Datenbank- und XML-Technologien im Projekt NAPROCHE

Gregor Büchel

1. Einleitung

Das Projekt NAPROCHE (=: Natural Language Proof Checking), das an der Universität Bonn gegründet wurde[1], untersucht Konzepte und Methoden der maschinellen Prüfung von mathematischen Beweisen, die in einer natürlichen Sprache (z.B. Deutsch oder Englisch) abgefasst sind und maschinenlesbar vorliegen. Das NAPROCHE-Projekt ist ein interdisziplinäres Vorhaben von Mathematikern, Linguisten und Informatikern.

Mathematische Beweise bedürfen der Schriftform, damit die Behauptungen neuer Sätze korrekt aus Axiomen oder aus bereits bewiesenen Sätzen deduziert werden können. Diese Deduktion ist eine Kette logischer Schlüsse, die die Evidenz der vorgelegten Behauptung herstellt. Wird der Text eines mathematischen Beweises durch einen Rechner erfasst, können zunächst die Vorteile genutzt werden, die für beliebige Texte, die mit Textverarbeitungsprogrammen erfasst werden, gelten (Reproduzierbarkeit, Erweiterbarkeit, umfangreiche Alphabete von Sonderzeichen usw.). Für Mathematiker stellt sich darüber hinausgehend die Frage nach einem Mehrwert: Unter welchen Bedingungen kann der Rechner als Beweisprüfer eingesetzt werden?

2. Der Rechner als Beweisprüfer: MIZAR

Dass der Rechner als Beweisprüfer eingesetzt werden kann, belegen Proof Checking Programme, wie z.B. MIZAR[2]. Bei der Nutzung solcher Programme werden jedoch direkt zwei Probleme sichtbar, die den mit der obigen Frage vermuteten Mehrwert schmälern:
(1) MIZAR, wie andere Proof Checker auch, verlangt, dass der Beweis in einer eigenen formalen Sprache, deren Grammatik bei MIZAR durch Produktionsregeln in Backus-Naur-Form (BNF) spezifiziert ist[3], erfasst wird. Faktisch bedeutet das, dass ein mathematischer Beweis, der mit einem beliebigen Texteditor E erfasst worden ist, unter MIZAR ein zweites Mal editiert werden muss. Dieses soll an einem Beispiel illustriert werden: Es soll folgende elementare Eigenschaft der Anordnungsrelation „<“ („kleiner als“) im Körper der reellen Zahlen bewiesen werden:

Lemma: „Gilt $x < y$ und ist $a < 0$, so folgt $ax > ay$.“
„***Beweis.*** Nach Voraussetzung ist $y-x > 0$ und $-a > 0$. Nach Axiom (A.3)[4] folgt daraus $(-a)(y-x) > 0$, d.h. $ax - ay > 0$, also $ax > ay$.“ [FORST, S.15]

Der Beweis dieses Lemmas kann in MIZAR als Text einer formalen Sprache, der MIZAR Beweissprache, die einer Programmiersprache ähnlich ist, editiert werden:

```
environ
vocabularies ARYTM, ARYTM_1;
notations REAL_1, XXREAL_0, NUMBERS;
constructors REAL_1, XXREAL_0;
registrations XREAL_1, XREAL_0,REAL_1, ORDINAL1;
requirements SUBSET, BOOLE, NUMERALS, REAL, ARITHM;
theorems XREAL_1;
begin
reserve x, y, a for Real;
 a < 0 & x < y implies a * x > a * y
proof
```

[1] Das Projekt NAPROCHE wurde gegründet von Prof. Dr. Peter Koepke (Lehrstuhl für Mathematische Logik, Universität Bonn) und Prof. Dr. Bernhard Schröder (Lehrstuhl für Germanistik/Linguistik, Universität Duisburg-Essen). Der Verfasser des Beitrags arbeitet seit SS2007 im Projekt NAPROCHE mit. Die Homepage des Projekts hat die URL: http://www.math.uni-bonn.de/people/naproche/
[2] http://www.mizar.org/
[3] http://www.mizar.org/language/syntax.xml
[4] Axiom (A.3) ist folgendes Anordnungsaxiom: „Sind $x > 0$ und $y > 0$, so folgt $xy > 0$.“ [FORST, S.14].

```
assume
A1:  a<0;
assume
A2:  x<y;
set l=-a;
set m=l*x;
set n=l*y;
    l>0 by A1, XREAL_1:60; then
    l*x<l*y by A2,XREAL_1:70; then
    -m>-n by XREAL_1:26;
hence a*x>a*y;
end;
```

Die MIZAR Schlüsselwörter entsprechen natürlich sprachlichen Wortformen, die typischerweise in einfach verfassten mathematischen Beweisen verwendet werden: assume / angenommen; implies / (so) folgt; set / setze; then / dann; usw. Der Beweis verlangt die Anwendung bereits bewiesener Sätze oder Axiome. In MIZAR geschieht dies durch Referenz auf Namen von Sätzen, die in MIZAR Artikeldateien bereits bewiesen sind (z.B. Satz XREAL_1:60 im obigen MIZAR Beweis).

(2) Damit die ATP-Komponente[5] von MIZAR insbesondere Satzreferenzen korrekt auflösen kann, müssen eine Reihe von Umgebungsinformationen (Environment) gesetzt sein, die auf Bibliotheken bereits in MIZAR definierter Begriffe, Operatoren und bewiesener Sätze referenzieren. D.h. ein menschlicher Erfasser muss umfassend die korrekten Environment Schlüsselwörter kennen, die zu seinem Beweis nötig sind. Dieses ist eine Schwierigkeit beim Verfassen von MIZAR Beweisen[6].

