypes als Quelle für a priori Wissen über Benutzer gezeigt.

[1] vgl. Kap. 2.2.3 *Planerkennung mit Beschreibungslogik*

Die Generierung eines Plans für das jeweilige Ziel des Benutzers ist immer dann von Vorteil, wenn die Vielzahl von potentiellen Zielen und Benutzereigenschaften nicht mehr sinnvoll durch Pläne einer statischen Planbibliothek abgedeckt werden kann. Es wurde gezeigt, daß dies in Anwendungsbereichen von Hilfesystemen praktisch immer der Fall ist. Darüber hinaus hat die Generierung eines Plans auch Vorteile für die benutzerspezifische *Präsentation* des Plans. Pläne einer Planbibliothek werden oft als Rezepte modelliert, die zwar die zeitliche Anordnung, aber keine Informationen über die Abhängigkeiten der einzelnen Schritte enthalten. Solche Abhängigkeiten, die bei der Generierung eines Plans ohnehin ermittelt werden, können bei der Präsentation eines Plans genutzt werden, um dem Benutzer zu erklären, welche Aufgaben einzelne Schritte im Plan haben. Auch stellen sie eine wichtige Information dar, wenn der Benutzer einen vom System präsentierten Plan noch weiterentwickeln oder modifizieren muß, weil der Plan noch nicht detailliert genug ist, oder wenn unerwartete Hindernisse der Planausführung entgegen stehen. Der Ansatz in dieser Arbeit ist damit klar im Vorteil gegenüber der Präsentation eines Rezeptes in konventionellen Hilfesystemen, bei denen der Benutzer nicht weiß, wozu die einzelnen Schritte erforderlich sind. Der Erfolg des Plans kann dort leicht durch ungeschickte Ausführung einzelner Schritte oder durch unerwartete Hindernisse verhindert werden, ohne daß der Benutzer eine Chance zur Korrektur hätte.

Andererseits ist es aber nicht notwendig dem Benutzer jede Abhängigkeit oder jedes Detail explizit zu präsentieren, wenn er die Information selbst leicht schließen kann. Vielmehr ist es sinnvoll, solche redundante Information zu unterdrücken, um den Umfang einer Präsentation möglichst gering zu halten. Im vierten Kapitel dieser Arbeit wurden Einflüsse auf die Entscheidungen untersucht, welche Komponenten eines Plans dem jeweiligen Benutzer explizit genannt werden und ob der Benutzer zusätzliche Detailinformation benötigt. Es wurde gezeigt, daß dies nicht auf die Frage beschränkt werden kann, ob der Benutzer ausreichende Kenntnisse hat, um anhand der Beschreibung *einen* korrekten Plan *ermitteln zu können*. Vielmehr wurde erstmals vorgeschlagen, das Benutzerwissen auch hinsichtlich *Nutzbarkeit* und *Verläßlichkeit* zu beurteilen. Eine solche Beurteilung wurde erst durch die im neu entwickelten Benutzermodell enthaltenen Annahmen über Handlungsmöglichkeiten und Planungswissen möglich. Ihre Berücksichtigung soll verhindern, daß der Benutzer seine Ziele *nicht* erreicht, weil die ihm bekannten Ausführungsmöglichkeiten jenseits seiner Handlungsmöglichkeiten liegen oder weil er eine falsche Ausführungsmöglichkeit wählt. Darüber hinaus wurde eine erste einfache Heuristik für eine Abwägung, ob die explizite Präsentation einer Plankomponente sinnvoll ist, entwickelt. Sie beruht auf einer Gegenüberstellung der Abschätzungen des Benutzeraufwands für das Lesen und Verstehen der Präsentation auf der einen und für die Rekonstruktion der unterdrückten Komponente auf der anderen Seite.

Insgesamt konnte das Ziel dieser Arbeit i.w. durch die folgenden innovativen Einzelbeiträge erreicht werden:

- das explizite Modell der Handlungsmöglichkeiten von Benutzern und dessen strikte Trennung vom Modell des Benutzerwissens

- die homogene beschreibungslogische Repräsentation von Planungswissen, insbesondere von Annahmen über Handlungsmöglichkeiten des Benutzers und über seine Annahmen über Veränderungsmöglichkeiten im Sachbereich

- die Strukturierung einer Benutzermodellierungswissensbasis, die ein Modell der Handlungsmöglichkeiten und des Wissens des jeweiligen Benutzers und der typischen Benutzergruppen im Anwendungsbereich repräsentiert

- die Erzeugung eines Planungssachbereichs aus dem Benutzermodell als Basis für die Erzeugung benutzerspezifischer Pläne durch ein anwendungsunabhängiges Planungssystem

- die Bestimmung des fehlenden Wissens, das ein Benutzer noch benötigt, um einen für ihn erzeugten Plan ausführen zu können, in Abhängigkeit vom Modell seines Wissens und seiner Handlungsmöglichkeiten

Ein weiterer Beitrag dieser Arbeit ist die Gegenüberstellung der Repräsentation und Verarbeitung von Plänen und der – teilw. sehr inhomogenen – Verwendung der zugehörigen Begrifflichkeit in verschiedenen KI-Forschungsbereichen.

Im Verlauf der Arbeit wurden auch bereits einige Möglichkeiten zur Weiterentwicklung des Ansatzes skizziert. Kap. 3.5.1 *Der Einfluß von Werkzeugen auf Handlungsmöglichkeiten* S. 160 zeigte bspw., wie die Idee des Handlungspotentials eines Benutzers auf Werkzeuge übertragen werden kann und so zum Verwendungspotential führt. Dieses kann sowohl für eine verbesserte Organisation des zu repräsentierten Wissens als auch für die Klassifikation von Objekten anhand ihrer *Verwendungsmöglichkeiten* genutzt werden. Auf weitere Chancen, die sich durch die explizite Modellierung von Handlungsmöglichkeiten in der Benutzermodellierungswissensbasis ergeben, wird noch im nachfolgenden Abschnitt eingegangen.

6.2 Ausblick

Naturgemäß ergeben sich aus der Einführung eines neuen Ansatzes eine Vielzahl von Ideen zu Anwendungs-, Erweiterungs- und Verbesserungsmöglichkeiten sowie zur Integration von Ideen und Ergebnissen aus anderen Forschungsgebieten. Dieser Abschnitt zeigt einige der Ideen auf, die i.w. zu den ersten Gruppen gehören, um einen Eindruck vom Entwicklungspotential des Ansatzes zu geben. Abschließend wird noch kurz darauf eingegangen, wie der Ansatz empirisch überprüft werden kann.

Als Ideen aus anderen Forschungsgebieten sei lediglich kurz auf die Arbeiten zur konkreten Ausgestaltung der Kommunikation zwischen Mensch und Computer hingewiesen. Je nach geplantem Einsatzbereich sind formale oder natürliche Sprachen in geschriebener oder gesprochener Form sowie Grafiken oder eine Kombination davon denkbar. Speziell für die Präsentation des Plans wurden einige Ansätze bis hin zu animierten Darstellungen zu Beginn von Kapitel 4 kurz angesprochen. Eine einfache Darstellung kann, wie bspw. in [Young, 1999] vorgeschlagen, auf der Basis von Textmustern realisiert werden. Die Benutzerziele die erreicht werden sollen können direkt vom Benutzer angegeben werden – bspw. als Aufforderung zur Hilfestellung – oder durch Beobachtung des Benutzers mithilfe von Plan- bzw. Zielerkennung gewonnen werden. Insbesondere im Rahmen von Fehlererkennung ist auch eine Kombination denkbar. Darüber hinaus erscheint es für eine Mensch-Computer-Schnittstelle wünschenswert, mit dem Benutzer über die Vorschläge des Systems verhandeln zu können, weil damit gerechnet werden muß, daß der Benutzer einen solchen Vorschlag nicht in jedem Fall akzeptiert. Gründe dafür können sowohl falsche Annahmen des Systems über die aktuellen Gegebenheiten im Anwendungsbereich oder über die Benutzereigenschaften einschließlich seiner Präferenzen sein als auch falsche Annahmen des Benutzers – einschließlich falscher Annahmen über seine eigenen Handlungsmöglichkeiten. Interessant ist dabei die Frage, welche Chancen sich durch ein Benutzermodell, das auch Annahmen über die Handlungsmöglichkeiten des Benutzers enthält, für bisher entwickelte Strategien zur Dialogführung über Pläne ergeben, wie sie bspw. in [Chu-Carroll und Carberry, 1998] bzw. generell im Bereich *Gemeinsames Planen*[2] beschrieben sind. Auf jeden Fall kann erwartet werden, daß die Annahmen selbst und auch die Strukturinformation, die bei der Erzeugung eines Plans anfällt, für die Begründung oder Rechtfertigung von Ratschlägen des Systems gegenüber dem Benutzer genutzt werden können.

[2] vgl. a. Kap. 2.3 *Gemeinsames Planen (mit einem Benutzer)*

6.2.1 Weitergehende Bewertung von Plänen

In Kapitel 4.3 *Inhaltsbestimmung und Bewertung einer Planpräsentation* wurden Plankomponenten bewertet, um eine Entscheidung darüber treffen zu können, welche Komponenten explizit in einer Planpräsentation enthalten sein sollen. Im Abschnitt 4.3.5 *Alternative Modelle zur Planbewertung* wurden darauf aufbauend einige Möglichkeiten zur Bewertung eines Gesamtplans angesprochen. Als Beispielkriterien wurden Benutzerpräferenzen und die Wahrscheinlichkeit, mit der der jeweilige Benutzer den Plan erfolgreich ausführt, genannt. Darüber hinaus sind mit der Ausführung von Handlungen generell Aufwände bzw. Kosten verbunden, die möglichst gering gehalten werden sollten. Im Beispielsachbereich dieser Arbeit können dies u.a. Kosten für die Beschaffung von Information (die dafür verbrauchte Arbeitszeit des Benutzers und/oder Benutzungsgebühren eines Auskunftssystems), Kosten für die Benutzung des Computers oder anderer Geräte oder Kosten für den Datentransfer über Netzwerke sein. Oft muß ein Benutzer nicht notwendigerweise alle Schritte eines Plans selbst ausführen, sondern er kann andere Akteure um Hilfe bitten. Diese Akteure würden dann den Aufwand für die Ausführung der Schritte übernehmen. Hinzu käme der Aufwand für die Beauftragung. Auch ein Akteurwechsel bei der Planausführung ist mit Aufwand verbunden. Für eine Gesamtbewertung des Plans sind verschiedene Modelle denkbar. Zum einen können die Aufwände für die Ausführung einzelner Schritte nicht nur von den Schritten, sondern auch vom jeweiligen Akteur abhängen – eine Handlung, die für einen Akteur ungewohnt und schwierig ist, kann für einen anderen tägliche Routinearbeit sein. Das Benutzermodell muß dafür zum Akteurmodell erweitert werden und auch die Aufwände für die Ausführung von Plan-Concepts enthalten. Die Bewertung kann zudem aus der Sicht eines einzelnen Akteurs oder als Gesamtsicht berechnet werden. Im ersten Fall würden möglichst viele Schritte an andere Akteure delegiert. Im zweiten nur dann, wenn der Aufwand für einen anderen Akteur um so viel geringer ist als der für den Benutzer, daß sich der Aufwand für Beauftragung und Akteurwechsel lohnt. In der Realität ist der Aufwand, den *ein* Akteur für die Ausführung eines Plans treiben kann, begrenzt. Werden solche Kapazitätsobergrenzen – ggf. akteurspezifisch – berücksichtigt, kann sich daraus die *Notwendigkeit* ergeben, einen Gesamtplan von verschiedenen Akteuren gemeinsam ausführen zu lassen.

6.2.2 “Bessere” Plan-Beschreibungen

6.2.2.1 Abstraktere Plan-Beschreibungen

Kriterien für gute Plan-Beschreibungen sind vor allem die Verständlichkeit der Beschreibung für den Benutzer sowie die Wahrscheinlichkeit für die korrekte Planausführung aufgrund der Beschreibung. Die in Kapitel 4.3.4 *Arbeiten zum Verstehen von Plänen und Anweisungen aus der Psychologie* angesprochenen Arbeiten haben bereits gezeigt, daß es kein einfaches Verfahren gibt, diese Kriterien für eine Beschreibung quantitativ zu fassen. Allerdings kann für eine grobe Abschätzung davon ausgegangen werden, daß der Aufwand für das Verstehen einer Beschreibung normalerweise mit dem (expliziten) Informationsgehalt steigt; d.h. die Verständlichkeit nimmt dementsprechend ab. Möglichkeiten die explizit präsentierte Informationsmenge einer Planbeschreibung unter Ausnutzung der (angenommen) Inferenzfähigkeiten des Benutzers zu reduzieren wurden in Kap. 4.4 *Berücksichtigung von Benutzer-Inferenzen* behandelt. Weitere Möglichkeiten zur Reduzierung der Informationsmenge ergeben sich durch abstraktere Beschreibungen. Da abstrakte Beschreibungen auf mehrere konkrete Pläne zutreffen können, muß geprüft werden, ob der jeweilige Benutzer aufgrund seines Wissens aus der abstrakten Beschreibung ausschließlich *korrekte* Konkretisierungen ableiten kann.

Eine Möglichkeit zur Abstraktion von Planbeschreibungen ist die Verwendung von abstrakteren Plan-Concepts. Im Beispielplan aus Abb. 4.1 S. 174 könnte bspw. anstelle des Plan-Concepts *ftp-dnload-bin* (FTP Download im Binärmodus) das abstraktere Plan-Concept *ftp-dnload* angegeben

werden, für das der Transfermodus nicht festgelegt ist. Die Informationsmenge wird durch die Verwendung des abstrakteren Plan-Concepts zwar nicht signifikant geringer, die abstrakte Beschreibung erscheint aber dennoch besser, weil die Spezialisierung durch den vorangegangenen Schritt (Einschalten des Binärmodus) redundant ist. Damit müssen allerdings die Kriterien für einen kohärenten Text überdacht werden, denn der Schritt *Einschalten des Binärmodus* wird ja nur für den Dateitransfer im Binärmodus benötigt, der in der Beschreibung nicht mehr explizit vorkommt. Außerdem muß das in Kap. 4.4 vorgestellte Verfahren für das Entfernen von Komponenten aus einer Planpräsentation berücksichtigen, daß der Benutzer die Notwendigkeit zur Rekonstruktion von Komponenten teilweise erst dann erkennen kann, wenn er explizit genannte Komponenten geeignet spezialisiert. Wird also der Schritt *Einschalten des Binärmodus* aus der Planpräsentation entfernt, kann der Benutzer die Notwendigkeit für das Rekonstruieren erst dann feststellen wenn er weiß, daß er die Datei im Binärmodus herunterladen muß – auch wenn dies nicht explizit in der Planbeschreibung enthalten ist. Der Rekonstruktionsaufwand wird damit größer, und der Benutzer muß die Kenntnisse über die Spezialisierung des abstrakten Plan-Concepts haben.

Anstelle der Abstraktion eines einzelnen Plan-Concepts, können auch mehrere Planschritte zu einem Makro-Schritt zusammengefaßt werden – in Umkehrung des Vorgehens beim Planen mit Operatorabstraktion[3]. Dieses Vorgehen ist besonders dann vorteilhaft, wenn der Benutzer ein (abstraktes) Plan-Concept kennt, das (nach seiner Ansicht) in die Folge der zusammengefaßten Schritte dekomponiert werden kann. Im Idealfall braucht die Planbeschreibung dann nur ein Exemplar des abstrakten Plan-Concepts – anstelle mehrerer einzelner Schritte – zu enthalten, was zu einer merklichen Reduzierung der explizit präsentierten Informationsmenge führt. Die Feststellung aus Kap. 4.2.1 *Dekomposition eines Plan-Concepts*, daß nur unvollständige Plan-Concepts für Schritte von Dekompositionen verwendet werden, wird damit natürlich obsolet. Bedingung für die Zusammenfassung von Planschritten zu einem Makro-Schritt ist, daß der Benutzer nur solche Dekompositionen kennt, die zu einem korrekten Gesamtplan führen. Deren Überprüfung ist vergleichbar zur Feststellung, ob ein Benutzer *verläßliches Wissen über die Ausführungsmöglichkeiten eines Plan-Concepts* hat.

Die Kenntnis von abstrakten Plan-Concepts und Dekompositionen wie hier vorgeschlagen, kann dem Benutzer auch bei der Rekonstruktion von Planschritten helfen, die vom Präsentationsalgorithmus aus der Planbeschreibung entfernt wurden. Enthält eine Planbeschreibung einen Schritt oder eine Folge von Schritten, die der Benutzer als Schritt(e) einer Dekomposition erkennt, kann er leicht evtl. fehlende Schritte dieser Dekomposition ergänzen. Kennt der Benutzer bspw. ein abstraktes Plan-Concept *Binär-Datei-mit-FTP-herunterladen* und dazu eine Dekomposition, die alle zugehörigen Schritte enthält, kann er aus dem Auftreten eines Planschritts *ftp-dnload-bin* schließen, daß der Schritt möglicherweise zu dem abstrakten Plan-Concept gehört und daß in diesem Falle für den Plan noch Schritte wie *Einschalten des Binärmodus* und *Öffnen der FTP-Verbindung* erforderlich sind[4]. Die Reduzierung des Rekonstruktionsaufwandes ist dabei um so deutlicher, je weniger Dekompositionen, die den Schritt oder die Folge enthalten, dem Benutzer bekannt sind. Ansonsten kann der Vorteil vom Aufwand für die Auswahl der "richtigen" Dekomposition geschmälert oder aufgezehrt werden.

6.2.2.2 Annahmen eines Benutzers über seine eigenen Handlungsmöglichkeiten

Wenngleich die Schwierigkeit ein zuverlässiges Benutzermodell aufzubauen, mit der Schachtelungstiefe der berücksichtigten Annahmen anwächst[5], seien hier zwei Möglichkeiten aufgezeigt, wie Annahmen des Benutzers über seine eigenen Handlungsmöglichkeiten zur Verbesserung einer Planbeschreibung genutzt werden können. Das in Kap. 4 behandelte Verfahren zur Präsentation eines Plans

[3] vgl. Kap. 2.1.2.2 *Operatorabstraktion*
[4] vgl. bspw. Abb. 5.11 S. 272
[5] vgl. Kap. 3.3.4 *Geschachtelte Annahmen über Handlungsmöglichkeiten*

garantiert zwar (sofern die Annahmen über den Benutzer korrekt sind), daß der Benutzer den Plan ausführen kann und auch das notwendige Wissen darüber hat, wie die einzelnen Schritte ausgeführt oder ggf. in Subschritte dekomponiert werden können. Glaubt der Benutzer aber nicht, daß er die Schritte des Plans ausführen kann, wird er möglicherweise dennoch den Rat des Systems in Zweifel ziehen und einen Klärungsdialog initiieren, wodurch der Gesamtaufwand für die Kommunikation erhöht wird. Noch ungünstiger ist, wenn der Benutzer einzelne Schritte des Plans für die Ausführung noch dekomponieren muß und er nur solche Dekompositionen kennt, die auch Schritte enthalten, von denen er nicht glaubt, daß er sie ausführen kann. Der Benutzer kann solche Dekomposition nicht verwenden, weil er sie nach seiner Ansicht nicht erfolgreich ausführen kann. Damit fehlt ihm also – nach seiner Meinung und entgegen der Ansicht des Systems – das notwendige Wissen *wie* die Schritte des Plans ausgeführt werden können. Eine solche Planbeschreibung reicht also *nicht* aus, um den Benutzer in die Lage zu versetzen seine Ziele zu erreichen. In beiden Fällen ist es sinnvoll, den Benutzer über die entsprechenden Fehlannahmen aufzuklären. Allerdings muß sorgfältig zwischen Risiken und Nutzen abgewogen werden, weil die Akzeptanz eines Systems, das den Benutzer häufig korrigiert, leiden dürfte. Dies gilt insbesondere, wenn die Annahmen des Systems über Fehlannahmen des Benutzers nicht sicher sind und Belehrungen durch das System dementsprechend selbst falsch sein können.

Fehlannahmen eines Benutzers werden in dieser Arbeit nicht explizit als "falsche Annahmen" repräsentiert. Vielmehr werden Benutzerannahmen vom System als falsch eingestuft, wenn sie mit den Systemannahmen nicht konform sind[6]. Um abzuschätzen, ob ein Benutzer glaubt, daß er einen Plan ausführen kann, müssen dementsprechend die Systemannahmen über die Handlungsmöglichkeiten des Benutzers (im Modell durch SBUC repräsentiert) mit den Systemannahmen über die Benutzerannahmen über seine eigenen Handlungsmöglichkeiten verglichen werden. Nach der Vorgehensweise aus Kap. 3.3.4 *Geschachtelte Annahmen über Handlungsmöglichkeiten* gehören solche Annahmen zu einem eigenen Assumption-Type[7] mit Kennung SBUBUC. In der Benutzermodellierungswissensbasis wird dafür eine Partition SBUBUC, als Sub-Partition von Partition $\mathrm{SBU}^{Dom}\mathrm{B}$ definiert. Gehen wir davon aus, daß der Benutzer einen Plan nur dann für ausführbar hält, wenn er glaubt, daß er *jeden* einzelnen Schritt ausführen kann, muß das System für jeden Schritt des Plans und der ggf. erforderlichen Dekompositionen prüfen, ob er nach Ansicht des Benutzers für ihn ausführbar ist. Die Prüfung kann durch den Test realisiert werden, ob SBUBUC ein Plan-Concept *pc* mit der gleichen Bezeichnung[8] enthält, wie das Plan-Concept aus dem der Planschritt erzeugt wurde und ob der Planschritt auch von *pc* subsumiert wird. Der zweite Teil der Bedingung stellt sicher, daß auch eventuelle Einschränkungen des Plan-Concepts berücksichtigt werden, die lediglich nach Annahme des Benutzers bestehen. Bspw. könnte ein Benutzer annehmen, daß er nur mit Schwarz-Weiß-Druckern drucken darf, während das System dem Benutzer zuspricht, daß er mit beliebigen Druckern drucken kann und darf. Enthält ein Plan dann einen Planschritt *drucken*, der als Drucker einen Farb-Drucker einsetzt, muß der Benutzer annehmen, daß er den Plan *nicht* ausführen kann. Darüber hinaus kann es sinnvoll sein, auch Annahmen des Benutzers über Handlungsmöglichkeiten die er *nicht* hat explizit zu repräsentieren. Pohls AsTRa-Formalismus[9] und die Realisierung in BGP-MS sehen ausdrücklich auch *negative Assumption-Types* vor. Annahmen des Benutzers über Handlungsmöglichkeiten die er *nicht* hat, können dann durch den Assumption-Type mit Kennung SBUB~UC repräsentiert werden.

Annahmen des Benutzers über seine eigenen Handlungsmöglichkeiten können auch dazu genutzt werden, die Zahl der Ausführungsalternativen bei der Planpräsentation zu reduzieren. In Kap. 4.2.2.3 *Verläßliches Wissen über die Ausführungsmöglichkeiten von Plan-Concepts* wurde einem Benutzer

[6] vgl. Kap. 3.1 *Handlungspotential und Wissen von Benutzern*
[7] vgl. Kap. 3.3.1 *Akteurmodelle*
[8] vgl. Kap. 3.3.2 *Repräsentation eines Benutzermodells mit Handlungspotential und Wissen im AsTRa-Formalismus*
[9] vgl. [Pohl, 1997] und Kap. 3.3.1 *Akteurmodelle*; für *negative Assumption-Types* insb. Kap. 5.4 in Pohls Arbeit.

verläßliches Wissen über ein Plan-Concept zugeschrieben, wenn er nur korrekte Dekompositionen kennt. Es wurde argumentiert, daß dem Benutzer keine weiteren Erklärungen zu einem solchen Plan-Concept gegeben werden müssen, weil er keine Möglichkeit kennt, einen dazu gehörenden Planschritt falsch auszuführen. Darüber hinaus kann aber auch erwartet werden, daß ein Benutzer keine Ausführungsvarianten wählt die Schritte enthalten, von denen er *nicht* glaubt, daß er sie ausführen kann – unabhängig von der Ansicht des Systems dazu. Deshalb kann das Wissen eines Benutzers über ein Plan-Concept auch dann – in diesem Sinne – als *verläßlich* eingestuft werden, wenn der Benutzer zwar falsche Dekompositionen des Plan-Concepts kennt, diese aber jeweils zumindest ein Plan-Concept enthalten von denen der Benutzer nicht glaubt, daß er es ausführen kann. Auf zusätzliche Erklärungen, die lediglich den Sinn haben den Benutzer vor einer falschen Ausführungsvariante zu bewahren, kann dann verzichtet werden.

6.2.3 Verallgemeinerung benutzerspezifischer Plangenerierung und -präsentation

In diesem Abschnitt werden zwei Möglichkeiten aufgezeigt, die benutzerspezifische Plangenerierung und -präsentation zu erweitern, um ihr ein größeres Einsatzfeld zu erschließen. Das Benutzermodell mit Annahmen über Wissen und Handlungsmöglichkeiten des Benutzer bleibt dabei unverändert.

6.2.3.1 Abstrakte Benutzerziele – unvollständiges Systemwissen

In dieser Arbeit wurde stets davon ausgegangen, daß ein konkreter Plan für die konkreten Ziele des Benutzers erzeugt werden soll. Ein erweiterter Anwendungsbereich ergibt sich, wenn das System darüber hinaus auch Fragen nach allgemeinen Lösungsmöglichkeiten beantworten kann. Eine Verallgemeinerung der in der Einleitung genannten Beispielfrage wäre: "*Wie kann ich einen Ausdruck von einer Textverarbeitungs-Datei bekommen, die sich auf einem fremden Computer befindet?*". Die Schwierigkeit der Beantwortung solcher Fragen besteht zum einen darin, geeignete Annahmen zu machen, die die Generierung eines Plans erst möglich machen – bspw. über den Speicherort der Datei und die Zugriffsmöglichkeiten auf den fremden Rechner. Außerdem müssen die Annahmen protokolliert und bei der Präsentation des Ergebnisses – i.a. abstrahiert – genannt werden. Ergeben sich durch die Annahmen mehrere alternative Lösungswege, sollten dem Benutzer möglichst alle Lösungen angeboten werden. Sind dies mehr als dem Benutzer zugemutet werden können, muß eine sinnvolle Auswahl getroffen werden. Im günstigen Fall können die Lösungen zu einer Gesamtlösung abstrahiert werden. Diese kann dem Benutzer dann zusammen mit den Annahmen und daraus resultierenden Spezialisierungen präsentiert werden. Beispielsweise könnte eine Antwort auf die Frage lauten: "*Starten Sie das Textverarbeitungsprogramm, bringen Sie die Datei ins lokale Dateisystem, öffnen Sie die Datei mit dem Textverarbeitungsprogramm, drucken Sie sie damit aus und holen Sie den Ausdruck am Drucker ab. Wenn die Datei über FTP zugänglich ist, können Sie sie mit FTP ins lokale Dateisystem kopieren. Wenn sie sich (ggf. indirekt) in einem freigegebenen Verzeichnis befindet, kann sie durch Mounten des freigegebenen Verzeichnisses ins lokale Dateisystem gebracht werden.*" Die Präsentation von Details der Lösungsalternativen sollte aus Übersichtlichkeitsgründen generell nur auf Anforderung des Benutzers erfolgen. Alternativ ist auch denkbar, die "*Auskunftsfreudigkeit*" des Systems durch einen Parameter einzustellen.

