Acquisition of Human Expert Knowledge for Rule-based Knowledge-based Systems using Ternary Grid

Vom Fachbereich Ingenieurwissenschaften der
Universität Duisburg-Essen
zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigte Dissertation

von

Yuliadi Erdani
aus
Indonesien

Referent: Prof. Dr.-Ing. A. Hunger
Korreferent: Prof. Dr.-Ing. H.-D. Kochs

Tag der mündlichen Prüfung: 24.06.2005

Acquisition of Human Expert Knowledge for Rule-based Knowledge-based Systems using Ternary Grid

to Ibu, Bapak, Istriku Endah dan Putriku Rumaisha

Acquisition of Human Expert Knowledge for Rule-based Knowledge-based Systems using Ternary Grid

– Summary –

Knowledge acquisition is a task in Artificial Intelligence (AI) concerned with eliciting and representing knowledge of human experts and the most important element in the development of expert system. It deals with extracting knowledge from sources of expertise and transferring it to a knowledge base. Knowledge acquisition is still a major and critical phase in the development of expert systems and the most difficult and error-prone task that knowledge engineer does while building a knowledge-based system or an expert system. It is not an easy task to acquire knowledge from human expert not trained in knowledge engineering. The performance of the knowledge due to the redundant and faulty information is performed by interaction between experts and knowledge engineer during acquisition process.

The need to overcome the knowledge acquisition bottleneck is also recognised beyond the knowledge acquisition community. In order to solve that problem and to achieve that mentioned performance, a rule-based knowledge acquisition system using Ternary Grid is developed.

The Ternary Grid is a model of rule-based knowledge in a grid format where every cell represents the relation between a rule and a fact. The cell of the grid can only allow value “0”, “1” or “2” that is called ternary value. Value “1” represents the condition part of a rule. Value 2 the conclusion part of a rule. Value “0” represents relation less between rule and fact. The Ternary Grid is convenient for processing the knowledge. They may be directly viewed as task domain and production rule structure. The grid has elements as *problem-solving domains,* which can be derived into sub domains or group of rules, rows as *rules*, columns as *facts* and values as *IF-THEN syntax*.

The Ternary Grid knowledge acquisition and representation work in a model domain using concept matrix or set operation. The organisation and logical content of expert knowledge in Ternary Grid can be easily inspected and analysed. Completion and recognition of patterns which consists of “0 or empty”, ”1”, “2”values in the grid are facilitated by the structure and relative compactness of the matrix representation. Representation in the Ternary Grid facilitates optimising and testing for conditions of ambiguity, redundancy, completeness and correctness.

Acknowledgments

I would like to express my thanks to the numerous people and parties who have helped me in completing this work.

First and foremost, I want to thank my supervisor Prof. A. Hunger for his guidance and many worthwhile advices. His constant quest for meaningful research and result has taught me invaluable lesson for my future career. I want to thank the members of my dissertation committee, Prof. H.-D. Kochs, Prof. D. Jäger, Prof. P. Jung, Prof. G. Krost. I also want to thank Stefan Werner with whom I did discussions concerning the content and the process of this work.

I want to thank Prof. H.-D. Kochs for his readiness to be my second reviewer (Korreferent) and Prof. W. Geisselhardt who has given me advice, motivation and written recommendation letters for extension my DAAD-scholarship. For support with my every-day lab work I want to thank Joachim Zumbrägel and Bernd Holzke, who helped me with supporting and cordially serving computer and network. For interesting discussion on all kind of topics I want to thank to Prof. Mahamod bin Ismail, Mr. Bagio Budiarjo, Henner Heck, Kerstin Luck, Marc Zimmermann, Frank Schwarz, Stefan Freinatis, Kalamullah Ramli, Ahmad Jaiz, Mrs. Riri Fitri, Sascha Mertens, Marius Rosu, Phan Cong Chinh, Astha Ekadianto, Iqbal Taswar and Robinson Fornge. For non technical supports, I want to thank to Mrs. Elvira Laufenburg.

For financial support, I would like to thank German Academic Exchange Service (DAAD), who paid for my scholarship. I would like also to thank the Institute for Multimedia and Software Engineering of the University Duisburg-Essen, who provided me all financial supports for many conferences and their trips.

I thank my parents Nanik Sularni and Erlan Dasuki for having me brought up to appreciate the value of respect, attitude and education and for their love and spiritual support. I thank my wife Endah Sugihartati for her love and patience throughout my time as a PhD student. I am very grateful with my daughter Rumaisha Muthiara Erdani who has been my best support. I also thank my parents-in-law Ratu Sugiharti, Sodikin and Ratu Eni Sekarwati for their love and spiritual supports.

Abstract

Knowledge acquisition is the most important part in the development of expert system. It deals with extracting knowledge from sources of expertise and transferring it to a knowledge base. Knowledge acquisition is major research field in knowledge engineering and still the most difficult and error-prone task for knowledge engineer while building an expert system. This situation influences the performance of the knowledge due to the quality of information and the reduction of error possibility.

It is not an easy task to acquire knowledge from human expert not trained in knowledge engineering. The performance of the knowledge is performed by interaction between experts and knowledge engineer or machine during acquisition process.

In most rule-based expert system, building of rules can easily be done. Knowledge engineer or expert does not have to do any work specifying rules and how they are linked to each other. Sometime the knowledge engineer or expert can reference rules or facts that have not yet been created. It seems to be a simple and an instant work. The problem due to the performance of the knowledge will not occur until the number of rules is getting higher. Some problem may appear in the form of inconsistent rules, unreachable rules, redundant rule and rotating chain of rules.

In order to solve that problem and to achieve that mentioned performance, a rule-based knowledge acquisition system using Ternary Grid is developed. This system acquires knowledge from human expert using grid or matrix system. Ternary Grid represents a model of rule-based knowledge in a grid or matrix format.

Keywords: knowledge acquisition, knowledge elicitation, rule-based knowledge-based system, rule-based expert system, grid application

Table of Contents

List of Tables

List of Figures

Chapter 1

Problem Situation in Knowledge Acquisition

1.1. The Nature of Knowledge

"Knowledge is Power", wrote the English philosopher and lawyer Francis Bacon in 1597. This term leads to the meaning that “In the knowledge lies the power". The term “knowledge” has been the motto of most Artificial Intelligence researchers since the 1970s. Philosopher, writer, book have attempted to answer the question *“what is knowledge?”* [FEM83] suggest clarifying this that knowledge is *not* synonymous with information. Rather, knowledge is information that has been interpreted, categorised, applied and revised. [WAP81] reported that knowledge can be exemplified by concept, constraint, and heuristic method for using probabilistic data, and principles that governs domain-specific operations. [HAY83] contends that domain knowledge consists of descriptions, relationships, and procedures. More specifically, knowledge consists of symbolic description of definitions, symbolic description of relationships, and procedures to manipulate both type of descriptions.

[FID91] stated that the successful practice of knowledge elicitation does not, paradoxically, rely on understanding the nature of knowledge. Such understanding has eluded philosophers for as long as epistemology has been an active branch of philosophy. As far as practising knowledge elicitation skills is concerned it is a very much smaller subset of human knowledge that is relevant.

Machines can not think. Machine can deal only with zeros and ones. Only natural things such as human experts can posses intelligence and understand the world view. We must understand what human experts know so that we can emulate some aspect of their performance on a machine. The machine does not need an entire world view, life experience, or even all expert specific knowledge. It performs a given task, within

constraints, for a given purpose. Nor does the machine need to carry out the task in exactly the same way experts do as long as it simulates a degree of accuracy agreed by those concerned.

The nature of knowledge itself in a philosophical sense, though it may be of interest to others, is not dealt with here. We maintain that the nature of expert knowledge is adequately defined, for our practical purposes, by the practice of attempting to understand it for machine emulation. Much the same considerations apply to knowledge of human psychology. As far as successfully practising knowledge elicitation skills is concerned, understanding the way the mind works is useful, but knowledge of the theories, that have ever been proposed will not necessarily guarantee adequate results from knowledge elicitation, either in obtaining the right information or the right amount of it. The practitioner needs to know where and how to apply this knowledge. This can best be learned by getting involved in a project and gaining real experience [FID91].

1.2. Difficulties in Knowledge Acquisition

Knowledge acquisition is a task in Artificial Intelligence (AI) concerned with eliciting and representing knowledge of human experts. Knowledge Acquisition is the transfer and transformation of potential problem solving expertise from some knowledge source to a program [BUC83]. Knowledge acquisition is phases of expert systems development that progress virtually together [IGN91]. These are basic for the development of an integrated rule base of the expert system to be built. Many professionals on this field have exposed a common view of that knowledge acquisition is an extremely hard task to accomplish.

The acquisition of knowledge is a major and critical phase in the development of expert systems and is still the most difficult and error-prone task that knowledge engineer does while building a knowledge-based system or an expert system [RHE01]. One of the most fundamental and still unsolved problems in knowledge acquisition goes by the name of knowledge acquisition bottleneck which is coined by [HAY83] and [FEI84].
Knowledge acquisition is considered by many to be the most difficult and precarious stage in the knowledge engineering process [SMI96], [TSA94] noted that this was

because knowledge acquisition involves communications between people with completely different backgrounds, human experts and knowledge engineers, who must formulate the concepts, relations and control mechanisms needed for the expert system. [HOF87], [ROO89], [BYR92], [LIE93], [HWA94], amongst others, stated that knowledge acquisition has often been described as the bottleneck in knowledge based systems development. [KEY89] noted that the process of gathering knowledge is a fuzzy process. [BYR92] has further described knowledge acquisition as a separate and distinct process from knowledge engineering. The cost and performance of the application depends directly on the quality of the knowledge acquired [RHE01].

The problem of the knowledge acquisition or elicitation from skilled person or expert is well known in the literature of psychology. [BAI79] has reviewed the difficulties of verbal debriefing and notes that there is no necessary correlation between verbal report and mental behaviour, and that many psychologist feel strongly that verbal data are useless. Other school of psychology see the problem not as methodological but as psychological and resulting from cognitive defences that obstruct internal communication for a variety of reasons [FRE14] and [ROG67].

The term *"knowledge acquisition bottleneck"* constricts the building of an expert system like an ordinary bottleneck constricts fluid flow into a bottle. It's a hard work (long and difficult) for an expert to explain (all) his knowledge and reasoning used to solve problems in a specific domain, and then code all this knowledge into facts and rules. Several techniques were developed in order to achieve this task, but no one can really explain all the knowledge used by the expert to solve a problem. Human reasoning is a complex task and we need to take into consideration that all knowledge - even though related to a very small problem - can't be obtained and/or may not be available from the expert, due to common sense, particular cases, implicit context relations, etc.

Two main reasons why the knowledge acquisition is difficult [HAR86]:

- The expert will usually have insufficient knowledge about programming and expert system techniques, and
- The expert will find it difficult to describe his knowledge completely and

correctly.

The difficulties of knowledge acquisition can appear during transferring and expressing the knowledge from human expert into computer or machine and structuring the knowledge [VIL99]. The reasons conduct to that difficulties are:

- Experts may lack time or knowledge,
- Testing and refining knowledge is complicated,
- Poorly defined method for knowledge elicitation
- The knowledge maybe incomplete
- Difficult to recognised specific knowledge when mixed with irrelevant data

In the connection to expert system, the difficulties of knowledge acquisition system influence the development process of expert system. [ONM89] attempted through surveys to demonstrate why the vast majority of expert systems fail. Some of the reasons noted by him and others include:

- The lack of user participation in design [REE96]
- The lack of structure and organization of knowledge acquisition [MCH89]
- Communication problems between the knowledge engineer and the domain expert [ONM89]
- Failure in identifying the right candidates for knowledge acquisition [STE93]
- Failure of verification and validation.

[MCH89] described in more detail the lack of structure and organization of knowledge acquisition. This lack can result in the following:

- Lack of traceability from knowledge source to code
- Redundancy of knowledge gathered
- Increased development time and cost
- Difficulties scoping the focus of knowledge acquisition
- Knowledge that is acquired at the wrong level or abstraction
- Lack of appropriate documentation
- Wasted knowledge engineer time

- Disgruntled domain expert

The problem of knowledge acquisition and knowledge representation are vital to the integrity of the rule base for the expert system to be constructed. Knowledge Acquisition can be extremely frustrated as well as time consuming. It has been stated by many Knowledge Engineers as the bottleneck of expert systems development. Because this phase is intimately related to domain experts, it involves all the problems of dealing with people. How does one best elicit the facts and rules within the human expert's knowledge base? The conclusion is that this process is mainly an art rather than a science [IGN91].

The need to overcome the knowledge acquisition bottleneck is also recognised beyond the knowledge acquisition community [CHK01]. For example DARPA's funding the Rapid Knowledge Formation (RFK) initiative [DAR00] seeks to address precisely this issues. The central objective of this Program is to enable experts to enter and modify knowledge directly and easily, without the need for specialised training in knowledge representation, acquisition, or manipulation. The resulting knowledge bases will be available to provide specific answers to questions and could be applied in many different problem-solving situations.

Many various techniques for knowledge acquisition used to acquire knowledge from the expert without considering lack on knowledge it self. Lack on knowledge mostly caused during elicitation process. There are some reasons are considered as causations for these lacks [PSU02]:

- Expert may lack in know-how technique
- Expert is unaware of knowledge used
- Expert is unable to verbalise the knowledge
- Expert may provide irrelevant knowledge
- Expert may provide incomplete knowledge
- Expert may provide incorrect knowledge
- Expert may provide inconsistent knowledge
- Expert may provide redundant knowledge

This is due to the endless problems encountered dealing with human beings. Knowledge acquisition technique has to meet and overcome problems such as domain expert's fears, incompetence, inability to communicate, and ultimately the open opposition from the domain expert.

1.3. Motivation

All situations mentioned in section 1.2 above refer to the fact that the bottleneck in creating an effective expert system lies in the acquisition process of the knowledge. Furthermore the difficulties of the knowledge acquisition affect the performance of the knowledge concerning the quality of information and the reduction of error possibility. This has been my motive to overcome those problems. We have been proposed the new technique for knowledge elicitation and acquisition called Ternary Grid knowledge elicitation/acquisition technique [EHW04]. This technique provides solutions to some of those problems mentioned above [EHW05a], [EHW05b], [EHW05c], [EHW05d].

There is not a best way to elicit and acquire knowledge as in a cooking recipe format because this is more an art than a science [IGN91]. However there are ways to improve the manner in which the knowledge engineer approaches the domain, the domain expert or experts and their own organisational ways to accomplish the task.

This paper describes some of the advantages than can be gained by rule-based knowledge acquisition with Ternary Grid elicitation technique. The term "Ternary Grid" comes from the combination of words "ternary" and "grid" that represent the model of rule-based knowledge structure in grid format. The grid has two axes. Vertical Axis represents the rule and horizontal axis represents the fact. The value of the grid can only contain ternary value i.e. "0", "1" or "2". Value "1" represents the condition part of the rule. Value "2" represents the conclusion part of the rule. Value "0" means, neither condition nor conclusion part of the rule is represented. Every appeared value maps the relation between the fact and the rule. The rule-based knowledge is represented in the IF-THEN format.

The idea of this work is inspired and can be traced back to knowledge acquisition systems using grid- or matrix-based technique, like Repertory Grid Analysis [KEL55] , Repertory Grid [Shaw, 1980], Formal Concept Analysis (FCA), first developed by [WIL82], is a mathematically based method of finding, ordering and displaying formal concepts [WIL92], Multiple Classification Ripple Down Rules (MCRDR) [KCP95],[KAN96], Knowledge Acquisition Tool based on Personal Construct Psychology [GAS95], WebGrid [GAS96] and Epistemic Matrix-based Technique [EPI04].

The experience of knowledge acquisition systems above suggested that grid or matrix technique and representation are convenient for processing the knowledge. In the past, knowledge engineer believed that rule-based knowledge could be acquired relative easily and applied into expert system immediately. In practice, they often found significant mismatches between the acquired knowledge and the knowledge needed for required purpose. This occurs when the number of knowledge is getting higher. The tracking of the rule becomes more complicated. Sometime some situations can miss and not covered by existing rules or conflict with another situation. The developed Ternary Grid techniques should be able to overcome from these problems. Furthermore the problem solving or reasoning method can be developed and implemented using this Ternary Grid.

With the goal of developing this knowledge acquisition system and believing in the importance of improving the elicitation process by using Ternary Grid elicitation technique I define the architecture for the knowledge acquisition system, implement the algorithms of elicitation process and develop the knowledge acquisition tool based on Ternary Grid. The interface for the acquisition process from the expert is built in this architecture as well.

Knowledge acquisition tool developed here provides support for creating new knowledge, modifying existing knowledge, transferring the knowledge into knowledge base and enable the possibility of adaptation of any knowledge format.

The example of expert system for applying this developed knowledge acquisition

system has been CongaXpert [HUW99], [EHW03]. It is a web-based expert system that is used for consultation system in academic area in the University of Duisburg-Essen Germany.

I motivate both the goal and methodology of the work presented here which covers the advantages of rule-based knowledge, the need for accomplishing the performance of the knowledge, the method of acquiring rule-based knowledge from human expert using Ternary Grid, the role of the expert interface in supporting the visual presentation of mentioned grid and the contribution of this research work in developing of rule-based knowledge acquisition system.

1.4. Objectives

The aim of knowledge acquisition system developed here is to achieve the performance of rule-based knowledge due to the quality of information and reduction of error possibility that is performed by direct interaction between experts and knowledge engineer during elicitation process.