NAPROCHE versucht das Problem der Mehrfacherfassung aufzuheben, indem der Vorgang der maschinellen Beweisprüfung direkt mit dem Editierwerkzeug für mathematische Beweise verbunden wird. Bei Mathematikern ist das Verfassen von mathematischen Texten in einer TeX-Sprache (TeX, LaTeX, TeXmacs, …) verbreitet. TeX-Sprachen verwalten mathematische Ausdrücke in Form von Klammerausdrücken.

3. Arbeitsgebiete des Projekts NAPROCHE

Im Projekt NAPROCHE können grob drei Arbeitsgebiete skizziert werden: (1) Texttechnologie: Im Zentrum steht die maschinelle Transformation von mathematischen Beweisen, die als Texte natürlicher Sprachen maschinenlesbar in Form von Klammerungssprachen (z. B. TeX-Sprachen) vorliegen, in XML Texte, deren Grammatik möglichst geeignet mathematische Beweisstrukturen abbildet. (2) Maschinelle Semantik[7]: Hierbei wird der in (1) gewonnene Text in eine Proof Representation Structure (PRS) überführt, die auf die Diskursrepräsenta-

[5] Automated theorem proving.

[6] Als Werkzeug hierfür wurde im Rahmen einer von mir betreuten Bachelor Arbeit ein Knowledge Management System für MIZAR (KMS-MIZAR) entwickelt [CHOU]. Das KMS-MIZAR besteht aus einem Datenbanksystem (DBS) zur Verwaltung von MIZAR Definitionen, Theoremen, Notationen und Registrierungen. Diese Entitäten wurden mit einer Akquisitionskomponente aus den MIZAR Abstractfiles extrahiert und in das DBS, das mit einem Oracle™ DBMS und Java/JDBC implementiert ist, eingefügt. Die Extraktion operiert mit vollständiger Erkennungsrate, da sie auf Grundlage der MIZAR BNF-Grammatik entwickelt wurde. Das DBS umfasst im Moment folgendes Mengengerüst:

Entitäten	**Anzahl**
Absfiles	1019
Definitions	11175
Theorems	47064
Notations	545
Registrations	7882

[7] Arbeitsgruppe von Prof. Dr. Schröder.

tionstheorie aufbaut ([KAMP], [SiKl], [FISS], [KOLE]). (3) Mathematische Logik[8]: Der im PRS Format vorliegende Beweis wird in ein FOL Format (First Order Logic) transformiert (z.B. TPTP[9]) [KÜHL]. Mit einem FOL Beweiser erhält der Anwender eine Rückmeldung, ob der Beweis korrekt ist oder in welchem Beweisschritt ein Fehler auftritt. Im folgenden wird auf Arbeiten im Arbeitsgebiet (1) eingegangen.

4. Grundlegende Formate, verwandte Arbeiten

MathML [MaML] wird als plattformunabhängiges Austauschformat für mathematische Texte eingesetzt. MathML beruht auf XML und stellt eine formale Semantik für mathematische Symbole, Formeln und Ausdrücke bereit. Zur Semantik von Beweisen wurde von [SrKo] eine auf XML beruhende Annotationssprache entwickelt (ProofML). ProofML wurde unter Berücksichtigung von OMDoc Konzepten [OMDo] weiterentwickelt [FISS]. Wichtige Ergebnisse zur Fachsprachenforschung auf dem Gebiet der Mathematik sind in [BECK] und [EISE] zusammengestellt.

5. Natürliche Sprache eines mathematischen Beweises als Transformationsgegenstand

Natürliche Sprache kommt in einem mathematischen Beweis vor und hat einige zentrale Aufgaben, wie den Fluss der mathematischen Argumente für potentielle Leser zu strukturieren, logische Figuren hervorzuheben und bestimmte Formelanteile zu erläutern. Das nachfolgende Beispiel ist der Beweis eines Satzes[10] aus der elementaren Differential- und Integralrechnung, dass sich zwei Stammfunktionen[11] einer gegebenen reellen Funktion nur um eine Konstante c unterscheiden:

Satz 2:
Voraussetzung: $F : I \rightarrow \Re$ ist eine Stammfunktion von $f : I \rightarrow \Re$.
Behauptung: $G: I \rightarrow \Re$ ist genau dann eine Stammfunktion von $f : I \rightarrow \Re$, wenn $F - G$ eine Konstante ist.
Beweis[12]**: „**a) [$\Leftarrow$] Sei $F - G = c$ mit einer Konstanten $c \in \Re$. Dann ist $G' = (F - c)' = F' = f$. [D.h. G ist damit Stammfunktion von f.]
b) [$\Rightarrow$] Sei G Stammfunktion von f, also $G' = f = F'$. Dann gilt $(F - G)' = 0$, daher ist $F - G$ konstant. (§16, Corollar3 [zum Satz von Rolle])."