Die gleiche Vorgehensweise kann angewendet werden, wenn dem System nicht alle für die Erzeugung eines Plans erforderlichen Fakten zur Verfügung stehen. D.h., daß außer der direkten Nachfrage beim Benutzer auch Annahmen getroffen und ggf. alternative Lösungsmöglichkeiten präsentiert werden können. Wichtig ist, daß dem Benutzer durch die Präsentation klar gemacht wird, daß und welche Annahmen getroffen wurden, damit er sie überprüfen und ggf. auch zurückweisen kann.

Auch bei der Ermittlung und Präsentation von verallgemeinerten Lösungsmöglichkeiten spielen die Benutzereigenschaften und insbesondere Handlungsmöglichkeiten des Benutzers eine Rolle. Allerdings kann es dabei auch sinnvoll sein, die Handlungsmöglichkeiten anderer Akteure mit einzubeziehen – insbesondere wenn ansonsten keine Lösung gefunden werden kann. Gehen wir von den Stereotypes in Abb. 3.18 S. 132 und einem Benutzer aus der Gruppe der *studentischen Hilfskräfte* aus, der (sofern er keine besonderen Rechte hat) keine fremden Verzeichnisse mounten darf, kann dieser höchstens die erste der beiden Varianten des o.g. Beispiels einsetzen. Ist die Datei nicht über FTP zugänglich, wäre für ihn der Hinweis nützlich, daß zwar er selbst das Ziel nicht erreichen kann (und ggf. warum), daß aber Mitglieder der Gruppe der *Angestellten* eine andere Möglichkeit haben, das Ziel zu erreichen. Er kann dann einen solchen Mitarbeiter um Hilfe fragen oder er weiß zumindest – ggf. als Rechtfertigung gegenüber seinem Auftraggeber – daß und warum er das Ziel *nicht* erreichen kann.

6.2.3.2 Lernbereitschaft des Benutzers – Adaptierbarkeit des Systems

In der Arbeit wurde mehrfach argumentiert, daß es sinnvoll sein kann, dem Benutzer zusätzliche Informationen zu geben, die er (im Idealfall) nicht unbedingt für das Erreichen der Ziele benötigt. Ein Beispiel waren die Abhängigkeiten zwischen Planschritten, die – wenn keine Plan-Verfeinerungen oder -Modifikationen vorkommen – für die Planausführung selbst nicht notwendig sind. Generell sollen solche zusätzlichen Informationen Zusammenhänge vermitteln, die der Benutzer für eigene Schlußfolgerungen nutzen kann. Werden – wie im vorangegangenen Abschnitt vorgeschlagen – abstrakte Fragen beantwortet, sind solche Schlußfolgerungen immer notwendig. In anderen Fällen kommt der Benutzer im günstigsten Fall zwar ohne sie aus. Die zu Beginn von Kap. 4.1 *Planstruktur* angegebenen Untersuchungen[10] zeigen allerdings, daß Benutzer normalerweise wissen wollen, *wozu* welcher Schritt im Plan ausgeführt werden muß. Die "Stärke" dieses Wunsches dürfte aber sowohl von der aktuellen Situation (hat der Benutzer die Zeit, Zusatzinformationen zu verarbeiten oder muß der dringend seine Ziele erreichen), als auch von der persönlichen *Lernbereitschaft* des Benutzers abhängen. Da solche Parameter nur schwierig zu ermitteln sind[11], bietet es sich an, das System einstellbar zu machen. Dabei ist eine Differenzierung nach Situation und Benutzereigenschaften denkbar. Für letztere macht es Sinn, sie in einem langfristigen Benutzermodell zu speichern.

Auch für Entscheidungen über Ausführungsalternativen beim Präsentationsprozeß (Kap. 4.3 *Inhaltsbestimmung und Bewertung einer Planpräsentation*) und für Planungsentscheidungen bei der in Kap. 4.5 vorgestellten optimierten Plangenerierung ist die Lernbereitschaft des Benutzers und die aktuelle Situation der Hilfeanforderung relevant. Der Präsentationsprozeß wurde dahingehend optimiert, daß möglichst wenig Erklärungen gegeben werden müssen. Bei der optimierten Plangenerierung konnten sogar (etwas) umständlichere Pläne bevorzugt werden, wenn dadurch Erklärungen vermieden werden. Ist zu erwarten, daß der Benutzer ähnliche Aufgaben in Zukunft wieder ausführen will, kann es dennoch sinnvoll sein, ihm eine bessere Lösung anzubieten, auch wenn dies in der aktuellen Situation zu einer längeren Erklärung führt.

Zusammen mit der im vorangegangenen Abschnitt behandelten Beantwortung abstrakter Fragen ist denkbar, daß für das System ein *tutorieller Modus* vorgesehen wird. In diesem Modus würde der Schwerpunkt von der Lösung des konkreten Problems für den jeweiligen Benutzer zu einer Unterstützung des Benutzers beim Erlernen der Zusammenhänge im Anwendungsbereich und der Abschätzung seiner Möglichkeiten, Ziele im Anwendungsbereich zu erreichen, verlagert. Der in dieser Arbeit vorgestellte Ansatz führt aber dennoch nicht (unmittelbar) zu einem Tutorsystem[12], das den Benutzer umfassend über ein Themengebiet informiert. Dafür fehlt die Möglichkeit didaktisches Wissen ein-

[10] [Gilbert und Jirotka, 1990], [Kidd, 1985]
[11] vgl. Kap. 4.3.2 *Gesamtbewertung einer Planpräsentation*
[12] vgl. bspw. [McCalla, 2000] für eine Übersicht und Perspektive des Forschungsbereichs

zubringen und ein zu vermittelndes Themengebiet insgesamt geeignet zu strukturieren. Es ist allerdings zu erwarten, daß die Erkenntnisse aus diesem Forschungsbereich für die Entscheidungen, wann welche zusätzlichen Informationen oder Korrekturen gegeben werden sollten, wertvoll ist.

6.2.4 Weitere Einsatz- und Anwendungsmöglichkeiten

6.2.4.1 Weitere Anwendungsbereiche

Der Beispiel-Anwendungssachbereich einer computerisierten Büroumgebung wurde für diese Arbeit gewählt, weil hierbei die Problemstellung durch die anwendungsübergreifenden Aufgaben und Benutzern mit unterschiedlichen Kenntnissen und Zuständigkeiten besonders deutlich wird. Zudem kann von Menschen, die ohnehin Computer einsetzen, die Bereitschaft erwartet werden, eine Unterstützung durch ein solches Hilfesystem auch zu akzeptieren.

Die steigende Verwendung von Computern in nahezu allen Lebensbereichen zeigt zudem, daß Computernutzung keineswegs auf die Arbeitswelt beschränkt ist. Vielmehr werden Computer – zumindest in den hoch technisierten Ländern – zunehmend für verschiedenste Zwecke von beinahe allen Menschen eingesetzt. Es muß also verstärkt damit gerechnet werden, daß Computerbenutzer sehr unterschiedliche Eigenschaften und Begriffswelten haben. Je nach Anwendung können die Benutzer auch stark vom Klischee eines typischen Computerbenutzers abweichen, bspw. Kinder (unterschiedlichen Alters) und Menschen mit – altersbedingten oder sonstigen – Behinderungen. Es ist sicherlich nicht möglich für all diese Benutzer eine adäquate Unterstützung anzubieten, ohne auf ihre speziellen Eigenschaften oder auch Einschränkungen einzugehen.

Als Beispiel für solche Anwendungen sei ein Computer im privaten Bereich genannt, auf dem ein Kind seine Hausaufgaben und Sprachübungen für die Schule machen darf. Die Eltern werden dem Kind (wahrscheinlich) selbst wenn es sehr gute Computerkenntnisse hat (oder vielleicht gerade deswegen) nicht den uneingeschränkten Zugriff auf alle Funktionalitäten erlauben. Eine Unterstützung, die dieser Einschränkung nicht Rechnung trägt, kann zu undurchführbaren Ratschlägen führen, die damit wertlos sind.

Benutzerspezifische Plangenerierung und -präsentation ist aber keineswegs auf die Arbeit mit einem Computer beschränkt, sondern läßt sich generell für die Erzeugung von Handlungsanweisungen einsetzen. Der damit verbundene Vorteil gegenüber vorgefertigten Rezepten für bekannte Aufgabenstellungen ist besonders stark, wenn die Variabilität möglicher Aufgabenstellung, die Kombination der Handlungsmöglichkeiten im Anwendungssachbereich und die Unterschiedlichkeit der Benutzereigenschaften eine große Zahl potentieller Lösungen implizieren.

6.2.4.2 Einsatz in der Planerkennung

Zu Beginn dieser Arbeit wurde festgestellt, daß die in Hilfesystemen und der Planerkennung übliche Repräsentation von Planungswissen in (weitgehend) statischen Planbibliotheken für die Ziele dieser Arbeit nicht ausreichend sind. Die Anwendung der vorgeschlagenen Repräsentation von Planungswissen in einer Benutzermodellierungswissensbasis wurde in den vorangegangenen Kapiteln ausführlich für Hilfesysteme behandelt. Hier sollen nun kurz einige Möglichkeiten für die Nutzung des Ansatzes für die Planerkennung angesprochen werden.

Es ist sicher unrealistisch *alle* Pläne, die der jeweilige Benutzer aufgrund seiner Kenntnisse erzeugen *könnte*, durch einen Plangenerator erzeugen zu lassen und dann als Referenz für die Planerkennung in einer Planbibliothek zu speichern. Hier kann der Ansatz also bestenfalls zur Ergänzung der Planbibliothek beitragen, wie dies bspw. auch im PHI-Projekt[13] vorgeschlagen wurde.

[13] [Bauer *et al.*, 1993]; s.a. Kap. 2.5.3 *Erzeugen von Plänen*

Ein schwieriges Problem der Planerkennung besteht darin, daß beobachtete Aktionen oft zu vielen Plänen passen und deshalb eine eindeutige Erkennung des Plans und damit der Ziele des Benutzers erst nach vielen Beobachtungen möglich ist. Einige Arbeiten (bspw. [Carberry, 1990b], [Bauer, 1996]) berücksichtigen die unterschiedlichen Wahrscheinlichkeiten, mit denen Benutzer bestimmte Ziele (ggf. auf bestimmten Wegen) anstreben, um die Zahl der sinnvollen Hypothesen frühzeitig einzuschränken. Die Wahrscheinlichkeiten können durch Wissen (des Bibliotheksentwicklers) über den Anwendungsbereich oder Beobachtung von Benutzergruppen bzw. des einzelnen Benutzers gewonnen werden.

Durch die in dieser Arbeit vorgeschlagene Repräsentation von Planungswissen in der Benutzermodellierungswissensbasis kann die Zahl der Erkennungsalternativen und die Planbibliothek selbst *benutzerspezifisch* weiter eingeschränkt werden. Ein erstes Beispiel für die Reduzierung von Erkennungsalternativen wurde in Abb. 2.4 S. 35 in Kap. 2.2.2 *Kautz' Theorie der Planerkennung* gezeigt. Dabei wurde von einer Planbibliothek ausgegangen, die nicht jeden Plan explizit, sondern in einer Dekompositionshierarchie repräsentiert. Da solche Darstellungen i.a. mehr als nur die korrekten Pläne implizieren, konnte die Zahl der Erkennungsalternativen reduziert werden, indem inkorrekte Alternativen identifiziert und verworfen wurden.

Eine weitere Reduzierung von Erkennungshypothesen ist i.a. unter der Annahme möglich, daß ein Benutzer nicht beliebige Pläne im Anwendungsbereich ausführen wird, sondern nur solche, die er aufgrund seines Wissens auch erstellen kann. Grundlage dafür sind die Annahmen des Benutzers über seine eigenen Handlungsmöglichkeiten (SBUBUC) oder, wenn diese nicht explizit im Benutzermodell repräsentiert sind, ersatzweise seine allgemeinen Annahmen über Handlungsmöglichkeiten (SBUB). Je nach vertretbarem Berechnungsaufwand können aus der Planbibliothek alle Pläne eliminiert werden, die sich anhand des Benutzerwissens nicht verifizieren lassen. Alternativ können auch einfach die Pläne eliminiert werden, die Planschritte enthalten, von denen der Benutzer nicht glaubt, daß er sie ausführen kann. Die erste – aufwendigere – Methode berücksichtigt auch unvollständige oder falsche Annahmen des Benutzers über Vorbedingungen und Effekte von Plan-Concepts.

Beide Methoden können unabhängig davon eingesetzt werden, ob die Planbibliothek unterschiedliche Auftretenswahrscheinlichkeiten für die möglichen Pläne repräsentiert. Die Wahrscheinlichkeiten werden allerdings durch das unterschiedliche Planungswissen des jeweiligen Benutzers beeinflußt, wenn er nicht alle Pläne der Planbibliothek erzeugen kann. Als Beispiel sei von Plänen p_1, p_2 die beide Ziel z_1 erreichen und p_3 der Ziel z_2 erreicht ausgegangen. Dabei sei p_1 ein Plan, der selten eingesetzt wird um z_1 zu erreichen und p_1 und p_3 mögen mit gleichen Schritten ($s_a, s_b, \ldots$) beginnen. Die Beobachtung von Schritt s_a würde also darauf hindeuten, daß wahrscheinlich p_3 ausgeführt wird, um z_2 zu erreichen, weil der ebenfalls mögliche Plan p_1 selten verwendet wird[14]. Kennt der Benutzer dagegen nicht alle für p_2 notwendigen Schritte, oder glaubt er nicht, daß er sie alle ausführen kann, bleibt ihm p_1 als einzige Möglichkeit z_1 zu erreichen. Dementsprechend muß die a priori Wahrscheinlichkeit für das Ausführen von p_1 für diesen Benutzer erhöht werden, wenn die Wahrscheinlichkeit für das Ziel z_1 unverändert angenommen wird. Damit ist die Wahrscheinlichkeit, daß der Benutzer Ziel z_1 anstrebt wenn Schritt s_a beobachtet wird, größer verglichen mit dem Fall, in dem der Benutzer glaubt, auch p_2 ausführen zu können. Gleichzeitig sinkt die Wahrscheinlichkeit, daß der Benutzer p_3 ausführt, um z_2 zu erreichen.

Der wesentliche Aspekt besteht darin, daß die Planbibliothek (auch) aufgrund der Annahmen über das Planungswissen des Benutzers an den Benutzer adaptiert werden kann, ohne daß es dazu notwendig wäre, die Häufigkeit der Ausführung der einzelnen Pläne durch den Benutzer beobachtet zu haben.

[14] vgl. [Bauer, 1996] für eine präzisere und ausführliche Darstellung eines solchen Planerkennungsprozesses

6.2.4.3 Einsatz in Dialogsystemen

Eng verwandt mit der Planerkennung ist die Aufgabe von Dialogsystemen, eine Äußerung des Benutzers zu interpretieren, d.h. seine Intention bzw. seine Ziele zu erkennen. In Kap. 3.3.4 *Geschachtelte Annahmen über Handlungsmöglichkeiten* wurde bereits darauf hingewiesen, daß die Zahl der Interpretationsmöglichkeiten oft durch den sogenannten Dialograhmen reduziert wird, indem Interpretationen, die nicht in die aktuelle Situation passen, von vorn herein ausgeschlossen werden. Klassisches Beispiel sind Fragen an einen Bahnbediensteten in einer Bahnhofs-Schalterhalle, die als Fragen zum Zugreiseverkehr interpretiert werden. Die Gegebenheiten der Umgebung in der ein Dialog stattfindet *an sich* sind aber nur eine Komponente, die Einfluß auf den Dialog und damit auf die Interpretation einzelner Äußerungen haben können. Wichtig sind auch die Vorstellungen, die ein Sprecher über seinen Dialogpartner hat. Wird von einem zielgerichteten Dialog ausgegangen, kann erwartet werden, daß der Sprecher nicht nur Begriffe verwendet, von denen er annimmt, daß sein Dialogpartner sie versteht, sondern er muß auch berücksichtigen, daß sein Dialogpartner die mit einer Äußerung verbundenen Handlungen ausführen kann. Für ein Dialogsystem heißt dies, daß die Benutzerannahmen über Kenntnisse und Handlungsmöglichkeiten des Systems für die Interpretation einer Benutzeräußerung berücksichtigt werden müssen. Gegenüber einem statischen Dialograhmen können so die unterschiedlichen Ansichten der verschiedenen Benutzer über die Eigenschaften des Systems berücksichtigt werden.

Die Vorgehensweise kann in naheliegender Weise erweitert werden, wenn mehrere Akteure an einem Dialog (bzw. Konversation) beteiligt sind. Werden dabei auch Menschen um Unterstützung gebeten, kann der Ansatz verfeinert werden, indem für die einzelnen Akteure nicht undifferenziert die gesamten Handlungsmöglichkeiten berücksichtigt werden, sondern Teilbereiche separiert werden, von denen *erwartet werden kann*, daß der Akteur solche Handlungen auch *für andere ausführen* wird oder muß. Letzteres ist bspw. für Handlungen im *Aufgabenbereich* von Mitarbeitern in einem Unternehmen der Fall. Beispielsweise wird eine Äußerung "*Ich benötige ein neues Benutzerkonto für Person XY.*" von einem Systemadministrator normalerweise als Aufforderung interpretiert, das Konto einzurichten, weil dies zu seinen Aufgaben gehört. Von einer studentischen Hilfskraft kann die gleiche Äußerung dagegen (bei naheliegender Aufgabenverteilung) lediglich als allgemeine Information oder als Aufforderung, den Systemadministrator zu informieren bzw. bitten, aufgefaßt werden. Dies gilt selbst dann, wenn die Hilfskraft ebenfalls das Benutzerkonto einrichten könnte – bspw. weil ihr das Administratorkennwort bekannt ist.

Insgesamt können äußere Einfüße auf die Interpretation von Dialogbeiträgen mithilfe des hier skizzierten Benutzer- bzw. Akteurmodells wesentlich präziser beschrieben werden, als durch die Berücksichtigung eines Dialograhmens.

6.2.4.4 Fehlerdiagnose

Ausgangspunkt für eine Fehlerdiagnose ist ein Plan, den der Benutzer ausgeführt hat oder noch ausführt, der aber nicht zum Ziel geführt hat bzw. der nach Ansicht des Systems nicht zum Ziel führen wird. Im ersten Fall könnte sich der Benutzer bspw. von sich aus mit der Bitte um Unterstützung an das System wenden. Der zweite Fall ist bei einem aktiven Hilfesystem denkbar, das den Benutzer beobachtet und dabei feststellt, daß er einen fehlerhaften Plan ausführt (vgl. a. 2.5.1 *Erkennung von falschen oder suboptimalen Plänen*). Gehen wir davon aus, daß der Benutzer dem System im ersten Fall (auch) seine Ziele mitteilt und das System die Benutzerziele im zweiten Fall durch Planerkennung ermittelt, kann für den Benutzer – so wie bisher in dieser Arbeit vorgeschlagen – ein Plan erzeugt und präsentiert werden, mit dem dieser seine Ziele erreichen kann. Dennoch bleibt die Lösung unbefriedigend, weil der Benutzer dadurch nicht weiß, warum sein Plan fehlerhaft ist und deshalb auch nicht abschätzen kann, ob sein Plan in einer ähnlichen Situation ebenfalls nicht funktionieren wird.

Im zweiten Fall kommt hinzu, daß der Benutzer keinen Anhaltspunkt dafür bekommt, wieso sein Plan überhaupt falsch sein könnte und deshalb einen Rat des Systems in Zweifel ziehen könnte.

Während die in Kap. 2.5.2 *Erkennung von Fehlerursachen* vorgestellten Ansätze auf Plänen einer Planbibliothek basieren, ermöglicht die in dieser Arbeit vorgeschlagene Repräsentation des Planungswissens die Analyse von beliebigen Plänen, die der Benutzer aufgrund seines spezifischen Wissens erzeugen kann. Dabei ist unerheblich, ob der Benutzer einen Plan tatsächlich konstruiert hat oder ob er lediglich auf einen ihm bekannten Plan zurückgegriffen hat.

Teilt der Benutzer dem System den zu analysierenden Plan (bspw. im Rahmen einer Hilfeanforderung) mit, kann davon ausgegangen werden, daß die Beschreibung lediglich die Folge der Schritte des Plans enthält. Für die Bestimmung der Fehlerursache(n) müssen zunächst die weiteren Bestandteile eines Plans bestimmt werden. Nach der Definition eines Plans in Kap. 2.1.1 also Causal Links, Bindungsbedingungen und die Zugehörigkeit der Planschritte zu den Plan-Concepts (bzw. Planoperatoren). Für das Ergebnis kann dann festgestellt werden, ob der Plan vollständig[15] ist bzw. wenn nicht, welche Vorbedingungen nicht erfüllt sind. Sowohl die Ergänzung des Plans als auch die Feststellung unerfüllter Vorbedingungen erfolgt anhand des Wissens des Systems. Die Fehlerbeurteilung stellt also die Ansicht des Systems dar. Um dem Benutzer den oder die Fehler treffend erklären zu können, muß diese Ansicht der Ansicht des Benutzers gegenübergestellt werden. Das heißt, daß die Ermittlung der Causal Links, Bindungsbedingungen und Plan-Concepts für den Plan des Benutzers zusätzlich anhand der Benutzerannahmen durchgeführt werden muß. Wird der fehlerhafte Benutzerplan durch Planerkennung ermittelt, kann in analoger Weise vorgegangen werden.

Die möglichen Fehlerursachen können grob in drei Gruppen eingeteilt werden:

- der Benutzer hat versucht ein Plan-Concept auszuführen, das nicht zu seinem Handlungspotential gehört. Mögliche Resultate sind bspw. unerwartete, unerwünschte oder gar keine Effekte.

- der Planungsprozeß des Benutzer war fehlerhaft. Bspw. kann der Benutzer eine Vorbedingung unberücksichtigt gelassen haben oder er kann übersehen haben, daß ein Planschritt eine bereits erfüllte Vorbedingung wieder zerstört.

- das System hat Unzulänglichkeiten des Plans festgestellt; die Analyse anhand des Benutzerwissens läßt ihn aber als vollständig und konsistent erscheinen. Die Ursache liegt also in falschen Annahmen des Benutzers. Diese können das Planungswissen selbst (falsche Vorbedingungen oder Effekte von Plan-Concepts) oder die Eigenschaften von Objekten (falsche Typzugehörigkeit, falsche Attribute) betreffen.

Es ist offensichtlich, daß diese Fehlertypen unterschiedliche Erklärungen für den Benutzer erfordern. In Kap. 3.4.2.4 *Ableitung von Abhängigkeiten aus dem Benutzermodell* wurde außerdem ein Beispiel angegeben das zeigt, daß die Erklärung von generellen gegenüber konkreten Fehlannahmen vorteilhaft sein kann. In dem Beispiel hatte der Benutzer die generelle Fehlannahme, daß jede Textverarbeitungsdatei eine Text-Datei ist und hatte deshalb für den FTP-Transfer einer Textverarbeitungsdatei ein (gemäß Systemwissen) falsches Plan-Concept verwendet. Eine sinnvolle Systemreaktion klärt den Benutzer über seine generelle Fehlannahme auf, anstatt ihn lediglich darauf hinzuweisen, daß die betreffende Datei eine Binär-Datei ist oder daß er anstelle des Transfers im Text-Modus den Binär-Modus verwenden muß.

Wichtig ist, daß für eine detaillierte Analyse der Fehlerursache als Grundlage für eine treffende Fehlererklärung das Planungswissen sowohl des Systems als auch des Benutzers berücksichtigt werden müssen.

[15] vgl. wiederum Kap. 2.1.1 *Least Commitment Planning*

Außer der Erklärung des Fehlers benötigt der Benutzer einen korrekten Plan mit dem er seine Ziele erreichen kann. Sinnvollerweise wird dafür nicht ein völlig neuer Plan erzeugt, sondern – sofern es eine Lösungsmöglichkeit gibt – der unvollständige Plan des Benutzers vervollständigt. Ggf. müssen zuvor Plan-Concepts eliminiert werden, die nicht zum Handlungspotential des Benutzers gehören. Da das in dieser Arbeit eingesetzte Planungsverfahren ohnehin mit einem unvollständigen Plan startet[16], kann das gesamte Verfahren zur Erzeugung und Präsentation eines benutzerspezifischen Plans auch für eine solche Planvervollständigung angewendet werden.

6.2.5 Empirische Überprüfung

Für die These dieser Arbeit, daß die Erzeugung einer sinnvollen Handlungsanweisung außer den Kenntnissen auch die spezifischen Handlungsmöglichkeiten des Akteurs berücksichtigen muß, ist keine empirische Überprüfung notwendig. Es ist offensichtlich, daß die Mißachtung der Handlungsmöglichkeiten des Akteurs zu Anweisungen führen kann, die dieser nicht ausführen kann und die damit wertlos sind. Gleiches gilt auch für die (Nicht-)Berücksichtigung der Handlungsmöglichkeiten bei Erklärungen, die zu den Ausführungsmöglichkeiten einzelner Schritte einer Handlungsanweisung gegeben werden können.

Die im Präsentationskapitel entwickelten Verfahren haben die Aufgabe, eine "möglichst gute" Beschreibung eines Plans zu erzeugen, der für einen Benutzer unter Berücksichtigung seiner Handlungsmöglichkeiten generiert wurde. Die Gewichtung der einzelnen Einflußfaktoren, die zur Bewertung einer solchen Beschreibung beitragen, wurden dabei lediglich abgeschätzt, weil die Verfahren in dieser Arbeit in erster Linie zeigen sollen, *daß* und *wie* das neue Benutzermodell auch für die Planpräsentation sinnvoll eingesetzt werden kann. Für die Abschätzung wurden zwar einige empirische Untersuchungen aus der Psychologie berücksichtigt (vgl. insb. Kap. 4.3.4), für eine gesicherte Festlegung der Gewichtungen und zur Bewertung des Gesamtmodells sind aber spezifischere Untersuchungen erforderlich. Im folgenden werden Möglichkeiten für solche Untersuchungen skizziert.