The objective is to design a knowledge acquisition system and develop its tool that can achieve that performance mentioned above. The system should be able to guide the expert in creating, modifying and maintaining the knowledge base and provide an environment for developing expert system that allows the experts to add new knowledge without needing to understand the details of system organisation and implementation.

1.5. Organisation of the Thesis

The main body of this thesis consists of nine chapters. Chapter 1 describes the background idea and the motivation of the development of the knowledge acquisition system using Ternary Grid technique. It describes the difficulties of knowledge acquisition and their impacts to the performance of knowledge. Chapter 2 describes

briefly theories in artificial intelligence that relate to this research work. Chapter 3 describes briefly related works that have given inspiration for this work. Chapter 4 describes the design of the developed knowledge acquisition system. It describes also the introduction to the whole concept of the system. Chapter 5 concerns the process of obtaining factual knowledge. Chapter 6 presents the core idea of this work. It deals with the elicitation process as main part of the knowledge acquisition process using Ternary Grid. The transferring process of knowledge is described in Chapter 7. Chapter 8 describes the implementation part of the whole concept. It deals with the development of algorithms and software application as knowledge acquisition tool named KasTerGrid (Knowledge Acquisition System using Ternary Grid). The experimental results that have been gained from the developed system are presented in chapter 9. Chapter 10 presents contributions of the research and summary of comparison to related work.

Chapter 2

A Brief Review of Related Theory In Artificial Intelligence

2.1. Artificial Intelligence

The research area of the Artificial Intelligence deals with computer engineering for solving problems that requires obviously humans intelligence for its accomplishment. Artificial Intelligence is a technique of the data processing which deals with the process of logical thinking, learning and perception.

Artificial intelligence (AI) is a broad field, and means different things to different people. It is concerned with getting computers to do tasks that require human intelligence. However, having said that, there are many tasks which we might reasonably think require intelligence - such as complex arithmetic - which computers can do very easily. Conversely, there are many tasks that people do without even thinking - such as recognising a face - which are extremely complex to automate. AI is concerned with these difficult tasks, which seem to require complex and sophisticated reasoning processes and knowledge [CAW03].

The following definition from [BAF81] is representative of opinion in the field:
"Artificial Intelligence (AI) is the part of computer science concerned with designing intelligent computer systems, that is, systems that exhibit the characteristics we associate with intelligence in human behaviour - understanding language, learning, reasoning, solving problems, and so on."

Of course, the term *intelligence* covers many cognitive skills, including the ability to solve problems, learn, and understand language; AI addresses all of those. But most

progress to date in AI has been made in the area of problem solving -- concepts and methods for building programs that analyse (*reasoning method)* problems rather than calculate a solution.

In other words, AI is concerned with programming computers to perform tasks that are presently done better by humans, because they involve such higher mental processes such as perceptual learning, memory organization and judgemental reasoning [MIN68]. Thus, writing a program to perform complicated statistical calculations would not be seen as an artificial intelligence activity, while writing a program to design experiments to test hypotheses would. Most people are not very good at doing long calculations by hand, whereas computers excel at such tasks. On the other hand, devising good experiments to test hypotheses is a skill that the research scientist derives partly from training and partly from experience. Programming a computer to perform such a task would be entirely non-trivial.

We can find almost as many definitions of artificial intelligence as we can find people working in the field. Some descriptions of artificial intelligence are [WIM86]:

- Artificial intelligence is the study of ways to make computers be intelligent.
- It is an attempt to make a computer respond like human being.
- Artificial intelligence is a misleading term that makes people expect to get something for nothing, that is, intelligence in a computer.
- Artificial intelligence is the ability of a manmade system to deal with unplanned realities and survive them.
- It is the study of how to make computers do things that, so far, people do better.
- Artificial intelligence is the ability of a computer to resolve uncertainly through whatever processes it has available to it.
- The term is becoming so overused that it is fast on its way to becoming a buzzword.
- Artificial intelligence is the science of modelling human intelligence.
- The term should be used only for programs that emulate the human thought process.

- The notion of self-awareness, or introspection, is a key component of artificial intelligence.

There are differences of outlook and emphasis among researchers, however, some incline towards the view that AI is a branch of engineering, since it is ultimately about building intelligent artefacts, such as robots [NIL71]. Others stress the link with cognitive science: a discipline which concerns itself with the study of human information processing, and sometimes uses computers to model or simulate such processing. Still other writers are interested in the overlap with problems of philosophy associated with knowledge and consciousness.

In the end, AI is about the emulation of human behaviour: the discovery of techniques that will allow us to design and program machines which simulate or extend our mental capabilities. It is therefore hardly surprising that the discipline should be closely related to a wide range of other academic subject areas such as computer science, psychology, philosophy, linguistics and engineering. The fact that AI crosses a number of traditional interdisciplinary boundaries sometimes causes friction, but is more often a source of inspiration and new ideas.

As an aid to the general reader, I will attempt to give a very brief overview of artificial intelligence research, insofar as it relates to the design and construction of expert systems. I will also try to explain in what way knowledge-based programming differs from both more conventional programming techniques and the general-purpose problem solving methods devised by the pioneers of AI research. For a more general introduction to AI, the reader is referred to textbooks cited in the Bibliographical notes at the end of the chapter.

In the classical period Artificial intelligence is scarcely younger than conventional computer science; the beginnings of AI can be seen in the first game-playing and puzzle-solving programs written shortly after 2nd World War. Game-playing and puzzle-solving may seem somewhat remote from expert systems, and insufficiently serious to provide a theoretical basis for real applications. However, a rather basic notion about computer-based problem solving can be traced back to early attempts to program computers to perform such tasks.

The mid-1960s to the mid-1970s represents what so called the Romantic Period in artificial intelligence research. At this time, people were very concerned with making machines 'understand', by which they usually meant the understanding of natural language, especially stories and dialogue. SHRDLU system was arguably the climax of this epoch [WIN72]: a program which was capable of understanding a quite substantial sub set of English by representing and reasoning about a very restricted domain (a world consisting of children's toy blocks).

The Modem Period stretches from the latter half of the 1970s to the present day. It is characterized by an increasing self-consciousness and self-criticism, together with a greater orientation towards techniques and applications. The flirtation with psychological aspects of understanding is somehow less central than it was.

The disillusionment with general problem-solving methods, such as heuristic search, has continued apace. Researchers have realized that such methods overvalue the concept of 'general intelligence', traditionally favoured by psychologists, at the expense of the domain-specific ability that human experts possess. Such methods also undervalue simple common sense, particularly the ability of humans to avoid, identify and correct errors.

2.2. Knowledge Engineering

The term "knowledge engineering" is often used to mean the process of designing, building, installing an expert system or other knowledge- based system. In other words, knowledge engineering is the whole process of making a Knowledge Base System or Expert System from the beginning to the end.

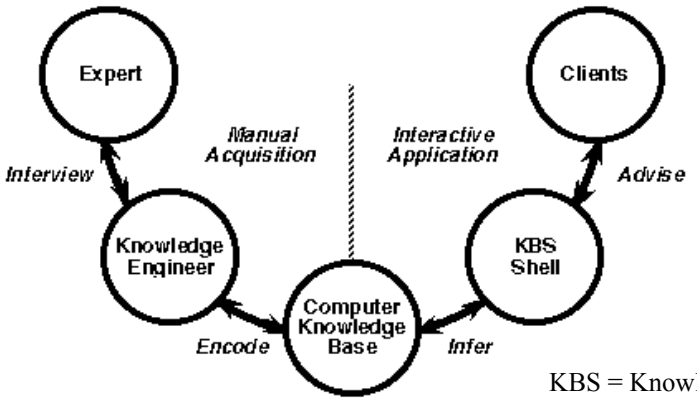
Knowledge engineering refers to the process of acquiring knowledge from a human expert and shaping it in such a way that it can be used efficiently in a knowledge-based system. As we see it, knowledge engineering can be divided into several distinct components: expert system project management, expert system design, knowledge

elicitation, knowledge programming and user interface design. These components are reflected in the chapters of this book. In addition, the knowledge engineer needs to consider what tools to use for building an expert system.

The knowledge engineer elicits knowledge about a particular problem such as how to invest wisely on the stock-market - and represents this knowledge in a computer program in such a way that it can be used on subsequent occasions to solve similar problems.

The basic model for knowledge engineering has been that the knowledge engineer mediates between the expert and knowledge base, eliciting knowledge from the expert, encoding it for the knowledge base, and refining it in collaboration with the expert to achieve acceptable performance. Figure 2.1 shows this basic model with manual acquisition of knowledge from an expert followed by interactive application of the knowledge with multiple clients through an expert system shell [GAS93]:

- The knowledge engineer interviews the expert to elicit his or her knowledge;
- The knowledge engineer encodes the elicited knowledge for the knowledge base;
- The shell uses the knowledge base to make inferences about particular cases specified by clients;
- The clients use the shell's inferences to obtain advice about particular cases.



KBS = Knowledge-based System

Figure 2.1 Basic model of knowledge engineering

However, the knowledge engineer in the role of an intermediary between the expert and the knowledge-based systems may create as many problems as he or she solves [GAI87]. The computer itself is an excellent tool for helping the expert to structure the knowledge domain and in recent year's research on knowledge acquisition has focused on the development of computer-based acquisition tools [GAB88], [BOO89].

Figure 2.2 specifies multiple knowledge engineers since the tasks above may require the effort of more than one person, and some specialization may be appropriate. Multiple experts are also specified since it is rare for one person to have all the knowledge required, and, even if this were so, comparative elicitation from multiple experts is itself a valuable knowledge elicitation technique [BOO87], [SHW87], [SHG88], [GAS89]. Validation is shown in Figure 2.2 as a global test of the shell in operation with the knowledge base that is of overall inferential performance. However, validation may also be seen as a local feature of each step of the knowledge engineers' activities: the experts' proper use of the tools needs validation; the operation of the tools themselves needs validation; the resultant knowledge base needs validation; and so on. Attention to quality control through validating each step of the knowledge acquisition process is the key to effective system development.

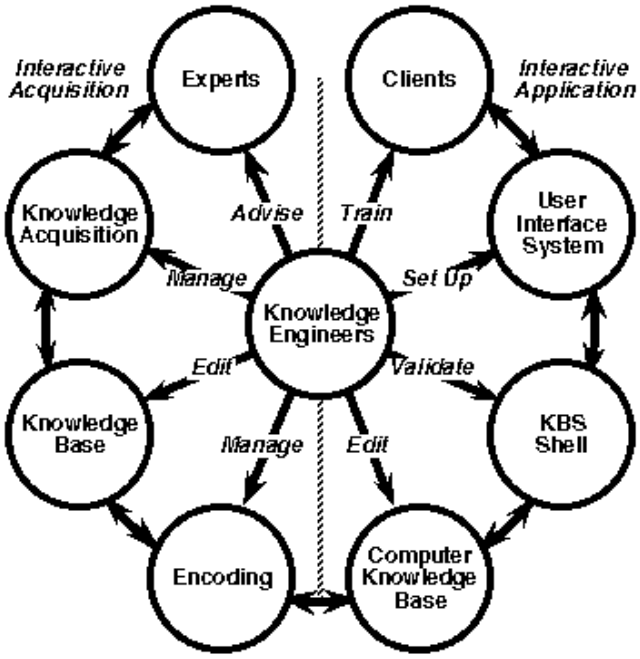


Figure 2.2 multiple knowledge engineer tasks

Knowledge engineering is a rapidly evolving discipline which depends on advances in theoretical areas such as artificial intelligence and cognitive science. Already there is talk of a second generation of expert systems in which more advanced techniques and principles are to be applied. There is no doubt that we will see many new developments in knowledge engineering theory and practice over the next few years

Knowledge Engineering was in the past primary concerned with building and developing knowledge-based systems, an objective which puts Knowledge Engineering in a niche of the world-wide research efforts - at best. This has changed dramatically: Knowledge Engineering is now a key technology in the upcoming knowledge society. Companies are recognizing knowledge as their key assets, which have to be exploited and protected in a fast changing, global and competitive economy. This situation has led to the application of Knowledge Engineering techniques in Knowledge Management. The demand for more efficient (business to) business processes requires the interconnection and interoperation of different information systems. But information integration is not an algorithmic task that is easy to solve: much knowledge is required to resolve the semantic differences of data residing in two information systems. Thus Knowledge Engineering has become a major technique for information integration. And, last but not least the fast growing World Wide Web generates an ever increasing demand for more efficient knowledge exploitation and creation techniques. Here again Knowledge Engineering technologies may become the key technology for solving the problem [STU00].

2.3. The Role of the Knowledge Engineer

The knowledge engineer must have the right inter-personal skills [FEM84], [WEL83]). Some of these are described as the role of the knowledge engineer below:

Good communication skills - The knowledge elicitation will involve many hours of discussion and argument. Results have to be recorded, and conclusions or models agreed with the expert. This necessitates the effective use of the spoken and written word, diagrammatic representation, and interpretation of body language. Above all there

must be a good relationship between the knowledge engineer and expert. A poor communicator cannot make a good knowledge engineer.

Intelligence - The knowledge engineer is continually learning. As he starts a new project he needs to be able to learn about a new knowledge domain, and understand enough of the terminology and principles to be able to discuss it fully with an acknowledged expert. He must keep up-to-date with advances in hardware and software. In addition, he needs to have knowledge of subjects such as formal logic, probability theory and psychology, and be able to appreciate the relevance of developments in these subjects. He needs to keep an open mind and be able to tackle problems in different ways.

Tact and diplomacy - The success of the project will depend on the cooperation of a small number (often one) of important experts. An expert who has been alienated by thoughtless or tactless treatment will tend to lose interest. Any suggestion that a program can replace or outperform the expert can be disastrous. So is an intimation that the expert is failing to provide the right information in an appropriate way.

Empathy and patience - The knowledge engineer and expert must work together as a team, each respecting the other. This means that the knowledge engineer must appreciate the problems faced by the expert. He needs to encourage without being patronizing, to argue without appearing self-opinionated, and ask for clarification without appearing critical. If he realizes the reasons for the expert's hesitancy or apparent incoherence then he will be able to exercise sufficient patience.

Persistence - Results may come slowly. Ultimately, there must be no gaps or inconsistencies, but during development there may be many. In order to resolve problems the knowledge engineer must persist; he must retain his enthusiasm and belief in the project. Despite setbacks, he must persist in the conviction that success will come.

Logicality - The inference mechanism of the expert system must be consistent and logical. During knowledge elicitation, especially the early stages, the expert's explanations may seem confused or fragmented. The elicitor needs to be able to argue reasonably, recognizing valid statements and providing meaningful counterexamples for possible errors. The completeness and consistency of the emerging model must also be assessed. All of this requires a level of clear thought and logicality.

Versatility and inventiveness - Until recognized methodologies are developed, or shells are produced with very flexible structures; the methods used to elicit a model of expertise will rely heavily on the knowledge engineer. Using his judgement he will have to select methods which seem appropriate, and abandon those which are not effective. He may have to discard early results or models, and if necessary invent representations which suit the expert and the domain. This requires an informed, versatile approach to the project, together with an ability and willingness to try new ideas.

Self confidence - The combination of these qualities and skills must be matched by self confidence. A shy or immature person, however technically able, would not be able to control a project. The development of an expert system is a challenge, and the knowledge engineer must have enough self confidence to sustain enthusiasm for the project. At the same time this confidence must not result in bombastic or patronizing behaviour.

Domain knowledge - The knowledge engineer has to talk to the expert using the expert's terminology. It would be advantageous, therefore, for the knowledge engineer to have some background knowledge of the domain, for example, the types of problems encountered, the terminology, accepted methods and tools.

Programming knowledge - The knowledge base and inference mechanisms used by the expert will be implemented in a program. It is advisable, but not essential, that the knowledge engineer understands programming and the various forms of knowledge representation available, e.g. semantic nets and frames. However, during knowledge elicitation an intelligent and versatile approach is most important, and some deficiencies in computer science experience can be tolerated.

It is unlikely that a knowledge engineer would have all these qualities, since personnel for a particular project are often sought from existing staff, rather than employing new specialists. This list is included to give an idea of the optimal person's talents and the ways in which they are likely to be needed. However, it is certain that the selection of the knowledge engineer will have a crucial effect on the success of a project. It is useful at this point to explore the tasks required of the knowledge engineer [HAR86].

2.4. Expert Systems and Knowledge Base Systems

An expert system is a set of programs that manipulate encoded knowledge to solve problems in a specialized domain that normally requires human expertise. An expert system's knowledge is obtained from expert sources and coded in a form suitable for the system to use in its inference or reasoning processes. The expert knowledge must be obtained from specialist or other sources of expertise, such as texts, journal, articles, and database. This type of knowledge usually requires much training and experience in some specialized field such as medicine, geology, system configuration, or engineering design. Once a sufficient body of expert knowledge has been acquired, it must be encoded in some form, loaded into a knowledge base, then tested, and refined continually throughout the life of the system.

There is no official definition for the term of expert system but there are some descriptions for it created by people working in the field of expert system.

"*Expert Systems are computer programs that are derived from a branch of computer science research called Artificial Intelligence (AI). AI's scientific goal is to understand intelligence by building computer programs that exhibit intelligent behaviour. It is concerned with the concepts and methods of symbolic inference, or reasoning, by a computer, and how the knowledge used to make those inferences will be represented inside the machine* [FEE93]".