Betrachtet man die sprachlichen Elemente dieses Satzes und seines Beweises, dann lassen sich diese in drei Bereiche einteilen: **A) Allgemeinsprachliche Wörter** der deutschen oder einer anderen natürlichen Sprache. Z.B.: also, dann, daher, genau, gilt, ist, sei, wenn. **B) Mathematische Begriffe**, die als fachsprachlich bestimmte Wörter einer natürlichen Sprache (z.B. der deutschen Sprache) vorliegen und eindeutig in einen entsprechenden Terminus einer anderen natürlichen Sprache (z.B. Englisch) übersetzt werden können. Z.B.: „Menge" / „set" , „Stammfunktion" / „antiderivative". **C) Mathematische Symbole**, **Ausdrücke** und **Formeln**. Z. B.: G' als Symbol der 1. Ableitungsfunktion der Funktion G. Die Bereiche A) und B) repräsentieren zusammengenommen den natürlich sprachlichen Anteil des Beweises. In dem obigen Beispiel können folgende Leistungsmerkmale der natürlichen Sprache in der Formulierung des Satzes und seines Beweises beobachtet werden: (1) **Erläuterung von Formeldaten**: Mit dem Wort „sei" wird typischerweise die Setzung eines Symbols

---

[8] Arbeitsgruppe von Prof. Dr. Koepke.
[9] http://www.tptp.org/
[10] Dieser Satz wird in diesem Beitrag aus Abkürzungsgründen durchgängig als „Satz 2" zitiert.
[11] Mit I wird ein Intervall in der Menge der reellen Zahlen $\Re$ bezeichnet. Es gilt $I \subseteq \Re$. Eine diffenenzierbare Funktion $F : I \rightarrow \Re$ heisst **Stammfunktion** einer Funktion $f : I \rightarrow \Re$, wenn für alle $x \in I$ gilt: $F'(x) = f(x)$ (hierbei ist $F'(x)$ die erste Ableitung von $F(x)$ an der Stelle x). Beispiel: Die Funktion $F(x) = x^3$ ist Stammfunktion der Funktion $f(x) = 3*x^2$.
[12] Der Beweis ist aus [FORST], S.140. Textergänzungen im Beweis sind in [...] gesetzt.

eingeleitet. Mit der Setzung „Sei F – G = c“ wird eine reelle Konstante c als Differenz von F und G deklariert. Die Rechtfertigung für diese Setzung folgt aus der Beweisrichtungsvoraussetzung, die besagt, das F – G konstant ist. (2) **Strukturierung der Argumentationsfolge**: Mit Formulierungen, wie „dann gilt“, „dann ist“ oder „man sieht“ können mehrere Einzelargumente pauschal zusammengefasst werden und als ein Schluss dargestellt werden. (3) **Hervorhebung logischer Figuren**: Die Verknüpfung „genau dann A, wenn B“ beschreibt eine Äquivalenz der Aussagen A und B, die auch durch die Formel A ⇔ B ausgedrückt werden kann.

Eine Aufgabe des NAPROCHE-Projekts besteht in der computerlinguistischen Untersuchung der natürlich sprachlichen Bestandteile mathematischer Beweise mit dem Ziel der maschinellen Klassifikation der erläuternden, der strukturierenden oder der logischen Funktion kurzer Wortfolgen in mathematischen Beweisen. Dieses Ziel ist Bestimmungsaspekt für die Gestaltung der Grammatik (DTD) des XML Zielformats der Transformation von TeX-Sprachen.

Der Umfang des allgemein sprachlichen Wortschatzes in mathematischen Lehrbüchern ist in der Regel klein. Das Lehrbuch „Grundlagen der Analysis“[13] von Edmund Landau [LAND] benötigt zur Formulierung von 301 mathematischen Sätzen mit zugehörigen Beweisen ein deutschsprachiges Vokabular mit ca. 580 Einträgen. Die Mehrzahl der Schlagwörter sind Wortformen der deutschen Allgemeinsprache. Ein weiteres Ziel der Texttechnologie im Rahmen des NAPROCHE Projekts ist die Entwicklung von Hilfsmitteln zur Trennung des allgemeinsprachlichen Anteils vom fachsprachlichen Anteil in mathematischen Beweisen. Zu diesem Zweck wird ein Datenbanksystem (DBS), das auf Landaus Deutsch-Englischem Glossar aufbaut, entwickelt.

6. Transformation eines maschinenlesbaren mathematischen Beweises in ein XML Format für natürlich sprachliche Beweise

In NAPROCHE wurde ein XML-Format definiert, das folgende Bedingungen erfüllen soll: (a) Es soll an die Gliederung von Standardtypen mathematischer Beweise (z.B. direkter Beweis, Widerspruchsbeweis, vollständige Induktion) angepasst sein. Es soll auf weitere Beweistypen erweiterbar sein (z.B. transfinite Induktion). (b) Es soll bezüglich mathematischer Formelanteile mit MathML übereinstimmen. (c) Es soll hinsichtlich PRS weiterverarbeitbar sein. Die Grammatik dieses XML-Formats kann mittels einer Dokumenttyp Definition (DTD) beschrieben werden (Satz.DTD)[14]. Die Hauptelemente dieser DTD werden nachfolgend erläutert:

(1) Das XML Wurzelelement ist <Satz>. Ein <Satz> kann mehrere <Voraussetzungen> beinhalten. Er hat eine Behauptung (<Beh>) und einen Beweis (<Bew>). Haupttestgegenstand der Entwicklung des Konverters waren Beweise, die innerhalb der Beweisschritte auf externe Hilfssätze bzw. Lemmata verwiesen. Für den Fall, dass der Beweis den Beweis integrierter Lemmata enthält, wurde für bestimmte Anwendungsfälle eine gesonderte DTD definiert (Bonn.DTD).