Als "Gütekriterium" für eine adaptierte Planbeschreibung bietet sich das Ziel der Benutzeradaption in dieser Arbeit an. Es muß also überprüft werden, ob die Benutzeradaption die Chancen verbessert, daß der Benutzer seine Ziele erreicht. Um lediglich die Adaption der Plan*beschreibung* zu bewerten, müssen verschiedene Beschreibungen des gleichen Plans verglichen werden. Sinnvollerweise sind dies Pläne, mit denen der Benutzer im Prinzip seine Ziele erreichen kann. Weitere Kriterien sind der Zeitaufwand, den der Benutzer für das Verstehen der Planbeschreibung und evtl. für das Ausführen des Plans benötigt und die subjektive Zufriedenheit des Benutzers. Letzteres ist ein wichtiges Kriterium für die Akzeptanz des Gesamtsystems.

Um vergleichbare Bedingungen zu schaffen, d.h. insbesondere um Wissen und Handlungsmöglichkeiten der Benutzer im Anwendungssachbereich sicher bestimmen zu können, bietet es sich an, die Untersuchungen anhand einer künstlichen Welt durchzuführen. Die Kenntnisse der Versuchspersonen können dann bestimmt werden, indem diese definierte Ausschnitte der künstlichen Welt mit den darin enthaltenen Handlungsmöglichkeiten lernen. Die Handlungsmöglichkeiten der Versuchspersonen können durch die Versuchsanordnung festgelegt werden.

In einem ersten Schritt müssen zunächst Werte für die Gewichtung der einzelnen Einflußfaktoren für die Bewertung einer Planbeschreibung ermittelt werden. In den Abschnitten 4.3.1, 4.3.3 und 4.4.2.2 des Präsentationskapitels wurden dafür die Konstanten k_{cl}, k_{hier}, k_{corr} und k_{check} eingeführt, die als Faktoren bzgl. der Basiseinheit k_{pres} verwendet wurden.

Es bietet sich an, analog zu den Untersuchungen zum Verstehen von Anweisungen vorzugehen; d.h. den Versuchspersonen Beschreibungen zu präsentieren und die Zeit für das Lesen der Beschreibung als Maß für den Aufwand für das Erfassen der Beschreibung anzusehen, falls die Anweisung

[16] normalerweise dem *Startplan* aus zwei initialen Schritten *Start* und *Ziel* (vgl. wieder Kap. 2.1.1)

korrekt wiedergegeben oder ausgeführt wird (vgl. bspw. [Dixon, 1982] und Kap. 4.3.4). Da für die Ausführung einer Anweisung nicht in jedem Falle alle Komponenten der Beschreibung benötigt werden, ist es für den ersten Schritt sinnvoll zu prüfen, ob die Versuchspersonen die Beschreibung korrekt wiedergeben können. Die Komponenten der Beschreibung sollten dabei zunächst für die Versuchspersonen unbekannt sein. Damit kann verhindert werden, daß eine Beschreibung nur teilweise gelesen, aber durch Schlußfolgerung der überlesenen Teile dennoch vollständig wiedergegeben wird. Als Gegenprobe und ggf. als Ansatz für eine Verbesserung der Präsentationsverfahren können dann die Aufwände bestimmt werden, wenn die Beschreibungen auch Komponenten enthalten, die den Versuchspersonen bekannt sind oder die sie durch Schlußfolgerung ermitteln können. In einer weiteren Untersuchung müssen dann gezielt die Aufwände für Schlußfolgerungen auf Plankomponenten bestimmt werden, die in der Beschreibung nicht explizit enthalten sind. Da die Versuchspersonen die notwendigen Schlußfolgerungen sowohl während des Lese- als auch während des Wiedergabevorgangs ziehen können, läßt sich der Aufwand nicht direkt bestimmen. Er kann aber aus der Gesamtzeit für Lesen und Wiedergabe errechnet werden, wenn der Aufwand für das Lesen und die Wiedergabe solcher Planbeschreibungen abgezogen wird, die keine Schlußfolgerungen erfordern.

Es ist zu erwarten, daß der Aufwand für das Erfassen der verschiedenen Komponenten einer Planbeschreibung von der Darstellungs*form* abhängt. Die Untersuchungen sollten deshalb mit mehreren Darstellungsformen – zumindest einer textuellen und einer grafischen – durchgeführt werden. Letztere kann sich bspw. an Abb. 5.11 S. 272 orientieren.

Die ermittelten Ergebnisse haben direkten Einfluß auf das Verhalten des Reduktionsalgorithmus aus Abb. 4.3 S. 206, der Komponenten aus einer Planpräsentation entfernt, die der Benutzer leicht inferieren kann. Hierbei wird der Aufwand des Benutzers für das Erfassen gegen den Aufwand für das Rekonstruieren einer Plankomponente abgewogen. Beim Präsentationsalgorithmus aus Abb. 4.2 S. 189 kann der Parameter k_{hier} bewirken, daß längere, strukturierte Erklärungen von Ausführungsmöglichkeiten von Plan-Concepts gegenüber kürzeren, aber unstrukturierten bevorzugt werden. Ergibt sich aus der Untersuchung, daß der Aufwand für das Erfassen eines Planschritts signifikant geringer ist, wenn der Benutzer das zugehörige Plan-Concept kennt, muß dies durch die Funktion *effort-plan-concept* berücksichtigt werden. Die Folge wäre, daß der Präsentationsalgorithmus Pläne und Erklärungen von Plan-Concepts bevorzugt, die bekannte Plan-Concepts enthalten. Außerdem würde der Reduktionsalgorithmus bekannte Planschritte weniger häufig aus einer Planbeschreibung entfernen, weil der daraus resultierende Nutzen durch den geringeren Aufwand für das Erfassen ebenfalls sinken würde.

In einem zweiten Schritt kann danach untersucht werden, welche Verbesserungen sich durch die in dieser Arbeit vorgeschlagene Benutzeradaption von Planbeschreibungen bzgl. der o.g. Gütekriterien ergeben. Die Untersuchung kann ähnlich wie die Untersuchung zur Bestimmung der Gewichtung der Einflußfaktoren durchgeführt werden. Allerdings geht es nun darum, daß die Versuchspersonen die Planbeschreibung in eine Folge von Handlungen umsetzen, mit der das oder die Ziele (in der künstlichen Welt) erreicht werden können. Die Handlungen, die eine Versuchsperson ausführen kann bzw. darf, werden von der Versuchsanordnung vorgegeben, wodurch das Handlungspotential dieser Versuchsperson festgelegt wird. Durch die Versuchsanordnung wird auch die Ausgangssituation bestimmt und damit festgelegt, ob der präsentierte Plan ohne Modifikation ausgeführt werden kann oder nicht. Alle Handlungen müssen dann in einer Simulation der künstlichen Welt ausgeführt werden. Handlungen, die eine Versuchsperson dabei direkt in einem Schritt ausführen kann, müssen ausgezeichnet werden, sodaß die Versuchsperson erkennen kann, bis zu welchem Detaillierungsgrad sie Planschritte dekomponieren muß.

Für die nicht adaptierten Planbeschreibungen, die als Bezugspunkt für die Beurteilung der adaptierten herangezogen werden, bieten sich mehrere Alternativen an, die sinnvollerweise alle in die Untersuchung einbezogen werden. Die Schritte des Plans müssen in jeder nichtadaptierten Planbe-

schreibung enthalten sein, da nicht bekannt ist, welche Schritte der Benutzer selbst ermitteln könnte, wenn sie nicht explizit genannt werden. Causal Links können dagegen auch weggelassen werden, weil sie für die Planausführung selbst nicht erforderlich sind. Alternativ können sie vollständig oder "auf Anforderung" genannt werden. Gleiches gilt auch für Dekompositionen von Planschritten. Hierbei kann weiter unterschieden werden, ob den Versuchspersonen alle Dekompositionen für einen Planschritt genannt werden oder nur eine, die bspw. zufällig ausgewählt wird. Dekompositionen, die die Versuchsperson bei gegebenem Handlungspotential *nicht* ausführen können, sollten allerdings auch bei einer an sich nicht adaptierten Planbeschreibung unterdrückt werden, weil auch ohne empirische Untersuchung klar ist, daß sie nutzlos sind.

Während Dauer und Fehlerhäufigkeit solcher Planausführungen direkt gemessen werden kann, muß die subjektive Zufriedenheit der Versuchspersonen mit den Planbeschreibungen durch Befragung ermittelt werden. Hierbei sind spezifische Fragen sinnvoll, die auch Rückschlüsse auf die Ursache mangelnder Zufriedenheit zulassen; bspw. zu viel oder zu wenig Information, ggf. zu welchem Aspekt der Planbeschreibung.

6.3 Abschließende Bemerkungen

Es ist unumstritten, daß Softwaresysteme, die nicht an den jeweiligen Benutzer angepaßt sind, für den einzelnen Benutzer unangemessen sein können. Dies gilt besonders dann, wenn nicht von einem homogenen Benutzerkreis ausgegangen werden kann. In dieser Arbeit wurde bspw. gezeigt, daß der Rat eines Hilfesystems völlig wertlos sein kann, wenn die Handlungsmöglichkeiten des Benutzers nicht berücksichtigt werden.

Dennoch wird der Einsatz von Benutzermodellierung keinesfalls uneingeschränkt positiv gesehen. Die Vorstellung, daß ihr Verhalten von einem Computer beobachtet, analysiert und daraus möglicherweise ein Persönlichkeitsprofil erstellt wird, bereitet vielen Benutzern Unbehagen. Solche Vorgänge finden zwar auch bei der Kommunikation zwischen Menschen statt – u.a. ebenfalls, um sich auf den Dialogpartner einzustellen. Im Vergleich dazu kann ein Computer allerdings wesentlich mehr Beobachtungen längerfristig speichern und aufwendigere Analysen anstellen. Für den Benutzer ist dabei oft nicht transparent, welche Informationen über ihn gesammelt werden, wie sie ausgewertet werden und wer Zugriff darauf hat.

Studien wie bspw. [For, 1999] oder [DePallo, 2000] (die speziell das Benutzerverhalten im Internet untersuchten) zeigen, daß mehr als zwei Drittel der Benutzer sehr besorgt darüber sind, wenn ihr Verhalten aufgezeichnet und persönliche Daten gesammelt werden. Andererseits sind Benutzer zumindest teilweise bereit persönliche Daten preiszugeben, wenn sie um Erlaubnis gefragt werden und wenn sie davon einen Nutzen haben. Bezogen auf die Benutzermodellierung heißt dies, daß es für die Akzeptanz eines Systems förderlich ist, wenn der Benutzer selbst bestimmen kann, ob und in welchem Maße sein Verhalten durch ein System analysiert werden soll. Außerdem sollte das System in der Lage sein dem Benutzer zu erklären, welche Informationen über ihn gespeichert sind und welche Konsequenzen sich daraus ergeben. Ohnehin bestehen solche Anforderungen durch die Datenschutzgesetze vieler Länder (vgl. bspw. [Kobsa, 2001]). Zusätzlich wird zumeist gefordert, daß der Benutzer falsche Information die über ihn gespeichert ist korrigieren oder löschen können muß. Für eine ausführliche Behandlung von Datenschutz und Wahrung der Privatsphäre bei der Benutzermodellierung sei auf [Schreck, 2000] verwiesen.

Ein weiterer kritischer Aspekt von adaptiven Systemen ist die (aus Benutzersicht oft unvorhersehbare) Veränderbarkeit ihres Verhaltens bzw. ihrer Ausgaben. Diese Veränderbarkeit ist einerseits notwendig, um sich an den jeweiligen Benutzer anzupassen. Andererseits ist es für einen Benutzer schwierig ein System zu bedienen, dessen Verhalten sich häufig ändert. Ein Beispiel ist die Anpas-

sung von Menü-Einträgen einer graphischen Bedienoberfläche an die Nutzungshäufigkeit durch den jeweiligen Benutzer. Häufig benutzte Einträge können so positioniert werden, daß der Benutzer sie möglichst leicht erreichen kann. Dennoch kann eine solche Anpassung störend sein, wenn ein Eintrag seine für den Benutzer vertraute Position verläßt. Die Handlungsmöglichkeiten eines Benutzers, die im Fokus dieser Arbeit standen, ändern sich zwar normalerweise nicht (häufig). Sehr wohl ändern kann sich allerdings die *Ansicht* des Systems darüber. Auch hier ist es also sinnvoll, den Benutzer über eine geänderte Ansicht des Systems zu informieren, um ein evtl. verändertes Systemverhalten für den Benutzer nachvollziehbar zu machen. Offen ist dabei, *wie* der Benutzer informiert wird, ohne vom eigentlichen Ziel des Dialogs abzulenken. Denkbar ist auch, daß das System sein Verhalten trotz geänderter Ansichten *un*verändert läßt, wenn der Benutzer dann dennoch seine Ziele erreichen kann und ansonsten langwierige Erklärungen erforderlich wären. Auf jeden Fall muß das System allerdings reagieren, wenn es zu der Annahme kommt, daß die Handlungsmöglichkeiten des Benutzers geringer sind als bisher angenommen. Ansonsten besteht die Gefahr, daß ein Plan erzeugt wird, den der Benutzer nicht ausführen kann.

Auch hierbei ist es wieder für die Akzeptanz des adaptiven Systems wichtig, daß der Benutzer nicht “zu seinem Glück gezwungen wird”, sondern selbst entscheiden kann ob und in welchem Maße ein System sich an ihn anpaßt. Damit ein Benutzer eine solche Entscheidung sinnvoll treffen kann, müssen ihm Informationen über die Funktionsweise des Systems sowie über Art und Umfang der Datenspeicherung und -nutzung offen gelegt werden. Dies beinhaltet auch Informationen über den Schutz der Daten gegenüber unbefugtem Zugriff.

Anhang A

Syntaxdefinitionen

A.1 *Extended Backus-Naur Form (EBNF)*

A.2 *Zusammenfassung syntaktischer Definitionen in EBNF*

A.3 *SB-ONE: Syntax der graphischen Notation*

A.1 Extended Backus-Naur Form (EBNF)

Die folgende Definition der Extended Backus-Naur Form (EBNF) ist [Jensen und Wirth, 1985, S.213f] entnommen. Dabei wurde die mit "[d)]" gekennzeichnete Zeile hinzugefügt.

An Extended Backus-Naur Form (EBNF) specification of the syntax of a programming language consists of a collection of rules or productions collectively called a "grammar" that describe the formation of sentences in the language. Each production consists of a non-terminal symbol and an EBNF expression separated by an equal sign and terminated with a period. The non-terminal symbol is a "meta-identifier" (a syntactic constant denoted by an English word), and the EBNF expression is its definition.

The EBNF expression is composed of zero or more terminal symbols, non-terminal symbols, and other metasymbols summarized in the table below:

Metasymbol	**Meaning**
=	is defined to be
|	alternatively
.	end of production
[X]	0 or 1 instance of X
{X}	0 or more instances of X
$\{X\}^+$	1 or more instances of X [d)]
(X | Y)	a grouping: either X or Y
"XYZ"	the terminal symbol XYZ
MetaIdentifier	the non-terminal symbol *MetaIdentifier*

As an example, EBNF can be used to define its own syntax.

Syntax	=	{ *Production* } .
Production	=	*NonTerminal* "=" *Expression* "." .
Expression	=	*Term* { "|" *Term* } .
Term	=	*Factor* {*Factor* } .
Factor	=	*NonTerminal* | *Terminal* | "(" *Expression* ")" |
		"[" *Expression* "]" | "{" *Expression* "}" .
Terminal	=	"""" *Character* { *Character* } """" .
NonTerminal	=	*Letter* { *Letter* | *Digit* } .

Notes:

1. A terminal symbol (literal) is always enclosed in quotation marks ("); if a " itself is enclosed, it is written twice. Thus in the Pascal EBNF below "["and "]" represent left and right brackets in a Pascal program, whereas [and] are meta-symbols in an EBNF expression that specify zero or one occurrence of whatever they enclose.
2. Every syntax has a *start symbol*, a meta-identifier from which all the sentences in the language are generated. The start symbol for the Pascal syntax is *Program*.

[d)] $\{X\}^+$ ist *nicht* in der Definition von Jensen und Wirth enthalten. $\{X\}^+$ kann durch X {X} ersetzt werden.

A.2 Zusammenfassung syntaktischer Definitionen in EBNF

Dieses Kapitel enthält die Syntax-Vorschriften, die für die Definition von Plan-Concepts und den damit zusammenhängenden Angaben eingehalten werden müssen. Es handelt sich um eine Zusammenfassung von Definitionen, die zum größten Teil bereits im Verlauf der Arbeit angegeben und erläutert wurden. Ausnahmen bilden Concept-Beschreibungen, sowie die Beschreibungen von potentiellen Axiomen und Facts, deren Syntax dem UCPOP-Manual [Barrett *et al.*, 1995] entnommen wurde. Für eine ausführliche Erläuterung der Konstrukte wird auf die entsprechenden Stellen dieser Arbeit bzw. des UCPOP-Manuals verwiesen.

Für die Beschreibung wird die erweiterte Backus-Naur-Form [Jensen und Wirth, 1985, S.213f] verwendet, die in Anhang A.1 beschrieben ist. Die folgenden nichtterminalen Symbole haben lediglich eine verbale Beschreibung:

any-string	ein String nach LISP-Syntax
number	eine vorzeichenlose Zahl nach LISP-Syntax
var	ein LISP-Name, der mit einem Fragezeichen beginnt
nichtterminale Symbole, deren Bezeichnung mit "*-name*" endet	ein Name nach LISP-Syntax

Die Definitionen der nichtterminalen Symbole *literal* und *predicate* finden sich in Abb. A.2.

A.2.1 Beschreibung von Concepts

Für die kompakte Beschreibung von SB-ONE-Concepts wurde die LISP-Structure *bgp-ms-concept-struc* eingeführt (vgl. a. Kap. 5.2.1 *Vorgehensweise*). Sie enthält Slots für alle Komponenten einer Concept-Beschreibung: *Typ* des Concepts, *Super-Concepts*, *Attribute*, *Rollen* und *Role-Value-Maps*. Für Dokumentationszwecke ist außerdem ein Kommentar-Slot vorgesehen. Abb. A.1 zeigt die Syntax für die Beschreibung. Die darin enthaltenen Schlüsselwörter sind gleichzeitig die Slot-Bezeichner von *bgp-ms-concept-struc*, sodaß die angegebene Syntax auch bei der Erzeugung der Structure mithilfe der zugehörigen Konstruktor-Funktion zur Anwendung kommt (vgl. die Beispiele in Abb. 5.2 S. 246 und in der Datei *office-world-domain-concepts.lisp* im Verzeichnis *KN-PP-Demo* des elektronischen Anhangs dieser Arbeit).

SB-ONE verzichtet auf das Beschreibungsmittel "Attribut", das gleichbedeutend mit einer notwendigen Rolle mit der Anzahlrestriktion [1,1] ist. Aus Gründen einer klareren Beschreibung wurde das Konstrukt in dieser Arbeit dennoch verwendet. Beim Eintrag in eine SB-ONE-Wissensbasis wird es wie eine entsprechend restringierte Rolle behandelt. (Vgl. dazu und für die Semantik der Beschreibungen die Abschnitte 3.2.1 *Beschreibungslogiken* und 3.2.2.1 *Spezifika der Repräsentation in SB-ONE* im Modellierungs-Kapitel.).

Durch die Funktion *bgp-ms-defconcepts-from-struclist* kann eine Liste solcher Concept-Beschreibungen in eine oder mehrere Views einer Benutzermodellierungswissensbasis eingetragen werden (s.a. die Beispiele in Kap. 5.2.3 *Initialer Aufbau einer Benutzermodellierungswissensbasis*).

A.2.2 Beschreibung von Plan-Concepts

Für die Definition von Plan-Concepts wurde in Kap. 3.2.2 *Planoperatoren in Beschreibungslogik* eine Beschreibung auf Basis von Planoperatoren vorgeschlagen, die i.w. ein Auszug aus der PDDL bzw. UCPOP-Syntax darstellt. Für die Definition eines Plan-Concepts ist die Erzeugung einer LISP-Structure *bgp-ms-plan-struc* erforderlich, wobei die Syntax aus Abb. A.2 einzuhalten ist[1]. Die Funktion

[1] vgl. auch das Bsp. in Abb. 5.3 S. 247 im Kapitel 5.2.1 *Vorgehensweise*.

conceptstruc-def	=	":name" "'" *concept-name*
		[":type" (":primitive" | ":defined" | ":singleton")]
		[":supers" "'(" {*concept-name*}$^+$ ")"]
		[":attributes" "'(" {*attribute-descr* }$^+$ ")"]
		[":roles" "'(" {*role-descr* }$^+$ ")"]
		[":rvms" "'(" {*rvm-descr* }$^+$ ")"]
		[":comment" *any-string*] .
attribute-descr	=	"(" *attribute-name* [*value-restr*] ")" .
role-descr	=	"(" *role-name* [*number-restr*] [*value-restr*] ")" .
number-restr	=	"(" *min* (*max* | "INF") ")" .
value-restr	=	*concept-name* .
min	=	*number* .
max	=	*number* .
rvm-descr	=	"(" ("eq" | "neq" | "sub") "(" {*role-name*}$^+$ ")" "(" {*role-name*}$^+$ ")" ")" .

Abbildung A.1: EBNF-Syntax für die Definition von SB-ONE-Concepts

knpp-make-plan-concept kann zur Umsetzung dieser Structure in eine SB-ONE-Concept-Beschreibung (s.o.) verwendet werden. Für den Eintrag eines Plan-Concepts in eine BMWB steht die Funktion *bgp-ms-def-plan-concepts-from-struclist* zur Verfügung, die in eine oder mehrere Views die Einträge macht, die sich aus der als Parameter angegebenen Liste solcher Structures ergibt. Beispiele dazu finden sich wieder im Abschnitt 5.2.3.

planstruc-def	=	":name" "'" *planconcept-name*
		[":supers" "'(" {*planconcept-name*}$^+$ ")"]
		[":precondition" "'" *plan-conjunction*]
		[":effect" "'" *plan-conjunction*]
		[":comment" *any-string*] .
plan-conjunction	=	*literal-conjunction* | *literal* | "(" ")" .
literal-conjunction	=	"(" "and" {*literal*}$^+$ ")" .
literal	=	*predicate* | "(" "not" *predicate* ")" .
predicate	=	"(" *predicate-name var* [*var*] ")" .

Abbildung A.2: EBNF-Syntax für die Definition von Plan-Concepts.

A.2.3 Beschreibung von Dekompositionen

Analog zur Definition eines Plan-Concepts mithilfe einer LISP-Structure ist für Dekompositionen die Structure *bgp-ms-decomp-struc* vorgesehen (vgl. Kap. 4.6.2 *Repräsentation von Dekompositionen in BGP-MS*). Abb. A.3 zeigt die entsprechende Syntax. Für den Eintrag von so beschriebenen Dekompositionen in eine BMWB ist die Funktion *bgp-ms-def-decomposition-from-struclist* vorgesehen, die

auch die Umsetzung in die beschreibungslogische Repräsentation durchführt. Aufrufbeispiele finden sich wieder im Abschnitt 5.2.3.

decompstruc-def = [“:decomposition” “”’ *decomposition-name*] “:of” “”’ *plan-concept-name*
“:steps” “’(” {“(” *step-name plan-concept-name* {*step-constraint*} “)”}$^+$ “)”
[“:precedence” “’(” {“(” *step-name step-name* “)”}$^+$ “)”]
[“:comment” *any-string*] .
step-constraint = “(” “=” *plan-concept-var-name step-var-name* “)” .

Abbildung A.3: EBNF-Syntax für die Definition von Dekompositionen eines Plan-Concepts.

A.2.4 Beschreibung von Axiomen für den Planungssachbereich

Die Syntax von UCPOP-Axiomen ist in Kap. 4.5 des UCPOP-Manuals [Barrett *et al.*, 1995] angegeben. In Kap. 3.4.2.4 *Ableitung von Abhängigkeiten aus dem Benutzermodell* wurde erörtert, welche Abhängigkeiten im Sachbereich durch Axiome beschrieben und aus dem Benutzermodell abgeleitet werden können. Abb. A.4 zeigt die daraus resultierenden syntaktischen Restriktionen in EBNF, wobei die Beschreibung eines Axioms die Form einer LISP-Liste mit drei Schlüsselwort-Wert Paaren für den Axiom-Namen, die Prämisse und die Konklusion hat. Die hochgestellten Ziffern geben dabei die Stelligkeit der Prädikate an.

axiom-def = “(” “:axiom” *axiom-name*
(1) “:context”
(2) (*predicate*^{1} |
(3) “(” “exists” *predicate*^{1} *predicate*^{2} “)” |
(4) “(” “exists” *predicate*^{2} *predicate*^{1} “)” |
(5) “(” “exists” *predicate*^{1} “(” “and” *predicate*^{2} *predicate*^{1} “)” “)” |
(6) “(” “exists” *predicate*^{1} “(” “and” *predicate*^{1} *predicate*^{2} “)” “)”)
(7) “:implies” *predicate*^{1} “)”.

Abbildung A.4: EBNF-Syntax für die Definition von potentiellen Axiomen.

Die Zeilen (3) und (4) bzw. (5) und (6) beschreiben logisch äquivalente syntaktische Varianten, die bei der Verarbeitung gleich behandelt werden, wenngleich in Kap. 3.4.2.4 nur die jeweils erstgenannten Varianten angegeben wurden.

Die Axiom-Beschreibung können der Funktion *bgp-ms-define-pd-dependency* als Parameter mitgegeben werden, womit sie in die Menge der potentiellen Axiome der BMWB eingetragen werden (vgl. 5.1.1 *Verarbeitung in der Modellierungsphase* insb. Fußnote 8 S. 230 und Kap. 5.2.3 *Initialer Aufbau einer Benutzermodellierungswissensbasis*). Um dem Entwickler des Sachbereichsmodells die Möglichkeit zu geben, Abhängigkeiten im Sachbereich durch Axiome zu beschreiben, die *nicht* aus dem Benutzermodell abgeleitet werden können oder sollen, wurden *Spezial-Axiome* eingeführt. Diese müssen für Prämisse und Konklusion lediglich die UCPOP-Syntax einhalten und werden durch das Schlüsselwort *:axiom-special* anstelle von *:axiom* gekennzeichnet. Sie können wie normale Axiome

in eine BMWB eingetragen werden und werden bei der Erzeugung des Planungssachbereichs *immer* an das Planungssystem weitergegeben.

A.2.5 Beschreibung von Facts für den Planungssachbereich

Zulässige Beschreibungen von Facts werden im UCPOP-Manual (Kap. 4.6) definiert. Die folgende EBNF-Syntax ist einzuhalten:

fact-def = "(" ":fact" *predicate LISP-function-body* ")".