"*Expert System is a system that employs human knowledge captured in a computer to solve problems that ordinarily require human expertise. Well designed systems imitate the reasoning processes experts use to solve specific problems* [TUR98]."

Expert systems differ from conventional computer system in several important ways.

- Expert systems use knowledge rather than data to control the solution process. "In the knowledge lays the power" is the important word of expert systems. Much of knowledge used is heuristic in nature rather than algorithmic.
- The knowledge is encoded and maintained as an entity separate from the control program. As such, it is not compiled together with the control program it self. This permits the incremental addition and modification (refinement) of the knowledge base without recompilation of the control program. Furthermore, it is possible in some case to use different types of expert systems. Such systems are known as expert system shells since they may be loaded with different knowledge bases.
- Expert systems are capable of explaining how a particular conclusion was reached, and why requested information is needed during a consultation. This is important as it gives the user a chance to assess and understand the system's reasoning ability, thereby improving the user's confidence in the system.
- Expert systems use symbolic representation for knowledge (rules, networks, or frames) and perform their inference through symbolic computation that closely resembles manipulation of natural.
- Expert systems often reason with meta-knowledge; that is, they reason with knowledge about themselves, and their own knowledge limits and capabilities.

Expert systems can perform some task which requires expertise. Such tasks often have one or more of the following characteristics.

- The task may be difficult to specify.
- The task may have incomplete or uncertain data.
- There may not always be an optimum solution.
- The task cannot be solved in a step-by-step manner
- Solutions are often obtained by using accumulated experience.

Expert systems can bring the following benefits.

- They can preserve valuable knowledge which would otherwise be lost when an expert system is no longer available.
- They can allow an expert to concentrate on more difficult aspect of the task.
- They can enforce consistency.
- They can perform dangerous tasks which would otherwise be carried out by humans.

Expert systems can be applied to a wide range of application. A suitable classification of these applications can often help to identify the methods that are best suited for implementing a particular application. A common classification produces the following groups.

- Diagnostic System / Fault diagnosis
- Design System
- Planning System
- Instruction System
- Consultation System (advisory system)
- Monitoring and Control System

The Knowledge-Based System (KBS) is the first realisation of research in the field of Artificial Intelligence, in the form of a software technology. For developers of application software, particularly in medical and engineering disciplines, it was a boon, as it addressed the decision-making process with the use of symbols rather than numbers. Tasks belonging to the classification and diagnosis category were the first to benefit from the emergence of KBS technology. Though AI researchers were carrying out symbolic processing much earlier, the results of such research could be taken from *lab to field* only when KBS was introduced as a software tool for addressing a class of problems that required simulation of the knowledge-based decision-making process.

The term expert systems are reserved for programs whose knowledge base contains the knowledge used by human experts, in contrast to knowledge gathered from textbooks or non-experts. More often than not, the two terms, expert systems and KBS, are used

synonymously. Taken together, they represent the most widespread type of AI application [FEE93].

KBS is computer programs designed to act as an expert to solve a problem in a particular domain. The program uses the knowledge of the domain coded in it and a specified control strategy to arrive at solutions. As knowledge base forms an integral, but implicitly understood part of a KBES, the adjective knowledge-based is often not used. The terms *expert system* and *knowledge-based expert system* can therefore be used interchangeably. An expert system is not ca11ed a program, but a system, because it encompasses several different components such as knowledge base, inference mechanisms, explanation facility etc. All these different components interact together in simulating the problem-solving process by an acknowledged expert of a domain [KRI96].

2.5. Production Rule and Rule-Based Knowledge

Production rule are simple but powerful forms of knowledge representation providing the flexibility of combining declarative and procedural representation for using them in a unified form. The term production rule came from production system which is developed by [NEW72]. A production system is a model of cognitive processing, consisting of a collection of rules (called *production rules*, or just *productions*). Each rule has two parts: a *condition* part and an *action (conclusion)* part. The meaning of the rule is that when the condition holds true, then the action is taken. A typical production rule is given below:

IF (mathematic score >= 60%) **AND** (physic score > 58%)
THEN student passed the examination

The statement of the rule above means that a student can pass the examination if he/she has got mathematic score more than or equal 60% and physic score more than 58%.

Production system or production rule provides appropriate structures for performing and describing search process. A production system has four basic components as enumerated below [KRI96]:

- A set of rules following the classical IF-THEN construct. If the conditions on the left-hand side are satisfied, the rule is fired, resulting in the performance of action on the right-hand side of the rule.
- A database of current facts established during the process of inference.
- A control strategy which specifies the order in which the rule are selected for matching of antecedents by comparing the facts in the database. It also specifies how to resolve conflicts in selection of rule or selection of facts.
- A rule firing module.

Instead of representing knowledge in a relatively declarative, static way (as a bunch of things that are true), rule-based system represent knowledge in terms of a bunch of rules that tell you what you should do or what you could conclude in different situations. A rule-based system consists of a bunch of IF-THEN *rules*, a bunch of *facts*, and some *interpreter* controlling the application of the rules, given the facts.

There are two broad kinds of rule system: *forward chaining* systems, and *backward chaining* systems. In a forward chaining system you start with the initial facts, and keep using the rules to draw new conclusions (or take certain actions) given those facts. In a backward chaining system you start with some hypothesis (or goal) you are trying to prove, and keep looking for rules that would allow you to conclude that hypothesis, perhaps setting new sub goals to prove as you go. Forward chaining systems are primarily data-driven, while backward chaining systems are goal-driven. We'll look at both, and when each might be useful.

Production systems have advantages over conventional programming languages that make them ideal for task domains in which knowledge may change or grow over time, and in which the initial problem state and final solution state may differ from user to user: they are flexible, modular, and plausible [SHA96].

Production Systems Are Flexible

Production systems use the same basic IF-THEN format to represent knowledge in very different domains. An informal Automated Tourist Guide rule, for example, expresses a very different content from the rules, given earlier, for closing Tweetie's cage or for applying suntan lotion.

Production Systems Are Modular

In an ordinary computer program one procedure calls another, in such a way that a change to one procedure may entail the modification of any others that call it. Simply removing a procedure may well result in the collapse of the whole program. In contrast, the functional units of a production system -- the set of rules in its rulebase -- are independent, self-contained chunks of knowledge, any one of which can be altered or replaced without disabling the entire production system and without requiring the modification of other rules. Such alterations might modify or restrict the behaviour of the system, but will not cripple it. This is because the rules in a production system are separate from the program that runs them: the rules do not interact with one another directly but only through changes to the working memory. whose head matches the relevant items in the database and whose body executes the appropriate action.

The most important development in production systems has been in the building of expert systems: computer programs that have the knowledge and expertise, in the form of hundreds or even thousands of rules, that will enable them to operate at the level of the human expert in some specialist domain. This makes them valuable as consultants in medicine, management, engineering, computer-system configuration, and chemical analysis, to name but a few areas where expert systems are in regular use. Such systems provide the support of expert knowledge that is relatively cheap (a full-time human consultant commands a much higher salary!), reliable (humans do make mistakes), portable (human experts are scarce, and sometimes too busy to come on call), and untiring (human experts have to sleep sometimes; eventually they die); and, because of the modularity of such systems, they can be extended to become more proficient than any human expert whose knowledge has been `written into' the rulebase. Expert systems need not store information uniquely in the form of production rules.

Production Systems Are Plausible

Human experts do not simply apply their knowledge to problems; they can also explain exactly why they have made a decision or reached a particular conclusion. This facility is also built into the expert systems that simulate human expert performance: they can be interrogated at any moment and asked, for example, to display the rule they have just used or to account for their reasons in using that rule. That a system is able to explain its reasoning is in itself no guarantee that the human user will understand the explanation: if the advice is to be of use, it is important that the system be able to justify its reasoning process in a cognitively plausible manner, by working through a problem in much the same way as a human expert would. Production systems, and the more sophisticated expert systems, can be made to reason either forwards, from initial evidence towards a conclusion, or backwards, from a hypothesis to the uncovering of the right kind of evidence that would support that hypothesis, or by a combination of the two. One significant factor which will determine whether a system will use forward or backward reasoning is the method used by the human expert.

If all the necessary data are either pre-given or can be gathered, and if it is possible to state precisely what is to be done in any particular set of circumstances, then it is more natural and likely that a human being -- and hence any machine modelling human performance -- would work forward from the data towards a solution.

Chapter 3

A Brief Review of Related Works

3.1. Repertory Grid

The Repertory Grid is a simple knowledge elicitation technique devised by clinical psychologists [KEL55]. After identifying a small set of *elements* (objects, entities), the user is asked to define some *constructs* (attributes, slots), which characterise those elements. Construct values can be given for each element on a limited scale between two range end-points (the left and right *poles*).

George Kelly introduced the *repertory grid* as a way of representing personal constructs as a set of distinctions made about elements relevant to the problem domain. In clinical psychology this domain will often be personal relationships, and the elements may be family members and friends. It is a systematic theory of human cognition based on the single primitive of a *construct*, or dichotomous distinction. This theory was developed in the context of clinical psychology with the goal of having techniques to bypass cognitive defences and to elicit the construct systems underlying the behaviour of an individual.

The term *repertory* derives, of course, from *repertoire* - the repertoire of constructs which the person had developed. Because constructs represent some form of judgement or evaluation, by definition they are scalar: that is, the concept *good* can only exist in contrast to the concept *bad*, the concept *gentle* can only exist as a contrast to the concept *harsh*. Any evaluation we make - when we describe a car as *sporty*, or a politician as *right-wing*, or a sore toe as *painful* - could reasonably be answered with the question 'Compared with what?' The process of taking three elements and asking for two of them to be paired in contrast with the third is the most efficient way in which the two poles of the construct can be elicited. The meaning of construct and element are:

A **Construct** is something that can be qualitative or quantitative,
i.e. bigger, longer, better, etc.
Elements are names of things such as, Mr. President, Monika or all Trades

A *repertory grid* is a two-way classification of data in which events are interlaced with abstractions in such a way as to express part of a person's system of cross-references between his *personal observations* or experience of the world (*elements*) and his *personal constructs* or classifications of that experience [SHG87].

In the context of an expert system, the elements are the things that are used to define the area of the topic, and can be concrete or abstract entities. They should be of the same type and level of complexity, and should span the topic as fully as possible. It is usual to start with about 6 to 12 elements. The *constructs* define the attributes along which the elements are distinct or similar.

The repertory grid technique is used in many fields for eliciting and analysing knowledge and for self-help and counselling purposes. The technique is essentially matrix-based although it is more complex than simply filling-in a matrix of elements. When used in knowledge engineering, the technique usually involves the following four main stages as described below [EPI04].

Stage 1

In stage 1 the concepts (called elements) are selected for the grid. For the technique to be successful and not take too much time to operate, the number chosen should be no less than about 7 and no more than about 15. A set of about the same number of attributes (called constructs) is also required. These should be such that the values can be rated on a continuous scale. The attributes can be taken from knowledge previously elicited or generated during the session using triadic elicitation.

Stage 2

Stage 2 involves the rating of each concept against each attribute. A numerical scale is often used, say 1 - 9. For instance, if the concepts are planets in the solar system, each might be rated on its distance from the sun (1 meaning close to the sun, 9 meaning far away), and so on through the other attributes.

Stage 3

In stage 3, the ratings are applied to a statistical calculation called cluster analysis to create a focus grid. These calculations ensure that concepts with similar scores are grouped together in the focus grid. Similarly, attributes that have similar scores across the concepts are grouped together in the focus grid. An example of a focus grid for planets of the solar system is shown in the figure 3.1 below.

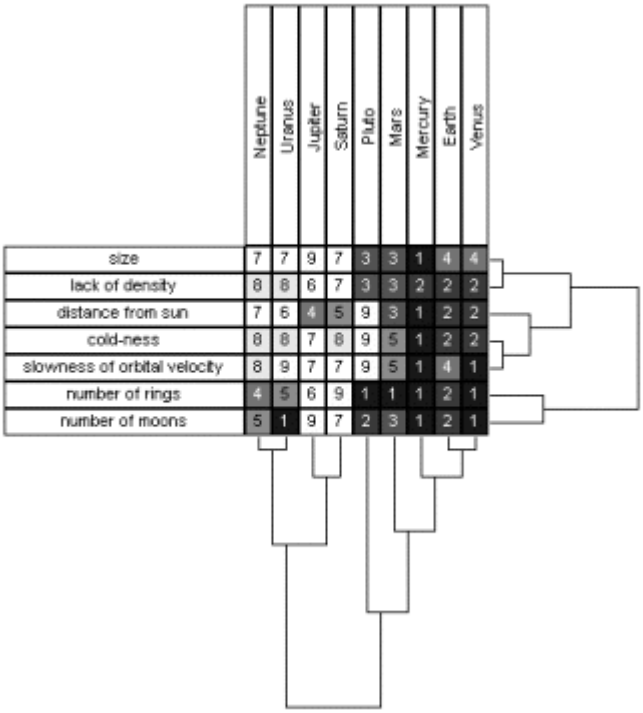


Figure 3.1 an example of a focus grid for planets of the solar system

The structures to the bottom and to the right of the gird shown above are dendrograms that indicate the strength of correlations. For instance, the lower dendrogram shows Neptune and Uranus as being very similar planets, and the right-hand dendrogram indicates a correlation between size and lack of density.

Stage 4

In stage 4, the knowledge engineer walks the expert through the focus grid gaining feedback and prompting for knowledge concerning the groupings and correlations shown. If appropriate, extra concepts or attributes are added and then rated to provide a

larger and more representative grid. In this way the technique can be used to uncover hidden correlations and causal connections.

Assessing Element Similarity

In order to assist in the elicitation of further attributes, or to refine the values already given to defined attributes, it is possible to assess the pair wise similarity of known elements. There are three different distance measures provided. In any of the given measures, the lower the distance between two elements, the more similar the elements are deemed to be.

The measures are [WHI97]:

Euclidean Distance

The number of attributes defines the number of dimensions of the space. The distance between two elements *in a single dimension* is the distance between the values of an attribute. In the current configuration, which has a granularity of 5, the maximum distance in any one dimension is 4 (= 5 - 1). To find the distance in n dimensions, use the Euclidean distance:

$$d = \sqrt{(x1 - y1)^2 + (x2 - y2)^2 + \ldots + (xn - yn)^2}$$

Manhattan Distance

This measure simply sums the distances in each dimension, so the distance measure in n dimensions is the following (in which the vertical bars '|' denote "the absolute value of"):

$$d = \sum_{i=1..n} |xi - yi|$$

Hamming Distance

This measure gives a value of 0 or 1 to the distance between two elements in any one dimension. The distance is 0 if the assigned values are the same and 1 otherwise. The

hamming distance over n dimensions is therefore the number of attributes which have different values:

$$d = \sum_{i=1..n} f(xi, yi)$$

Where f(xi,yi) 0 if xi=yi, otherwise 1

3.2. Web Grid

WebGrid is a knowledge acquisition and inference server on the World Wide Web that uses an extended repertory grid system for knowledge acquisition, inductive inference for knowledge modelling, and an integrated knowledge-based system shell for inference.

WebGrid offer knowledge elicitation, analysis, comparison, modelling and inference services to remote users access it through the Internet using a standard web browser such as Netscape or Internet Explorer. Its user interface is entirely through the interactive generation of active HTML documents using standard web capabilities such as typographic text, forms, images, links and clickable maps. It is designed to minimize server administration by conforming to the web's stateless protocol and storing the data being elicited in hidden fields within the documents returned to the user. This enables the user to manage their own data storage as if they were operating a local application by storing the HTML documents returned by WebGrid on their local disks.

A description of WebGrid, associated applications and example applications, can be found in our associated paper [GAS96]. WebGrid is a port of the KSS0/RepGrid knowledge acquisition tools to operate as a server on the World Wide Web [CPC91], allowing a web client on any platform world-wide to be used for knowledge modelling and inference. The system is interesting for a number of reasons:

- It provides widely available access to knowledge-based system development tools
- It is open in its architecture and designed to support integration with other systems

- The repertory grid technology is extended to support data types other than rating scales, such as categorical data, integers, floats and dates
- The inductive modelling methodology can generate rules, rules with exceptions, factored rules with exceptions and ripple-down rules
- The performance engine is integrated so that test cases may be checked and, if appropriate, corrected and posted into the dataset to change the model
- The acquisition, modelling and inference tools are designed to reason correctly with open data having don't care or unknown values

Figure 3.2 shows some features of WebGrid.