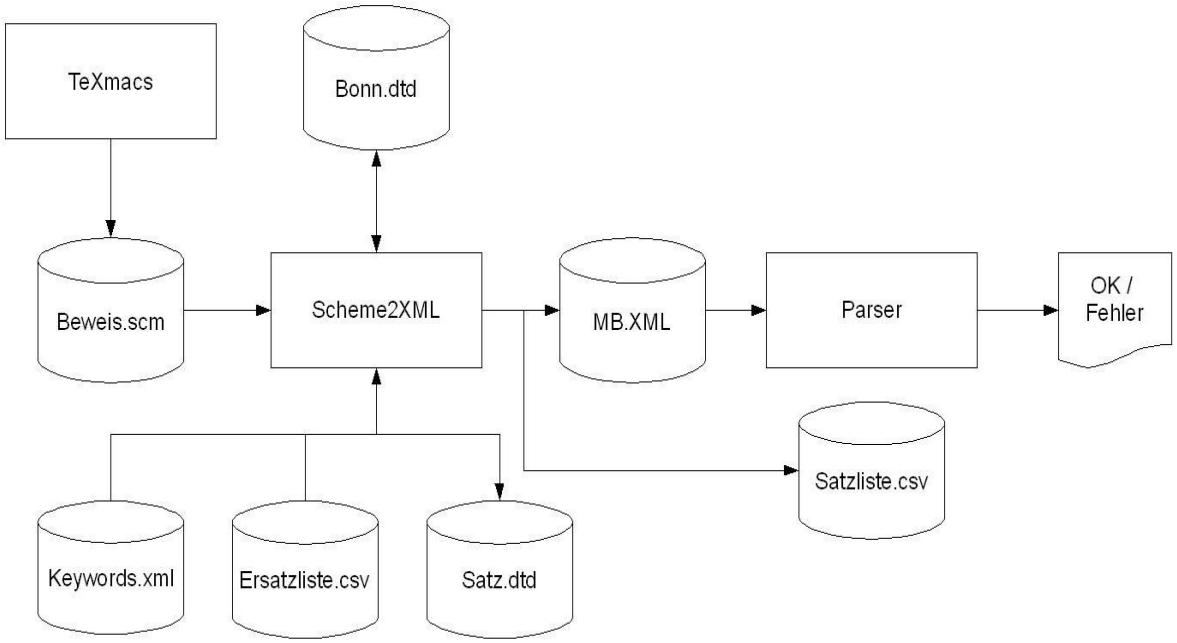
(2) Der Beweis folgt in seinem Aufbau einem Beweismodus (<BewModus>). Der Modus des Beweises von Satz 2 ist direkt (<DirBew>). Weitere Modi sind <IndBew> für Widerspruchsbeweise und <VollstIndukt> für die vollständige Induktion. Die Liste der Modi ist erweiterbar. Durch den Beweismodus wird gesteuert, ob der Beweis eine gegenüber dem direkten Beweis abweichende Substruktur hat, z.B. die Invertierung (<Invertierung>)

[13] Das Buch enthält ein nahezu vollständiges Deutsch-Englisches Vokabular, das anglophonen Studierenden ohne Deutschkenntnisse das leichte Erlernen der deutschen Sprache in mathematischen Texten ermöglichen soll.
[14] Diese DTD ist im Anhang dokumentiert.

(Annahme des logischen Gegenteils) beim indirekten Bewies oder die Induktionsverankerung bei der vollständigen Induktion.

(3) Jeder Beweis besteht aus einer Folge von Argumenten (<Arg>). Ein Argument besteht aus mindestens einer <Statementfolge>. Eine Statementfolge ist eine Kombination aus natürlich sprachlichem Text (<TEXT>) und mathematischen Formelanteilen. Die Formelanteile können einzelne mathematischen Symbole(<MATH>) oder aus mehreren Zeichen aufgebaute mathematischen Formeln sein (<MATHmodus>, <MATHreihe>, <MATHreihen>)[15]. In dem mit <TEXT> markierten XML-Element befindet sich der Hauptteil des in natürlicher Sprache verfassten Anteils der Formulierung eines Beweisschritts. Innerhalb der mathematischen Formeln (z.B. <MATHmodus> oder <MATHreihen>) können einzelne Textzeichen (z.B. „f“ als Funktionssymbol oder „a“, „x“ etc. als Formelbuchstaben) oder Wörter (z.B. „aus“, „Länge“, „in“ usw.) bzw. Wortfolgen auftreten. Diese Textelemente einer Formel werden durch das XML Element <FTEXT> verwaltet. Textsequenzen in <FTEXT> können ihrerseits neben Wörtern oder Buchstaben auch wiederum zusammengesetzte mathematische Symbole (z.B. Brüche <BRUCH>) oder einfache mathematische Symbole (z.B. das Symbol für die Menge der natürlichen Zahlen <bbb-N/>) enthalten.