LISP-function-body steht dabei für den Rumpf einer LISP Funktionsdefinition. In dieser Arbeit werden Facts allerdings ausschließlich zur Erzeugung neuer Variablen eingesetzt. D.h. in die BMWB wird nur genau ein Fact eingetragen und zwar genau in der in Kap. 3.4.2.5 *Erzeugen von Objekten* beschriebenen Form. Der Eintrag erfolgt wieder mit Funktion *bgp-ms-define-pd-dependency*.

A.2.6 Beschreibung von impliziten Effekten

Implizite Effekte können angegeben werden, um negative Veränderungen, die sich als kausale Folge einer positiven Veränderung ergeben, für den Planungssachbereich explizit zu machen (vgl. Kap. 3.4.2.4.4 *Ausschließende Eigenschaften bzw. Typzugehörigkeiten*). Da hierfür nur einstellige Prädikate verwendet werden können, die jeweils das gleiche Objekt beschreiben, brauchen (und dürfen) Variablen nicht angegeben werden. Bei der Erzeugung der Planoperatoren für den Planungssachbereich werden die entsprechenden Variablen automatisch erzeugt. Die Beschreibung von impliziten Effekten hat deshalb die in Abb. A.5 gezeigte syntaktische Form.

impl-effect-def = "(" ":conseq" *predicate-name* ":implies" "(" {*lit*}$^+$ ")" ")" .
lit = "(" "not" *predicate-name* ")" .

Abbildung A.5: EBNF-Syntax für die Definition von impliziten Effekten.

Solche Beschreibungen können wieder der Funktion *bgp-ms-define-pd-dependency* mitgegeben werden, um sie in die BMWB einzutragen, und zwar in die *Menge der potentiellen impliziten Effekte* (vgl. wieder Kap. 5.1.1). In dieser Arbeit werden implizite Effekte eingesetzt, um zu erreichen, daß neu erzeugte Objekte nicht mehr zu den "noch zu erzeugenden" Objekten gehören[2]. Da im Beispielsachbereich die Erzeugung von Papierdokumenten (*paperdoc*), Dateien (*fs-file*) und Dateinamen (*fs-name*) möglich ist, sind die folgenden impliziten Effekte modelliert:

```
(:conseq paperdoc :implies ((not to-create)))
(:conseq fs-file  :implies ((not to-create)))
(:conseq fs-name  :implies ((not to-create)))
```

A.2.7 Beschreibung von Vorrang-Mustern

Vorrang-Muster dienen dazu, die Reihenfolge der Vorbedingungskomponenten von Planoperatoren, die für den Planungssachbereich erzeugt werden, zu beeinflussen (vgl. Kap. 3.4.2.7 *Erzeugung des aktiven Teils eines Planungssachbereichs*). Dies ist immer dann sinnvoll, wenn ein Planungssystem eingesetzt wird, das mit unterschiedlichem Zeit und/oder Speicherplatzverbrauch auf die Reihenfolge

[2] s. Kap. 3.4.2.5 *Erzeugen von Objekten*

dieser Komponenten reagieren kann. Da durch Vorrang-Muster nur solche Vorbedingungskomponenten beeinflußt werden, die vom zugehörigen Plan-Concept abgeleitet werden können, sind Vorrang-Muster oft für mehrere ähnliche Plan-Concepts bzw. Operatoren verwendbar. Die folgende syntaktische Form sieht deshalb die Angabe von Literalen für ein oder mehrere Plan-Concepts/-Operatoren vor. Beispiele dazu finden sich im Anhang B.4 *Vorrang-Muster*.

prec-pattern-def = “(” “:prec-pattern” {*literal*}$^+$ “:for” “(” {*planop-name*}$^+$ “)” “)” .

A.3 SB-ONE: Syntax der graphischen Notation

Vgl. dazu auch Kap. 3.2.2.1 *Spezifika der Repräsentation in SB-ONE*.

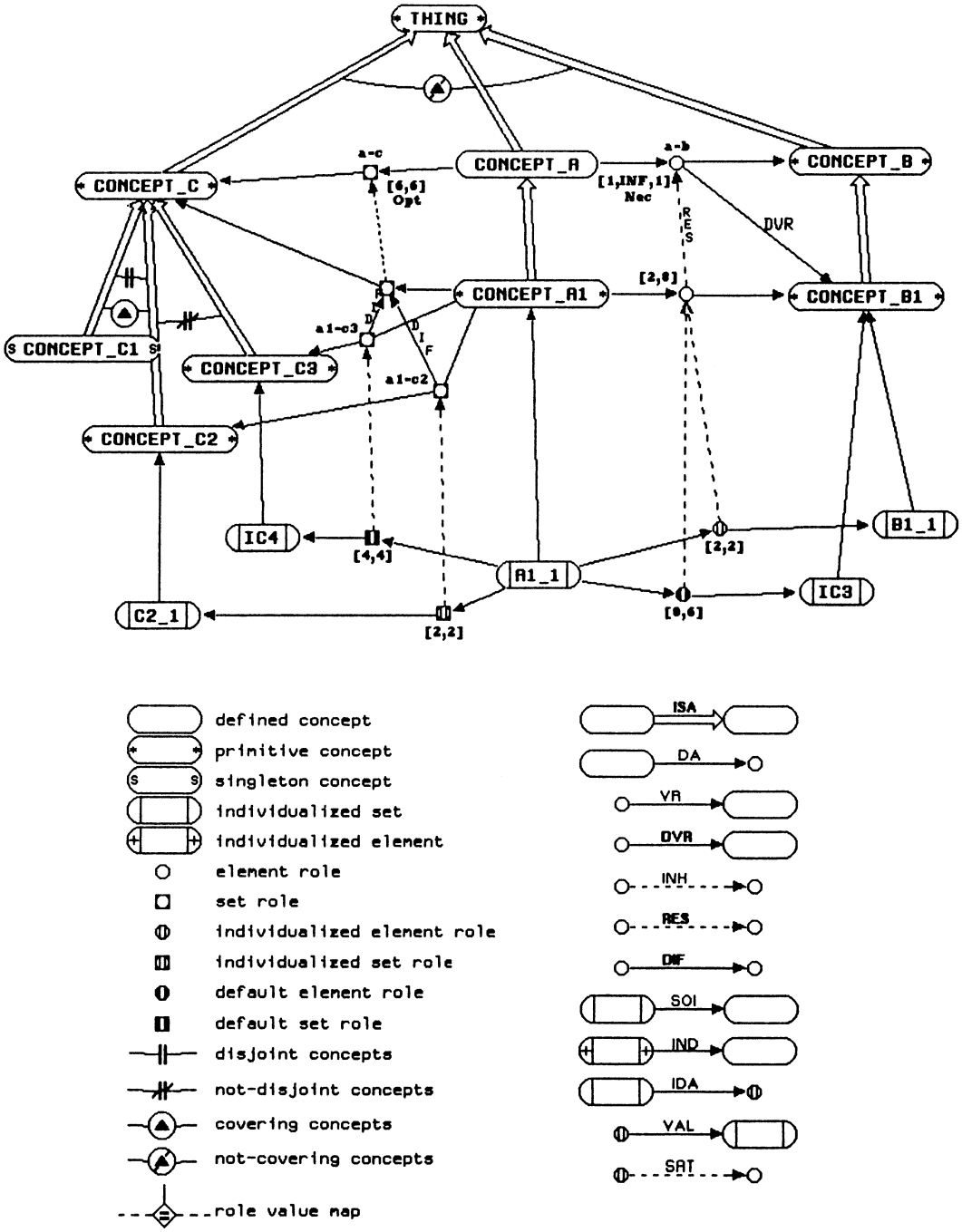

Abbildung A.6: Graphische Darstellung von SB-ONE-Netzen aus [Profitlich, 1990, Anhang B]

Anhang B

Sachbereichsmodell

Die folgenden Abschnitte enthalten – mit Ausnahme von potentiellen impliziten Effekten und Facts – formale Beschreibungen aller Teile des Beispiel-Sachbereichs dieser Arbeit. Aus logischer Sicht sind dies lediglich die Domain- und Plan-Concepts – letztere einschließlich der Dekompositionen. Für die Erzeugung eines Planungssachbereichs ist es außerdem erforderlich festzulegen, welche Abhängigkeiten des Sachbereichs beim Planungsprozeß berücksichtigt werden sollen. In dieser Arbeit sind dafür potentielle Axiome und potentielle implizite Effekte vorgesehen. Hinzu kommen Vorrang-Muster[1], mit denen die Reihenfolge der Vorbedingungskomponenten von Planoperatoren bei der Erzeugung des Planungssachbereichs beeinflußt werden kann, und Facts, die in dieser Arbeit bei der Modellierung der Erzeugung von Objekten eine Rolle spielen. Die für den Beispiel-Sachbereich modellierten potentiellen impliziten Effekte und Facts wurden bereits vollständig in Anhang A.2.6 *Beschreibung von impliziten Effekten* bzw. Kap. 3.4.2.5 *Erzeugen von Objekten* angegeben. Die folgenden formalen Beschreibungen entsprechen den syntaktischen Vorgaben die im Anhang A.2 zusammengestellt sind. Sie können entweder direkt als Parameter der Funktion *bgp-ms-define-pd-dependency* in die Benutzermodellierungswissensbasis (BMWB) eingetragen werden oder sie werden über den Zwischenschritt der Erzeugung von LISP-Structures in die BMWB eingetragen. Auf die Angabe der entsprechenden Funktionsaufrufe wird hier aus Platzgründen verzichtet – es sei stattdessen auf den elektronischen Anhang dieser Arbeit verwiesen. Ebenfalls aus Platzgründen wurde für die Beschreibung der Domain-Concepts eine kompakte beschreibungslogische Notation gewählt. Die konkrete Form, die für den Aufbau der BMWB zum Einsatz kommt, ist – bis auf Beispiele in Kap. 5.2.1 *Vorgehensweise* – nur im elektronischen Anhang enthalten und zwar in der Datei *office-world-domain-concepts.lisp* im Verzeichnis *KN-PP-Demo*. Im gleichen Verzeichnis sind die Dateien *office-world-plan-concepts.lisp* und *office-world-pd-dependencies.lisp* enthalten. Die erste enthält die Funktionsaufrufe für die Erzeugung der LISP-Structures für Plan-Concepts und Dekomposition sowie für den Eintrag der Vorrang-Muster in die BMWB. Die zweite enthält die Funktionsaufrufe für den Eintrag der potentiellen Axiome, potentiellen impliziten Effekte und des Facts in die BMWB.

B.1 Domain-Concepts

Die Beschreibung der Concepts folgt der beschreibungslogischen Notation aus [Patel-Schneider und Swartout, 1993]. Die Beispiele aus Kap. 5 *Der Demonstrations-Prototyp* gehen davon aus, daß alle hier beschriebenen Concepts in allen Views der BMWB enthalten sind. Einzig für die Demonstration der Auswirkungen von Fehlannahmen in Kap. 5.3.2.4 *Pläne für Nick* wird für die View SBUB davon abgewichen. Die Wurzel der Begriffshierarchie ist THING.

[1] vgl. Kap. 3.4.2.7 *Erzeugung des aktiven Teils eines Planungssachbereichs*

domain-concept	$\sqsubseteq$	THING
information	$\sqsubseteq$	*domain-concept*
pot-object	$\sqsubseteq$	*domain-concept*
object	$\sqsubseteq$	*pot-object*
non-existent	$\sqsubseteq$	*pot-object*
to-create	$\sqsubseteq$	*non-existent*
new-var	$\sqsubseteq$	*non-existent*
name-c	$\sqsubseteq$	*domain-concept*
op-sys-c	$\sqsubseteq$	*domain-concept*
os-type-win	$\sqsubseteq$	*op-sys-c*
os-type-unix	$\sqsubseteq$	*op-sys-c*
os-type-mac-os	$\sqsubseteq$	*op-sys-c*
file-type-c	$\sqsubseteq$	*domain-concept*
file-type-text	$\sqsubseteq$	*file-type-c*
file-type-bin	$\sqsubseteq$	*file-type-c*
file-type-tpx	$\sqsubseteq$	*file-type-bin*
file-type-executable	$\sqsubseteq$	*file-type-bin*
fs-object	$\sqsubseteq$	*object*
fs-container	$\sqsubseteq$	*fs-object* $\sqcap$ $\forall$*contains*:*fs-name*
named-fs-object	$\sqsubseteq$	*fs-object* $\sqcap$ $\forall$*fso-named*:*fs-name*
fs-root	$\sqsubseteq$	*fs-container*
local-fs-root	$\doteq$	*fs-root* $\sqcap$ $\forall$*contains*:*local-fs-name*
fs-dir	$\sqsubseteq$	*fs-container* $\sqcap$ *named-fs-object*
local-named-fs-object	$\sqsubseteq$	*named-fs-object* $\sqcap$ $\forall$*contains*:*local-fs-name*
fs-file	$\sqsubseteq$	*named-fs-object* $\sqcap$ *fso-contents*:*information* $\sqcap$ *file-type*:*file-type-c*
text-file	$\doteq$	*fs-file* $\sqcap$ *file-type*:*file-type-text*
bin-file	$\doteq$	*fs-file* $\sqcap$ *file-type*:*file-type-bin*
tpx-file	$\doteq$	*bin-file* $\sqcap$ *file-type*:*file-type-tpx*
exe-file	$\doteq$	*bin-file* $\sqcap$ *file-type*:*file-type-executable*
fs-name	$\sqsubseteq$	*object* $\sqcap$ *fsn-val*:*name-c* $\sqcap$ *fsn-contained-in*:*fs-container* $\sqcap$ $\forall$*designates*:*named-fs-object* $\sqcap$ ≤ 1 *designates*
local-fs-name	$\sqsubseteq$	*fs-name* $\sqcap$ $\forall$*designates*:*local-named-fs-object*
fs-name-for-mount	$\sqsubseteq$	*fs-name*
non_fs-name-for-mount	$\sqsubseteq$	*fs-name*
computer	$\sqsubseteq$	*object* $\sqcap$ *comp-name*:*name-c* $\sqcap$ *fs-root-of-comp*:*fs-root* $\sqcap$ *op-system*:*op-sys-c*
local-comp	$\sqsubseteq$	*computer* $\sqcap$ *fs-root-of-comp*:*local-fs-root*

paperdoc	$\sqsubseteq$	*object* $\sqcap$ *pd-contents*:*information*
application	$\sqsubseteq$	*object*
internet-application	$\sqsubseteq$	*application*
ftp-prog	$\sqsubseteq$	*internet-application*
office-application	$\sqsubseteq$	*application*
tpx	$\sqsubseteq$	*office-application*
printer	$\sqsubseteq$	*object*
user	$\sqsubseteq$	*object*
launched	$\sqsubseteq$	*application*
exports	$\sqsubseteq$	*fs-dir*
online	$\sqsubseteq$	*internet-application*
non_online	$\sqsubseteq$	*internet-application*
down	$\sqsubseteq$	*computer*
non_down	$\sqsubseteq$	*computer*
ftp-binary-mode	$\sqsubseteq$	*ftp-prog*
non_ftp-binary-mode	$\sqsubseteq$	*ftp-prog*

Die folgenden Concepts sind ausschließlich für Werterestriktionen von Rollen oder Attributen von Plan-Concepts notwendig, weil SB-ONE lediglich Concepts, nicht aber Concept-Ausdrücke als Werterestriktion zuläßt (vgl. Kap. 3.2.2.1 *Spezifika der Repräsentation in SB-ONE*).

and_fs-dir_and_local-named-fs-object	$\doteq$	*fs-dir* $\sqcap$ *local-named-fs-object*
and_fs-file_and_local-named-fs-object	$\doteq$	*fs-file* $\sqcap$ *local-named-fs-object*
and_ftp-prog_and_launched_and_non_online	$\doteq$	*ftp-prog* $\sqcap$ *launched* $\sqcap$ *non_online*
and_ftp-prog_and_launched	$\doteq$	*ftp-prog* $\sqcap$ *launched*
and_ftp-binary-mode_and_launched	$\doteq$	*ftp-binary-mode* $\sqcap$ *and_ftp-prog_and_launched*
and_launched_and_non_ftp-binary-mode	$\doteq$	*non_ftp-binary-mode* $\sqcap$ *and_ftp-prog_and_launched*
*and_launched_and*_tpx	$\doteq$	*launched* $\sqcap$ tpx
and_local-named-fs-object_and_tpx-file	$\doteq$	*local-named-fs-object* $\sqcap$ *tpx-file*
and_non_ftp-binary-mode_and_online	$\doteq$	*non_ftp-binary-mode* $\sqcap$ *online*

B.2 Plan-Concepts

Dieser Abschnitt enthält Beschreibungen der Plan-Concepts einschl. Dekompositionen, die in den Beispielen verwendet wurden. Die Plan-Concepts des ersten Teils können für die Plangenerierung eingesetzt werden oder sind Super-Concepts davon. Zusätzlich zur formalen Beschreibung ist hierfür auch eine verbale Beschreibung angegeben. Wegen der Einfachheit der Beschreibungen von unvollständigen Plan-Concepts[2] und Dekompositionen im zweiten und dritten Teil wurde dafür auf eine zu-

[2] vgl. Kap. 4.2.1 *Dekomposition eines Plan-Concepts*

sätzliche verbale Erläuterung verzichtet. Abgesehen davon werden unvollständige Plan-Concepts wie andere Plan-Concepts behandelt. Dekompositionen unterscheiden sich dagegen sowohl in Beschreibung als auch im Eintrag in die BMWB von Plan-Concepts, die für die Plangenerierung verwendet werden können (s.u.).

Der Abschnitt endet mit einer Auflistung von Abkürzungen für Plan-Concept-Namen, die in einigen Abbildungen dieser Arbeit verwendet wurden.

B.2.1 Plan-Concepts für die Plangenerierung

Die in Schriftart Courier gesetzten formalen Beschreibungen der Plan-Concepts genügen der Syntax aus Abb. 3.6 S. 114 und können für die Erzeugung von *bgp-ms-plan-struc*tures verwendet werden, auf deren Basis ein Eintrag von Plan-Concepts in die Benutzermodellierungswissensbasis erfolgt (vgl. Kap. 5.2.1 *Vorgehensweise* und 5.2.3 *Initialer Aufbau einer Benutzermodellierungswissensbasis*).

Das allgemeinste Plan-Concept *plan-concept* hat weder Vorbedingung noch Effekt und ist kein (echtes) Sub-Concept von *plan-concept*. Seine beschreibungslogische Darstellung ist:

$$\textit{plan-concept} \quad \sqsubseteq \quad THING$$

Es hat damit Eigenschaften wie ein Domain-Concept und wird auch wie ein solches definiert und in die BMWB eingetragen.

B.2.1.1 open-file-tpx

Öffnet eine Datei mit Textverarbeitungssystem vom Typ X.
Vorbedingung: ein aktives Textverarbeitungssystem vom Typ X und eine dazu passende lokale Datei.
Effekt: die Datei wird aktives Dokument des Textverarbeitungssystems.

```
:name         'open-file-tpx
:precondition '(and (tpx-file ?f) (local-named-fs-object ?f)
                    (tpx ?app) (launched ?app))
:effect       '(active ?app ?f)
```

B.2.1.2 mount

Mountet ein exportiertes Verzeichnis in das lokale Dateisystem.
Vorbedingung: ein exportiertes Verzeichnis, ein (zum Mounten) freier Dateisystem-Name und der lokale Computer.
Effekt: der Dateisystem-Name ist nicht mehr frei, sondern bezeichnet das exportierte Verzeichnis (das damit lokal wird s.Abschnitt 5.2.2.1.3 *Lokale Dateien*).

```
:name         'mount
:precondition '(and (exports ?export-dir)
                    (fs-name-for-mount ?free-fs-name) (local-comp ?comp))
:effect       '(and (designates ?free-fs-name ?export-dir )
                    (not (fs-name-for-mount ?free-fs-name)))
```

B.2.1.3 mount-win

Windows-Variante von *mount*.
Zusätzliche Vorbedingung: i.w. der lokale Computer arbeitet unter einem Windows-Betriebssystem.

```
:name         'mount-win :supers '(mount)
:precondition '(and (local-fs-root ?loc-root)
                    (contains ?loc-root ?free-fs-name )
                    (os-type-win ?os) (op-system ?comp ?os))
```

B.2.1.4 mount-ux

UNIX-Variante von *mount*.
Zusätzliche Vorbedingung: i.w. der lokale Computer arbeitet unter einem UNIX-Betriebssystem.

```
:name         'mount-ux :supers '(mount)
:precondition '(and (designates ?free-fs-name ?free-dir )
                    (local-named-fs-object ?free-dir) (os-type-unix ?os)
                    (op-system ?comp ?os) (fs-dir ?free-dir) )
:effect        '(not (designates ?free-fs-name ?free-dir ))
```

B.2.1.5 app-print

Aus einem Anwendungsprogramm drucken.
Vorbedingung: ein Anwendungsprogramm mit einem aktiven Dokument, ein Drucker.
Effekt: der Drucker hat (besitzt) ein neues Papierdokument mit dem Inhalt des aktiven Dokuments.

```
:name         'app-print
:precondition '(and (fso-contents ?doc ?doc-c)
                    (application ?app) (active ?app ?doc)
                    (printer ?pr)      (non-existent ?pap-doc))
:effect       '(and (paperdoc ?pap-doc) (pd-contents ?pap-doc ?doc-c)
                    (has ?pr ?pap-doc))
```

B.2.1.6 user-get

Der aktuelle Benutzer übernimmt den Besitz an einem Objekt.
Vorbedingung: ein Objekt, das sich im Besitz von irgend jemandem/etwas befindet.
Effekt: der Benutzer besitzt das Objekt, der ursprüngliche Besitzer dagegen nicht mehr.

```
:name         'user-get
:precondition '(and (has ?oo ?obj) (user ?ag))
:effect       '(and (not (has ?oo ?obj)) (has ?ag ?obj))
```

B.2.1.7 launch

Starten eines Anwendungsprogramms.
Vorbedingung: ein Anwendungsprogramm.
Effekt: das Anwendungsprogramm ist aktiv (gestartet).

```
:name         'launch
:precondition '(application ?app)
:effect       '(launched ?app)
```

B.2.1.8 launch-office-app

Starten einer Office-Anwendung als Spezialisierung von *launch*.
Vorbedingung: ein Office-Programm.
Effekt (geerbt): das Programm ist aktiv (gestartet)

```
:name         'launch-office-app :supers '(launch)
:precondition '(office-application ?app)
```

B.2.1.9 com-copy

Oberbegriff zu allen Datei-Kopier-Operationen: eine Datei wird in ein lokales Verzeichnis kopiert.
Vorbedingung: ein lokales Verzeichnis und eine Datei mit Name, Inhalt und Dateityp.
Effekt: in dem lokalen Verzeichnis befindet sich eine neue Datei mit gleichem Inhalt und Namen wie die alte (wobei ein neuer Datei-Name erzeugt wird, der aber den gleichen Wert hat wie der alte).

```
:name         'com-copy
:precondition '(and
    (fso-contents ?src ?src-info) (designates ?src-name ?src)
    (fsn-val ?src-name ?src-name-val) (file-type ?src ?src-type)
    (fs-file ?src) (fs-dir ?dest-dir) (local-named-fs-object ?dest-dir)
    (non-existent ?dest) (non-existent ?dest-name) (neq ?dest ?dest-name))
:effect       '(and
    (fs-file ?dest) (fso-contents ?dest ?src-info)
    (file-type ?dest ?src-type) (fs-name ?dest-name)
    (designates ?dest-name ?dest) (fsn-val ?dest-name ?src-name-val)
    (contains ?dest-dir ?dest-name))
```

B.2.1.10 copy

Kopieren einer Datei als Spezialisierung von *com-copy*.
Zusätzliche Vorbedingung: die zu kopierende Datei muß sich im lokalen Dateiverzeichnis befinden.

```
:name         'copy :supers '(com-copy)
:precondition '(local-named-fs-object ?src)
```

B.2.1.11 ftp-download-file

FTP-Download als Spezialisierung von *com-copy*.
Zusätzliche Vorbedingung: i.w. ein aktives FTP-Programm und jeweils ein aktives Verzeichnis auf dem lokalen und dem fremden Rechner. Letzteres muß die zu kopierende Datei enthalten.

```
:name         'ftp-download-file :supers '(com-copy)
:precondition '(and
    (neq ?src ?dest) (neq ?src-name ?dest-name) (neq ?src-dir ?dest-dir)
    (contains ?src-dir ?src-name ) (ftp-prog ?prog) (launched ?prog)
    (active-r-dir ?prog ?src-dir) (active-l-dir ?prog ?dest-dir))
```

B.2.1.12 ftp-download-file-bin

FTP-Download im Binär-Modus als Spezialisierung von *ftp-download-file*.
Zusätzliche Vorbedingung: das FTP-Programm muß sich im Binär-Modus befinden und die zu kopierende Datei muß eine Binärdatei sein.

```
:name         'ftp-download-file-bin :supers '(ftp-download-file)
:precondition '(and
    (ftp-binary-mode ?prog) (launched ?prog) (bin-file ?src))
```

B.2.1.13 ftp-download-file-text

FTP-Download im Text-Modus als Spezialisierung von *ftp-download-file*.
Zusätzliche Vorbedingung: das FTP-Programm muß sich im Text-Modus befinden und die zu kopierende Datei muß eine Textdatei sein.

```
:name         'ftp-download-file-text :supers '(ftp-download-file)
:precondition '(and
    (not (ftp-binary-mode ?prog)) (launched ?prog) (text-file ?src))
```

B.2.1.14 remote-copy

Eine Umbenennung des allgemeinen Kopierbegriffs *com-copy*.

```
:name         'remote-copy :supers '(com-copy)
```

B.2.1.15 download-file

Eine weitere Datei-Kopier-Variante: gegenüber *com-copy* wird von den Dateinamen abstrahiert.

```
:name         'download-file
:precondition '(and
    (fso-contents ?src ?src-info) (file-type ?src ?src-type)
    (fs-file ?src) (fs-dir ?dest-dir) (local-named-fs-object ?dest-dir)
    (non-existent ?dest) (non-existent ?dest-name) (neq ?dest ?dest-name))
:effect       '(and
    (fs-file ?dest) (fso-contents ?dest ?src-info)
    (file-type ?dest ?src-type) (fs-name ?dest-name)
    (designates ?dest-name ?dest) (contains ?dest-dir ?dest-name))
```