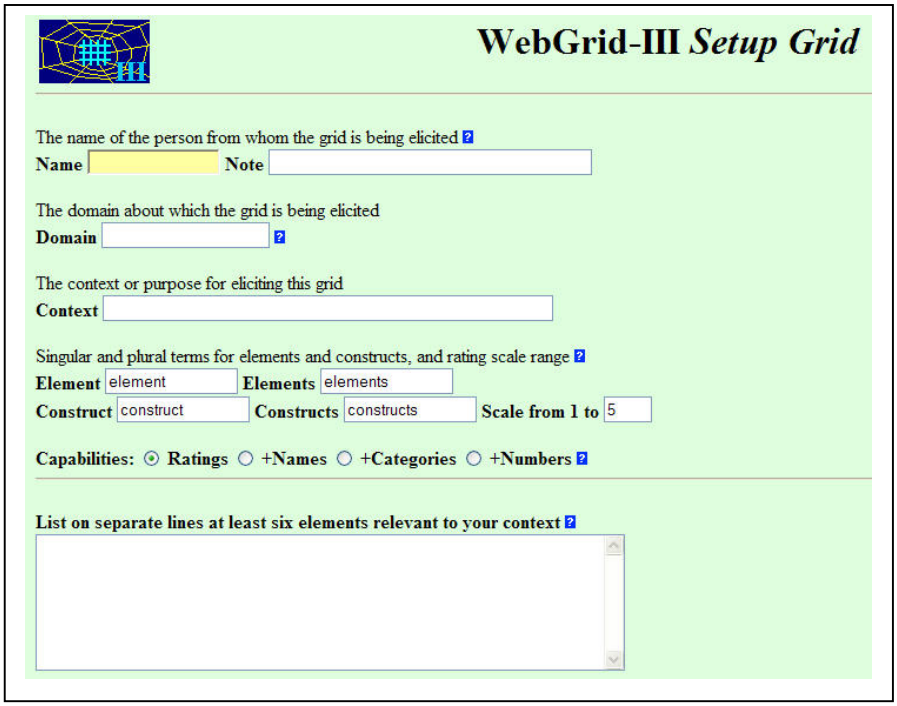


Figure 3.2 Presentation of WebGrid III

3.3. Formal Concept Analysis (FCA)

[GAW99] said that The Formal Concept Analysis (FCA) project was born around when a research group in Darmstadt Germany begun to systematically develop a frame work for lattice theory applications. It was first presented to the mathematical public in a programmatic lecture given at the 1981 Banff conference on Ordered Sets [WIL82].

Since then, several hundred articles have been published including a textbook on the mathematical foundations [GAW96]. The Darmstadt group alone has participated in more than a hundred application cooperation projects Former members of that team have founded a small firm and now make their living from such applications.

Furthermore [GAW99] explain that the sophisticated name of "Formal Concept Analysis" needs to be explained The method is mainly used for the analysis of data, i.e. for investigating and processing explicitly given information Such data will be structured into units which are formal abstractions of concepts of human thought allowing meaningful and comprehensible interpretation. [GAW99] use the prefix formal to emphasize that these formal concepts are mathematical entities and must not be identified with concepts of the mind. The same prefix indicates that the basic data format that of a formal context is merely a formalization that encodes only a small portion of what is usually referred to as a "context".

All following descriptions and examples below are taken from [RIC98]. It is described that Formal Concept Analysis (FCA), first developed by [WIL82], is a mathematically based method of finding, ordering and displaying formal concepts [WIL92]. A concept in FCA is comprised of a set of objects and the set of attributes associated with those objects. The set of objects forms the extension of the concept while the set of attributes forms the intension of the concept. Knowledge is seen as applying in a context and can be formally defined as a cross table as Table 3.1 below. The rows are objects and the columns are attributes. An X indicates that a particular object has the corresponding attribute. This cross table is used to find formal concepts. We interpret an (Multiple Classification Ripple Down Rules (MCRDR) Knowledge Base System [KCP95][KAN96] as a cross table by treating each rule pathway (which includes the conditions on parent nodes) as an object and the conditions as attributes. The MCRDR KBS in Figure 4 has been converted to a cross table in Table 3.1.

The following description of FCA follows [WIL82] and the screen dumps shown in Figures 3.3 and 3.4 are our implementation, called MCRDR/FCA, which is an enhancement of the current MCRDR for a Windows system. A formal context (K) has a set of objects G (for *Gegenstande* in German) and set of attributes M (for *Merkmale* in

German) which are linked by a binary relation I which indicates that the object g (from the set G) has the attribute m (from the set M) and is defined as: K = (G,M,I). Thus in figure 5 we have the context K of "MCRDR contact lens rules" with G = {0-%LENSN, 1-%LENSS, 2-%LENSN, 3-%LENSH, 4-%LENSH}, where the object is referred to by the rule number and its conclusion, and M = {1=1, astigmatic=no, tear_production=normal, age=presbyopic, prescription=myope, astigmatic=yes, age=young}. Note: 1=1 is the condition for the default rule with the default conclusion as was seen in Figure 3. The crosses show where the relation I exists, thus I = {(0-%LENSN,1=1), (1-%LENSS,1=1), (1-LENSS, astigmatic=no),…,(4-%LENSH, age=young)}.

	1=1	astigmatic = no	Tear_production = normal	age = presbyopic	prescripton = myope	astigmatic = yes	age = young
1-%LENSN	X						
2-%LENSS	X	X	X				
3-%LENSN	X	X	X	X	X		
4-%LENSH	X		X		X	X	
5-%LENSH	X		X			X	X

Table 3.1 Context of “MCDRD Contact Lens Rule”

A formal concept is a pair (X,Y) where X is the *extent*, the set of objects, and Y is the *intent*, the set of attributes, for the concept. The derivation operators:

$$X^{G}: X \mapsto X' := \{m \in M \mid gIm \text{ for all } g \in X\}$$
$$Y^{\subseteq M}: Y \mapsto Y' := \{g \in G \mid gIm \text{ for all } m \in Y\}$$

are used to construct all formal concepts of a formal context, by finding the pairs (X'',X') and (Y',Y''). We can obtain all extents X' by determining all row-intents $\{g\}'$ with $g \in G$ and then finding all their intersections using (3). Alternatively Y' can be

obtained by determining all column-extents {m}' with m ∈M and then finding all their intersection (4). This is specified as:

$$X' = \bigcap_{g \in X} (g), \qquad Y' = \bigcap_{m \in Y} (m)$$

Less formally, in Figure 3.3 nine formal concepts have been derived for the formal context in Table 3.1 by finding the intersection of sets of attributes for each object and then finding the sets of objects that have those attributes.

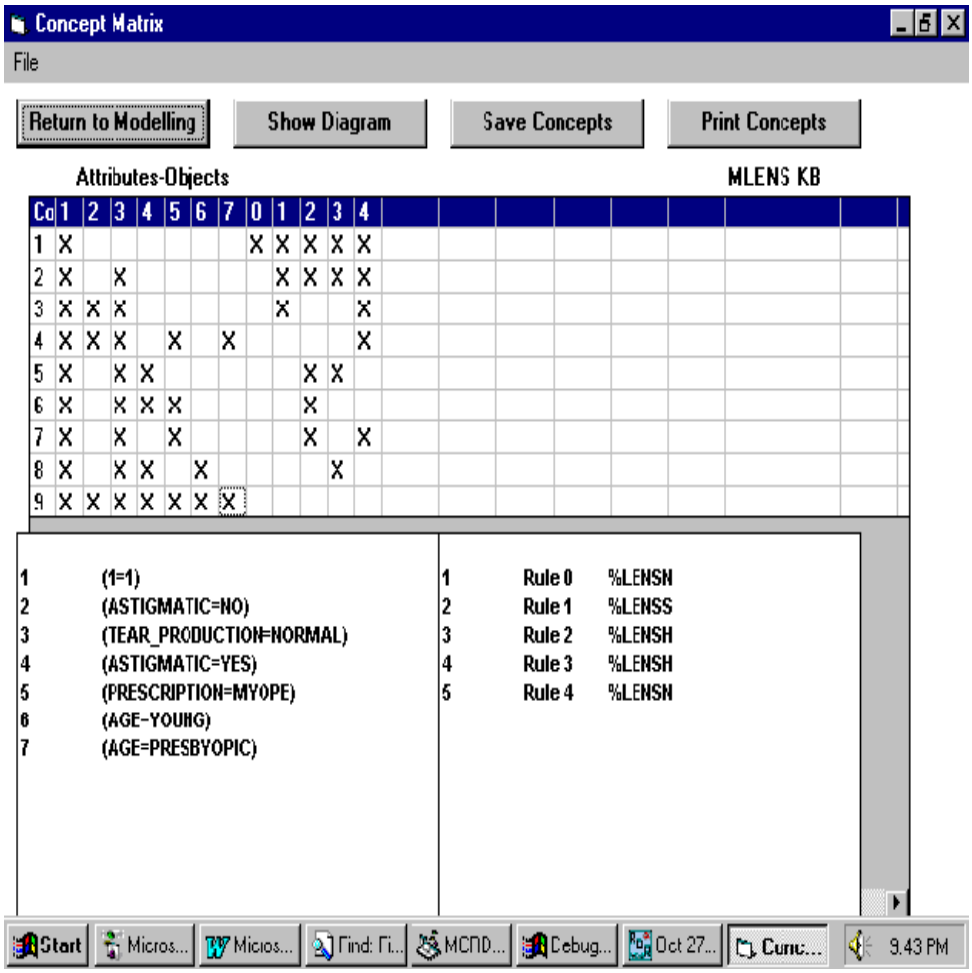


Figure 3.3 The concept matrix screen from MCRDR/FCA

From the figure 3.3, nine (9) concepts have been found. Each row represents a concept. The columns show the attributes, which are listed first, followed by the objects. As was shown in the formal context in Table 3.1, there are seven attributes and five objects. The

attribute labels have been converted to sequential numbers and the object labels correspond to the rule number to allow the relationships between concepts and the possible patterns to be more readily seen. Full labelling can be obtained by using the pop-up windows as shown in this figure or by clicking on the attribute, object or concept number. The concepts have been ordered to show the subsumption relations that exist. The extent of the top concept, No 1, includes all objects. The intent of the bottom concept, No 9, includes all attributes.

To present a visualisation of our ordered set of concepts as a line diagram it is necessary to compute the predecessors and successors of each concept. Predecessors are found by determining the largest sub concept of the intents for each concept. Successors are found by determining the smallest super concept of the intents. A super concept is a set that has all of the members of another set and additional members. A sub concept is a set that has fewer members than another set but all the members it has are contained in the other set. We only concentrate on finding sub or super concepts of the intents or extents because they are inversely related and using either set will give the same result. In MCRDR/FCA the successor list was used to identify concepts higher in the diagram, the parents, and the predecessor list identified concepts lower in the diagram, the children. The number of levels of parents and children are used to layout the line diagram and an algorithm is given in [RCO97]. However, just as users have different views of their knowledge, there is not one fixed way of drawing line diagrams and often a number of different layouts should be used [WIL92].

More formally, we use the subsumption relation $\leq$ on the set of all concepts formed such that $(X_1, Y_1) \leq (X_2, Y_2)$ if $X_1 \subseteq X_2$. From Lattice Theory, the ordered concept set can be used to form a complete lattice, called a concept lattice and denoted (K). For a family (X_i, Y_i) of formal concepts of K the greatest sub concept, the join (5), and the smallest super concept, the meet (6), are respectively given by:

$$\bigvee_{i \in I} (X_i,B_i) := \left(\left(\bigcup_{i \in I} A_i \right)'' , \bigcap_{i \in I} B_i \right) \qquad \bigwedge_{i \in I} (X_i,B_i) := \left(\bigcap_{i \in I} A_i, \left(\bigcup_{i \in I} B_i \right)'' \right)$$

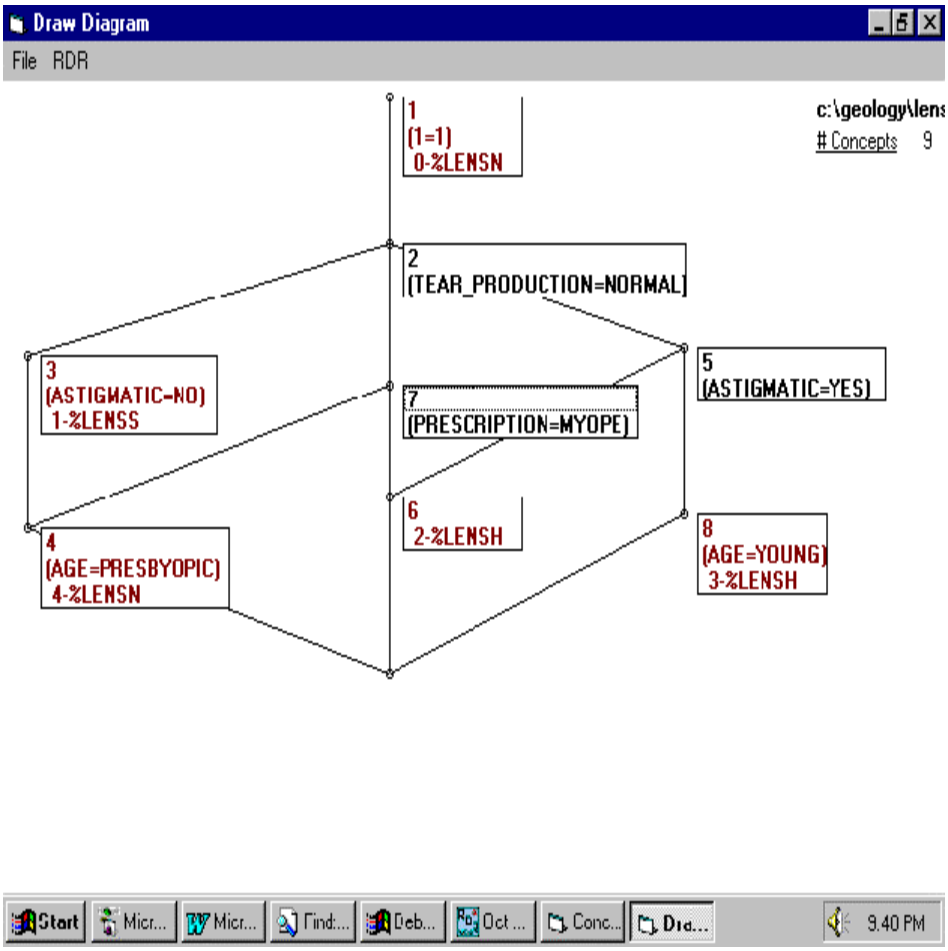


Figure 3.4 The Diagram Screen in MCRDR/FCA which shows the Concept Lattice for the Formal Context "MCRDR Contact Lens Rules" given in Table 3.1

In Figure 3.4 the concepts are shown as small circles and the sub or super concept relations as lines. Each concept has various intents and extents associated with it. The labelling has been reduced for clarity by removing all attributes that can be reached by ascending paths and all objects that can be reached by descending paths from the concept. In MCRDR\FCA it is possible to display the concept, attribute/s or object/s belonging to each node or all three dimensions can be displayed concurrently, as in Figure 3.4.

The incorporation of FCA into Ripple Down Rules (RDR) [ERD93] has provided a powerful tool for browsing the knowledge base providing explanations of the knowledge in terms of relationships between primitive concepts and higher level

concepts which were previously hidden in the RDR structure. This means that RDR is able to support a wider range of activities beyond those for which it had typically been used.

3.4. Matrix-Based Technique

In the Linear Algebra a Matrix is a composition of numerical values (but also other objects such as operators) in tabular form. One speaks of the columns and lines of the matrix, and calls also vectors (i.e. line vectors and column vectors). One calls the objects, which are arranged in the matrix, components or elements of the matrix. The Designation “Matrix” is introduced by an English Mathematician names James Joseph Sylvester in 1850.

Sylvester did important work on matrix theory. In 1851 he discovered the Discriminant of a cubic equation and first used the name "Discriminant" for such expressions of quadratic equations and those of higher order. He used matrix theory to study higher dimensional geometry. He also contributed to the creation of the theory of elementary divisors of lambda matrices.

Matrix is proved to be very useful in solving scientific problems. The speciality of the Matrix is the connection to linear illustrations. Each linear illustration can be assigned a Matrix and each Matrix corresponds to a linear illustration. One calls this connection also as Homeomorphism. Homeomorphism is an illustration between two structures; by which the parts of a structure is clearly illustrated on the same sense parts of the other structure.

Between the quantity of the linear Matrix and the quantity of the linear illustrations exists a Bijection. A bijective function is a function that illustrates different elements of its definition range on different elements of the range of values (injective) and arises for each additional element of the range of values as image (surjective). A bijective function has therefore always a completely defined inverse function.

Matrix-based techniques involve the construction of grids indicating such things as problems encountered against possible solutions. Important types include the use of frames for representing the properties of concepts and the repertory grid technique used to elicit, rate, analyse and categorise the properties of concepts [EPI04].

These techniques involve the construction and filling-in of a 2-dimensional matrix (grid, table). Useful examples are:

- Concepts v Properties (attributes and values)
- Problems v Solutions
- Hypotheses v Diagnostic techniques
- Tasks v Resources

The elements within the matrix can contain:

- Symbols (ticks, crosses, question marks)
- Colours
- Numbers
- Text

[EPI04] developed Matrix Tool in PCPACK v4. It is an integrated suite of seven knowledge tools designed to support the acquisition and use of knowledge. It has the flexibility to support various methodologies. The Matrix tool allows two types of matrix (grid) to be created and edited: an Attribute Matrix and a Relationship Matrix.

Basic Features of Matrix Tool in PCPACK v4

- Attribute Matrix or Relationship Matrix
- Simple clicking operation for assigning properties (in an Attribute Matrix) or asserting relationships (in a Relationship Matrix)
- Inheritance of attributes/values (in an Attribute Matrix), and indication of conflicts (if multiple inheritance is present)
- Selectable displays for relationship attributes

- Designable 'user forms' for data entry to a Relationship Matrix
- Customisable symbols for numerical relationship values in a Relationship Matrix
- Charts for displaying numerical values of a Relationship Matrix

Attribute Matrix

An Attribute Matrix is a way of associating properties (attributes and values) to knowledge objects. It does this by presenting a matrix of knowledge objects on the vertical axis and attributes/values on the horizontal axis. A simple clicking operation associates attributes and/or values with the relevant object. An example of an Attribute Matrix is shown below.