(4) Mathematische Formeln (<MATHmodus>, <MATHreihe>, <MATHreihen>) können mit beliebiger Verschachtelungstiefe auftreten. Um dieses in XML rekursiv verwalten zu können wurde die ENTITY %Mathe eingeführt. Bei den XML Elementen für mathematische Symbole wurde sich an den Elementen von MathML orientiert. Z.B. wird das Symbol <bbb-R/> für das mathematische Symbol der Menge der reellen Zahlen verwendet. Enthält eine mathematische Formel eine Gleichung oder Ungleichung, wird dieses im Element <Statementtyp> notiert. Weiterhin wird dort festgehalten, ob innerhalb des Beweisschritts ein prädikatenlogischer Ausdruck auftritt. Tritt dabei ein Ausdruck mit Quantor auf, ist dieser als XML Element in der Formel ausgewiesen (Allquantor: <forall/> bzw. Existenzquantor: <exists/>).



Systemübersichtsdiagramm Scheme2XML

[15] Die Tatsache, dass es hier drei Tags zur Beschreibung mathematischer Formeln gibt, ist eine Folge unterschiedlicher Editieroptionen für komplexe Formeln in TeXmacs.

Auf Grundlage dieser DTD wurde ein Code-Transformator (Scheme2XML) spezifiziert. Hierbei wird von einem maschinenlesbaren Beweis in einem TeX Format ausgegangen. In Bezug auf das Referenzwerkzeug von NAPROCHE wurde als Eingangsformat der TeXmacs Scheme Code ausgewählt (Scheme2XML := TeXmacs Scheme Code to XML Converter). Eine erste Version des Code-Transformators wurde bereits implementiert[16]. Die Validitätskontrolle des Transformators wurde bisher auf einem Korpus von ca. 20 Beweisen aus der Algebra und der Analysis ausgeführt[17]. Zur Unterstützung der Beweistextsegmentierung wurde eine Schlüsselwortdatei verwendet (Keywords.xml). Hierin sind Wörter und Symbole notiert, die typischerweise Anfangs- und Endpositionen von Beweistextteilen markieren: Z.B. Beweisendesymbole: `qed.; q.e.d.; Qed.; Q.e.d.; qed;`

7. Aufbau eines DBS zu Landaus Deutsch-Englischem Glossar

Das Lehrbuch „Grundlagen der Analysis“ von Edmund Landau [LAND] enthält ein Deutsch-Englisches Glossar, das 301 mathematischen Sätzen mit zugehörigen Beweisen beinhaltet, abdeckt. Die Mehrzahl der Schlagwörter des Glossars sind deutsche Wortformen. Daneben treten als Schlagwörter auch Präfixe und Suffixe auf. Beispiele für Glossareinträge sind folgende:

aufzählen, (auf+zählen) to enumerate, to count, to list.
dann, then; **_und nur_**, if and only if.
ge-, prefix forming past participles of verbs.
heben, to lift (cog., heave).
lernend, (present participle of: lernen) learning.

Das Design der Datenbank basiert auf der syntaktischen Analyse der Glossar-Einträge in Form der Data Dictionary Notation (DDN) der Strukturierten Analyse [BALZ]. Nachfolgend ist ein Auszug aus den DDN-Produktionsregeln zu den Glossar-Einträgen gegeben:

LGEVART:=SW+(SYN_ANG)+(MORPH_ANG)+TR_LIST
TR_LIST:=TR_LIST1|TR_LIST2|…
TR_LIST1:={ENGLW+(ETYM_ANG)}
TR_LIST2:=TR_LIST1+{SUB_SW+TR_LIST1}
SW := catchword;
SYN_ANG := syntactic information;
MORPH_ANG := morphological information;
TR_LIST: list of translation;
ENGLW := English word;
ETYM_ANG := etymological information;
SUB_SW := “sub catchword”.

Das Datenbanksystem zu Landaus Glossar ist in erster Stufe erstellt. Es enthält den deutschen Schlagwortbestand (SW) mit Attributen, die Verweise auflösen (SUB_SW), Abkürzungen kennzeichnen, Schlagwörter als Partizipien ausweisen (SYN_ANG), die gfs. Angaben zum Schlagwort als Kompositum (MORPH_ANG) bzw. zur Wortherkunft machen (MORPH_ANG) und Präfixe bzw. Suffixe kennzeichnen. Weiterhin sind Tabellen zur englischen Übersetzung des deutschen Schlagworts aufgebaut, sowohl zur Verwaltung der englischen Zielphrasen als auch zur Verwaltung der Mehrdeutigkeit.

In der nächsten Ausbaustufe soll sowohl beim deutschen Schlagwortbestand als auch bei den englischen Phrasen in bestimmten Fällen notiert werden, ob es sich um Repräsentanten allgemeinsprachlicher Wörter oder um mathematische Fachwörter handelt. Bei den

[16] Sebastian Zittermann: „Entwicklung eines TeXmacs-to-XML-Parsers“, Bachelorarbeit, FH Köln, Institut für Nachrichtentechnik, September 2008.
[17] Ein Beispiel eines erzeugten XML Beweiscodes ist im Anhang gegeben.

allgemeinsprachlichen Wörtern soll in bestimmten Fällen markiert werden, ob sie logische Funktionen haben, Setzungen oder Annahmen (generell Beweisstrukturaspekte) markieren.

In der nächsten Scheme2XML-Version soll dieses DBS-Modul dann u. a. zur Generierung von XML-Tags eingesetzt werden, die bestimmte erläuternde, strukturierende oder logische Funktionen von allgemeinsprachlichen Wortfolgen in Beweisen kennzeichnen sollen.

8. Referenzen

[BALZ] Balzert, Helmut: "Lehrbuch der Software-Technik", Heidelberg [etc.] (Spektrum) 1996.