B.2.1.16 ftp-open-connection

Öffnen einer FTP-Verbindung.
Vorbedingung: Ein aktives FTP-Programm, das nicht *online* sein darf, sowie ein fremder und der lokale Computer mit ihren Dateisystemen.
Effekt: das FTP-Programm ist im Text-Modus (Standardeinstellung) mit dem fremden Computer verbunden (*online*). Die Wurzelverzeichnisse der beiden beteiligten Computer werden die aktiven Fremd- bzw. Lokal-Verzeichnisse.

```
:name         'ftp-open-connection
:precondition '(and
    (ftp-prog ?prog) (launched ?prog) (not (online ?prog))
    (fs-root-of-comp ?remote-comp ?rem-root) (local-fs-root ?loc-root))
:effect       '(and
    (active-r-dir ?prog ?rem-root) (active-l-dir ?prog ?loc-root)
    (connected ?prog ?remote-comp) (online ?prog)
    (not (ftp-binary-mode ?prog)))
```

B.2.1.17 ftp-close-connection

Schließen einer FTP-Verbindung
Vorbedingung: Ein aktives FTP-Programm, das *online* ist, sowie aktive Fremd- bzw. Lokal-Verzeichnisse.
Effekt: das FTP-Programm verläßt den *online*-Modus, ist nicht mehr mit dem fremden Computer verbunden und die aktiven Fremd- bzw. Lokal-Verzeichnisse des FTP-Programms verlieren diese Beziehung zu dem FTP-Programm.

```
:name         'ftp-close-connection
:precondition '(and
    (ftp-prog ?prog) (launched ?prog) (active-r-dir ?prog ?rem-dir)
    (active-l-dir ?prog ?loc-dir) (connected ?prog ?remote-comp))
:effect       '(and
    (not (active-r-dir ?prog ?rem-dir))
    (not (active-l-dir ?prog ?loc-dir))
    (not (connected ?prog ?remote-comp)) (not (online ?prog)))
```

B.2.1.18 ftp-chmode-binary

Binär-Modus eines FTP-Programms einschalten.
Vorbedingung: ein FTP-Programm, das *online* (und damit auch aktiv) ist.
Effekt: das FTP-Programm ist im Binär-Modus.

```
:name         'ftp-chmode-binary
:precondition '(and (ftp-prog ?prog) (launched ?prog))
:effect       '(ftp-binary-mode ?prog)
```

B.2.1.19 ftp-chmode-text

Text-Modus eines FTP-Programms einschalten.
Vorbedingung: ein FTP-Programm, das *online* (und damit auch aktiv) ist.
Effekt: das FTP-Programm ist nicht im Binär-Modus und damit im Text-Modus (Binär-Modus und Text-Modus sind komplementär).

```
:name         'ftp-chmode-text
:precondition '(and (ftp-prog ?prog) (launched ?prog))
:effect       '(not (ftp-binary-mode ?prog))
```

B.2.1.20 ftp-cd

Wechseln des aktiven Fremd-Verzeichnisses eines FTP-Programms.
Vorbedingung: ein FTP-Programm, das mit einem fremden Computer verbunden ist, sowie ein aktives Fremd-Verzeichnis.
Effekt: anstelle des bisherigen Fremd-Verzeichnis wird ein anderes Verzeichnis des fremden Computers aktives Fremd-Verzeichnis.

```
:name         'ftp-cd
:precondition '(and
    (active-r-dir ?prog ?old-dir) (neq ?old-dir ?new-dir)
    (connected ?prog ?remote-comp) (fs-dir ?new-dir)
    (local-fs-object-of-comp ?remote-comp ?new-dir) (ftp-prog ?prog)
    (launched ?prog))
:effect       '(and
    (not (active-r-dir ?prog ?old-dir)) (active-r-dir ?prog ?new-dir))
```

B.2.1.21 ftp-lcd

Wechseln des aktiven Lokal-Verzeichnisses eines FTP-Programms.
Vorbedingung: ein FTP-Programm, sowie ein aktives Lokal-Verzeichnis.
Effekt: anstelle des bisherigen aktiven Lokal-Verzeichnis wird ein anderes lokales Verzeichnis aktives Lokal-Verzeichnis.

```
:name         'ftp-lcd
:precondition '(and
    (active-l-dir ?prog ?old-dir) (neq ?old-dir ?new-dir)
    (fs-dir ?new-dir) (local-named-fs-object ?new-dir) (ftp-prog ?prog)
    (launched ?prog))
:effect       '(and
    (not (active-l-dir ?prog ?old-dir)) (active-l-dir ?prog ?new-dir))
```

B.2.1.22 shutdown-pc

Herunterfahren eines Computers.
Vorbedingung: ein Computer.
Effekt: der Computer ist heruntergefahren.

```
:name         'shutdown-pc
:precondition '(computer ?comp)
:effect       '(down ?comp)
```

B.2.2 Unvollständige Plan-Concepts

Die folgende Liste enthält die unvollständigen Plan-Concepts (vgl. Kap. 4.2.1 *Dekomposition eines Plan-Concepts*), die bei der Plangenerierung keine Berücksichtigung finden. Sie werden lediglich für die Schritte von Dekompositionen eingesetzt oder sind – wie bspw. *incomplete* – lediglich für die Modellierung erforderlich[3]. Dementsprechend sind keine Vorbedingungen oder Effekte modelliert. Einzige Ausnahme ist *select-file-menu-item*, dessen Vorbedingung (der auszuwählende Name) mit der Vorbedingung der Dekompositionen von *ftp-download-file-bin* und *ftp-download-file-text* gekoppelt wird (vgl. dazu das Beispiel in Abb. 4.10 S. 222).

[3] vgl. Kap. 4.6.2 *Repräsentation von Dekompositionen in BGP-MS*

```
:name 'incomplete
:name 'click-button                :supers '(incomplete)
:name 'click-button-ok             :supers '(click-button)
:name 'click-button-start          :supers '(click-button)
:name 'click-button-open           :supers '(click-button)
:name 'click-button-download       :supers '(click-button)
:name 'click-button-transfer       :supers '(click-button)
:name 'push-2key                   :supers '(incomplete)
:name 'push-2key-alt-h             :supers '(push-2key)
:name 'push-2key-alt-o             :supers '(push-2key)
:name 'push-2key-alt-t             :supers '(push-2key)
:name 'push-3key                   :supers '(incomplete)
:name 'push-3key-ctrl-alt-del      :supers '(push-3key)
:name 'select-file-menu-item       :supers '(incomplete)
      :precondition '(name-c ?file-name-val)
:name 'select-menu-item            :supers '(incomplete)
:name 'select-menu-item-terminate  :supers '(select-menu-item)
:name 'select-menu-item-shutdown   :supers '(select-menu-item)
```

B.2.3 Dekompositionen

Die Plan-Concepts *decomposition* und *atomic* sind für die Modellierung von Dekompositionen erforderlich (vgl. wieder Kap. 4.6.2), stellen aber selbst weder Dekompositionen eines anderen Plan-Concepts dar noch werden sie bei der Plangenerierung berücksichtigt. Sie werden deshalb wie "normale" Plan-Concepts gemäß der Syntax aus Abb. 3.6 S. 114 definiert und in die BMWB eingetragen:

```
:name 'decomposition
:name 'atomic
```

Für alle anderen Dekompositionen ist eine Beschreibung nach der Syntax aus Abb. 4.9 S. 222 erforderlich, die insb. die Dekompositions*schritte* berücksichtigt. Analog zur Beschreibung von Plan-Concepts können die Angaben zur Erzeugung von LISP-Structures – hier vom Typ *bgp-ms-decomp-struc* – verwendet werden, auf deren Basis ein Eintrag von Plan-Concepts in die Benutzermodellierungswissensbasis erfolgt (vgl. Kap. 5.2.1 *Vorgehensweise* und 5.2.3 *Initialer Aufbau einer Benutzermodellierungswissensbasis*).

```
:decomposition 'd1-ftp-bin  :of 'ftp-download-file-bin
:steps '((s1 select-file-menu-item (= ?src-name-val ?file-name-val))
         (s2 push-2key-alt-o))
:precedence '((s1 s2))

:decomposition 'd2-ftp-bin  :of 'ftp-download-file-bin
:steps '((s1 select-file-menu-item (= ?src-name-val ?file-name-val))
         (s2 click-button-download) (s3 click-button-transfer))
:precedence '((s1 s2) (s2 s3))

:decomposition 'd3-ftp-bin  :of 'ftp-download-file-bin
:steps '((s1 select-file-menu-item (= ?src-name-val ?file-name-val))
         (s2 push-2key-alt-t))
:precedence '((s1 s2))
:comment "haeufige Fehlannahme: Alt-T anstelle von Alt-O. 2002-02-20"

:decomposition 'd1-ftp-txt  :of 'ftp-download-file-text
:steps '((s1 select-file-menu-item (= ?src-name-val ?file-name-val))
```

```
         (s2 push-2key-alt-o))
:precedence '((s1 s2))

:decomposition 'd2-ftp-txt  :of 'ftp-download-file-text
:steps '((s1 select-file-menu-item (= ?src-name-val ?file-name-val))
         (s2 click-button-download) (s3 click-button-transfer))
:precedence '((s1 s2) (s2 s3))

:decomposition 'd3-ftp-txt  :of 'ftp-download-file-text
:steps '((s1 select-file-menu-item (= ?src-name-val ?file-name-val))
         (s2 push-2key-alt-t))
:precedence '((s1 s2))
:comment "haeufige Fehlannahme: Alt-T anstelle von Alt-O. 2002-02-20"

:decomposition 'd-open-file-tpx :of 'open-file-tpx
:steps '((s1 click-button-open) (s2 select-file-menu-item)
         (s3 click-button-ok))
:precedence '((s1 s2) (s2 s3))

:decomposition 'shutdown-pc-by-key :of 'shutdown-pc
:steps '((s1 push-3key-ctrl-alt-del) (s2 push-2key-alt-h))
:precedence '((s1 s2))

:decomposition 'shutdown-pc-by-menu :of 'shutdown-pc
:steps '((s1 click-button-start) (s2 select-menu-item-terminate)
         (s3 select-menu-item-shutdown) (s4 click-button-ok))
:precedence '((s1 s2) (s2 s3) (s3 s4))
```

B.2.4 Abkürzungen für Plan-Concepts

Im Verlauf der Arbeit wurden folgende Abkürzungen für Plan-Concepts verwendet:

Abkürzung	vollständiger Name
dnload-file	*download-file*
ftp-bin	*ftp-chmode-binary*
ftp-close	*ftp-close-connection*
ftp-dn	*ftp-download-file*
ftp-dn-bin	*ftp-download-file-bin*
ftp-dn-txt	*ftp-download-file-text*
ftp-open	*ftp-open-connection*
ftp-txt	*ftp-chmode-text*
rem-copy	*remote-copy*

B.3 Potentielle Axiome

Potentielle Axiome beschreiben *denkbare* Abhängigkeiten im Sachbereich, die für die Plangenerierung relevant sind (vgl. Kap. 3.4.2.4 *Ableitung von Abhängigkeiten aus dem Benutzermodell* und 5.1.1 *Verarbeitung in der Modellierungsphase*). Abb. 5.8 S. 252 enthält die potentiellen Axiome, die für die

Beispiele eingesetzt wurden, in prädikatenlogischer Notation. Die folgenden Beschreibungen entsprechen der Syntax aus Anhang A.2.4 S. 303. Sie können der Funktion *bgp-ms-define-pd-dependency* als Parameter mitgegeben werden und werden dadurch in die BMWB eingetragen.

```
(:axiom ax-text-file
 :context (exists (file-type-text ?t)
                  (and (file-type ?f ?t) (fs-file ?f)))
 :implies (text-file ?f))

(:axiom ax-tpx-file
 :context (exists (file-type-tpx ?t)
                  (and (file-type ?f ?t) (fs-file ?f)))
 :implies (tpx-file ?f))

(:axiom ax-exe-file
 :context (exists (file-type-executable ?t)
                  (and (file-type ?f ?t) (fs-file ?f)))
 :implies (exe-file ?f))

(:axiom ax-bin-file1
 :context (exe-file ?f)
 :implies (bin-file ?f))

(:axiom ax-bin-file2
 :context (tpx-file ?f)
 :implies (bin-file ?f))

(:axiom ax-wrong-tpx-text-file
 :context (tpx-file ?f)
 :implies (text-file ?f))

(:axiom ax-local-fs-root
 :context (exists (local-comp ?loccomp)
                  (fs-root-of-comp ?loccomp ?locfsroot))
 :implies (local-fs-root  ?locfsroot))

(:axiom ax-local-fs-name2
 :context (exists (contains ?x ?y) (local-named-fs-object ?x))
 :implies (local-fs-name  ?y))

(:axiom ax-local-fs-name1
 :context (exists (contains ?x ?y) (local-fs-root ?x))
 :implies (local-fs-name  ?y))

(:axiom ax-local-named-fs-object
 :context (exists (designates ?y ?z) (local-fs-name ?y))
 :implies (local-named-fs-object ?z))

(:axiom-special ax-local-fs-object-of-comp
 :context (or (fs-root-of-comp ?comp ?obj)
              (exists (designates ?obj-name ?obj)
                      (exists (contains ?dir ?obj-name)
                              (local-fs-object-of-comp ?comp ?dir))))
 :implies (local-fs-object-of-comp ?comp ?obj))
```

```
(:axiom ax-find-to-create-object
 :context (to-create ?obj)
 :implies (non-existent ?obj))

(:axiom ax-create-new-var
 :context (new-var ?obj)
 :implies (non-existent ?obj))
```

B.4 Vorrang-Muster

Durch Vorrang-Muster kann Einfluß auf die Reihenfolge der Vorbedingungskomponenten von Planoperatoren bei der Erzeugung des Planungssachbereichs genommen werden (vgl. Kap. 3.4.2.7 *Erzeugung des aktiven Teils eines Planungssachbereichs*). Da die Beeinflussung auf Vorbedingungskomponenten beschränkt ist, die aus der beschreibungslogischen Repräsentation der Plan-Concepts abgeleitet wurden, können die Muster oft für mehrere Plan-Concepts verwendet werden. Für den Beispielsachbereich werden die folgenden Beschreibungen von Vorrang-Mustern (nach der Syntax aus Abb. 3.24 S. 158) der Funktion *bgp-ms-define-pd-dependency* als Parameter mitgegeben und dadurch – analog zu potentiellen Axiomen – view-unabhängig in die BMWB eingetragen.

```
(:prec-pattern
  (fs-file ?f)       (tpx-file ?f) (local-named-fs-object ?f)
  (application ?app) (tpx ?app)    (launched ?app)
:for (open-file-tpx))

(:prec-pattern
  (exports ?export-dir) (fs-name-for-mount ?free-fs-name)
  (local-fs-root ?loc-root) (contains ?loc-root ?free-fs-name )
  (designates ?free-fs-name ?free-dir )
  (local-named-fs-object ?free-dir) (local-comp ?comp)
  (os-type-win ?os) (os-type-unix ?os) (op-system ?comp ?os)
  (fs-dir ?free-dir)
:for (mount mount-win mount-ux))

(:prec-pattern
  (fso-contents ?doc ?doc-c) (application ?app) (active ?app ?doc)
  (printer ?pr) (non-existent ?pap-doc)
:for (app-print))

(:prec-pattern
  (fso-contents ?src ?src-info) (designates ?src-name ?src )
  (fsn-val ?src-name ?src-name-val) (file-type ?src ?src-type)
  (fs-file ?src) (local-named-fs-object ?src) (fs-dir ?dest-dir)
  (local-named-fs-object ?dest-dir) (non-existent ?dest)
  (non-existent ?dest-name) (neq ?dest ?dest-name)
:for (com-copy copy remote-copy download-file ))

(:prec-pattern
  (ftp-prog ?prog) (launched ?prog) (not (online ?prog))
  (fs-root-of-comp ?remote-comp ?rem-root) (local-fs-root ?loc-root)
:for (ftp-open-connection))
```

```
(:prec-pattern
  (ftp-prog ?prog) (active-r-dir ?prog ?rem-dir)
  (active-l-dir ?prog ?loc-dir) (connected ?prog ?remote-comp)
:for (ftp-close-connection))

(:prec-pattern
  (active-r-dir ?prog ?old-dir) (active-l-dir ?prog ?old-dir)
  (neq ?old-dir ?new-dir) (connected ?prog ?remote-comp)
  (fs-dir ?new-dir) (local-fs-object-of-comp ?remote-comp ?new-dir)
  (local-named-fs-object ?new-dir) (ftp-prog ?prog)
:for (ftp-cd ftp-lcd))

(:prec-pattern
  (neq ?src ?dest) (neq ?src-name ?dest-name) (neq ?src-dir ?dest-dir)
  (neq ?dest ?dest-name) (fso-contents ?src ?src-info)
  (ftp-binary-mode ?prog)  (not (ftp-binary-mode ?prog))
  (designates ?src-name ?src ) (fsn-val ?src-name ?src-name-val)
  (contains ?src-dir ?src-name ) (file-type ?src ?src-type)
  (fs-file ?src) (bin-file ?src) (text-file ?src) (ftp-prog ?prog)
  (active-r-dir ?prog ?src-dir) (fs-dir ?dest-dir)
  (local-named-fs-object ?dest-dir) (active-l-dir ?prog ?dest-dir)
  (non-existent ?dest) (non-existent ?dest-name)
:for (ftp-download-file ftp-download-file-bin ftp-download-file-text))
```

Anhang C

Systemumgebung und ausgewählte Protokolle

Dieser letzte Teil des Anhangs enthält Angaben zu den Software-Komponenten und zum Betrieb des Prototyps sowie die vollständige Ausgabe des Planungssystems zu zwei Beispiel-Planungsproblemen.

C.1 Systemumgebung und -voraussetzungen

Der Demonstrations-Prototyp baut auf den Systemen BGP-MS und UCPOP auf. Beide Systeme sind im Common LISP Quellcode verfügbar. Da diese Arbeit eine Erweiterung von BGP-MS darstellt, sind alle neuen Programmteile ebenfalls in Common LISP implementiert. Als LISP-System wurde CMU Common LISP[1] eingesetzt. Die Angaben zum Betrieb des Gesamtsystems beziehen sich auf eine UNIX-Umgebung. Der Prototyp wurde unter SunOS 5.8 implementiert und betrieben.

Der Programmcode der für den Prototyp neu entwickelten Funktionen (kurz KN-PP-Code) findet sich im elektronischen Anhang dieser Arbeit[2], und zwar in Unterverzeichnissen vom KN-PP-Verzeichnis *Systeme/KN-PP*. Die weiteren, zum Betrieb des Prototyps notwendigen Teilsysteme sowie das CMU-LISP-System sind im Internet frei verfügbar:

BGP-MS	`ftp://ftp.gmd.de/gmd/bgp-ms/`
UCPOP	`ftp://ftp.cs.washington.edu/pub/ai/ucpop41.tar.gz`
CMU Common LISP	`http://www.cons.org/cmucl/`

Für den Betrieb des Prototyps müssen das LISP-System, BGP-MS und UCPOP installiert und der KN-PP-Code – d.h. das KN-PP-Verzeichnis mit allen Unterverzeichnissen – auf die Zielmaschine kopiert sein. Außerdem müssen dem LISP-System die Pfade zu den verschiedenen Teilkomponenten und zu den Beschreibungen der Beispiele bekannt gemacht werden. Dies geschieht über die in Tabelle C.1 angegebenen Umgebungsvariablen des Betriebssystems. Im ursprünglichen BGP-MS-System werden zum Auffinden der Teilkomponenten LISP-Suchpfade eingesetzt, deren Werte beim Start des LISP-Systems aus Systemvariablen bestimmt werden. Die dazu notwendigen Funktionsaufrufe sind in der LISP-Startdatei *init.lisp* enthalten. Für das Planungssystem und den KN-PP-Code wurde diese Technik übernommen und dafür die Suchpfade *ucpop:*, *knpp-src:*, *knpp-plansrc:* und *knpp-aux:* definiert. Tabelle C.1 zeigt die Bedeutung der Suchpfade, die zugehörigen Systemvariablen und für den KN-PP-Code und die Beispiele zusätzlich ihre Lokation im elektronischen Anhang. Das Verzeichnis

[1] eine Common LISP Implementation der Carnegie Mellon University, School of Computer Science, Pittsburgh, PA.

[2] Der elektronische Anhang kann vom miless-Server der Universität Essen (→ http://miless.uni-essen.de/ Suche nach: "Küpper Benutzermodellierung" Detailanzeige) oder über den Autor (Detlef.Kuepper@FH-Aalen.de) bezogen werden.

Systemvariable	LISP-Suchpfad	Pfad zu(m)	Verzeichnis im el. Anhang
CMUCLHOME		LISP-System	
CMUCLLIB		Unterverzeichnis *lib* von CMUCL	
BGPMSHOME	bgp-ms:	BGP-MS	
BGPMSDATA	bgp-ms-data:	Daten für BGP-MS	KN-PP-Demo
UCPOP	ucpop:	Unterverzeichnis *dist* von UCPOP	
KNPP_SRC	knpp-src:	allgemeinen Programmdateien dieser Arbeit	Systeme/KN-PP/lisp
KNPP_PLANSRC	knpp-plansrc:	Programmdateien für die Ankopplung von UCPOP an BGP-MS	Systeme/KN-PP/lisp-plansys
KNPP_AUX	knpp-aux:	nützlichen Funktionen; für den Betrieb nicht unbedingt erforderlich	Systeme/KN-PP/lisp-aux

Tabelle C.1: Systemvariablen und Suchpfade für den Betrieb des Prototyps

Prototyp-Konfig des elektronischen Anhangs enthält ein Beispiel für eine Datei *init.lisp* und für ein Shell-Skipt mit dem die Systemvariablen belegt werden können.

Ist das Unterverzeichnis *bin* von CMUCL im Suchpfad des Betriebssystems enthalten, kann das LISP-System durch Angabe von "`lisp`" von der Kommandozeile oder durch "`M-x run-lisp`" aus emacs aufrufen werden. Der Prototyp kann dann durch

```
(load "knpp-src:start-bgp-ms-pp")
```

geladen werden. Seine Funktionen gehören zum Package *BGP-MS-PROGRAMMER*. Der Ladevorgang beinhaltet nicht das Laden des Planungssystems und der optionalen Hilfsfunktionen, die sich im Unterverzeichnis *lisp-aux* des KN-PP-Verzeichnisses befinden. Bei Bedarf können sie durch

```
(load "knpp-plansrc:start-plansys")    bzw.
(load "knpp-aux:knpp-aux")
```

geladen werden. Die Datei *demo-init-bmwb.lisp* im Verzeichnis *KN-PP-Demo* des elektronischen Anhangs zeigt den Ablauf, der für die Beispiele in Kap. 5 *Der Demonstrations-Prototyp* gewählt wurde (s.a. Fußnote 45 S. 259). Da die Funktionalität des Kernsystems für den initialen Aufbau der Benutzermodellierungswissensbasis ausreicht, wurde zunächst nur dieses sowie – für Demonstrationszwecke – die Hilfsfunktionen geladen. Das Planungssystem kam erst für die Erzeugung der benutzerspezifischen Pläne hinzu.

C.2 UCPOP-Ausgabe des Planungsresultats

Die Ergebnisdarstellung des Planungssystems enthält nicht nur die Schritte eines Plans in der richtigen Reihenfolge, sondern dokumentiert auch die Causal Links ; d.h. die Abhängigkeiten, die zwischen den einzelnen Schritten bestehen[3]. Für jeden Schritt werden die Formeln angegeben, die für die Ausführung des Schritts wahr sein müssen, und es wird angegeben durch welchen Schritt sichergestellt ist, *daß* die Formel wahr ist. Die Referenz wird dabei durch den Rang in der Erzeugungsreihenfolge der Schritte angegeben – nicht durch die Schrittnummer. Im ersten Beispiel unten besagt die Zeile

[3] vgl. Kap. 2.1.1 *Least Commitment Planning*

```
4  -> (launched mytp)
```

in Schritt 3, daß dessen Vorbedingung *launched (mytp)* durch den Schritt erreicht wird, der im Plangenerierungsprozeß als viertes erzeugt wurde. Der Rang in der Erzeugungsreihenfolge wird im Protokoll nach dem Schlüsselwort "*created*" angegeben. Die Vorbedingung wird im Beispiel also durch Schritt 1 erreicht. Eine Referenz 0 bedeutet, daß die zugehörige Formel bereits in der Startsituation wahr war.

Die folgenden beiden Abschnitte enthalten die Ergebnisse, die UCPOP für die Pläne für Oscar (O-C-e und O-C-ne) ausgibt (vgl. a. Kap. 5.3.2.2 *Pläne für Oscar*). Die Ausgabe ist wieder in Kleinschreibweise gewandelt. Bei den Referenzen ist zu beachten, daß UCPOP Vorbedingungen die sich mithilfe von Axiomen ableiten lassen auflöst. D.h. die Darstellung der Planschritte enthält u.U. *nicht* die Formeln, die für den zugehörigen Planoperator als Vorbedingung angegeben sind, sondern Formeln, aus denen die Vorbedingung abgeleitet wurde und die nur *primitive Prädikate*[4] enthalten. Schritt 8 von Plan O-C-ne zeigt bspw., daß durch Schritt 7 sichergestellt wurde, daß die Formeln

```
(fs-file obj-426)
(file-type obj-426 sc-file-type-tpx)
```

wahr sind. Die Vorbedingung von *open-file-tpx* enthält aber keine der beiden Formeln als Komponenten; stattdessen muß für die zu öffnende Datei (hier durch die Konstante obj-426 referenziert) das Prädikat *tpx-file* wahr sein (s. S. 310). Das Planungssystem hat diese Bedingung durch das Axiom

$$\forall x\ (\exists y\ \textit{file-type-tpx}(y) \wedge \textit{file-type}(x,y) \wedge \textit{fs-file}(x)) \rightarrow \textit{tpx-file}(x)$$

aus den beiden obigen Formeln sowie der Formel *file-type-tpx(sc-file-type-tpx)*, die bereits in der Startsituation wahr ist, abgeleitet.