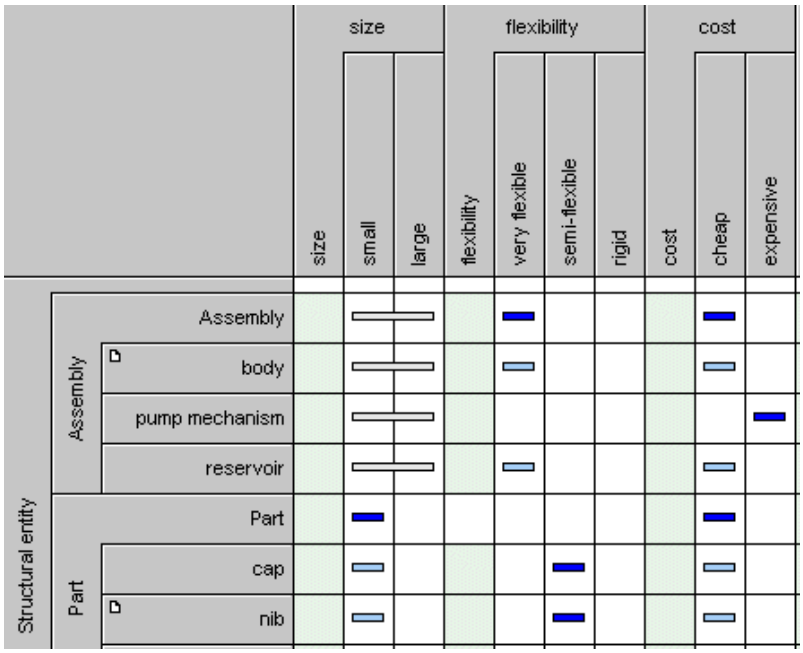


Figure 3.5 Attribute Matrix

Relationship Matrix

A Relationship Matrix is a way of associating certain knowledge objects (on the X-axis) to certain knowledge objects (on the Y-axis) using a specified relation, such as "followed by", "has constraint" or "requires". One relation per matrix is used. If the selected relation has properties (attributes and values) then special symbols can be used

to show each relationship's properties. User forms can be designed to aid entry of data, and charts can be created to display the numerical properties of relationships in graphical form.

An example of a Relationship Matrix is shown below. This matrix shows the numerical values of a relation attribute.

Test Tasks		Computers						
			Computer					
			Centaur	Hydra	Gorgon	Typhon	Cerberus	Pegasus
	Test Task	Spreadsheet	5	7	4	6	8	3
		2D Graphics	7	6	4	5	7	3
		3D Graphics	4	-1	3	6	8	0
		Database	6	5	-1	7	6	4

Figure 3.6 Relation Matrix

Chapter 4

Design and Architecture of Knowledge Acquisition System

4.1. Concept and Methodology

Key to the success of the design and development of the expert system is the choice of the correct or suitable technique for knowledge acquisition. Much research is available discussing the various techniques [NIE96]. Several knowledge acquisition techniques include structured and unstructured interviews. [BCZ92] list includes observation of the expert in action, unstructured elicitation which corresponds to unstructured interviews.

The process of capturing knowledge is defined as the collection, organization, evaluation, and incorporation of knowledge within a working expert system [LIC93]. In designing expert systems, the process of eliciting information has been termed knowledge acquisition. According to [HOF87], knowledge acquisition, also known as knowledge elicitation, involves extracting problem-solving expertise from knowledge sources, which are usually domain experts. [WAT85] defines knowledge acquisition as the process of extracting, structuring and organizing knowledge from several sources, usually human domain experts, so it can be used in a program. [SMI96] noted that knowledge acquisition involves the elicitation of data from the expert, interpretation of the data to deduce the underlying knowledge and creation of a model of the expert's knowledge in terms of the most appropriate knowledge representation. This knowledge acquisition process involves one or more knowledge engineers interacting with one or more domain experts, each of which brings a certain set of attributes to this interaction with the goal of developing a shared representation or model of the expert’s problem solving processes [FEL87]. [SIA97] noted that knowledge acquisition process consists

of the process of identifying, obtaining, and organizing the knowledge that is to be incorporated into an expert system and knowledge elicitation.

Refer to the statements above and consider the character of Ternary Grid elicitation technique I have defined the methodology for knowledge acquisition system that is developed here into:

- Organising the knowledge base
- Obtaining the factual knowledge
- Eliciting the judgemental knowledge
- Transferring the knowledge into knowledge base

4.2. System Architecture

The basic feature of the system architecture is to organise the independent and sequential obtaining process of the factual knowledge and the elicitation process of judgmental knowledge using Ternary Grid. The overall systems architecture is presented in terms of collection of functions providing effective acquisition, processing, transferring and flexible transformation of knowledge. This section gives an overview of the system that shows the design approach of the system and the concept of acquisition process.

The Factual Knowledge Processor and Knowledge Base Transformer are of particular interest. The Factual Knowledge Processor demonstrates the obtaining process of the factual knowledge from the expert with new approach. The Knowledge Base Transformer demonstrates the transformation process of the knowledge into any knowledge base. It enables the knowledge acquisition system adaptable into different format of knowledge base.

The most interest is Ternary Grid Knowledge Elicitation (KE) System because it demonstrates the main part of knowledge acquisition system. It is the elicitation process of knowledge using new approach of grid application which has been named here Ternary Grid.

Figure 4.1 and 4.2 show the system architecture of knowledge acquisition system. The knowledge acquisition system consists of three main parts, i.e. Knowledge Elicitation system, Expert's Interface and Knowledge Transformer. Additionally the system provides the knowledge base in Ternary Grid format that can be directly connected to the Ternary Grid format supported inference engine.

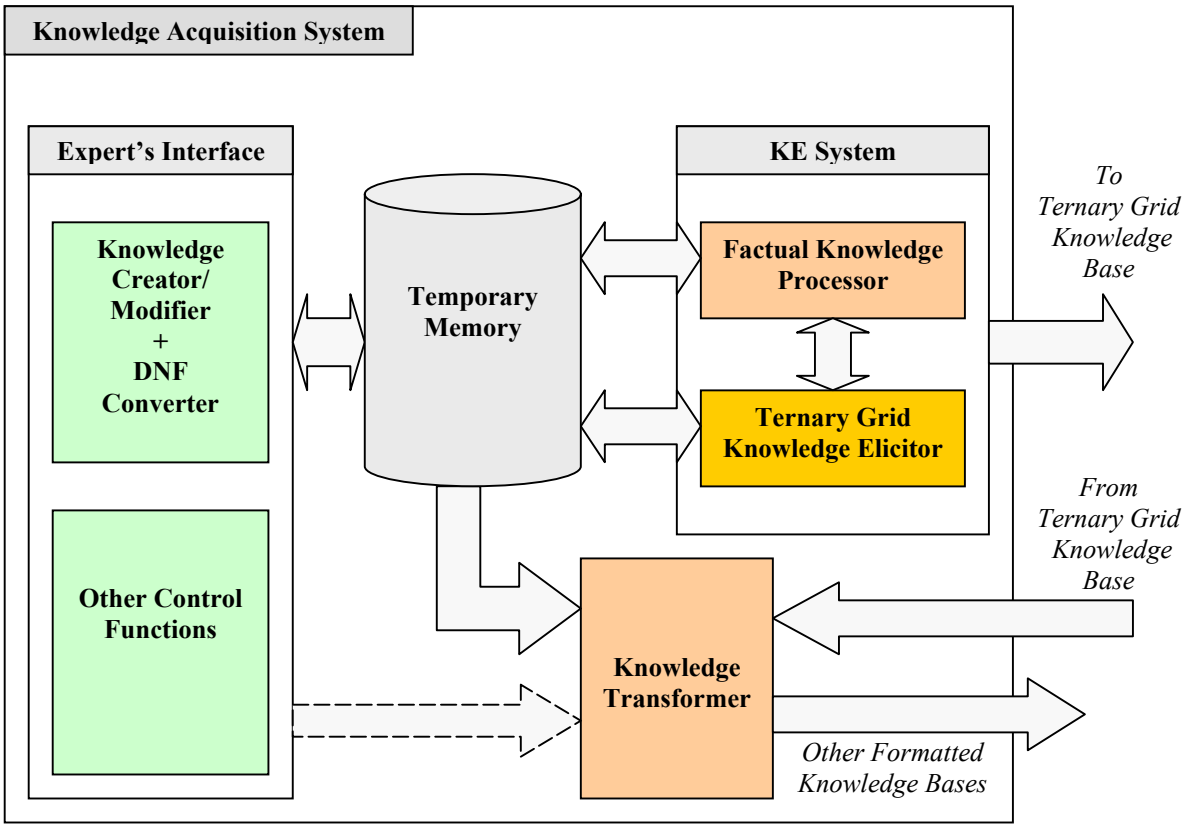


Figure 4.1 System Architecture

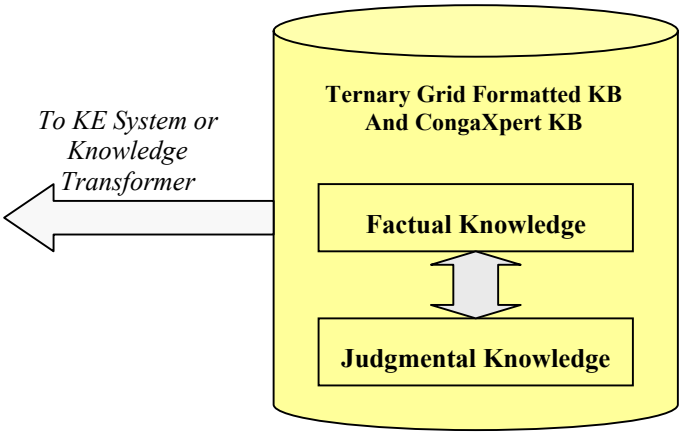


Figure 4.2 Ternary grid formatted Knowledge Base and CongaXpert Knowledge Base

The task of knowledge elicitation system is divided into two independent and sequential sub tasks i.e. Factual Knowledge Processor and Ternary Grid Knowledge Elicitor. The task of the Factual Knowledge Processor is to define, collect, process and transform the factual knowledge. The task of Ternary Grid knowledge elicitor is to define, validate and optimise the judgmental knowledge or rule-based knowledge by means of factual knowledge that has been defined before by the Factual Knowledge Processor.

In most rule-based system the knowledge engineer can reference rules or facts that have not yet been defined or created. The integrity of knowledge is therefore violated and not guaranteed. This will lead to a situation of unorganised rule if the number of rules is getting higher. The mechanism of knowledge elicitor developed here should be able to overcome this problem.

The independent process of knowledge elicitation provides flexible and user-friendly elicitation to the experts while they are doing their job. The sequential process of knowledge elicitation provides stepwise and structured elicitation so that unorganised rules will not occur.

Knowledge Transformer enables the transformation of Ternary Grid format into another format of knowledge. The transformation procedure of every format must be programmed and integrated into the knowledge transformer. By the time of development the knowledge transformer will be able to transform knowledge into knowledge format of CongaXpert [EHW03]. Furthermore additional transformation procedure for other formats will be integrated into knowledge transformer. Even a method of problem solution for transformation procedure is still being explored in order to get better solution.

Expert's Interface enables and is responsible for the interaction or communication between human expert and the system. It consists of two main parts i.e. knowledge creator/modifier with inclusive DNF (Disjunctive Normal form)-Converter, and one additional part that can consists of some control functions for the system.

The Ternary Grid Editor facilities the creation of the new knowledge and modification of the existing knowledge in a grid form. They provide the following functions like creating new, adding, changing and deleting knowledge. The Rule Editor is an editor for building rule in classical form. The DNF-Converter converts that rules into another format of rule, i.e. Disjunctive Normal Form. It is a disjunction of conjunctions of literals. This form is also called *sum of products* [CON01] [WIK04]. This Rule Editor provides also the facility for the creation of the new knowledge and the modification of the existing knowledge, but not in a grid form.

There was a time that acquisition tools permit experts to modify only factual knowledge but they provide limited support for modifying problem solving or judgmental knowledge. EXPECT can automatically derive the interdependencies between problem solving and factual knowledge [SWG95]. The knowledge Modifier that is developed here enables the experts to modify both factual and judgmental knowledge.

Fact Editor enables the expert to create new, add, change or delete the fact. The additional part of Expert's Interface supports other control functions of interface like setting database, setting the tool profile, help function etc.

The Ternary Grid formatted knowledge base consists of factual knowledge or facts and judgmental knowledge or rules. This knowledge base format can be transformed into another format of knowledge by knowledge transformer or processed directly by a Ternary Grid supported Inference Engine.

4.3. Knowledge Base Organisation

This section concerns with the structure organisation or representation of knowledge that is stored in knowledge base after acquisition process and used for expert system. To create more powerful knowledge acquisition system, we not only need better acquisition tools, but we need also to change the architecture of the knowledge based system or expert system we created, so their structure will provide better support for acquisition. [SWG95]. A major prerequisite for the support of knowledge acquisition is to define an

adequate representation system [MOR93]. [GRC87] consequently describe the problem of knowledge acquisition as being caused by incongruity of representation formalisms and the expert's manner of formulating the problem.

Considering the building block of the expert system [FEE93], this statement leads to the term that the knowledge base and the reasoning part of the expert system are concerned. Refer to the role of the of knowledge acquisition system in the development of expert system [MCH89], the change or modification of the architecture of the expert system focuses on the knowledge base only. It is what this paper concerns.

The knowledge base contains both factual and judgmental knowledge. The factual knowledge or fact is defined by its name and contains a certain value. An operator builds the relation between name and value. The interesting thing of this structure is the transformation of classical factual format into *logical term* of fact. Figure 4.3 shows the structure of factual knowledge. This logical term should support better performance of elicitation process using Ternary Grid. With logical term the knowledge elicitor does not have to know the meaning of logical term related to the fact. In another words, which fact to which logical term belongs. The knowledge elicitor has only to know the value of logical term which is true or false. This elicitation process is a Boolean operation.

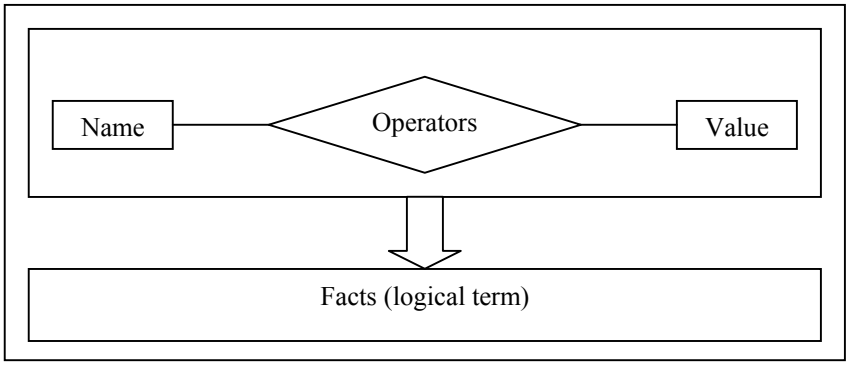


Figure 4.3 Structure of factual knowledge

The judgmental knowledge is rule-based knowledge or production rule. Figure 4.4 shows the structure of judgmental knowledge. The term production rule came from

production system which is developed by [NEW72]. A production system is a model of cognitive processing, consisting of a collection of rules (called *production rules*, or just *productions*). Each rule has two parts: a *condition* part and an *action (conclusion)* part. The meaning of the rule is that when the condition holds true, then the action is taken.

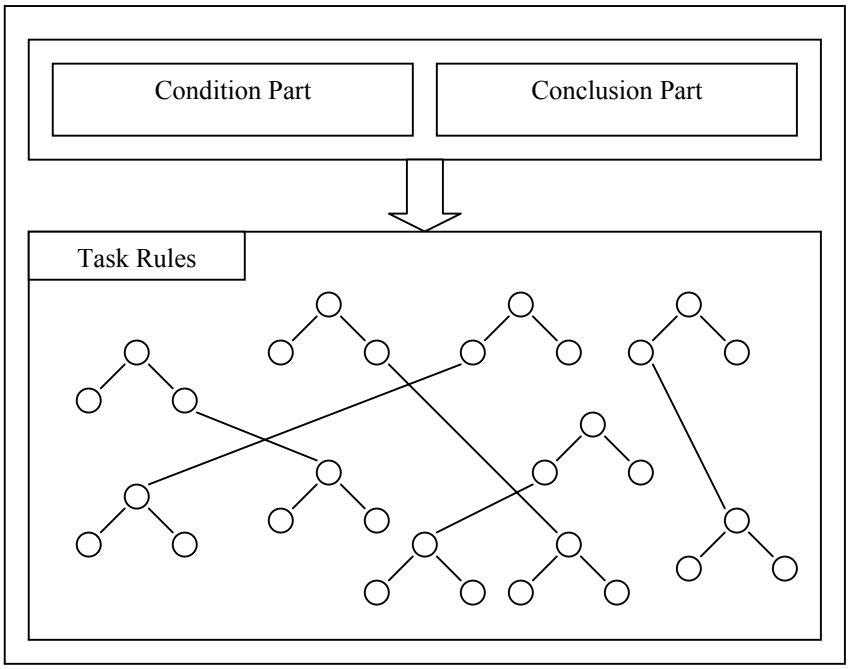


Figure 4.4 Structure of judgmental knowledge

The typically form of production rule:

IF <condition> THEN <conclusion>

The condition and conclusion may consist of many logical terms, in this case logical entities which represent facts:

IF <Fact_1> and <Fact_2> and … THEN <Fact_9>

Example:

Rule 1: IF <F1> and <F2> and <F3> THEN <F9>

In mathematical form we describe:

In Rule 1, F9 will be true if ***only*** <F1> and <F2> and <F3> equals true

The organisation of production rule can be easily represented in Ternary Grid which has the following structure:

	Fj
Ri	0/1/2

Table 4.1 Ternary Grid basic structure

Ri: Rule i (i is the number of rule)

Fj: Fact as logical term j (j is the number of logical term)

i = {1, 2, 3, …, I}

j = {1, 2, 3, …, J}

J >= I + 1

The Value of every grid box is 0, 1 or 2

0 = It is not displayed but represented by empty grid box.