[BECK] Becker, Holger: "Semantische und lexikalische Aspekte der mathematischen Fachsprache des 19. Jahrhunderts", (Diss.) Universität Oldenburg 2005. http://docserver.bis.uni-oldenburg.de/publikationen/dissertation/2006/becsem05/becsem05.html

[CHOU] Chouaffé, Franck Edmond: „Entwicklung eines KMS (Knowledge Management System) für die Entwicklung von Mizar-Beweisen (KMS-MIZAR)" Bachelorarbeit FH Köln, Dezember 2008.

[EISE] Eisenreich, Günther: "Die neuere Fachsprache der Mathematik seit Carl Friedrich Gauß" in: L. Hoffmann, H. Kalverkämper, H. E. Wiegand et. al. (Hg.): "Fachsprachen – ein internationales Handbuch zur Fachsprachenforschung und Terminologiewissenschaft" , 1. Halbband, Berlin, New York (de Gruyter) 1998, S. 1222-1230.

[FISS] Fisseni, Bernhard: "Die Entwicklung einer Annotationssprache für natürlichsprachlich formulierte mathematische Beweise", (Magisterarbeit) Universität Bonn 2003.

[FORST] Forster, Otto: „Analysis I – Differential- und Integralrechnung einer Veränderlichen" Braunschweig/Wiesbaden (Vieweg) 1992.

[KAMP] Kamp, Hans; Reyle, U.: "From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory", Dordrecht (Kluwer) 1993.

[KOLE] Kolev, Nickolay: "Generating Proof Representation Structures for the Project NAPROCHE" (Magisterarbeit) Universität Bonn 2008.

[KÜHL] Kühlwein, Daniel: "Ein Kalkül für Proof Representation Structures" (Diplomarbeit) Universität Bonn 2008.

[LAND] Landau, Edmund: "Grundlagen der Analysis – With a Complete German-English Vocabulary", New York (Chelsea) 1965.

[MaML] Mathematical Markup Language (MathML) Version 2.0 (Second Edition), W3C Recommendation 21 October 2003: http://www.w3.org/TR/MathML2/ .

[OMDo] Kohlhaase, Michael: "OMDoc – An open markup format for mathematical documents" LNAI Nr.4180, Berlin (Springer) 2006.

[SiKl] Schielen, Michael; Klabunde, Ralf: "3.5.4 Diskursrepräsentationstheorie" in: K.-U. Carstensen et. al. (Hg.): "Computerlinguistik und Sprachtechnologie – Eine Einführung", Heidelberg (Spektrum) 2004, S.302-313.

[SrKo] Schröder, Bernhard; Koepke, Peter: „ProofML – eine Annotationssprache für natürliche Beweise". In: LDV-Forum, Nr.1,2 2003, ISSN 0175-1336, S.428-441.

6. Anhang

Auszug aus dem vom TCM2XML generierten XML Code des Beweises von Satz 2:

```
<Beweis>
 <BewModus>DirBew</BewModus>
  <BewArgFolge>
   <DirBew>
    ...
```

```
    <Arg ID= "1">
      <Statementfolge>
       <TEXT>(a) Sei</TEXT>
       <MATHmodus><FTEXT>F - G = c</FTEXT></MATHmodus>
       <Statementtyp>0;gl</Statementtyp>
       <TEXT>mit der Konstanten</TEXT>
       <MATHmodus><FTEXT>c <epsilon/> <bbb-R/></FTEXT></MATHmodus>
       <TEXT>.</TEXT>
      </Statementfolge>
    </Arg>
    <Arg ID= "2">
      <Statementfolge>
       <TEXT> Dann ist</TEXT>
       <MATHmodus><FTEXT>G'=(F-c)'=F'=f</FTEXT></MATHmodus>
       <Statementtyp>0;gl,gl,gl</Statementtyp>
       <TEXT>.</TEXT>
      </Statementfolge>
    </Arg>
   ...
  </DirBew>
 </BewArgFolge>
</Beweis>
```