C.2.1 Plan O-C-e für Oscar und Ausgangssituation 1

Der Computer, auf dem sich die gesuchte Datei befindet, exportiert ein Verzeichnis, in dem die Datei indirekt enthalten ist. Oscar kann das Ziel deshalb (u.a.) durch Mounten dieses Verzeichnisses erreichen.

```
Step 1: (launch mytp)          created 4
         0  -> (application mytp)
Step 2: (mount-win comp-pc pc-root sc-os-type-win kn-home fn-pc-drive-p)
                                                                created 3
         0  -> (op-system comp-pc sc-os-type-win)
         0  -> (os-type-win sc-os-type-win)
         0  -> (local-comp comp-pc)
         0  -> (contains pc-root fn-pc-drive-p)
         0  -> (fs-root-of-comp comp-pc pc-root)
         0  -> (local-comp comp-pc)
         0  -> (fs-name-for-mount fn-pc-drive-p)
         0  -> (exports kn-home)
Step 3: (open-file-tpx f-kn-xy-doc mytp)   created 2
         4  -> (launched mytp)
         0  -> (tpx mytp)
         0  -> (fs-root-of-comp comp-pc pc-root)
         0  -> (local-comp comp-pc)
         0  -> (contains pc-root fn-pc-drive-p)
         3  -> (designates fn-pc-drive-p kn-home)
```

[4] vgl. Kap. 3.4.2.3 *Nichtatomare Formeln im Planungssachbereich*

```
         0  -> (contains kn-home fn-kn-otto)
         0  -> (designates fn-kn-otto kn-otto)
         0  -> (contains kn-otto fn-kn-docs)
         0  -> (designates fn-kn-docs kn-docs)
         0  -> (contains kn-docs fn-kn-xy-doc)
         0  -> (designates fn-kn-xy-doc f-kn-xy-doc)
         0  -> (fs-file f-kn-xy-doc)
         0  -> (file-type f-kn-xy-doc sc-file-type-tpx)
         0  -> (file-type-tpx sc-file-type-tpx)
Step 4: (app-print fco-kn-xy-doc loc-pr f-kn-xy-doc dx mytp)   created 1
         0  -> (to-create dx)
         0  -> (printer loc-pr)
         2  -> (active mytp f-kn-xy-doc)
         0  -> (application mytp)
         0  -> (fso-contents f-kn-xy-doc fco-kn-xy-doc)
Step 5: (user-get loc-pr dx sc-user)   created 5
         0  -> (user sc-user)
         1  -> (has loc-pr dx)

Goal  : (exists (?infogoal) (fso-contents f-kn-xy-doc ?infogoal)
         (and (paperdoc dx) (pd-contents dx ?infogoal) (has sc-user dx)))
         5  -> (has sc-user dx)
         1  -> (pd-contents dx fco-kn-xy-doc)
         1  -> (paperdoc dx)
         0  -> (fso-contents f-kn-xy-doc fco-kn-xy-doc)
Facts:
Complete!

UCPOP Stats: Initial terms = 124;   Goals = 3 ;  Success (5 steps)
      Created 211 plans, but explored only 125
      CPU time:    0.0000 sec
      Branching factor:  1.680
      Working Unifies: 931
      Bindings Added: 517 #plan<S=6; O=0; U=0; F=0>
```

C.2.2 Plan O-C-ne für Oscar und Ausgangssituation 2

Der Computer, auf dem sich die gesuchte Datei befindet, exportiert kein Verzeichnis. Oscar kann das Ziel durch Herunterladen der Datei via FTP erreichen.

```
Step 1 : (launch mytp)          created 9
          0  -> (application mytp)
Step 2 : (launch myftp)         created 4
          0  -> (application myftp)
Step 3 : (ftp-open-connection myftp comp-kn pc-root kn-root)   created 6
          0  -> (fs-root-of-comp comp-pc pc-root)
          0  -> (local-comp comp-pc)
          0  -> (fs-root-of-comp comp-kn kn-root)
          0  -> (not (online myftp))
          4  -> (launched myftp)
          0  -> (ftp-prog myftp)
Step 4 : (ftp-lcd myftp pc-drive-c pc-root)   created 7
          4  -> (launched myftp)
          0  -> (ftp-prog myftp)
```

```
          0  -> (fs-root-of-comp comp-pc pc-root)
          0  -> (local-comp comp-pc)
          0  -> (contains pc-root fn-pc-drive-c)
          0  -> (designates fn-pc-drive-c pc-drive-c)
          0  -> (fs-dir pc-drive-c)
          6  -> (active-l-dir myftp pc-root)
Step 5 : (ftp-cd comp-kn myftp kn-docs kn-root)   created 5
          4  -> (launched myftp)
          0  -> (ftp-prog myftp)
          0  -> (fs-root-of-comp comp-kn kn-root)
          0  -> (contains kn-root fn-kn-home)
          0  -> (designates fn-kn-home kn-home)
          0  -> (contains kn-home fn-kn-otto)
          0  -> (designates fn-kn-otto kn-otto)
          0  -> (contains kn-otto fn-kn-docs)
          0  -> (designates fn-kn-docs kn-docs)
          0  -> (fs-dir kn-docs)
          6  -> (connected myftp comp-kn)
          6  -> (active-r-dir myftp kn-root)
Step 6 : (ftp-chmode-binary myftp)   created 3
          0  -> (ftp-prog myftp)
          4  -> (launched myftp)
Step 7 : (ftp-download-file-bin myftp f-kn-xy-doc pc-drive-c fn-kn-xy-doc
 sc-file-type-tpx obj-426 kn-docs fco-kn-xy-doc obj-427 str-xy-doc)
                                                                 created 2
          4  -> (launched myftp)
          7  -> (active-l-dir myftp pc-drive-c)
          0  -> (fs-root-of-comp comp-pc pc-root)
          0  -> (local-comp comp-pc)
          0  -> (contains pc-root fn-pc-drive-c)
          0  -> (designates fn-pc-drive-c pc-drive-c)
          0  -> (fs-dir pc-drive-c)
          5  -> (active-r-dir myftp kn-docs)
          0  -> (fs-file f-kn-xy-doc)
          0  -> (file-type f-kn-xy-doc sc-file-type-tpx)
          0  -> (file-type-tpx sc-file-type-tpx)
          0  -> (file-type f-kn-xy-doc sc-file-type-tpx)
          0  -> (contains kn-docs fn-kn-xy-doc)
          0  -> (fsn-val fn-kn-xy-doc str-xy-doc)
          0  -> (designates fn-kn-xy-doc f-kn-xy-doc)
          3  -> (ftp-binary-mode myftp)
          0  -> (fso-contents f-kn-xy-doc fco-kn-xy-doc)
Step 8 : (open-file-tpx obj-426 mytp)   created 8
          9  -> (launched mytp)
          0  -> (tpx mytp)
          0  -> (fs-root-of-comp comp-pc pc-root)
          0  -> (local-comp comp-pc)
          0  -> (contains pc-root fn-pc-drive-c)
          0  -> (designates fn-pc-drive-c pc-drive-c)
          2  -> (contains pc-drive-c obj-427)
          2  -> (designates obj-427 obj-426)
          2  -> (fs-file obj-426)
          2  -> (file-type obj-426 sc-file-type-tpx)
          0  -> (file-type-tpx sc-file-type-tpx)
```

```
Step 9 : (app-print fco-kn-xy-doc loc-pr obj-426 dx mytp)   created 1
          0  -> (to-create dx)
          0  -> (printer loc-pr)
          8  -> (active mytp obj-426)
          0  -> (application mytp)
          2  -> (fso-contents obj-426 fco-kn-xy-doc)
Step 10: (user-get loc-pr dx sc-user)   created 10
          0  -> (user sc-user)
          1  -> (has loc-pr dx)

goal   :(exists (?infogoal) (fso-contents f-kn-xy-doc ?infogoal)
         (and (paperdoc dx) (pd-contents dx ?infogoal) (has sc-user dx)))
         10 -> (has sc-user dx)
         1  -> (pd-contents dx fco-kn-xy-doc)
         1  -> (paperdoc dx)
         0  -> (fso-contents f-kn-xy-doc fco-kn-xy-doc)
Facts:
Complete!

UCPOP Stats: Initial terms = 123;   Goals = 3 ;  Success (10 steps)
      Created 4401 plans, but explored only 2440
      CPU time:    0.0000 sec
      Branching factor:  1.793
      Working Unifies: 16077
      Bindings Added: 12208#plan<S=11; O=0; U=0; F=0>
```

Literaturverzeichnis

[AAAI, 1986] *Proc. of 5th National Conference on Artificial Intelligence*, Philadelphia, PA, 01-11/15, 1986.

[AAAI, 1991] *Proc. of 9th National Conference on Artificial Intelligence*, Anaheim, CA, 07-14/19, 1991.

[AAAI, 1996] *Proc. of 13th National Conference on Artificial Intelligence*, Portland, OR, 08-04/08, 1996.

[AAAI, 1998] *Proc. of 15th National Conference on Artificial Intelligence*, Madison, WI, 07-26/30, 1998.

[ACL, 1985] *Proc. of 23th Annual Meeting of the Association for Computational Linguistics*, Chicago, IL, 1985.

[AIPS, 1994] K. Hammond (Hrsg.). *Proc. of the Second International Conference on Artificial Intelligence Planning Systems*, Chicago, IL, 06-13/15, 1994.

[AIPS, 1996] B. Drabble (Hrsg.). *Proc. of the Third International Conference on Artificial Intelligence Planning Systems*, Edinbourgh, GBR, 05-29/31, 1996.

[Allen *et al.*, 1990] J. F. Allen, J. Hendler und A. Tate (Hrsg.). *Readings in Planning*. Morgan Kaufmann, San Mateo, CA, 1990.

[Allen *et al.*, 1991] J. F. Allen, H. A. Kautz, R. N. Pelavin und J. D. Tenenberg (Hrsg.). *Reasoning about Plans*. Kaufmann, San Mateo, CA, 1991.

[Allen *et al.*, 1995] J. F. Allen, L. K. Schubert, G. Ferguson, P. Heeman, C. H. Hwang, T. Kato, M. Light, N. G. Martin, B. W. Miller, M. Poesio und D. R. Traum. The TRAINS Project: A case study in building a conversational planning agent. *Journal of Experimental and Theoretical Artificial Intelligence*, 7:7–48, 1995.

[Allen und Ferguson, 1994] J. F. Allen und G. Ferguson. Events and actions in interval temporal logic. *Journal of Logic and Computation*, 4(5):531–579, 1994.

[Allen und Perrault, 1980] J. F. Allen und C. R. Perrault. Analyzing intention in utterances. *Artificial Intelligence*, 15:143–178, 1980.

[Allen und Schubert, 1991] J. F. Allen und L. K. Schubert. The TRAINS projekt. Memo 91-1, Univ. of Rochester, Rochester, NY, 1991.

[Allen, 1979] J. F. Allen. *A Plan–Based Approach to Speech Act Recognition*. PhD thesis, Department of Computer Science, University of Toronto, Toronto, CAN, 1979.

[Allen, 1984] J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, 1984.

[Anderson *et al.*, 1985] J. R. Anderson, C. F. Boyle und G. Yost. The geometry tutor. In *9th International Joint Conference on Artificial Intelligence*, 1–7, Los Angeles, CA, 08-18/23, 1985.

[André, 1995] E. André. *Ein planbasierter Ansatz zur Generierung multimedialer Präsentationen*. Infix, St. Augustin, DEU, 1995.

[Appelt und Kronfeld, 1987] D. E. Appelt und A. Kronfeld. A computational model of referring. In IJCAI [1987], 640–647.

[Appelt, 1985] D. E. Appelt. *Planning English Sentences*. Cambridge University Press, Cambridge, GBR, 1985.

[Artale *et al.*, 1996a] A. Artale, E. Franconi und N. Guarino. Open problems with part–whole relations. In *Proc. of the 1996 International Workshop on Description Logics*, Cambridge, MA, Nov., 1996.

[Artale *et al.*, 1996b] A. Artale, E. Franconi, N. Guarino und L. Pazzi. Part–whole relations in object–centered systems: An overview. *Data & Knowledge Engineering*, 20(3):347–383, 1996.

[Austin, 1962] J. L. Austin. *How To Do Things with Words*. Harvard University Press, Cambridge, MA, 1962.

[Baader *et al.*, 1990] F. Baader, H.-J. Bürckert, J. Heinsohn, B. Hollunder, J. Müller, B. Nebel, W. Nutt und H.-J. Profitlich. Terminological knowledge representation: A prosposal for a terminological logic. Memo TM-90-04, DFKI GmbH, Kaiserslautern, DEU, 1990.

[Baader, 1996] F. Baader. Logik–basierte Wissensrepräsentation. *Künstliche Intelligenz*, (3):8–16, 1996.

[Bacchus und Yang, 1994] F. Bacchus und Q. Yang. Downward refinement and the efficiency of hierarchical problem solving. *Artificial Intelligence*, 71:43–100, 1994.

[Balkanski, 1990] C. T. Balkanski. Modelling act–type relations in collaborative activity. Tech.Rep 23-90, Harvard Univ., Cambridge, MA, 1990.

[Ballim und Wilks, 1991] A. Ballim und Y. Wilks. Beliefs, stereotypes and dynamic agent modeling. *User Modeling and User-Adapted Interaction*, 1(1):33–65, 1991.

[Barrett *et al.*, 1995] A. Barrett, D. Christiansen, M. Friedman, C. Kwok, K. Golden, S. Penberthy, Y. Sun und D. S. Weld. UCPOP User's Manual (V.4.0). Tech.Rep 93-09-06d, Univ. of Washington, Seattle, WA, 1995.

[Bauer *et al.*, 1993] M. Bauer, S. Biundo, D. Dengler, J. Koehler und G. Paul. PHI: A logic–based tool for intelligent help systems. In IJCAI [1993], 460–466.

[Bauer *et al.*, 1996] M. Bauer, S. Biundo, D. Dengler, H. Feibel, J. Koehler und G. Paul. RAP – Reasoning about plans – A progress report. In *Proc. of 10. Workshop 'Planen und Konfigurieren' (PuK96)*, 193–204, Bonn, DEU, 04-15/17, 1996.

[Bauer und Dengler, 1999] M. Bauer und D. Dengler. TrIAs: Trainable information assistants for cooperative problem solving. In *Proc. of Agents'99*, 260–267, 1999.

[Bauer, 1988] J. Bauer. *Konzepte und Prototypen interaktiver Hilfesysteme*. Dissertation, Univ. Stuttgart, DEU, 1988.

[Bauer, 1996] M. Bauer. *Ein evidenztheoretischer Ansatz zur Planerkennung*. Dissertation, Univ. des Saarlandes, Saarbrücken, DEU, 1996.

[Bäckström und Jonsson, 1995] C. Bäckström und P. Jonsson. Planning with abstraction hierarchies can be exponentially less efficient. In IJCAI [1995], 1599–1604.

[Berthold und Jameson, 1999] A. Berthold und A. Jameson. Interpreting symptoms of cognitive load in speech input. In *User Modeling: Proceedings of the 7th International Conference, UM'99*, 235–244, Banff, CAN, 06-20/24, 1999.

[Bibel, 1986] W. Bibel. A deductive solution for plan generation. *New Generation Computing*, 4:115–132, 1986.

[Biundo *et al.*, 1992] S. Biundo, D. Dengler und J. Köhler. Deductive planning and plan reuse in a command language environment. In ECAI [1992], 628–632.

[Biundo, 1993] S. Biundo. Present–day deductive planning. In *Current Trends in AI Planning*, 1–5, Vadstena, SWE, 12-09/11, 1993.

[Blum und Furst, 1995] A. L. Blum und M. L. Furst. Fast planning through planning graph analysis. In IJCAI [1995], 1636–1642.

[Blum und Furst, 1997] A. L. Blum und M. L. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90(1-2):281–300, 1997.

[Borgida *et al.*, 1989] A. Borgida, R. J. Brachman, D. L. McGuinness und L. A. Resnick. CLASSIC: A structural data model for objects. In *Proc. of the 1989 ACM SIGMOD International Conference on Management of Data*, 59–67, June, 1989.

[Brachman *et al.*, 1991] R. Brachman, D. McGuinness, P. Patel-Schneider, L. Resnik und A. Borgida. Living with CLASSIC: When and how to use a KL–ONE–like language. In Sowa [1991], 401–456.

[Brachman und Schmolze, 1985] R. J. Brachman und J. G. Schmolze. An overview of the KL–ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[Brachman, 1977] R. J. Brachman. *A Structural Paradigm for Representing Knowledge*. PhD thesis, Harvard University, Cambridge, MA, 1977.

[Brachman, 1979] R. J. Brachman. On the espistemological status of semantic networks. In N. V. Findler (Hrsg.), *Associative Networks: Representation and Use of Knowledge by Computers*, 3–50. Academic Press, New York, NY, 1979.

[Bratman, 1983] M. E. Bratman. Taking plans seriously. *Social Theory and Practice*, 9:271–287, 1983.

[Bratman, 1987] M. E. Bratman. *Intention, Plans, and Practical Reason*. Harvard University Press, Cambridge, MA, 1987.

[Breuker, 1990] J. Breuker (Hrsg.). *EUROHELP: Developing Intelligent Help Systems*. EC, Kopenhagen, DNK, 1990.

[Bro, 1996] *Der Brockhaus – Die Enzyklopädie in 24 Bänden*. Brockhaus, Leipzig, Mannheim, DEU, 1996.

[Brown und Burton, 1978] J. S. Brown und R. R. Burton. Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science*, 2:155–192, 1978.

[Buchberger *et al.*, 1982] E. Buchberger, I. Steinacker, R. Trappl, H. Trost und E. Leinfellner. VIE–LANG – A German language understanding system. In *Proc. of of 6th European Meeting on Cybernetics and Systems Research*, Wien, AUT, 04-13/16, 1982.

[Bylander, 1994] T. Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69:165–204, 1994.

[Calistri-Yeh, 1991] R. J. Calistri-Yeh. Utilizing user models to handle ambiguity and misconceptions in robust plan recognition. *User Modeling and User-Adapted Interaction*, 1(4):289–322, 1991.

[Carberry *et al.*, 1992] S. Carberry, Z. Kazi und L. Lambert. Modeling discourse, problem–solving, and domain goals incrementally in task–oriented dialogue. In *Proc. of 3rd International Conference on User Modeling*, 192–201, Dagstuhl Castle, DEU, 08-10/13, 1992.

[Carberry, 1985] S. Carberry. *Pragmatic Modeling in Information System Interfaces*. PhD thesis, Univ. of Delaware, Newark, USA, 1985.

[Carberry, 1986] S. Carberry. User models: The problem of disparity. In *Proc. of 11th International Conference on Computational Linguistics*, 29–34, Bonn, DEU, 1986.

[Carberry, 1990a] S. Carberry. *Plan Recognition in Natural Language Dialogue*. MIT Press, Cambridge, MA, 1990.

[Carberry, 1990b] S. Carberry. Incorporating default inferences into plan recognition. In *Proc. of 8th National Conference on Artificial Intelligence*, 471–478, Boston, MA, 1990.

[Chien *et al.*, 1995] S. Chien, R. H. Jr., X. Wang, T. Estlin, K. Fayyad und H. Mortenson. Why real–world planning is difficult: A tale of two applications. In EWSP [1995], 287–298.

[Chin, 1989] D. N. Chin. KNOME: Modeling what the user knows in UC. In Kobsa und Wahlster [1989], 74–107.

[Chu-Carroll und Carberry, 1998] J. Chu-Carroll und S. Carberry. Collaborative response generation in planning dialogues. *Computational Linguistics*, 24(3):355–400, 1998.

[Clark und Clark, 1968] H. H. Clark und E. V. Clark. Semantic distinctions and memory for complex sentences. *The Quarterly Journal of Experimental Psychology*, 20:129–138, 1968.

[Cohen *et al.*, 1982] P. R. Cohen, R. C. Perrault und J. F. Allen. Beyond question answering. In Lehnert und Ringle [1982], 245–274.

[Cohen *et al.*, 1990] P. R. Cohen, J. Morgan und M. E. Pollack (Hrsg.). *Intentions in Communication*. MIT Press, Cambridge, MA, 1990.

[Cohen und Levesque, 1985] P. R. Cohen und H. J. Levesque. Speech acts and rationality. In ACL [1985], 49–59.

[Cohen, 1978] P. R. Cohen. *On knowing what to say: Planning speech acts*. PhD thesis, Dept. of Computer Science, Univ. of Toronto, Toronto, CAN, 1978.

[Currie und Tate, 1991] K. Currie und A. Tate. O–Plan: The open planning architecture. *Artificial Intelligence*, 52:49–86, 1991.

[Dale und Reiter, 1995] R. Dale und E. Reiter. Computational interpretations of the Gricean Maxims in the generation of referring expressions. *Cognitive Science*, 19(2):233–263, 1995.

[Davidson, 1980] D. Davidson. Intending. In D. Davidson (Hrsg.), *Essays on Actions and Events*, 83–102. Oxford University Press, New York, NY, 1980.

[Dean und Boddy, 1988] T. Dean und M. Boddy. An analysis of time–dependent planning. In *Proc. of 7th National Conference on Artificial Intelligence*, 49–54, St. Paul, MN, 1988.

[Decker *et al.*, 1999] S. Decker, M. Erdmann, D. Fensel und R. Studer. Ontobroker: Ontology based access to distributed and semi–structured information. In *Semantic Issues in Multimedia Systems, Proc. of 8th Working Conference on Database Semantics*, 351–369, Rotorua, NZL, 01-04/08, 1999.

[Dempster, 1967] A. Dempster. Upper an lower probabilities induced by a multivalued mapping. *Annals of Mathematical Statistics*, S. 88, 1967.

[Dengler, 1994] D. Dengler. An adaptive deductive planning system. In ECAI [1994], 610–614.

[Dengler, 1996] D. Dengler. Customized plans transmitted by flexible refinement. In *Proc. of 12th European Conference on Artificial Intelligence*, 609–613, Budapest, HUN, 1996.

[DePallo, 2000] M. DePallo. *AARP National Survey on Consumer Preparedness and E–Commerce: A Survey of Computer Users Age 45 and Older*. AARP, Washington, DC, 2000.

[desJardins *et al.*, 1999] M. E. desJardins, E. H. Durfee, C. L. Ortiz, Jr. und M. J. Wolverton. A Survey of research in distributed, continual planning. *AI Magazine*, 20(4):13–22, 1999.

[Devanbu und Litman, 1991] P. T. Devanbu und D. J. Litman. Plan–based terminological reasoning. In *Principles of Knowledge Representation and Reasoning*, 128–138, Cambridge, MA, 04-22/25, 1991.

[Devanbu und Litman, 1996] P. Devanbu und D. J. Litman. Taxonomic plan reasoning. *Artificial Intelligence*, 84(July):1–35, 1996.

[Dixon, 1982] P. Dixon. Plans and written directions for complex tasks. *Journal of Verbal Learning and Verbal Behavior*, 21:70–84, 1982.

[Doane *et al.*, 1992] S. M. Doane, S. M. Mannes, W. Kintsch und P. G. Polson. Modeling user action planning: A comprehension based approach. *User Modeling and User-Adapted Interaction*, 2(3):249–285, 1992.

[Donini *et al.*, 1997] F. M. Donini, M. Lenzerini, D. Nardi und W. Nutt. The complexity of concept languages. *Information and Computation*, 134:1–58, 1997.

[Drabble und Tate, 1995] B. Drabble und A. Tate. O–PLAN: A situated planning agent. In EWSP [1995], 247–259.

[ECAI, 1992] B. Neumann (Hrsg.). *Proc. of 10th European Conference on Artificial Intelligence*, Wien, AUT, Aug., 1992.

[ECAI, 1994] A. G. Cohn (Hrsg.). *Proc. of 11th European Conference on Artificial Intelligence*, Amsterdam, NLD, 08-08/12, 1994.

[Erol *et al.*, 1994a] K. Erol, J. Hendler und D. S. Nau. Semantics for hierarchical task–network planning. Tech.Rep CS-TR-3239, Computer Science Dept., University of Maryland, 1994.

[Erol *et al.*, 1994b] K. Erol, J. Hendler und D. S. Nau. UMCP: A sound and complete procedure for hierarchical task–network planning. In AIPS [1994], 249–254.

[Erol *et al.*, 1994c] K. Erol, D. S. Nau und J. Hendler. HTN planning: Complexity and expressivity. In *Proc. of 12th National Conference on Artificial Intelligence*, 1123–1128, Seattle, WA, 08-01/04, 1994.

[Erol *et al.*, 1995] K. Erol, D. S. Nau und V. Subrahmanian. Complexity, decidability and undecidability results for domain–independent planning. *Artificial Intelligence*, 76(1-2 July):75–88, 1995.

[Etzioni *et al.*, 1992] O. Etzioni, S. Hanks, D. S. Weld, D. Draper, N. Lesh und M. Williamson. An approach to planning with incomplete information. In KR [1992], 115–125.

[Etzioni und Weld, 1994] O. Etzioni und D. S. Weld. A softbot–based interface to the internet. *Communications of the ACM*, 37(7):72–76, 1994.

[EWSP, 1995] M. Ghallab und A. Milani (Hrsg.). *New Directions in AI Planning*, Assisi, ITA, 09-27/29, 1995.

[Ferguson *et al.*, 1996a] G. Ferguson, J. F. Allen und B. Miller. TRAINS–95: Towards a mixed–initiative planning assistant. In AIPS [1996], 70–77.

[Ferguson *et al.*, 1996b] G. M. Ferguson, J. F. Allen, B. W. Miller und E. K. Ringger. The design and implementation of the TRAINS–96 system: A prototype mixed–initiative planning assistant. Memo 96-5, Univ. of Rochester, Rochester, NY, 1996.

[Ferguson und Allen, 1994] G. Ferguson und J. F. Allen. Arguing about plans: Plan representation and reasoning for mixed–initiative planning. In AIPS [1994], 43–48.

[Ferguson und Allen, 1998] G. Ferguson und J. F. Allen. TRIPS: An integrated intelligent problem–solving assistant. In AAAI [1998], 567–573.

[Ferguson, 1994] G. Ferguson. *Knowledge Representation and Reasoning for Mixed–Initiative Planning*. PhD thesis, Univ. Rochester, Rochester, NY, 1994.

[Fiedler, 1999] A. Fiedler. Using a cognitive architecture to plan dialogs for the adaptive explanation of proofs. In *Proc. of 16th International Joint Conference on Artificial Intelligence*, 358–363, Stockholm, SWE, 07-31/08-06, 1999.

[Fikes und Nilsson, 1971] R. E. Fikes und N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[Finin, 1983] T. W. Finin. Providing help and advice in task oriented systems. In *Proc. of 8th International Joint Conference on Artificial Intelligence*, 176–178, Karlsruhe, DEU, 08-08/12, 1983.

[Finin, 1989] T. W. Finin. GUMS – A general user modeling shell. In Kobsa und Wahlster [1989], 411–430.

[For, 1999] *The Privacy Best Practise*. Forrester Research, Cambridge, MA, 1999.

[Georgeff und Lansky, 1987] M. P. Georgeff und A. L. Lansky. Reactive reasoning and planning. In *Proc. of 6th National Conference on Artificial Intelligence*, 677–682, Seattle, WA, 1987.

[Georgeff, 1987] M. P. Georgeff. Planning. *Annual Review of Computer Science*, 2:359–400, 1987.

[Gerevini und Schubert, 1996] A. Gerevini und L. K. Schubert. Accelerating partial–order planners: Some techniques for effective search control and pruning. *Journal of Artificial Intelligence Research*, 5:95–137, 1996.