1 = The condition part of rule Rn (LHS = Left Hand Side)

2 = The conclusion part of Rn (RHS) = Right-Hand Side)

Example for n=3 and m=7 as follows:

	F1	**F2**	**F3**	**F4**	**F5**	**F6**	**F7**
R1							
R2							
R3							

Table 4.2 Ternary Grid (3 x 7)

If the following rules are given:

Rule 1: IF <F1> and <F3> THEN <F4>

Rule 2: IF <F4> and <F5> THEN <F6>

Rule 3: IF <F6> and <F7> THEN <F2>

The grid becomes as follows:

	F1	F2	F3	F4	F5	F6	F7
R1	1	0	1	2	0	0	0
R2	0	0	0	1	1	2	0
R3	0	2	0	0	0	1	1

Table 4.3 Filled (3 x 7) Ternary Grid

In order to simplify the representation of the grid, the "0" value is represented by empty boxes. The grid becomes as follows:

	F1	F2	F3	F4	F5	F6	F7
R1	1		1	2			
R2				1	1	2	
R3		2				1	1

Table 4.4 Filled (3 x 7) Ternary Grid, without "0" value

4.4. Knowledge Optimisation Stages

The knowledge optimisation stages are illustrated in Figure 4.5. Collecting expert's knowledge; this step requires extracting knowledge from the expert. This extraction is mainly done through conducting interviews with the experts. Interpreting deals with reviewing collected information, picking out relevant information and storing information in Ternary Grid. Analysing and optimising: the concept describes how the knowledge is organised and formed into optimal content and structure in the form of Ternary Grid. Designing and transferring knowledge provide support for further problem-solving strategies. An appropriate knowledge structure in the form of Rule-based knowledge is designed and the contents of knowledge in previous structure are

transferred into new structure. Furthermore the inference engine of expert system processes this Rule-based knowledge and provides expert system's knowledge to the user.

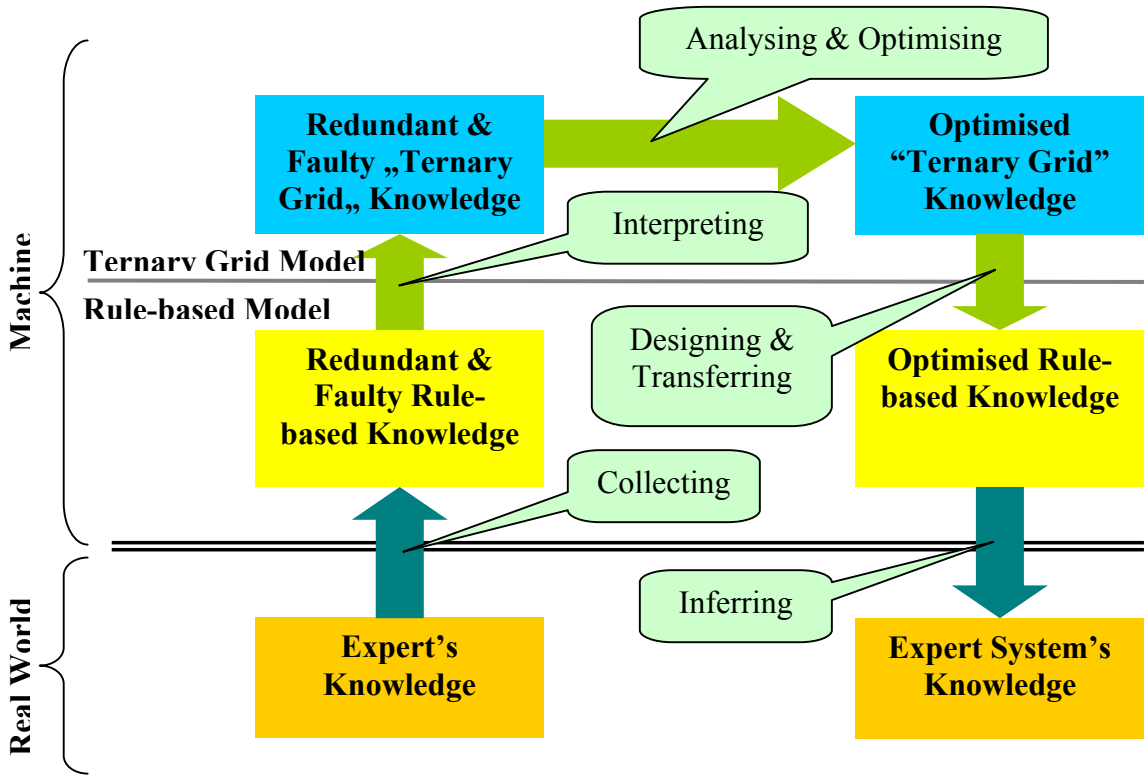


Figure 4.5 Knowledge optimisation stages

4.5. Obtaining Knowledge

Obtaining the knowledge is the step to get information in the form of factual knowledge from the expert. It has an important role in knowledge acquisition process. In this work, obtain the knowledge concerns with defining and collecting the factual knowledge from the expert, processing and transforming the knowledge into logical entity for supporting knowledge elicitation process using Ternary Grid.

4.6. Eliciting the Knowledge

Knowledge elicitation is the most important part of knowledge acquisition process. There are various techniques for Knowledge Elicitation, which have been used to obtain

the information required to solve problems in knowledge acquisition process. One common way is Direct Method. In the Direct Method the performance of the knowledge due to the quality of information and the reduction of error possibility is performed by direct interaction between experts and knowledge engineer during elicitation process. In order for these methods to be successful, we need a method that can achieve that performance.

Knowledge Elicitation is the process of obtaining knowledge from a domain expert that describes how they perform a specific task and/or describes what general knowledge they have about the domain. More specifically, KE refers to obtaining knowledge from a person in order to transfer it to a computer program [MCH89]. The methods of Knowledge Elicitation are mainly ad hoc and non-scientific. Knowledge engineers are computer specialist's people without adequate training in various psychological techniques, thus contributing to the fact why these methods are vague and instated [WRA87].

Knowledge Elicitation is not an easy task, the reasons of which appear later on in the report. But this can be concluded by fact that Knowledge Acquisition is the greatest bottleneck in the construction of expert systems [WRA87]. There is not a best way to elicit knowledge as in a cooking recipe format because this is more an art than a science. However there are ways to improve the manner in which the knowledge engineer approaches the domain, the domain expert or experts and their own organisational ways to accomplish the task [IGN91]. The suitability of an elicitation method depends on the kind of information being extracted. Each method has its own limitations

Different Knowledge Elicitation techniques have been developed in order to obtain knowledge [BOO89], [COR89]. They can be classified in many dimensions. The most common one used is "direct" versus "indirect," where direct techniques are used to obtain information that is easily verbalized and indirect techniques are used to obtain information that is not easily verbalized [HUD97]. Classification can also be based on the type of interaction with the subject and the type of knowledge most commonly obtained.

This paper concerns with the direct method technique for Knowledge Elicitation using Ternary Grid which derives the information given by human expert to obtain the performance of the knowledge due to the quality of information and the reduction of error possibility, mentioned above.

4.7. Transferring the Knowledge

The last process of knowledge acquisition is to transfer the knowledge into knowledge base after elicitation process. It concerns with the encoding and storing the knowledge into knowledge base. The system should be able to transfer knowledge into Ternary Grid Knowledge Base or other formatted knowledge base by using Knowledge Transformer. The storage organisation that is used for the knowledge base is facilitated by a database.

Chapter 5

Obtaining Factual Knowledge

The knowledge acquisition system supports the obtaining process of factual knowledge or facts in two steps. In the first one, the system defines and collects the factual knowledge from the expert and stores the knowledge in memory in the form of grid or matrix. The last one, the system transforms the factual knowledge into logical terms and processes it using grid or matrix concept. At last the system transfers the knowledge into database. The system provides verification of fact interconnection. It can avoid repeating facts and build the relationship of facts. This is done by Factual Knowledge Processor.

Before I describe more about obtaining process of factual knowledge, I want to describe first the following definitions concerning the type of factual knowledge. There are two types of factual knowledge, i.e.: *prospective facts* and *definitive fact*.

A prospective fact is actually not a real fact but it can be a real fact and is called definitive fact. There is no relation operator between fact-name and fact-value. The fact-values that belong to a fact-name are called “possible fact-value”. Table 5.1 shows the some examples of prospective facts.

Fact-name	Possible fact-value
Temperature	20
	25
	30
Weather	cloudy
	sunny
	shower

Table 5.1 some examples of prospective facts

From the example in table 5.1 one can see that the temperature can be 20, 25 or 30 (degrees). In this phase one does not know which one will be selected.

A definitive fact is a real fact. This occurs if a certain possible fact-value is related into a corresponding fact-name by a fact-operator. Table 5.2 shows the some examples of definitive facts.

No. / Fact-Id	Fact-name	Operator	Fact-value
1	Temperature	=	20
2	Temperature	>=	25
3	Weather	IS	cloudy
4	Weather	ISNOT	sunny

Table 5.2 some examples of definitive facts

If a prospective fact becomes a definitive fact, a fact-id will be assigned into this definitive fact. This fact-id is unique. It means, there will not be a same fact-id for different definitive facts.

From the table 5.2 one can see the different situation from the table 5.1. The value “20” (which means 20 degrees) has been selected or related to the fact-name “Temperature”. The relation between fact-name “Temperature” and fact-value “20” is done by “=” operator. In verbal expression one says “Temperature is 20 degrees”. The fact-id “1” has been assigned to this statement. Fact-id “2” expresses the statement that “Temperature is greater that 25 degrees”. Fact-id “3” expresses the statement that “Weather is cloudy” and Fact-id “4” expresses the statement that “Weather is not sunny”

5.1. Defining and Collecting Facts

Defining and collecting the factual knowledge is a process to create prospective facts. It consists of following steps:

- Defining the fact-name and possible fact-values.

- Building the relation between facts and values (many-to-many relation)

Fact-name and fact-value can not be separately created. Every time if the expert creates a fact-name, he will be asked to give a possible fact-value for this fact-name. The expert can create a new fact-value or take from the possible values that have been existent.

A fact-name-value number will be assigned to this prospective fact. This number serves as identification number for this fact. Table 5.3 shows an example of prospective facts that have been inputted by the expert. This table shows us that a fact-name can have some possible fact-values.

Fact-Name-Value Number	Fact-Name	Fact-Value
1	Temperature	20
2	Temperature	25
3	Temperature	30
4	Weather	cloudy
5	Weather	sunny
6	Weather	shower
7	Weather	rainy
8	Summer time	true
9	Summer time	false

Table 5.3 Fact-name and fact-value as prospective fact

Furthermore fact-name and fact value will be stored separately in the knowledge base. Fact-name will be numbered. This number is unique and serves as identification number for the fact-name. See table 5.4.

Fact-Name Number	Fact-Name
1	Temperature
2	Weather
3	Summer time

Table 5.4 Fact-name and its identification number

Same case like for fact-name, the fact-value will be numbered and stored in knowledge base. This number is also unique and serves as identification number. See table 5.5.

Fact-Value Number	Fact-Value
1	20
2	25
3	30
4	cloudy
5	sunny
6	shower
7	rainy
8	true
9	false

Table 5.5 Fact-value and its identification number

According to the table 5.4 and 5.5, the fact-names, fact-values and their relations will not be stored as in the table 5.3. Only the fact-name number and fact-value number will only be stored in the relation table. It is shown in table 5.6.

Fact-Name-Value Number	Fact-Name Number	Fact-Value Number
1	1	1
2	1	2
3	1	3
4	2	4
5	2	5
6	2	6
7	2	7
8	3	8
9	3	9

Table 5.6 Fact-name and fact-value as prospective fact

This organisation of the knowledge base will save a lot of space for data storage.

5.2. Transforming and Processing Facts

Transforming and processing the factual knowledge deals with creating definitive facts. It consists of following steps:

- Selecting the fact-name and its corresponding fact-values.
- Selecting the fact-operator to build relationship between fact-name and corresponding fact-value.

Table 5.7 shows the list of fact-operators that are used for building relationship between fact-name and its corresponding fact-value.

Code	Operator	Data Type
1	=	NUMERIC
2	>	NUMERIC
3	>=	NUMERIC
4	<	NUMERIC
5	<=	NUMERIC
6	<>	NUMERIC
7	BETWEEN	NUMERIC
8	AND	NUMERIC
9	FROM	NUMERIC
10	TO	NUMERIC
11	EXIST IN	NUMERIC
21	IS	TEXT
22	ISNOT	TEXT
23	EXIST IN TEXT	TEXT
24	==	BOOLEAN

Table 5.7 List of fact-operators

Every time after the expert creating definitive fact, a fact-id will be assigned to that definitive fact. This fact-id is unique and serves as identification number for this fact. Furthermore this fact-id has important role in building Ternary Grid rule. This fact-id will be used as pointer key to the definitive fact. The Ternary Grid rule has only to know the fact-id without having to know the meaning or content of that fact-id. The

fact-id represents a logical term of information about circumstances that exist or are real occurrence. The fact-id can be viewed as the interface between classical knowledge format and Ternary Grid knowledge format.

The fact-operators listed in table 5.7 enable the expert to create a definitive fact with multiple fact-values. In numerical data type, multiple fact-values can be built by the fact-operator "Between-And", "From-To" and "Exist in". In text data type, multiple fact-values can be built by the fact-operator "Exist in Text".
Example of definitive facts can be seen in table 5.8 as follows:

No.	Fact-Id	Fact-Name	Fact-Operator	Fact-Value
1	F1	Temperature	=	20
2	F2	Temperature	>=	25
3	F3	Temperature	BETWEEN… AND	20, 30
4	F4	Temperature	FROM ...TO	20, 30
5	F5	Temperature	EXIST IN	20, 25, 30
6	F6	Weather	IS	cloudy
7	F7	Weather	ISNOT	sunny
8	F8	Weather	EXIST IN TEXT	shower, rainy
9	F9	Summer time	==	true
10	F10	Summer time	==	false

Table 5.8 example of definitive facts

Fact-id F1, F2, F6, F7, F9 and F10 represent definitive facts that have single fact-value. Other fact-ids represent definitive facts that have multiple fact-values. Table 5.9 shows the content of table 5.8 in verbal expression:

The difference between fact-operator "Between-And" in F3 and "From-To" in F4 can be explained in mathematical expression as follows:

$$\text{F3: } y = \{x \mid x > 20 \cap x < 30,\ x \in \mathrm{N},\ y \in \mathrm{N}\}$$

$$\text{F4: } y = \{x \mid x \geq 20 \cap x \leq 30,\ x \in \mathrm{N},\ y \in \mathrm{N}\}$$

No.	Fact-Id	Expression
1	F1	Temperature equals 20 (degrees)
2	F2	Temperature is greater than 25 (degrees)
3	F3	Temperature is between 20 and 30 (degrees)
4	F4	Temperature is from 20 to 30 (degrees)
5	F5	Temperature can be 20, 25 or 30 (degrees)
6	F6	Weather is cloudy
7	F7	Weather is not sunny
8	F8	Weather can be shower or rainy
9	F9	It is summer time
10	F10	It is not summer time

Table 5.9 example of definitive facts in verbal expression

Due to the complexity of data in table 5.8, the data can not be easily stored in knowledge. This occurs with the facts that have multiple fact-values, like F3, F4, F5 and F8. Every data must be stored individually as shown table 5.10 as follows:

No.	Fact-Id	Fact-Name	Fact-Operator	Fact-Value
1	F1	Temperature	=	20
2	F2	Temperature	>=	25
3	F3	Temperature	BETWEEN	20
4	F3	Temperature	AND	30
5	F4	Temperature	FROM	20
6	F4	Temperature	TO	30
7	F5	Temperature	EXIST IN	20
8	F5	Temperature	EXIST IN	25
9	F5	Temperature	EXIST IN	30
10	F6	Weather	IS	cloudy
11	F7	Weather	ISNOT	sunny
12	F8	Weather	EXIST IN TEXT	shower
13	F8	Weather	EXIST IN TEXT	rainy
14	F9	Summer time	= =	true
15	F10	Summer time	= =	false

Table 5.10 example of data from definitive facts that must be stored individually

5.3. Avoiding Duplication

To avoid fact duplication I have developed the following concept by using cross table and look up table. The concept consists of two steps as follows:

1. Finding a relationship between fact-id and fact-name in cross table. The relation is done by using fact-operator.
2. Verifying the corresponding fact-value in look-up table by using fact-id as the key.

Table 5.11 shows the cross table mentioned above. This table describes the relationship between fact-id and fact-name. The numbers in the table are the fact-operator that makes relationship between fact-id and fact-name. Table 5.12 show the look-up table for finding corresponding fact-value by using fact-id. The data from look-up table will verify whether a definitive fact has been already existed or not.