Satz.DTD, eine Grammatik, die das Zielformat der Konvertierung von TeXmacs Scheme Code nach XML bestimmt(Scheme2XML):

```
<!-- DTD  : Satz.DTD                                          -->
<!-- Zweck: Eine DTD fuer einfach strukturierte               -->
<!--        mathematische Beweise                             -->
<!-- Verf.: Gregor Buechel                                    -->
<!-- Ueberarbeitung: Sebastian Zittermann                     -->
<!-- Stand: 27.11.2008                                        -->
<!-- Version: 3.x (Neue Versionszaehlung)                     -->
<!-- ##################################################### -->
<!-- Die XML-Wurzel ist: Satz.                                -->
<!--     Ein Satz braucht eine ID.                            -->
<!--     Ein Satz hat SatzInformationen.                      -->
<!--     Ein Satz kann mehrere Voraussetzungen haben.         -->
<!--     Ein Satz hat eine Behauptung.                        -->
<!--     Ein Satz hat einen Beweis.                           -->
<!-- ##################################################### -->

<!-- Tagalternativen (Wg. dynam. Ergaenzung einzelner Tags) -->
<!ENTITY % Mathe "MATH|MATHmodus|MATHreihe|MATHreihen|FTEXT">
<!ENTITY % TagsMitInhalt
"FTEXT|MSUP|MSUB|Auflistung|KLAMMERN|BRUCH|int
|uplimit|lowlimit|Ableitung|nicht|WURZEL|REIHE|CELL">
<!ENTITY % EinzelneTags "epsilon|bbb-R|gtr|bbb-N|forall|exists|geq
|bbb-Q|OR|bbb-Z|wedge|alpha|less|rightarrow|xi|leq|varepsilon
|leqslant|subset|searrow|nearrow|assign|infty|ldots|frak-a|neq
|in|thicksim|lambda|Opt-Voraussetzungen">

<!ELEMENT SATZ
(SatzInformationen,Voraussetzungen*,Behauptung,Beweis)>
<!ATTLIST SATZ
    ID CDATA #REQUIRED
>
<!-- ##################################################### -->
<!-  SatzInformationen      :                                -->
<!-- Der Satztitel muss enthalten sein.                       -->
<!-- Das mathematische Teilgebeit kann angegeben werden.      -->
<!-- Der Autor kann angegeben werden.                         -->
<!-- Es koennen mehrere Bemerkungen hinterlegt werden.        -->
```

```
<!-- ##################################################    -->
<!ELEMENT SatzInformationen
(Satztitel,Mathemteilgebiet?,Quelle?,Bemerkung*)>
<!ELEMENT Satztitel (TEXT|%Mathe;)*>
<!ELEMENT Mathemteilgebiet (TEXT|%Mathe;)*>
<!ELEMENT Quelle (TEXT|%Mathe;)*>
<!ELEMENT Bemerkung (TEXT|%Mathe;)*>

<!-- ##################################################### -->
<!-- Voraussetzungen:                                     -->
<!-- Eine Voraussetzung besteht aus einer einzelnen       -->
<!-- Voraussetzung (Vor).                                 -->
<!-- Einer einzelnen Voraussetzung wird                   -->
<!-- eine ID zugeordnet.                                  -->
<!-- ##################################################### -->
<!ELEMENT Voraussetzungen (Vor+)>
<!ELEMENT Vor (Statementfolge)>
<!ATTLIST Vor
    ID CDATA #REQUIRED
>

<!-- ################################################## -->
<!-- Behauptung:                                        -->
<!-- Behauptung besteht aus einer Statementfolge.       -->
<!-- ################################################## -->
<!ELEMENT Behauptung (Statementfolge)>

<!-- ################################################## -->
<!-- Beweis:                                            -->
<!-  Drei verschiedene Beweismodi sind derzeit spezifi-  -->
<!-  ziert. Jeder Beweismodus hat seine eigene          -->
<!-  BewArgFolge.                                       -->
<!-- ################################################## -->
<!ELEMENT Beweis (BewModus,BewArgFolge)>
<!ELEMENT BewModus (#PCDATA)>
<!ELEMENT BewArgFolge (VollstIndukt|IndBew|DirBew)>

<!-- ################################################## -->
<!-- direkter Beweis:                                   -->
<!-- ################################################## -->
<!ELEMENT DirBew (Arg+)>
<!ELEMENT Arg (Statementfolge+)>
<!ATTLIST Arg
    ID CDATA #REQUIRED
>

<!-- ################################################## -->
<!-- indirekter Beweis:                                 -->
<!-- ################################################## -->
<!ELEMENT IndBew
(Statementfolge?,Invertierung,InvertBew,Widerspruch)>
<!-- Entspricht dem Aufbau der Behauptung               -->
<!ELEMENT Invertierung (Statementfolge)>
<!-- Entspricht dem Aufbau des Beweises                 -->
<!ELEMENT InvertBew (Arg*)>