[Gerstl und Pribbenow, 1996] P. Gerstl und S. Pribbenow. A conceptual theory of part–whole relations and its applications. *Data & Knowledge Engineering*, 20(3):305–322, 1996.

[Gertner *et al.*, 1998] A. S. Gertner, C. Conati und K. VanLehn. Procedural help in Andes: Generating hints using a bayesian network student model. In AAAI [1998], 106–111.

[Ghallab *et al.*, 1998] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. S. Weld und D. Wilkins. PDDL – The planning domain definition language. Tech.Rep CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, Yale, 1998.

[Gilbert und Jirotka, 1990] G. Gilbert und M. Jirotka. Planning procedural advice. *Interacting with Computers*, 2:313–329, 1990.

[Golden und Weld, 1996] K. Golden und D. S. Weld. Representing sensing actions: The middle ground revisited. In *Principles of Knowledge Representation and Reasoning: Proc. of 5th International Conference (KR'96)*, 174–185, Cambridge, MA, 11-05/08, 1996.

[Grabowski, 1991] J. Grabowski. *Der propositionale Ansatz der Textverständlichkeit: Kohärenz, Interessantheit und Behalten*. Aschendorff, Münster, DEU, 1991.

[Graesser *et al.*, 1980] A. Graesser, S. Roberston, E. Lovelace und D. Swineheart. Answers to why questions expose the organization of story plot and predict recall of actions. *Journal of Verbal Learning and Verbal Behavior*, 19:110–119, 1980.

[Grice, 1971] H. Grice. Intention and uncertainty. In *Proc. of British Academy Vol. 57*, 263–279. Oxford University Press, 1971.

[Grice, 1975] H. P. Grice. Logic and conversation. In P. Cole und J. Morgan (Hrsg.), *Syntax and Semantics*, 41–58. Academic Press, New York, NY, 1975.

[Grosz und Kraus, 1996] B. J. Grosz und S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.

[Grosz und Kraus, 1999] B. J. Grosz und S. Kraus. The evolution of SharedPlans. In M. Wooldridge und A. Rao (Hrsg.), *Foundations and Theories of Rational Agency*, 227–262. Kluwer Academic Publishers, Dordrecht, NLD, 1999.

[Grosz und Sidner, 1990] B. J. Grosz und C. L. Sidner. Plans for discourse. In Cohen et al. [1990], 417–444.

[Grosz, 1981] B. Grosz. Focusing and description in natural languege dialogues. In A. Joshi, B. Webber und I.Sag (Hrsg.), *Elements of Discourse Understanding*, 84–105. Cambridge University Press, Cambridge, 1981.

[Haas, 1987] A. W. Haas. The case for domain–specific frame axioms. In *Proc. of the 1987 Workshop: The Frame Problem in Artificial Intelligence*, 343–348, 04-12/15, 1987.

[Hayes, 1980] P. J. Hayes. The logic of frames. In D. Metzing (Hrsg.), *Frame Conceptions and Text Understanding*. DeGruyter, Berlin, DEU, 1980.

[Heinsohn *et al.*, 1994] J. Heinsohn, B. Nebel, D. Kudenko und H.-J. Profitlich. An empirical analysis of terminological representation systems. *Artificial Intelligence*, 68(2):367–397, 1994.

[Hendrix, 1975] G. G. Hendrix. Expanding the utility of semantic networks through partitioning. In *Proc. of 4th International Joint Conference on Artificial Intelligence*, 115–121, Tiblisi, GEO, 09-03/08, 1975.

[Hertzberg, 1989] J. Hertzberg. *Planen: Einführung in die Planerstellungsmethoden der Künstlichen Intelligenz*. BI-Wissenschaftsverlag, Mannheim, DEU, 1989.

[Hertzberg, 1993] J. Hertzberg. KI–Handlungsplanung – Woran wir arbeiten, und woran wir arbeiten sollten. In *Grundlagen und Anwendungen der Künstlichen Intelligenz. 17. Fachtagung für Künstliche Intelligenz*, 3–27, Berlin, DEU, 09-13/16, 1993.

[Hintikka, 1962] J. Hintikka. *Knowledge and Belief*. Cornell Univ. Press, Ithica, NY, 1962.

[Hintikka, 1969] J. Hintikka. Semantics for propositional attitutes. In J. D. et al. (Hrsg.), *Philosophical Logic*, 21–45. Reidel, Dordrecht, NLD, 1969.

[Hobbs und Moore, 1985] J. Hobbs und R. Moore (Hrsg.). *Formal Theories of the Commensense World*. Ablex Publishing, Norwood, NJ, 1985.

[Hobbs, 1982] J. R. Hobbs. Towards an understanding of coherance in discourse. In Lehnert und Ringle [1982], 223–243.

[Hobbs, 1985] J. R. Hobbs. Ontological promiscuity. In ACL [1985], 61–69.

[Hollunder und Nutt, 1990] B. Hollunder und W. Nutt. Subsumption algorithms for concept languages. Tech.Rep RR-90-04, DFKI, Kaiserslautern, DEU, 1990.

[Hopcroft und Ullman, 1979] J. E. Hopcroft und J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, MA, 1979.

[Hoppe-Graff, 1984] S. Hoppe-Graff. Verstehen als kognitiver Prozeß. *Zeitschrift für Literaturwissenschaft und Linguistik*, 14(55):10–37, 1984.

[Horacek, 1994] H. Horacek. How to avoid explaining obvious things (without omitting central information). In ECAI [1994], 520–524.

[Horacek, 1997] H. Horacek. A model for adapting explanations to the user's likely inferences. *User Modeling and User-Adapted Interaction*, 7(1):1–55, 1997.

[Hovy, 1988] E. H. Hovy. Planning coherent multisentential text. In *Proc. of 26th Annual Meeting of the Association for Computational Linguistics*, 163–169, 1988.

[Hovy, 1991] E. H. Hovy. Approaches to the planning of coherent text. In C. L. Paris, W. R. Swartout und W. C. Mann (Hrsg.), *Natural Language in Artificial Intelligence and Computational Linguistics*, 83–102. Kluwer Academic Publishers, Boston, MA, 1991.

[Hovy, 1993] E. H. Hovy. Automated discourse generation using discourse structure relations. *Artificial Intelligence*, 63(1-2):341–386, 1993.

[Hustadt, 1995] U. Hustadt. Introducing epistemic operators into a description logic. In A. Laux und H. Wansing (Hrsg.), *Knowledge and Belief in Philosophy and Artificial Intelligence*, 65–86. Akademie-Verlag, Berlin, DEU, 1995.

[IJCAI, 1987] *Proc. of 10th International Joint Conference on Artificial Intelligence*, Milano, ITA, 08-23/28, 1987.

[IJCAI, 1993] *Proc. of 13th International Joint Conference on Artificial Intelligence*, Chambery, FRA, 08-28/09-03, 1993.

[IJCAI, 1995] *Proc. of 14th International Joint Conference on Artificial Intelligence*, Montreal, CAN, 08-22/25, 1995.

[Iris *et al.*, 1988] M. Iris, B. Lutowitz und M. Evens. Problems of the part–whole relation. In M. W. Evens (Hrsg.), *Relational Models of the Lexicon: Representing Knowledge in Semantic Networks*, 261–288. Cambridge University Press, Cambridge, 1988.

[Jameson *et al.*, 1994] A. Jameson, B. Kipper, A. Ndiaye, R. Schäfer, T. Weis und D. Zimmermann. Cooperating to be noncooperative: The dialog system PRACMA. In *KI–94: Advances in Artificial Intelligence, 18th Annual German Conference on Artificial Intelligence*, 106–117, Saarbrücken, DEU, 09-18/09-23, 1994.

[Jarke und Koch, 1984] M. Jarke und J. Koch. Query optimization in database systems. *ACM Computing Surveys*, 16:111–152, 1984.

[Jensen und Wirth, 1985] K. Jensen und N. Wirth. *PASCAL User Manual and Report (3rd Edition)*. Springer, New York, Berlin, Heidelberg, Tokyo, 1985.

[Johnson-Laird, 1980] P. Johnson-Laird. Mental models in cognitive science. *Cognitive Science*, 4:71–115, 1980.

[Kambhampati *et al.*, 1993] S. Kambhampati, M. Cutkosky, J. Tenenbaum und S. Lee. Integrating general purpose planners and specialized reasoners: Case study of a hybrid planning architecture. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(6):1503–1518, 1993.

[Kautz und Allen, 1986] H. A. Kautz und J. F. Allen. Generalized plan recognition. In AAAI [1986], 32–37.

[Kautz und Ladkin, 1991] H. A. Kautz und P. B. Ladkin. Integrating metric and qualitative temporal reasoning. In AAAI [1991], 241–246.

[Kautz und Selman, 1992] H. A. Kautz und B. Selman. Planning as satisfiability. In ECAI [1992], 359–363.

[Kautz und Selman, 1996] H. A. Kautz und B. Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In AAAI [1996], 1194–1201.

[Kautz, 1987] H. A. Kautz. *A Formal Theory of Plan Recognition*. PhD thesis, Univ. of Rochester, Rochester, NY, 1987.

[Kautz, 1991] H. A. Kautz. A formal theory of plan recognition and its implementation. In Allen et al. [1991], 69–126.

[Kidd, 1985] A. L. Kidd. The consultative role of an expert stystem. In P. Johnson und S. Cook (Hrsg.), *People and Computers: Designing the Interface*, 248–254. Cambridge University Press, 1985.

[Kieras und Bovair, 1984] D. E. Kieras und S. Bovair. The role of a mental model in learning to operate a device. *Cognitive Science*, 8:255–273, 1984.

[Kintsch und Keenan, 1973] W. Kintsch und J. M. Keenan. Reading rate and retention as a function of the number of propositions in the base structure of sentences. *Cognitive Psychology*, 5:257–274, 1973.

[Kintsch und van Dijk, 1978] W. Kintsch und T. A. van Dijk. Toward a model of text comprehension and production. *Psychological Review*, 85:363–394, 1978.

[Kintsch und Vipond, 1979] W. Kintsch und D. Vipond. Reading comprehension and readability in educational practice and psychological theory. In L.-G. Nilsson (Hrsg.), *Perspectives on memory research*, 329–365. Erlbaum, Hillsdale, NJ, 1979.

[Kintsch, 1988] W. Kintsch. The role of knowledge in discourse comprehension: A construction–integration model. *Psychological Review*, 95:163–182, 1988.

[Kintsch, 1998] W. Kintsch. *Comprehension: A Paradigm for Cognition*. Cambridge University Press, Cambridge, 1998.

[Knoblock *et al.*, 1991] C. A. Knoblock, J. D. Tenenberg und Q. Yang. Characterizing abstraction hierarchies for planning. In AAAI [1991], 692–697.

[Knoblock, 1994] C. A. Knoblock. Automatically generating abstractions for planning. *Artificial Intelligence*, 68:243–302, 1994.

[Knoblock, 1996] C. A. Knoblock. Building a planner for information gathering: A report from the trenches. In AIPS [1996], 134–141.

[Kobsa *et al.*, 1994] A. Kobsa, D. Müller und A. Nill. KN–AHS: An adaptive hypertext client of the user modeling system BGP–MS. In *Proc. of 4th International Conference on User Modeling*, 99–106, Hyannis, MA, 08-15/19, 1994.

[Kobsa und Pohl, 1995] A. Kobsa und W. Pohl. The user modeling shell system BGP–MS. *User Modeling and User-Adapted Interaction*, 4(2):59–106, 1995.

[Kobsa und Wahlster, 1989] A. Kobsa und W. Wahlster (Hrsg.). *User Models in Dialog Systems*. Springer, Berlin, Heidelberg, 1989.

[Kobsa, 1985] A. Kobsa. *Benutzermodellierung in Dialogsystemen*. Springer, Berlin, Heidelberg, 1985.

[Kobsa, 1989] A. Kobsa. A taxonomy of beliefs and goals for user models in dialog systems. In Kobsa und Wahlster [1989], 52–69.

[Kobsa, 1990a] A. Kobsa. Modeling the user's conceptual knowledge in BGP–MS, a user modelling shell system. *Computational Intelligence*, 6:193–208, 1990.

[Kobsa, 1990b] A. Kobsa. The SB–ONE knowledge representation workbench. Memo 50, Univ. des Saarlandes, Saarbrücken, DEU, 1990.

[Kobsa, 1991] A. Kobsa. Utilizing knowledge: The components of the SB–ONE knowledge representation workbench. In Sowa [1991], 457–486.

[Kobsa, 2001] A. Kobsa. Tailoring privacy to users' needs. In UM [2001], 303–313.

[Koehler, 1998] J. Koehler. Planning under resource constraints. In *Proc. of 13th European Conference on Artificial Intelligence*, 489–493, Brighton, GBR, 08-23/28, 1998.

[Kowalski und Sergot, 1986] R. Kowalski und M. Sergot. A logic–based calculus of events. *New Generation Computing*, 4:67–95, 1986.

[Kowalski, 1979] R. Kowalski. *Logic for Problem Solving*. Elsevier, Amsterdam, London, New York, 1979.

[Küpper *et al.*, 1992] D. Küpper, D. Rösner und A. Striegl-Scherer. Natürlichsprachliche Zugangssysteme mit Deutsch als Interaktionssprache – eine vergleichende Studie. *Sprache und Datenverarbeitung*, 16:29–66, 1992.

[KR, 1989] R. J. Brachman, H. J. Levesque und R. Reiter (Hrsg.). *Proc. of First International Conference on Principles of Knowledge Representation and Reasoning*, Toronto, CAN, 05-15/18, 1989.

[KR, 1992] B. Nebel, C. Rich und W. Swartout (Hrsg.). *Principles of Knowledge Representation and Reasoning: Proc. of 3rd International Conference (KR'92)*, Cambridge, MA, 10-25/29, 1992.

[Kraus, 1997] S. Kraus. Negotiation and cooperation in multi–agent environments. *Artificial Intelligence*, 94(1):79–97, 1997.

[Krause *et al.*, 1993] J. Krause, A. Hirschmann und E. Mittermaier. The intelligent help system COMFO-HELP: Towards a solution of the practicability problem for user modeling and adaptive systems. *User Modeling and User-Adapted Interaction*, 3(3):249–282, 1993.

[Kröger, 1987] F. Kröger. *Temporal Logic of Programs*. Springer, Heidelberg, DEU, 1987.

[Kripke, 1963a] S. A. Kripke. Semantical analysis of modal logic. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 67–96, 1963.

[Kripke, 1963b] S. A. Kripke. Semantical considerations on modal logic. *Acta Philosophica Fennica*, 16:83–94, 1963.

[Kushmerick *et al.*, 1995] N. Kushmerick, S. Hanks und D. S. Weld. An algorithm for probabilistic planning. *Artificial Intelligence*, 76(1-2):239–286, 1995.

[Kwok und Weld, 1996] C. Kwok und D. S. Weld. Planning to gather information. In AAAI [1996], 32–39.

[Lambert und Carberry, 1991] L. Lambert und S. Carberry. A tripartite plan–based model for dialogue. In *Proc. of 29th Annual Meeting of the Association for Computational Linguistics*, 47–54, 1991.

[Lambrix, 2000] P. Lambrix. *Part–Whole Reasoning in an Object–Centered Framework*. Springer, Berlin, Heidelberg, DEU, 2000.

[Lehnert und Ringle, 1982] G. Lehnert und A. Ringle (Hrsg.). *Strategies for Natural Language Processing*. Lawrence Erlbaum, Hillsdale, NJ, 1982.

[Lifschitz, 1986] V. Lifschitz. On the semantics of STRIPS. In *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*, 1–9, Timberline, OR, 06-30/07-2, 1986.

[Lin, 1998] F. Lin. On measuring plan quality. In *Principles of Knowledge Representation and Reasoning: Proc. of 6th International Conference (KR'98)*, 224–233, Trento, ITA, 06-02/05, 1998.

[Litman und Allen, 1984] D. J. Litman und J. F. Allen. A plan recognition model for clarification subdialogues. In *Proc. of 10th International Conference on Computational Linguistics*, 302–311, Stanford, CA, 1984.

[Litman, 1985] D. J. Litman. *Plan Recognition and Discourse Analysis: An Integrated Approach for Understanding Dialogues*. PhD thesis, Univ. of Rochester, Rochester, NY, 1985.

[Lochbaum, 1995] K. E. Lochbaum. The use of knowledge preconditions in language processing. In IJCAI [1995], 1260–1266.

[Long, 2000] D. Long. The AIPS–98 planning competition. *AI Magazine*, 21(2):13–34, 2000.

[Loui, 1987] R. Loui. Defeat among arguments: A system of defeasible inference. *Computational Intelligence*, 3(2):100–106, 1987.

[MacGregor, 1991] R. MacGregor. The evolving technology of classification–based knowledge representation systems. In Sowa [1991], 385–400.

[Mann und Thompson, 1987] W. C. Mann und S. A. Thompson. Rhetorical Structure Theory: A theory of text organisation. In L. Polanyi (Hrsg.), *The Structure of Discourse*. Ablex, Norwood, NJ, 1987.

[Mannes und Kintsch, 1991] S. M. Mannes und W. Kintsch. Routine computing tasks: Planning as understanding. *Cognitive Science*, 15:305–342, 1991.

[Mays *et al.*, 1991] E. Mays, R. Dionne und R. Weida. K–Rep system overview. *SIGART ACM Bulletin*, 2(3):93–97, 1991.

[McAllester und Rosenblitt, 1991] D. McAllester und D. Rosenblitt. Systematic nonlinear planning. In AAAI [1991], 634–639.

[McCalla, 2000] G. McCalla. The fragmentation of culture, learning, teaching and technology: Implications for the artificial intelligence in education research agenda in 2010. *International Journal of Artificial Intelligence in Education*, 11:177–196, 2000.

[McCarthy und Hayes, 1969] J. McCarthy und P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer und D. Michie (Hrsg.), *Machine Intelligence Vol. 4*, 463–502. Edinburgh University Press, Edinburgh, GBR, 1969.

[McCarthy, 1977] J. McCarthy. Epistemological problems in artificial intelligence. In *Proc. of 5th International Joint Conference on Artificial Intelligence*, 1038–1044, Cambridge, MA, 08-22/25, 1977.

[McCarthy, 1980] J. McCarthy. Circumscription – A form of non–monotonic reasoning. *Artificial Intelligence*, 13(1-2):27–39, 1980.

[McCoy, 1985] K. F. McCoy. *Correcting Object–Related Misconceptions*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, Phiadelphia, PA, 1985.

[McCoy, 1989] K. F. McCoy. Generating context–sensitive responses to object–related misconceptions. *Artificial Intelligence*, 41:157–195, 1989.

[McDermott und Hendler, 1995] D. McDermott und J. Hendler. Planning: What it is, what it could be, an introduction to the special issue on planning and scheduling. *Artificial Intelligence*, 76(July):1–16, 1995.

[McDermott, 1985] D. McDermott. Reasoning about plans. In Hobbs und Moore [1985], 269–318.

[McDermott, 2000] D. McDermott. The 1998 planning systems competition. *AI Magazine*, 21(2):35–56, 2000.

[McKeown *et al.*, 1990] K. R. McKeown, M. Elhadad, Y. Fukumoto, J. Lim, C. Lombardi, J. Robin und F. Smadja. Natural language generation in COMET. In R. Dale, C. Mellish und M. Zock (Hrsg.), *Current Research in Natural Language Generation*. Academic Press, London, GBR, 1990.

[McKeown, 1985] K. R. McKeown. *Text Generation*. Cambridge University Press, Cambridge, MA, 1985.

[McTear, 1993] M. McTear. User modelling for adaptive computer systems: A survey of recent developments. *Artificial Intelligence Review*, 7:157–184, 1993.

[Mellish und Evans, 1989] C. Mellish und R. Evans. Natural language generation from plans. *Computational Linguistics*, 15(4):233–249, 1989.

[Mey, 1971] *Meyers enzyklopädisches Lexikon*. Bibliographisches Institut Mannheim, DEU, 1971.

[Miksch *et al.*, 1998] S. Miksch, R. Kosara, Y. Shahar und P. Johnson. AsbruView: Visualization of time–oriented, skeletal plans. In *Proc. of the 4th International Conference on Artificial Intelligence Planning Systems*, 11–18, Pittsburgh, PA, 06-07/10, 1998.

[Miller und Kintsch, 1980] J. Miller und W. Kintsch. Readability and recall of short prose passages: A theoretical analysis. *Journal of Experimental Psychology: Human Learning and Memory*, 6:335–354, 1980.

[Miller, 1956] G. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63:81–97, 1956.

[Minsky, 1975] M. Minsky. A framework for representing knowledge. In P. Winston (Hrsg.), *The Psychology of Computer Vision*. McGraw-Hill, New York, NY, 1975.

[Mittermaier, 1995] E. Mittermaier. *Planbasierte intelligente Hilfe als Baustein eines symbiotischen Gesamtsystems*. Univ. Verlag Konstanz, Konstanz, DEU, 1995.

[Müller *et al.*, 2001] C. Müller, B. Großmann-Hutter, A. Jameson, R. Rummer und F. Wittig. Recognizing time pressure and cognitive load on the basis of speech: An Experimental Study. In UM [2001], 24–33.

[Moore und Paris, 1989] J. D. Moore und C. L. Paris. Planning text for advisory dialogues. In *Proc. of 27th Annual Meeting of the Association for Computational Linguistics*, 220–227, 1989.

[Moore und Paris, 1993] J. D. Moore und C. L. Paris. Planning text for advisory dialogues: Capturing intentional and rhetorical information. *Computational Linguistics*, 19(4):651–694, 1993.

[Moore, 1980] R. C. Moore. *Reasoning About Knowledge and Action*. PhD thesis, SRI International, Menlo Park, CA, 1980.

[Moore, 1985] R. C. Moore. A formal theory of knowledge and action. In Hobbs und Moore [1985].

[Moore, 1995] J. D. Moore. *Participating in Explanatory Dialogues*. MIT Press, Cambridge, MA, 1995.

[Morgenstern, 1987] L. Morgenstern. Knowledge preconditions for actions and plans. In IJCAI [1987], 867–874.

[Myers, 1996] K. L. Myers. Advisable planning systems. In A. Tate (Hrsg.), *Advanced Planning Technology: Technological Achievements of the ARPA/Rome Laboratory Planning Initiative*, 206–209. AAAI Press, Menlo Park, CA, 1996.

[Nau *et al.*, 1998] D. S. Nau, S. J. J. Smith und K. Erol. Control strategies in HTN planning: Theory versus practice. In AAAI [1998], 1127–1133.

[Newell und Simon, 1961] A. Newell und H. A. Simon. GPS, A program that simulates human thought. In H. Billing (Hrsg.), *Lernende Automaten*, 109–124. Oldenburg, München, DEU, 1961.

[Nilsson, 1980] N. Nilsson. *Principles of Artificial Intelligence*. Morgan Kaufmann, San Mateo, CA, 1980.

[NLG, 1994] *7th Workshop on Natural Language Generation*, Kennebunkport, ME, 1994.

[Nothdurft, 1984] W. Nothdurft. *...äh folgendes Problem äh... – Die interaktive Ausarbeitung 'des Problems' in Beratungsgesprächen*. Narr, Tübingen, DEU, 1984.

[Paris und Linden, 1996] C. L. Paris und K. V. Linden. DRAFTER: An interactive support tool for writing multilingual manuals. *IEEE Computer*, 27(7):49–56, 1996.

[Paris, 1993] C. L. Paris. *User Modeling in Text Generation*. Pinter, London, GBR, 1993.

[Pasula, 1996] H. Pasula. *Design of a Collaborative Planning System*. Senior honors thesis, Harvard University, Cambridge, MA, 1996.

[Patel-Schneider und Swartout, 1993] P. F. Patel-Schneider und B. Swartout. Description–Logic knowledge representation system specification from the KRSS group of the ARPA knowledge sharing effort. KRSS group of the ARPA, 1993.

[Pednault, 1988] E. P. D. Pednault. Synthesizing plans that contain actions with context-dependent effects. *Computational Intelligence*, 4:356–372, 1988.

[Pednault, 1989] E. P. Pednault. ADL: Exploring the middle ground between STRIPS and the situation calculus. In KR [1989], 324–332.

[Pelavin, 1991] R. N. Pelavin. Planning with simultaneous actions and external events. In Allen et al. [1991], 127–212.

[Penberthy und Weld, 1992] J. S. Penberthy und D. S. Weld. UCPOP: A sound, complete, partial order planner for ADL. In KR [1992], 103–114.

[Peter und Rösner, 1994] G. Peter und D. Rösner. User–model–driven generation of instructions. *User Modeling and User-Adapted Interaction*, 3(4):289–319, 1994.

[Pohl, 1997] W. Pohl. *Logic–Based Representation and Inference for User Modeling Shell Systems*. Dissertation, Infix, St. Augustin, DEU, 1997.

[Pollack, 1986] M. Pollack. *Inferring domain plans in question–answering*. PhD thesis, Univ. of Pennsylvania, Phiadelphia, PA, 1986.

[Pollack, 1990] M. E. Pollack. Plans as complex mental attitudes. In Cohen et al. [1990], 77–103.

[Pollack, 1992] M. E. Pollack. The uses of plans. *Artificial Intelligence*, 57:43–68, 1992.

[Pollock, 1987] J. L. Pollock. Defeasible reasoning. *Cognitive Science*, 11:481–518, 1987.

[Pollock, 1992] J. L. Pollock. How to reason defeasibly. *Artificial Intelligence*, 57:1–42, 1992.

[Profitlich, 1989] H.-J. Profitlich. Das SB–ONE Handbuch. Version 1.0. Memo 33, Univ. Saarbrücken, DEU, 1989.

[Profitlich, 1990] H.-J. Profitlich. *SB–ONE: Ein Wissenrepräsentationssystem basierend auf KL–ONE*. Diplomarbeit, Univ. Saarbrücken, DEU, 1990.

[Quilici, 1989] A. Quilici. Detecting and responding to plan–oriented misconcepitons. In Kobsa und Wahlster [1989], 108–132.

[Quillian, 1968] R. Quillian. Semantic memory. In M. Minsky (Hrsg.), *Semantic Information Processing*, 216–270. MIT-Press, Cambridge, MA, 1968.

[Reiter, 1980] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[Reiter, 1992] R. Reiter. The projection problem in the situation calculus: A soundness and completeness result, with an application to database updates. In *Proc. of the First International Conference on Artificial Intelligence Planning Systems*, 198–203, College Park, MD, 06-15/17, 1992.

[Reiter, 1994] E. Reiter. Has a consensus NL generation architecture appeared, and is it psycholinguistically plausible? In NLG [1994], 163–170.

[Rich, 1979] E. Rich. User modeling via stereotypes. *Cognitive Science*, 3:329–354, 1979.