The following algorithm has been derived from the concept:

1. For all column in cross table, finding the same fact-operator in each column.
2. If it is found, take the fact-id as key and uses that key to verify the fact-value in look-up table.

Fact-Id\Fact-Name	Temperature	Weather	Summer Time
F1	1		
F2	3		
F3	7		
F4	9		
F5		21	
F6		22	
F7		23	
F8			24
F9			24

Table 5.11 Cross table for identifying fact duplication

Fact-Id	Fact-Value
F1	20
F2	25
F3	20
F3	30
F4	20
F4	30
F5	20
F5	25
F5	30
F6	cloudy
F7	sunny
F8	shower
F8	rainy
F9	true
F10	false

Table 5.12 Look up table for verifying fact duplication

In the implementation, the data from cross table (table 5.11) can be stored using record-type as follows:

Temperature	F1	F2	F3	F4
	1	3	7	9
Weather	F5	F6	F7	
	21	22	23	
Summer Time	F8	F9		
	24	24		

Table 5.13 cross table record data type format

In table 5.12 the system needs 9x3 matrixes but effectively the system only uses 9 matrix elements (9 spaces). The memory space efficiency due to the size of matrix is therefore about 33% or the redundancy is 67%. This can be calculated by using formula as follows:

$$\eta = \frac{\#\,Facts}{Size\ of\ Matrix} \cdot 100\%$$

According to table 5.13 the size of memory space that is required here (S) will be:

$$S = 2\ x\ \#\,fact\text{-}ids$$

Instead of:

$$S = \#\,fact\text{-}ids\ x\ \#\,fact\text{-}name$$

means "number of"

5.4. Memory Space Comparison

Scenario 1: table 5.10 needs space (S1) about 60 units (S1 = 15 x 4)
Scenario 2: table 5.11 and 5.12 need space (S2) about 57 units (S2 = 9 x 3 + 15 x 2)
Scenario 3: table 5.12 and 5.13 need space (S3) about 30 units (S3 = 9 x 2 + 15 x 2)

The result of comparison shows that scenario 3 save more memory space than other scenarios. The knowledge base organisation concerning scenario 3 is therefore the most efficient. In comparison to the original knowledge base organisation in scenario 1, scenario 3 can save memory space up to 50%.

5.5. Fact Interconnection

The investigation of fact interconnection is done in order to build the interconnection between one fact and other facts that have the same fact-name. This interconnection is needed for the system during analysing and optimising the knowledge using Ternary Grid technique. Building fact interconnection is executed while the expert creates definitive fact. As example, the following facts are created by the expert. He puts the fact into knowledge base step-by-step (see column step-no).

Step-No.	Definitive Fact	Fact. No
1	Score Math = 3	1
2	Score Math > 2	2
3	Score Math > 4	3
4	Score Math <> 3	4
5	Score Math EXIST IN [1,2,3]	5
6	Score Math = 5	6

Table 5.14 given definitive facts as example

Process	Fact Interconnection
After 2nd step	Fact 1 is part of Fact 2
After 4th step	Fact 1 is complement of Fact 4
After 5th step	Fact 1 is part of Fact 5
After 6th step	Fact 6 is part of Fact 2 Fact 6 is part of Fact 3 Fact 6 is part of Fact 4

Table 5.15 building fact interconnection

Explanation:

The process after 2nd step, interconnection between Fact 1 and Fact 2 is done as "part-of" interconnection because the value of Fact 1 is part of the value set of Fact 2. It can be explained in mathematical expression as follows:

$$\text{Fact 1} \subset \text{Fact 2}$$

(Subset)

The process after 5th step and after 6th step has the same case like the process after 2nd step. In the process after 6th step, Fact 6 is interconnected to several facts.

The process after 4th step, interconnection between Fact 1 and Fact 4 is done as "complement of" interconnection because the value of Fact 1 is complement of the value set of Fact 4. It can be explained in mathematical expression as follows:

$$\text{Fact 1} = \text{Fact 4}^{C}$$

Or

$$C_{Fact1}(Fact\,4) = \{x \mid x \in Fact1 \wedge x \notin Fact4\}$$

(All elements of Fact 1 are not elements of Fact 4)

If conclusion part of a rule contains a fact that has interconnection to other facts, this rule will have multiple conclusions. For example, if "Fact 1" is fired then "Fact 22" is fired as well.

Chapter 6

Knowledge Elicitation using Ternary Grid

6.1. Task of Knowledge Elicitation

This section concerns the tasks of knowledge elicitation that are concluded from [NEG02], [LUS98], [WIS92], [RIK91], [BCF89], [INM97]. The knowledge elicitation in this work is carried out into following tasks:

- Collecting
- Interpreting
- Analysing
- Designing

The dataset used in this paper is taken from an example of admission process in the University. The following elicitation process will demonstrate the potential power of Ternary Grid knowledge elicitation technique in handling admission process.

6.2. Collecting Knowledge

This step requires extracting knowledge from the expert. This extraction is mainly done through conducting interviews with the experts [INM97]. The expert gives information to the knowledge engineer. The expert fills the tabular form which is given by knowledge engineer to express their knowledge. It is shown in table 6.1. This information is called “training rules”.

No.	Requirement to Student	Result
1	GPA >= 2,75 AND language test passed	Allowed to do selection test
2	Allowed to do selection test	Pre-Selection passed
3	Pre-Selection passed AND Selection test passed	Get admission to study
4	TOEFL >= 550 (paper base) AND Valid Duration <= 2 years	Language test passed (language test is not required)
5	TOEFL >= 550 (paper base) AND Valid Duration <= 2 years AND English course duration >= 2 year	language test passed
6	GPA between 2,5 and 2,75 AND Language test passed AND Mathematic score >= B	Language test passed
7	GRE >= 900 (paper base) AND Valid Duration <= 5 years	Language test passed
8	GPA >= 2,75 AND language test passed	Allowed to take preparation courses for study
9	Selection test passed AND Pre-Selection passed	Get admission to study

Table 6.1 Expert's expression as training rules

6.3. Interpreting

The collected information now has to be reviewed and the relevant information picked out. At first the information will be general and the knowledge engineer will only be able to define an overall specification which establishes the project goals, constraints and scope [INM97].

Knowledge engineer reviews of collected information and identification of key pieces of knowledge. The following results are shown as follows. This yields 14 factual knowledge or facts as shown in table 6.2.

From the table 6.2 one can see that there is not duplication factual knowledge or fact. Every fact (F1 to F14) represents one unique verbal expression.

Verbal Expression	Fact
GPA >= 2,75	F1
Language test passed	F2
Allowed to do selection test	F3
Pre-Selection passed	F4
Selection test passed	F5
Get admission to study	F6
TOEFL >= 550 (paper base)	F7
Valid Duration <= 2 years	F8
English course duration >= 2 year	F9
GPA between 2,5 and 2,75	F10
Mathematic score >= B	F11
GRE >= 900 (paper base) AND	F12
Valid Duration <= 5 years	F13
Allowed to take preparation courses for study	F14

Table 6.2 Set of factual knowledge from expert

The tabular form which is filled by expert is converted into IF-THEN rule notation. It is shown in table 6.3 as follows:

No.	IF	THEN
1	F1 AND F2	F3
2	F3	F4
3	F4 AND F5	F6
4	F7 AND F8	F2
5	F7 AND F8 AND F9	F2
6	F10 AND F2 AND F11	F2
7	F12 AND F13	F2
8	F1 AND F2	F14
9	F5 AND F4	F6

Table 6.3 Rules in IF-THEN notation

Knowledge engineer converts this notation into the following Ternary Grid. It is shown in Table 6.4. Every element of condition part is represented by value “1”. Every element

of conclusion part is represented by value “2”. If an element of condition part and an element of conclusion part meet each other, the value is summed up and yields value “3”. Value “0” means that there is no relation between rule and fact.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14
R1	1	1	2	0	0	0	0	0	0	0	0	0	0	0
R2	0	0	1	2	0	0	0	0	0	0	0	0	0	0
R3	0	0	0	1	1	2	0	0	0	0	0	0	0	0
R4	0	2	0	0	0	0	1	1	0	0	0	0	0	0
R5	0	2	0	0	0	0	1	1	1	0	0	0	0	0
R6	0	3	0	0	0	0	0	0	0	1	1	0	0	0
R7	0	2	0	0	0	0	0	0	0	0	0	1	1	0
R8	1	1	0	0	0	0	0	0	0	0	0	0	0	2
R9	0	0	0	1	1	2	0	0	0	0	0	0	0	0

Table 6.4 Filled Ternary Grid with “0” value

After neglecting “0” values or changing it into empty grid box, the whole grid becomes as follows:

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14
R1	1	1	2											
R2			1	2										
R3				1	1	2								
R4		2					1	1						
R5		2					1	1	1					
R6		3								1	1			
R7		2										1	1	
R8	1	1												2
R9				1	1	2								

Table 6.5 Filled Ternary Grid without “0” value

6.4. Analysing

The key pieces of knowledge defined in the previous stages will provide an insight into forming theories on how the knowledge will be organised and what problem-solving strategies will be used. Early attempts will identify the important concepts used by the expert. Relationships between concepts and how the expert uses them to solve problems will be looked at in more detail at a later stage [INM97].

Knowledge engineer analyses the organization of knowledge and forms it into an optimal structure. The analysing process consists of following steps:

6.4.1. Consistency and Attainability

Analyse the consistency deals with the evaluating and eliminating the inconsistent of rules. Inconsistent rule is a rule that does not have real meaning because the conclusion part is also put back into condition part. The condition part of these rules can only be satisfied if the conclusion part is known. But the goal of Production Rule is to gain the conclusion. It is of no account to process the condition part if the conclusion part has already been known. If this rule is applied, it will produce different value in the same fact. That's way it is inconsistent.

Analyse the attainability deals with evaluating and eliminating unattainable of rules. The case mentioned above will also lead to an unattainable or unreachable rule. The conclusion part of these rules will never be satisfied because the condition part will never be satisfied as well. This occurs because the conclusion part is a part of condition part.

The inconsistent or unattainable Rule has the following formula structure:

IF <…> & <…> & <**Y**> THEN <**Y**> + <…> + <…>

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14
R1	1	1	2											
R2			1	2										
R3				1	1	2								
R4		2					1	1						
R5		2					1	1	1					
R6		3								1	1			
R7		2										1	1	
R8	1	1												2
R9				1	1	2								

Table 6.6 Inconsistent or unattainable rule

Using Ternary Grid, this phenomenon can be immediately recognised by evaluating the value of grid, see table 6.6. If the value is "3", it means the rule is inconsistent or unattainable. Rule R6 is an inconsistent or unattainable rule because the value of F2 is "3". Rule R6 is therefore not accepted.

Unattainable rule can also occur if at least two or more facts that have the same fact-name appear in the condition part of rule. For example if the following rule is given:

IF <*F1*> AND <*F2*> AND <*F10*> **THEN** <*F14*>

This rule is unattainable, because F1 and F10 appear in the condition part of rule and both rules have the same fact-name, i.e. "GPA". The reason is that the "true" condition for both rules can not happen at the same time. This rule will also never be satisfied or fired.

Another unattainable rule can occur if condition part and conclusion part of rule contain a different fact but the same fact name. The following facts are given as example in the following table:

Verbal Expression	Fact
Physic Score > 80%	F1
Physic Score = 75%	F2
Mathematic Score > 80%	F3
Examination passed	F4

Table 6.7 given set of facts

The following rule is given:

IF <*F1*> AND <*F3*> **THEN** <*F2*> OR <*F4*>

This rule is also unattainable, because the same fact-name appears in condition and conclusion part of rule. In condition part is represented by F1 and in conclusion part by F2. This rule will not be accepted by the system.

6.4.2. Redundancy

Analysing redundancy deals with investigating and eliminating redundant rules that can appear in the form of:

- Repeating Fact
- Repeating Rule
- Rule with Unnecessary Condition
- Transitive Rule (Transitive Property of Equality [WIK03])
- Rule with Repeating Condition

(1)Repeating Fact

A repeating fact is the same fact that appears within a condition part of a rule. Using the Ternary Grid, repeating facts will never occur. For example:

Rule R1: **IF** <*F1*> AND <*F3*> AND <*F1*> **THEN** <*F4*>

F1 appears two times in rule R1. This repeating fact can be avoided by using logic operator “OR” for assigning value into Ternary Grid. The following table shows the process of value assignment into fact.

No.	Process	Result
1	Initial value	F1 = 0
2	First Assignment: F1 = F1 OR 1	F1 = 0 OR 1 F1 = 1
3	Second Assignment F1 = F1 OR 1	F1 = 1 OR 1 F1 = 1

Table 6.8 Assignment process in Ternary Grid

Rule R1 will always be stored in Ternary Grid as follows:

	F1	F2	F3	F4
R1	1		1	2

Table 6.9 There is no repeating fact in Ternary Grid

This means:

Rule R1: **IF** <*F1*> AND <*F3*> **THEN** <*F4*>

The logical OR-operator will be used for all assignment process for this Ternary Grid Technique.

(2)Repeating Rule

Repeating rule is a rule that is identical with another existing rule. Using Ternary Grid, a repeating rule can easily be recognised by evaluating the rows that has the same value. Table 6.10 shows the repeating rule.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14
R1	1	1	2											
R2			1	2										
R3				1	1	2								
R4		2					1	1						
R5		2					1	1	1					
R6														
R7		2										1	1	
R8	1	1												2
R9				1	1	2								

Table 6.10 Repeating rules

The following description explains how the repeating rule can be recognised. Find column in which the number of value “2” appears more than one time. In this case, in column F2 and F6 value “2” appears more than one time. According to column F2, rows R4, R5 and R7 will be investigated. According to column F6, rows R3 and R9 will be investigated. The number and position of value “1” in every row is compared. Due to column F2, comparison is done through row R4, R5 and R7. Consequently for column F6 comparison is done through row R3 and R9. If the number and position of value “1” of a row is the same like the number and position of value “1” of another row, the condition part of that rule is identical with the condition part of another rule. This means, those rules are identical. Row R3 and R9 are identical. One of them is therefore to be eliminated. In this case row R9 is eliminated.

(3)Rule with Unnecessary Condition

Redundant rule can also appear if for the same conclusion part, the number of consecutive values “1” of a rule has more than other rule. It occurs to rules that are represented by the row R4 and R5 in table 6.11. The number of values “1” in row R5 appears more than the number of values “1” in row R4.

In this Ternary Grid, these redundant rules can easily be recognised by investigating the number of value “1” in rows where the value “2” appears in the same column. Find

column in which the number of value “2” appears more than one time. According to column F2, row R4, R5 and R7 appear, then compare the number and position of value "1" of those rows. The number of consecutive values “1” in row R5 is more than the number of consecutive values “1”in row R4. Row R5 is redundant and therefore eliminated.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14
R1	1	1	2											
R2			1	2										
R3				1	1	2								
R4		2					1	1						
R5		2					1	1	1					
R6														
R7		2										1	1	
R8	1	1												2
R9														

Table 6.11 Rule with unnecessary condition

(4)Transitive Rule (Transitive Property of Equality)

Due to the principle of the equality (if a=b, b=c then a=c) [WIK03], the effect of rule R2 that is represented by row R2 is only caused by the conclusion part of rule R1 (see table 6.12). We can determine that rule R2 is caused by the condition part of rule R1 as well. If it occurs, a new rule will be generated to accommodate that statement.

In this Ternary Grid a transitive rule can be easily recognised by investigating the number of values “1” in a row. If a rule has only one fact in condition part, the number of value “1” in corresponding row appears only one time. If this fact is influenced by conclusion part of another rule, then this fulfils the principle of the equality.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14
R1	1	1	2											
R2			1	2										
R3				1	1	2								
R4		2					1	1						
R5														
R6														
R7		2										1	1	
R8	1	1												2
R9														

Table 6.12 Transitive rule

After generating the transitive rule, the whole Ternary Grid becomes as follows:

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14
R1	1	1	2											
R2			1	2										
R3				1	1	2								
R4		2					1	1						
R5														
R6														
R7		2										1	1	
R8	1	1												2
R9														
R10	1	1		2										

Table 6.13 A new rule R10 is generated

(5)Rule with Repeating Condition

Another redundant rule can appear in term “repeating condition of rule”. It is explained in the following example and description.

Rule R1: **IF** <*F1*> AND <*F2*> **THEN** <*F3*>

Rule R2: **IF** <*F1*> AND <*F2*> AND <*F3*> **THEN** <F4>

After converting into Ternary Grid, it becomes as follows:

	F1	F2	F3	F4
R1	1	1	2	
R2	1	1	1	2

Table 6.14 Rule with repeating condition

The set of condition part of rule R1 (R11) is subset of condition part of rule R2 (R21). The conclusion part of rule 1 (fact F3 or R12) influences the condition part of rule R2. In mathematical expression, it is described as follow:

$$R11 = \{a_{1j} \mid j = 1\},\ R12 = \{a_{1j} \mid j = 2\},\ R21 = \{a_{2j} \mid j = 1\}$$

$$(R11 \cup R12) \subset R21$$

In this Ternary Grid, this phenomenon can easily be recognised by comparing the condition part of rule R2 with all element of rule R1. The result will be a new rule R3 that has multiple conclusions. The previous rules R1 and R2 are deleted. It is shown in table 6.15:

	F1	F2	F3	F4	
R1	1	1	2		⇐ deleted
R2	1	1	1	2	⇐ deleted
R3	**1**	**1**	**2**	**2**	

Table 6.15 A new rule R3 with multiple conclusion

6.4.3. Multiple Conclusions

The discussion concerns the situation if two rules or more have identical condition part but different conclusion part. This situation will actually not lead to a serious problem.