<!ELEMENT Widerspruch (Statementfolge)>

<!-- ################################################## -->
<!-- vollst.Induktion:                                  -->
<!-- ################################################## -->
<!ELEMENT VollstIndukt (Induktionsverankerung,Induktionsschluss+)>
<!ELEMENT Induktionsverankerung (Statementfolge,Statementtyp?)>
```

```
<!ELEMENT Induktionsschluss
(Statementfolge?,Induktionsannahme,Induktionsbeh,Induktionsbeweis)>
<!-- Induktionsannahme entspricht Voraussetzungen        -->
<!ELEMENT Induktionsannahme (Statementfolge+)>
<!ELEMENT Induktionsbeh (Statementfolge+)>
<!ELEMENT Induktionsbeweis (Arg*)>

<!-- kann in beliebiger Reihenfolge mathematische Teile  -->
<!-- wie z.B. praedikatenlogische Ausdruecke enthalten   -->
<!ELEMENT Statementfolge
((TEXT|((MATH|MATHmodus|MATHreihen|MATHreihe),Statementtyp?))+)>

<!-- Ein Textbereich -->
<!ELEMENT TEXT (#PCDATA)>

<!-- Statementtyp: Aufbauf wie CSV -->
<!-- erste Stelle (0 oder 1) gibt an, ob es eine praedikatenlogische
Aussage beinhaltet -->
<!-- hinter Semikolon steht, ob es sich um eine Gleichung (gl) oder
Ungleichungen (ungl) handelt -->
<!ELEMENT Statementtyp (#PCDATA)>

<!-- "Inhalt" von den drei "MATH"s wird dynamisch anhand der
Fehlerexceptions bestimmt-->
<!ELEMENT MATH (%Mathe;|%TagsMitInhalt;|%EinzelneTags;)*>
<!ELEMENT MATHmodus (%Mathe;|%TagsMitInhalt;|%EinzelneTags;)*>
<!ELEMENT MATHreihe (%Mathe;|%TagsMitInhalt;|%EinzelneTags;)*>
<!ELEMENT MATHreihen (%Mathe;|%TagsMitInhalt;|%EinzelneTags;)*>
<!ELEMENT FTEXT (#PCDATA|%TagsMitInhalt;|%EinzelneTags;)*>

<!-- "verkettete" Elemente -->
<!ELEMENT MSUP (mi+)>
<!ELEMENT MSUB (mi+)>
<!ELEMENT mi (#PCDATA|%TagsMitInhalt;|%EinzelneTags;)*>

<!ELEMENT Auflistung (Item+)>
<!ELEMENT Item (#PCDATA|%TagsMitInhalt;|%EinzelneTags;)*>
<!ATTLIST Item
    ID CDATA #REQUIRED
>
<!ELEMENT KLAMMERN (#PCDATA|%TagsMitInhalt;|%EinzelneTags;)*>
<!ATTLIST KLAMMERN
    typ CDATA #REQUIRED
>
<!ELEMENT KlammerInd (#PCDATA|%TagsMitInhalt;|%EinzelneTags;)*>
<!ELEMENT KlammerExp (#PCDATA|%TagsMitInhalt;|%EinzelneTags;)*>

<!ELEMENT BRUCH (Zaehler,Nenner)>
<!ELEMENT Zaehler (#PCDATA|%TagsMitInhalt;|%EinzelneTags;)*>
<!ELEMENT Nenner (#PCDATA|%TagsMitInhalt;|%EinzelneTags;)*>

<!ELEMENT lowlimit (#PCDATA|%TagsMitInhalt;|%EinzelneTags;)*>
<!ELEMENT uplimit (#PCDATA|%TagsMitInhalt;|%EinzelneTags;)*>

<!-- einzelne Tags mit Inhalt-->
<!ELEMENT Ableitung (#PCDATA|%TagsMitInhalt;|%EinzelneTags;)*>
<!ELEMENT nicht (#PCDATA|%TagsMitInhalt;|%EinzelneTags;)*>
<!ELEMENT WURZEL (#PCDATA|%TagsMitInhalt;|%EinzelneTags;)*>
<!ELEMENT REIHE (#PCDATA|%TagsMitInhalt;|%EinzelneTags;)*>
<!ELEMENT CELL (#PCDATA|%TagsMitInhalt;|%EinzelneTags;)*>

<!-- einzelne Zeichen -->
```

```
<!-- element von -->
<!ELEMENT epsilon (#PCDATA)>
<!-- Menge reelle Zahlen -->
<!ELEMENT bbb-R (#PCDATA)>
<!-- groesser als (>) -->
<!ELEMENT gtr (#PCDATA)>
<!-- Menge der natuerlichen Zahlen -->
<!ELEMENT bbb-N (#PCDATA)>
<!-- "Fuer Alle"-Symbol -->
<!ELEMENT forall (#PCDATA)>
<!-- Existiert-Symbol -->
<!ELEMENT exists (#PCDATA)>
<!-- groesser-gleich -->
<!ELEMENT geq (#PCDATA)>
<!-- Menge der rationalen Zahlen -->
<!ELEMENT bbb-Q (#PCDATA)>
<!-- oder-Symbol  -->
<!ELEMENT OR (#PCDATA)>
<!-- Menge der ganzen Zahlen -->
<!ELEMENT bbb-Z (#PCDATA)>
<!-- /\ -Symbol (und) -->
<!ELEMENT wedge (#PCDATA)>
<!-- Alpha -->
<!ELEMENT alpha (#PCDATA)>
<!-- kleiner als -->
<!ELEMENT less (#PCDATA)>
<!-- Pfeil nach rechts -->
<!ELEMENT rightarrow (#PCDATA)>
<!-- Symbol griech. Buchstabe XSI -->
<!ELEMENT xi (#PCDATA)>
<!-- kleiner gleich -->
<!ELEMENT leq (#PCDATA)>
<!-- Symbol Epsilon -->
<!ELEMENT varepsilon (#PCDATA)>
<!-- kleiner gleich (schraeges gleich) -->
<!ELEMENT leqslant (#PCDATA)>
<!-- Teilmenge von (wie C) -->
<!ELEMENT subset (#PCDATA)>
<!-- Pfeil nach unten-rechts (Southeast) -->
<!ELEMENT searrow (#PCDATA)>
<!-- Pfeil nach oben-rechts (Northeast) -->
<!ELEMENT nearrow (#PCDATA)>
<!-- ":=" -->
<!ELEMENT assign (#PCDATA)>
<!-- Unendlichkeits-Symbol -->
<!ELEMENT infinity (#PCDATA)>
<!-- 3 Punkte (...) -->
<!ELEMENT ldots (#PCDATA)>
<!-- altdeutsches a -->
<!ELEMENT frak-a (#PCDATA)>
<!-- unlgeich (durchgestrichenes =) -->
<!ELEMENT neq (#PCDATA)>
<!-- epsilon fuer "ist Element von" -->
<!ELEMENT in (#PCDATA)>
<!-- Integralzeichen -->
<!ELEMENT int (#PCDATA)>
<!-- Tilde (~) -->
<!ELEMENT thicksim (#PCDATA)>
<!-- Lambda-Zeichen -->
<!ELEMENT lambda (#PCDATA)>
<!ELEMENT Opt-Voraussetzungen (#PCDATA)>
```