[Rich, 1983] E. Rich. Users are individuals: Individualizing user models. *International Journal of Man-Machine Studies*, 18:199–214, 1983.

[Rich, 1989] E. Rich. Stereotypes and user modeling. In Kobsa und Wahlster [1989], 35–51.

[Rosner und Pnueli, 1986] R. Rosner und A. Pnueli. A choppy logic. In *Proc. of Symposium on Logic in Computer Science*, 306–313, Cambridge, MA, 06-16/18, 1986.

[Rösner und Stede, 1992] D. Rösner und M. Stede. TECHDOC: A system for the automatic production of multilingual technical documents. In *Proc. of KONVENS–1992*, 1992.

[Russell und Norvig, 1995] S. Russell und P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, NJ, 1995.

[Sacerdoti, 1974] E. D. Sacerdoti. Planning in a Hierarchy of Abstraction Spaces. *Artificial Intelligence*, 5:115–135, 1974.

[Sacerdoti, 1977] E. D. Sacerdoti. *A Structure for Plans and Behavior*. Elsevier/North-Holland, Amsterdam, London, New York, 1977.

[Sandewall, 1994] E. Sandewall. *Features and Fluents*. Clarendon Press, Oxford, 1994.

[Sarner und Carberry, 1992] M. Sarner und S. Carberry. Generating tailored definitions using a multifaceted user model. *User Modeling and User-Adapted Interaction*, 2(3):181–210, 1992.

[Sattler, 1998] U. Sattler. *Terminological Knowledge Representation Systems in a Process Engineering Application*. Dissertation, LuFG Theoretical Computer Science, RWTH Aachen, DEU, 1998.

[Schmauks und Reithinger, 1988] D. Schmauks und N. Reithinger. Generating multimodal output: Conditions, advantages and problems. In *Proc. of 12th International Conference on Computational Linguistics*, 584–588, Budapest, HUN, 1988.

[Schmidt *et al.*, 1978] C. Schmidt, N. Sridharan und J. Goodson. The plan recognition problem: An intersection of psychology and artificial intelligence. *Artificial Intelligence*, 11:45–82, 1978.

[Schmidt-Schauß, 1989] M. Schmidt-Schauß. Subsumption in KL–ONE is undecidable. In KR [1989], 421–431.

[Schreck, 2000] J. Schreck. *Security and Privacy in User Modeling*. Dissertation, Univ. Essen, DEU, 2000.

[Schubert, 1990] L. Schubert. Monotonic solution of the frame problem in the situation calculus: An efficient method for worlds with fully specified actions. In H. E. Kyburg, Jr., R. P. Loui und G. N. Carlson (Hrsg.), *Knowledge Representation and Defeasible Reasoning*, 23–68. Kluwer Academic Publishers, Dordrecht, NLD, 1990.

[Searle, 1969] J. R. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, GBR, 1969.

[Searle, 1990] J. R. Searle. Collective intentions and actions. In Cohen et al. [1990], 401–415.

[Shafer, 1976] G. Shafer. *A Mathematical Theorie of Evidence*. Princeton, New Jersey, 1976.

[Shanahan, 1989] M. Shanahan. Prediction is deduction but explanation is abduction. In *Proc. of 11th International Joint Conference on Artificial Intelligence*, 1055–1060, Detroit, MI, 08-20/25, 1989.

[Shrager und Finin, 1982] J. Shrager und T. W. Finin. An expert system that volunteers advice. In *Proc. of Second National Conference on Artificial Intelligence*, 339–340, Pittsburgh, PA, 08-18/20, 1982.

[Sidner, 1985] C. Sidner. Plan parsing for intended response recognition in discourse. *Computational Intelligence*, 1:1–10, 1985.

[Sleeman, 1985] D. Sleeman. UMFE: A user modeling front–end subsystem. *International Journal of Man-Machine Studies*, 23:71–88, 1985.

[Sowa, 1991] J. Sowa (Hrsg.). *Principles of Semantic Networks*. Morgan Kaufmann, San Mateo, CA, 1991.

[Speel und Patel-Schneider, 1994] P.-H. Speel und P. F. Patel-Schneider. CLASSIC extended with physical whole–part relations. In *Proc. of the 1994 International Workshop on Description Logics*, 45–50, Bonn, DEU, 05-28/29, 1994.

[Stefik, 1981] M. Stefik. Planning and Meta–Planning (MOLGEN: Part 2). *Artificial Intelligence*, 16(2):141–170, 1981.

[Sussman, 1975] G. J. Sussman. *A Computer Model of Skill Acquisition*. Elsevier/North-Holland, Amsterdam, London, New York, 1975.

[Tate *et al.*, 1990] A. Tate, J. Hendler und M. Drummond. A review of AI planning techniques. In Allen et al. [1990], 26–49.

[Tate *et al.*, 1994] A. Tate, B. Drabble und R. Kirby. O–Plan2: An architecture for command, planning and control. In M. Fox und M. Zweben (Hrsg.), *Intelligent Scheduling*. Morgan Kaufmann, 1994.

[Tate, 1976] A. Tate. Project planning using a hierarchic non–linear planner. Research Report 25, Univ. of Edinburgh, Department of Artificial Intelligence, 1976.

[Taylor *et al.*, 1996] J. A. Taylor, J. Carletta und C. Mellish. Requirements for belief models in cooperative dialogue. *User Modeling and User-Adapted Interaction*, 6(1):23–68, 1996.

[Tenenberg, 1991] J. D. Tenenberg. Abstraction in planning. In Allen et al. [1991], 213–283.

[Thies, 1994] M. A. Thies. *Planbasierte Hilfeverfahren für direkt–manipulative Systeme*. Dissertation, Univ. Stuttgart, DEU, 1994.

[Trabasso und Sperry, 1985] T. Trabasso und L. L. Sperry. Causal relatedness and importance of story events. *Journal of Memory and Language*, 24:595–611, 1985.

[Trabasso und van den Broeck, 1985] T. Trabasso und P. van den Broeck. Causal thinking and the representation of narrative events. *Journal of Memory and Language*, 24:612–630, 1985.

[UM, 2001] M. Bauer, J. Vassileva und P. Gmytrasiewicz (Hrsg.). *User Modeling: Proceedings of the 8th International Conference, UM'01*, Sonthofen, DEU, 07-13/17, 2001.

[UMUAI, 2001] *User Modeling and User-Adapted Interaction*, 11(1-2), Ten Year Anniversary Issue, 2001.

[van den Broeck, 1988] P. van den Broeck. The effects of causal relations and hierarchical position on the importance of story statements. *Journal of Memory and Language*, 27:1–22, 1988.

[van Dijk und Kintsch, 1983] T. A. van Dijk und W. Kintsch. *Strategies of Discourse Comprehension*. Academic Press, New York, NY, 1983.

[Vere, 1983] S. A. Vere. Planning in time: Windows and durations for activities and aoals. *Pattern Analysis and Machine Intelligence*, 5:246–267, 1983.

[Vilain und Kautz, 1986] M. Vilain und H. A. Kautz. Constraint propagation algorithms for temporal reasoning. In AAAI [1986], 373–381.

[Virvou und du Boulay, 1999] M. Virvou und B. du Boulay. Human plausible reasoning for intelligent help. *User Modeling and User-Adapted Interaction*, 9(4):323–377, 1999.

[Wahlster *et al.*, 1988] W. Wahlster, M. Hecking und C. Kemke. SC: Ein intelligentes Hilfesystem für SINIX. In W. Gollan, W. Paul und A. Schmitt (Hrsg.), *Innovative Informationsinfrastrukturen*. Springer, Berlin, DEU, 1988.

[Wahlster *et al.*, 1993] W. Wahlster, E. André, W. Finkler, H.-J. Profitlich und T. Rist. Plan–based integration of natural language and graphics generation. *Artificial Intelligence*, 63:387–427, 1993.

[Wahlster und Kobsa, 1989] W. Wahlster und A. Kobsa. User models in dialog systems. In Kobsa und Wahlster [1989], 4–34.

[Walker, 1996] M. A. Walker. The effect of resource limits and task complexity on collaborative planning in dialogue. *Artificial Intelligence*, 85(1-2):181–243, 1996.

[Webber *et al.*, 1995] B. Webber, N. Badler, B. D. Eugenio, C. Geib, L. Levison und M. Moore. Instructions, Intentions and Expectations. *Artificial Intelligence*, 73(1-2 February):253–269, 1995.

[Weida und Litman, 1992] R. Weida und D. J. Litman. Terminological reasoning with constraint networks and an application to plan recognition. In KR [1992], 282–293.

[Weida und Litman, 1994] R. Weida und D. J. Litman. Subsumption and recognition of heterogeneous constraint networks. In *Proc. of The 10th IEEE Conference on Artificial Intelligence for Applications*, 381–388, San Antonio, TX, Mar., 1994.

[Weida, 1996] R. Weida. *Closed terminologies and temporal reasoning in description for concept and plan recognition*. PhD thesis, Department of Computer Science, Columbia University, New York, NY, 1996.

[Weld *et al.*, 1998] D. S. Weld, C. R. Anderson und D. E. Smith. Extending graphplan to handle uncertainty & sensing actions. In AAAI [1998], 897–904.

[Weld, 1994] D. S. Weld. An introduction to least commitment planning. *AI Magazine*, 4(4):27–61, 1994.

[Weld, 1999] D. S. Weld. Recent trends in planning. *AI Magazine*, 20(2):93–123, 1999.

[Wilensky *et al.*, 1988] R. Wilensky, D. Chin, M. Luria, J. Martin, J. Mayfield und D. Wu. The Berkeley UNIX consultant project. *Computational Linguistics*, 14(4):35–84, 1988.

[Wilensky, 1981] R. Wilensky. Meta–Planning: Representing and using knowledge about planning in problem solving and natural language understanding. *Cognitive Science*, 5:197–233, 1981.

[Wilkins *et al.*, 1995] D. E. Wilkins, K. L. Myers, J. D. Lowrance und L. P. Wesley. Planning and reacting in uncertain and dynamic environments. *Journal of Experimental and Theoretical Artificial Intelligence*, 7(1):197–227, 1995.

[Wilkins, 1988] D. E. Wilkins. *Practical Planning – Extending the classical AI planning paradigm*. Morgan Kaufmann, 1988.

[Wilkins, 1993] D. E. Wilkins. Using the SIPE–2 planning system: A manual for version 4.3. Artificial Intelligence Center, Menlo Park, CA, 1993.

[Winston *et al.*, 1987] M. E. Winston, R. Chaffin und D. Herrmann. A taxonomy of part–whole relations. *Cognitive Science*, 11:417–444, 1987.

[Wirth, 1999] N. Wirth. *Algorithmen und Datenstrukturen*. Teubner, Stuttgart, DEU, 1999.

[Woods, 1975] W. A. Woods. What's in a link: Foundations for semantic networks. In D. G. Bobrow und A. Collins (Hrsg.), *Representation and Understanding: Studies in Cognitive Science*, 35–82. Academic Press, New York, NY, 1975.

[Yang, 1990] Q. Yang. Formalizing planning knowledge for hierarchical planning. *Computational Intelligence*, 6:12–24, 1990.

[Yang, 1997] Q. Yang. *Intelligent Planning: A Decomposition and Abstraction Based Approach*. Springer, Berlin, DEU, 1997.

[Young *et al.*, 1994a] R. M. Young, J. D. Moore und M. E. Pollack. Towards a principled representation of discourse plans. In *Proc. of 16th Conference of the Cognitive Science Society*, Atlanta, GA, 1994.

[Young *et al.*, 1994b] R. M. Young, M. E. Pollack und J. D. Moore. Decomposition and causality in partial–order planning. In AIPS [1994], 188–193.

[Young und Moore, 1994] R. M. Young und J. D. Moore. DPOCL: A principled approach to discourse planning. In NLG [1994], 13–20.

[Young, 1999] R. M. Young. Using Grice's maxim of quantity to select the content of plan descriptions. *Artificial Intelligence*, 115(2):215–256, 1999.

[Zoeppritz, 1984] M. Zoeppritz. Datenabfrage in natürlicher Sprache: Diskussion von Eigenschaften der natürlichen Sprache aufgrund von Beobachtungsdaten. In *Proc. of ONLINE '84*, 1984.

[Zukerman und McConachy, 1993a] I. Zukerman und R. McConachy. Consulting a user model to address a user's inferences during content planning. *User Modeling and User-Adapted Interaction*, 3(2):155–185, 1993.

[Zukerman und McConachy, 1993b] I. Zukerman und R. McConachy. Generating concise discourse that addresses a user's inferences. In IJCAI [1993], 1202–1207.

[Zukerman und McConachy, 1995] I. Zukerman und R. McConachy. Generating discourse across several user models: Maximizing belief while avoiding boredom and overload. In IJCAI [1995], 1251–1259.

Index

abgeleitetes Prädikat, *siehe* Prädikat
Abhängigkeiten
    im Sachbereich, **140**
    potentielle, **147**, 230, 250
ABox, 110
Absichten
    gemeinsame, 55
Abstraktionshierarchie, 33, 116
ABSTRIPS, 19
Action, 140
Action Schema, 140
aktiver Teil des Planungssachbereichs, *siehe* Planungssachbereich
Allens Zeitrelationen, Komplexität, 33
Annahmen
    über Handlungsmöglichkeiten, **124**
    einfache, **119**
    falsche, 74, **92**, 105, 147, 187
    gemeinsame, 54, **122**
    reflexive, **121**
    und Wissen, 92
Anwendbarkeitsvorbedingung, **98**
Application Domain, **140**
Argumentations-System, 48
Assertionale Komponente, 110
Assumption Type, **123**
Assumption Type Representation, **123**
AsTRa, **123**, 229, 288
atomar (Prädikat), **177**, 222
atomarer Planschritt, *siehe* Planschritt
atomares Plan-Concept, *siehe* Plan-Concept
atomic (Plan-Concept), **223**
Attribut
    beschreibungslogisches, 106
    in SB-ONE, **117**
Attribute-Descriptions in SB-ONE, **116**
Ausführungsmöglichkeit
    eines Plan-Concepts, **171**
Axiome
    im Planungssachbereich, 16, 140, **145**
    in PDDL, **145**
    in UCPOP, **145**
    potentielle, **230**, 250, 317
    Syntax für die Definition, **303**
    terminologische, **107**

Bedrohung eines Causal Links, **18**, 214
before (Zugriffsfunktion), **180**
begriffliches Wissen, *siehe* Wissen
Begriffshierarchie, 116
Beispielbenutzer, **129**
Beliefs
    and Knowledge, *siehe* Annahmen und Wissen
    Mutual, 54, **122, 123**
    Shared, **123**
Benamung von Stereotype-Partitionen, **133**
Benamungsschema
    für Assumption Types, **123**
    für Partitionen, **120**
Benutzer, typische, 119
Benutzergruppen, typische, 119
Benutzermodell, 119
    Aufbau, **105**
Benutzermodellierungswissensbasis, **120**, 140, 227
Beschreibungslogik, 104, **105**
beschreibungslogisches Statement, **107**
Beurteilungsprädikat, **180**
Bewertung einer Planpräsentation, 187, 188, 191
Bezeichner eines Concepts, **126**, 165
BGP-MS, 10, **122**, 229, 321
BMWB, *siehe* Benutzermodellierungswissensbasis
Bug-List, **71**, 75

Capability, 124
Causal Links, **18**, 322
    irrelevante, **175**
    relevante, **175**
Closed-World Annahme, **14**, 139
Concept-Hierarchie, 116
Concepts, **106**
    definierte, **108**, 155
    disjunkte, 112, **118**
    komplementäre, **118**
    primitive, **108**
    Syntax für die Definition, **302**
constraints (Zugriffsfunktion), **180**

decomposition (Plan-Concept), **221**
Decomposition Link, 67
Dekomposition, 316
    eines Operators, **22**
    eines Plan-Concepts, 178, **179**

falsche, 179, **181**
korrekte, **181**
Syntax für die Definition, **222**
Dekompositions-Concept, **221**
Dekompositions-Hierarchie, **22**, 33
Zyklenfreiheit, **180**
Dekompositions-Relation, **22**
Description Logic, *siehe* Beschreibungslogik
Dialograhmen, 134, 293
Domain, **140**
einer Interpretation, **106**
Domain-Concept, **112**, 124, 307
Downward Refinement Property, **21**
Downward Solution Property, **20**

EBNF, **300**
Effekt
bedingter, **100**
Beobachtungs-, 83
einer Aktion, **15**
impliziter, **145**, **151**, 230, 252, 258, 262
impliziter, Syntax für die Definition, **151**
Effekt-Attribut, **113**
Effekt-Vererbung, **23**
Effektkomponente eines Planoperators, **17**
Endzustand, **14**
Entscheidbarkeit des Subsumptionsproblems, 110
Ereignis, 47
Ereigniskalkül, 48
Erzeuger eines Literals, 18
Event, **31**
Hierarchy, **31**
Types, **31**
Expansion-Critics Zyklus, 62
Explanation Closure, 51

Fähigkeit, **92**
Fähigkeitspartition, **124**
Fähigkeitsview, **124**
Feature Chain, **110**, 114
Fehlannahmen, 80
eines Benutzers, 74
typische, **129**
Fehler, typische, **71**, 182
Frame Problem, *siehe* Rahmen-Problem
Frames, 104, 105

General Problem Solver, 13
GPS, *siehe* General Problem Solver
Grice'sches Kooperativitäts-Prinzip, *siehe* Kooperativitäts-Prinzip

Handlungsmöglichkeiten, 1, 7, 89, **93**, 95, 96
Handlungspotential, **93**, 95, 124
Konstanz des, **102**
werkzeugspezifisches, **162**, **166**
Hauptzweck eines Kommandos, 80
Hierarchical Task Network Planning, *siehe* Planning
Hilfe
aktive, **70**
passive, **70**
Hilfestellung
benutzerspezifische, **80**
kooperative, 74
Hilfesysteme, planbasierte, **70**
Hinzu-Liste, **16**
HTN, *siehe* Planning, Hierarchical Task Network

Identifier eines Concepts, *siehe* Bezeichner
Ignore-Aktionen, 82
Impliziter Effekt, *siehe* Effekt
incomplete (Plan-Concept), **223**
Inheritance Abstraction, **20**
Intend-That, 56
Intend-To, 56
Interpretation, **106**
Interpretationsbereich einer, **106**
Interpretationsfunktion einer, **106**
inverse Subsumptions-Abbildung, **43**
ISA-Kante, **116**

Kenntnis-Lücke, **103**
Kenntnisse eines Benutzers, **92**, 96
KL-ONE, 106, 116
KL-ONE-artige Systeme, 104, 106, 116
Klassifikation eines Concepts, **116**
Knowledge and Belief, *siehe* Annahmen und Wissen
Kohärenz von Texten, 63
Komplementäre Typen, *siehe* Typen
Komplexität von Beschreibungslogiken, 106
Komponente eines Plans, **18**
konkrete Absicht einen Plan auszuführen, 53
Können eines Benutzers, **95**
Kontext einer Hilfestellung, **70**
Kooperativitäts-Prinzip, Grice'sches, **29**
Kooperativitätsbedingungen, Grice'sche, 74
Korrektheit einer Dekomposition, **181**
KRSS, **106**

Linearisierung eines Plans, **18**
Linearitätsannahme, **26**
Lösch-Liste, **16**

Maßeinheit für Aufwand, **188**
Modallogischer Ansatz
zur Akteurmodellierung, **120**
Mögliche-Welten
Ansatz, 99

Semantik, 61
Momentaufnahme der realen Welt, **87**, 139
Mutual Beliefs, *siehe* Beliefs

nicht-atomarer Planschritt, *siehe* Planschritt
nicht-atomares Plan-Concept, *siehe* Plan-Concept
Nutzbarkeit
einer Dekomposition, **183**
eines Plan Concepts, **183**
selbständige ~ einer Dekomposition, **184**
Nutzen
durch das Entfernen einer Komponente, **204**

Objektbereich einer Interpretation, **106**
Objektbezogenes Wissen, *siehe* Wissen
Operator, *siehe* Planoperator
Operatorabstraktion, 19, **21**, 233

Partial Order Planning, *siehe* Planning
Partition, 61
Geschwister-, **122**
Stereotype-, **123**
Werkzeug-, **166**
Partitionierung eines Concepts, **118**
Partitionsansatz, **120**
passiver Teil des Planungssachbereichs, *siehe* Planungssachbereich
PDDL, **16**, **110**
Plan, **17**
falscher, 31, **71**, 74
gemeinsamer, **55**
konsistenter, **18**
partiell geordneter, **17**
partieller, 17, **18**
suboptimaler, **71**
typischer falscher, 53
unvollständiger, 17
vollständiger, **18**
plan space, *siehe* Plan-Suchraum
Plan-Concept, **112**, 124, 309
atomares, **177**
Dekomposition eines, *siehe* Dekomposition
Konstruktion, **115**
nicht-atomares, **177**
Syntax für die Definition, **114**
unvollständiges, **178**, 263, 315
werkzeugspezifisches, **162**
plan-concept (Zugriffsfunktion), **190**
plan-concepts (Zugriffsfunktion), **180**
Plan-Netzwerk, 62
Plan-Suchraum, 17
Planbibliothek, 31, 71, 72
Planen
deduktives, **77**, 80, 234
hierarchisches, **19**, 233
nichtlineares, 19
Voraussetzungen beim, **25**
Plangenerierungskomponente, 96
Planning
Decompositional Partial-Order Causal Link, **67**
Hierarchical Task Network, **22**
Least Commitment, **17**, 67
Partial Order, **19**, 233
Planoperator, **15**, 140
primitiver, **22**
Syntax für die Definition, **111**
Planschritt, **17**
atomarer, **177**
nicht-atomarer, **177**
Planungsproblem, **138, 139**, 266
Lösung eines, **17**
Planungssachbereich, 16, 17, **138**, **140**
aktiver Teil, **140**
passiver Teil, **140**
POP, *siehe* Planning, Partial Order
potentielle Abhängigkeiten, *siehe* Abhängigkeiten
potentielle Axiome, *siehe* Axiome
Prädikat
abgeleitetes, 17
abgeleitetes in UCPOP, **146**, 259
primitives in UCPOP, **145**
Präsentationskomponente, 96
Precondition Elimination, 19
primitives Prädikat, *siehe* Prädikat
Procedural Network, 62
Propagationsmechanismus in BGP-MS, **122**

Qualifikationsproblem, **3**, **15**, 26, 48
Qualität eines Plans, 82
Qualitative, Metric and Equality Networks, 43

Rahmen-Axiome, **15**, 78
Rahmen-Problem, **15**
Ramification Problem, 145
Realisierungsmöglichkeiten
von Planoperatoren, **22**
Rekonstruktionsaufwand, **202**
Relaxed Models, **21**
Restriktion, **106**
Rezept, 53, 178, 225
Rhetorical Structure Theory, **64**
Role Chain, *siehe* Rollenkette
Role-Value-Map, **109**
Rolle, **106**
notwendige, **116**
optionale, **116**
Rollenfüller, **108**
Rollenkette, **109**
RVM, *siehe* Role-Value-Map

Sachbereich, statischer, *siehe* Universe, static
SB-ONE, **115**
Semantik von Beschreibungslogik, **106**
Semantisches Netz, 32, 104, 105, 116
    partitioniertes, 120
Shared Beliefs, *siehe* Beliefs
SharedPlan, 54, **55**
Situation, **14**
situation space, *siehe* Situations-Suchraum
Situations-Suchraum, 17
Situationsabstraktion, **19**
Situationskalkül, 14
SNLP, 19, 67
Spezialisierung, **106**
Startplan, **18**, 214
Startsituation, **14**, **17**
steps (Zugriffsfunktion), **180**
Stereotypes, 10, 80, 105, **128**
    in AsTRa, **129**
    Triggerung, **129**
STRIPS, **15**, 30, 37
    Annahme, 15, **26**
    System, **16**
Strukturierte Vererbungsnetze, 105
Subsumption, **109**
    potentielle, **42**
Subsumptions-Abbildung, **42**
Subsumptionsproblem, **109**

TBox, **110**
Temporal Constraint Network, **41**, 224
Terminologie, **110**
Threat, *siehe* Bedrohung
Tutor-Systeme, 70
Typ eines Objekts, **107**
Typen, komplementäre, **144**, **262**
Typisierung des Sachbereichs, **141**

Überbeantwortung, **30**
Überdeckung von Concepts, **118**
UCPOP, 97, **233**, 321
Unique Main Subaction, **24**
Universe
    dynamic, **142**
    static, 16, **142**
unvollständig
    Plan, *siehe* Plan
    Plan-Concept, *siehe* Plan-Concept, 315
Upward Solution Property, **20**, 24

Value Restriction, *siehe* Werterestriktion
Veränderungen
    durch Ausführung eines Operators, 89
Veränderungsmöglichkeiten, 89
Verbraucher eines Literals, 18
Vererbungsmechanismus in BGP-MS, **122**
verläßliches Benutzerwissen, **184**
Verwendungsmöglichkeiten, 160
    eines Werkzeugtyps, **161**
Verwendungspotential eines Werkzeugtyps, **161**
View, **122**, 141
Vorbedingung
    Anwendbarkeits-, **98**
    einer Aktion, 14, **15**
    Wissens-, 83, **99**
Vorbedingungs-Attribut, **113**
Vorbedingungs-Vererbung, **23**
Vorbedingungskomponente
    eines Planoperators, **17**, 113
Vorrang-Muster, **158**, 230, 247, 319
    Syntax für die Definition, **158**

Wahrheit einer Aussage, 92
Werkzeug-Attribut, **163**
Werkzeug-Concept, 161, **162**
Werkzeug-Partition, **166**
Werkzeugtyp, **161**
Werterestriktion, **108**
Wissen
    begriffliches, **104**
    objektbezogenes, **125**, 245, 260, 262
    und Annahmen, *siehe* Annahmen und Wissen
    verläßliches, *siehe* verläßliches Benutzerwissen
Wissensrepräsentationssystem
    terminologisches, 37, 104, **110**

Zielsituation, **14**, **17**
Zustand-Concept, 112

Abkürzungen

AsTRa
: Assumption Type Representation

AT
: Assumption Type

BGP-MS
: Belief, Goal and Plan Maintenance System

BMWB
: Benutzermodellierungswissensbasis

DFKI
: Deutsches Forschungszentrum für Künstliche Intelligenz

EBNF
: Extended Backus-Naur Form

KRSS
: Knowledge Representation System Specification

PDDL
: Planning Domain Definition Language

POP
: Partial Order Planning

RVM
: Role-Value-Map

STRIPS
: Stanford Research Institute Problem Solver

TCN
: Temporal Constraint Network

TWRS
: Terminologisches Wissensrepräsentationssystem