But if this situation can be optimised, it will give better structure for knowledge base, mainly during reasoning process. The conclusion part of those rules can be merged and yields a new rule with multiple conclusions. Fact F14 is moved into rule R1 and rule R9 is eliminated. It is shown in table 6.16.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14
R1	1	1		2										
R2														
R3				1	1	2								
R4		2					1	1						
R5														
R6														
R7		2										1	1	
R8	1	1												2
R9														
R10	1	1		2										

Table 6.16 Rule with multiple conclusions

6.4.4. Rotating Chain

Rotating chain is very crucial problem. If the chain of rule rotates, it will lead to endless situation and produces wrong decision. This situation will also put the reasoning process of inference engine at risk. The inference engine will never end the chaining process. A correct chain of rules will never rotate. There must be at least one “end-rule” or “top-rule”. It is a rule which its conclusion part does not influence another rule. If there is no end-rule, the rule chain rotates (endless chain). The whole rules are therefore not accepted and must be created from the beginning by the expert or knowledge engineer.

In this Ternary Grid the end-rule can be easily recognised by finding columns in which only the value “2” appears. A rule is only an end rule, if the rule’s fields only contain the value “2” within these columns. Table 6.17 shows set of rules that have correct chain. Rule R3 is an end-rule because in column F6 only value “2” of rule R3 appears.

Even value “2” appears alone in column F14 but rule R1 is not an end-rule because it has another conclusion in column F4. In this column F4 value “2” and “1” appear.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14
R2	1	1		2										2
R2														
R3				1	1	2								
R4		2					1	1						
R5														
R6														
R7		2										1	1	
R8														
R9														
R10	1	1		2										

Table 6.17 Set of rules with correct chain

Example of set of rule that have rotating chain is given as follow:

Rule R1: **IF** <*F1*> and <*F4*> **THEN** <*F3*>
Rule R2: **IF** <*F3*> and <*F7*> **THEN** <*F6*>
Rule R3: **IF** <*F6*> and <*F8*> **THEN** <*F5*>
Rule R4: **IF** <*F2*> and <*F5*> **THEN** <*F1*>

After converting into Ternary Grid, it becomes as follows:

	F1	F2	F3	F4	F5	F6	F7	F8
R1	1		2	1				
R2			1			2	1	
R3					2	1		1
R4	2	1			1			

Table 6.18 Set of rules that have rotating chain

From table 6.18 one can not see that there is a column, in which only value “2” appears. It means there is not end-rule. This set of rules has therefore rotating chain.
The illustration of rotating chain can be seen in the figure 6.1.

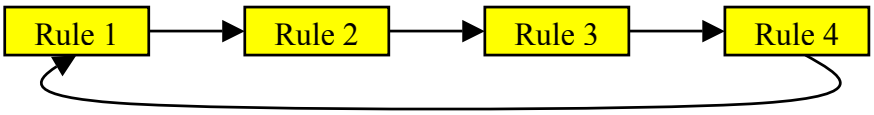


Figure 6.1 Illustration of rotating chain

6.5. Designing Knowledge

After the three main steps above have been completed, the knowledge engineer will have a new understanding of the problem domain and will form new concepts and problem-solving strategies that will need further investigation. This new understanding will help the knowledge engineer in designing new ways of collecting more specific knowledge from the expert. The whole process then starts again. This is really a prototype cycle [INM97]

To provide support for further problem-solving strategies, an appropriate knowledge content and structure are required. To achieve that performance, optimising the knowledge structure is carried out by compressing the knowledge. Illustration of knowledge will explain the expert an overview of knowledge structure. This section deals only in providing support for further concept and problem-solving strategies through compression and illustration of knowledge.

6.5.1. Compression

Compression of knowledge optimises the content and the structure of the knowledge. Unused rows and columns are eliminated. The name of corresponding rules and facts will be updated as well. The grid space will be therefore as much space as one needs.

In table 6.19 shows that the grids with line pattern represent unused rows and columns. The rows and columns that have this pattern can be therefore eliminated.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14
R1	1	1		2										2
R2														
R3				1	1	2								
R4		2					1	1						
R5														
R6														
R7		2										1	1	
R8														
R9														
R10	1	1		2										

Table 6.19 Unused rows and columns with line pattern

The Ternary Grid after compression becomes (table 6.20):

	F1	F2	F4	F5	F6	F7	F8	F12	F13	F14
R1	1	1	2							2
R3			1	1	2					
R4		2				1	1			
R7		2						1	1	
R10	1	1	2							

Table 6.20 Ternary Grid after compression

The name of rules and facts which is corresponds each other are changed or updated as follows (table 6.21):

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
R1	1	1	2							2
R2			1	1	2					
R3		2				1	1			
R4		2						1	1	

Table 6.21 Updated rule and fact name

The list of the factual knowledge is updated as well. It is shown in the following table (table 6.22):

Explanation	Fact
GPA >= 2,75	F1
Language test passed	F2
Pre-Selection passed	F3
Selection test passed	F4
Get admission to study	F5
TOEFL >= 550 (paper base)	F6
Valid Duration <= 2 years	F7
GRE >= 900 (paper base) AND	F8
Valid Duration <= 5 years	F9
Allowed to take preparation courses for study	F10

Table 6.22 Updated factual knowledge

The set of rules become (table 6.23):

No.	IF	THEN
1	F1 AND F2	F3, F10
2	F3 AND F4	F5
3	F6 AND F7	F2
4	F8 AND F9	F2

Table 6.23 Updated set of rules

6.5.2. Illustration

The updated rules are now valid and ready for transferring and processed by inference engine. Illustration of knowledge can provide information to content and structure of knowledge. Due to the text illustration [HAS02] noted that an illustration should reflect the contents of the text (in this case the knowledge), illustrations should reflect the user's interest, or the system's beliefs on user's interests and the generated illustration as well as the text should be furnished with interaction points. In this section the

illustration of knowledge deals with visualisation of content and structure of rules. It explains which rules the knowledge contains and how the chain of rules is constructed. These can be illustrated in the following diagram (figure 6.2):

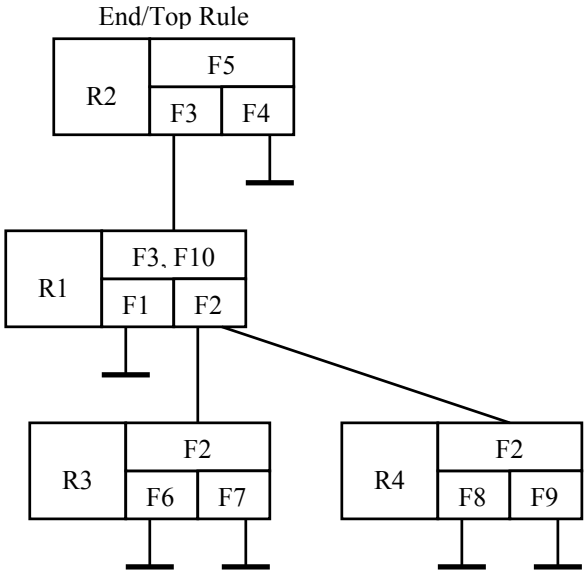


Figure 6.2 Illustration of knowledge

Explanation to diagram notation:

Rn	Yn
	Xn

Rn : Rule n

Xn : Condition part of Rn, Yn : Conclusion part of Rn

———— Rule chain

⊥ Input fact. It is a fact that is not produced by any rule.

6.6. Development of Algorithms

The algorithms that are developed here concern elimination of redundancy, modification of rule to build multiple conclusions and investigation of rule set error in the form of rotating chain using Ternary Grid. Elimination of unattainable rules and repeating fact is not discussed here because it is done while the expert creates a new

rule. Unattainable rules can be immediately recognised if a new rule is created. Repeating fact will never occur because the system uses “OR” operator for assigning value grid into Ternary Grid.

Ternary Grid can be considered as matrix. The following mathematical descriptions in the form of set operations explain developed algorithms. Matrix A represents discussed Ternary Grid.

	$F1$	$F2$	…	FJ
$R1$	a_{11}	a_{12}		a_{1j}
…				
RI	a_{i1}	a_{i2}		a_{ij}

Table 6.24 Ternary Grid in Matrix notation

$$i = \{1,..,I, I \in \mathrm{N}\}$$
$$j = \{1,...,J, J \in \mathrm{N}\}$$
$$J \geq I + 1$$
$$a_{ij} = \{0,1,2\}$$

Value 0 is represented by empty matrix cell.

The following value sets are needed for next stages:

- The set of condition parts in row i is determined as follows:
$$Ri1 = \left\{ j \mid a_{ij} = 1 \right\}$$
- The set of conclusion parts in row i is determined as follows:
$$Ri2 = \left\{ j \mid a_{ij} = 2 \right\}$$
- The set of condition parts in column j is determined as follows:
$$Fj1 = \left\{ i \mid a_{ij} = 1 \right\}$$
- The set of conclusion parts in column j is determined as follows:
$$Fj2 = \left\{ i \mid a_{ij} = 2 \right\}$$

For the expert all formulas above can be illustrated as follows:

$$R = \{R1, ..., RI\}$$
$$Ri1 = \left\{ Fj \;\middle|\; a_{ij} = 1 \right\}$$
$$Ri2 = \left\{ Fj \;\middle|\; a_{ij} = 2 \right\}$$

$$F = \{F1, ..., FJ\}$$
$$Fj1 = \left\{ Ri \;\middle|\; a_{ij} = 1 \right\}$$
$$Fj2 = \left\{ Ri \;\middle|\; a_{ij} = 2 \right\}$$

(1) Elimination of Inconsistent Rule

Mathematical description:

- Find rows, in which value 3 appears:

$$B = \left\{ i \;\middle|\; a_{ij} = 3 \right\}$$

- Remove row duplication

$$C = \bigcup_{b \in B} b$$

Derived algorithm:

1. For each number in row, starting with the first and ending with the last row.
2. For each number in column, starting with the first and ending with the last column.
   2.1. Look at the grid value of the column; Find value 3; Put the row number into the list B.
3. For each number in list L, starting with the first and ending with the last number.
   3.1. Remove duplicate from list B; put new list C without duplicate.
   3.2. Remove row, where its number exist in list C.

Another alternative to solve this problem can be as follow:

$$B = \left\{ i \;\middle|\; \left(\sum_j \left(a_{ij} \; div \; 3 \right) \right) > 0 \right\}$$

1. For each number in row, starting with the first and ending with the last row.
2. For each number in column, starting with the first and ending with the last column

2.1. Look at the grid value of the column; Find value 3; Put the row number into the list B

3. For each number in list B, starting with the first and ending with the last number

3.1. Remove row, where its number exist in list B

(2) Elimination of Repeating Rule

Mathematical description:

- Find rows in column j, where value 2 appears more than once:

$$Bj = \left\{ b \,\middle|\, b \in Fj2, |Fj2| \geq 2 \right\}$$

- Find pairs of rows for comparison, which has the same number of logic term in condition part

$$Cj = \left\{ (p,q) \,\middle|\, p \in Bj, q \in Bj, p < q, |Bj| > 0, |Rp1| = |Rq1| \right\}$$

- Remove duplication of row pairs

$$C = \bigcup_{j \in \mathrm{N}} Cj$$

- Find pairs of rows, which has the same condition part

$$D = \left\{ (p,q) \,\middle|\, (p,q) \in C, Rp1 = Rq1 \right\}$$

Derived algorithm

1. For each number in column, starting with the first and ending with the last column.

1.1. For each number in row, starting with the first row and ending with the second last row.

1.1.1. Look at the grid value; Find value 2; Let R1 be the row number; Let X1 be number of value 1.

1.1.2. For each number in row, starting with the second row and ending with the last row.

1.1.2.1. Look at the next grid value; Find the value 2; Let R2 be the row number; Let X2 be number of value 1.

1.1.2.2. Compare X1 with X2.

1.1.2.3. If X1 = X2, then compare every value 1 in R1 with every value 1 in R2.

1.1.2.4. If (R1 is identic with R2), then compare every value 2 in R1 with every value 2 in R2. Let Y be number of found value. Let Y1 be number of value 2 in R1; Let Y2 be number of value 2 in R2.

1.1.2.5. If (Y = Y1) and (Y2 > Y), then remove R1.

1.1.2.6. Else If (Y = Y1) and (Y = Y2), then remove R2.

1.1.2.7. Else If (Y = Y2) and (Y1 > Y), then remove R2

(3) Elimination of Rule with Unnecessary Condition

Mathematical description:

- Find rows in column j, where value 2 appears more than once:

$$Bj = \left\{ b \mid b \in Fj2, |Fj2| \geq 2 \right\}$$

- Find pairs of rows for comparison, which has different number of logic term in condition part

$$Cj = \left\{ (p,q) \mid p \in Bj, q \in Bj, p < q, |Bj| > 0, |Rp1| \neq |Rq1| \right\}$$

- Remove duplication of row pairs

$$C = \bigcup_{j \in \mathrm{N}} Cj$$

- Find pairs of rows, where a part of pair is subset of another part

$$D = \left\{ (p,q) \in C, Rp1 \in Rq1 \; or \; Rq1 \in Rp1 \right\}$$

$$E = \left\{ p \mid p \in (p,q), (p,q) \in D, Rq1 \in Rp1 \right\} \text{ or}$$

$$E = \left\{ q \mid q \in (p,q), (p,q) \in D, Rp1 \in Rq1 \right\}$$

Derived algorithm:

1. For each number in column, starting with the first and ending with the last column.
   1.1. For each number in row, starting with the first row and ending with the second last row.
      1.1.1. Look at the grid value; Find value 2; Let R1 be the row number; Let X1 be number of value 1.

1.1.2. For each number in row, starting with the second row and ending with the last row.

1.1.2.1. Look at the next grid value; Find the value 2; Let R2 be the row number; Let X2 be number of value 1.

1.1.2.2. Compare X1 with X2.

1.1.2.3. If X1 ≠ X2, then compare every value 1 in R1 with every value 1 in R2; Let Y be number of equal values.

1.1.2.4. If Y = X1, then Remove row R2.

1.1.2.5. If Y = X2, then Remove row R1.

(4) Elimination of Transitive Rule

Mathematical description:

- Find pair of rows and columns, where value 1 appears only once in any row:

$$B = \left\{ (i, j) \,\middle|\, a_{ij} = 1, |Ri1| = 1 \right\}$$

- Find rows that its value 2 corresponds to value 1 of row i in B.

$$C = \left\{ x \,\middle|\, x \in Fj2, j \in (i, j), (i, j) \in B \right\}$$

New rule can be produced as follows:

$$Rn1 = \left\{ j \,\middle|\, j \in Ri1, n \in \mathrm{N}, i \in C \right\}$$

$$Rn2 = \left\{ j \,\middle|\, j \in Ri2, n \in \mathrm{N}, i \in B \right\}$$

Derived algorithm:

1. For each number n1 in row, starting with the first row and ending with the last row.

1.1. If number of value 1 of row(n1) is equal 1, then let F be column number in which value 1 exists.

1.1.1. For each number n2 in row, starting with the first row and ending with the last row.

1.1.1.1. If (n2 ≠ n1), then

1.1.1.1.1. Look at the grid value, find value 2.

1.1.1.1.2. Let X be the number of row, where value 2 is found.

1.1.1.1.3. Create new row, copy all value 1 of row(X) and all value 2 of row(n2) into new row.

(5) Elimination of Rule with Repeating Condition

Mathematical description:

- Find rows, their condition and conclusion part is subset of condition part a rule.

$$Bi = \left\{ p \;\middle|\; |Ri1| \geq |Rp1| + |Rp2|, i \neq p, p \in \mathrm{N} \right\}$$

$$Ci = \left\{ p \;\middle|\; (Rp1 \cup Rp2) \in Ri1, i \neq p, p \in Bi \right\}$$

A new rule can be produced as follows:

$$Rn1 = \left\{ p \;\middle|\; p \neq q, p \in Ri1, q \in Rs2, s \in Ci, n \in N \right\}$$

$$Rn2 = \left\{ p \;\middle|\; p \in \left(Ri2 \cup \bigcup_{q \in Ci} Rq2 \right), n \in N \right\}$$

All rules that have repeating condition Ci can be removed.

Derived algorithm:

1. For each number n1 in row, starting with the first row and ending with the second last row.
   1.1. For each number n2 in row, starting with the second row and ending with the last row.
      1.1.1. Let V1 be number of value 1 of row(n1).
      1.1.2. Let V2 be number of value 1 of row(n2).

      1.1.3. If (V1 > V2), then
         1.1.3.1. Let X1 be sum of number of value 1 and value 2 of row(n2).
         1.1.3.2. Let X2 be sum of number of value 1 of row(n1).
         1.1.3.3. Compare value 1 and value 2 of row(n2) with value 1 of row(n1) correspondingly; Let X be number of comparison result.
         1.1.3.4. If (X = X1), then

1.1.3.4.1. Subtitute value 1 of row(n1) with value 2 of row(n2) correspondingly

1.1.3.4.2. Remove row(n2)

1.1.4. If (V1 < V2), then

1.1.4.1. Let X1 be sum of number of value 1 and value 2 of row(n1).

1.1.4.2. Let X2 be sum of number of value 1 of row(n2).

1.1.4.3. Compare value 1 and value 2 of row(n1) with value 1 of row(n2) correspondingly; Let X be number of comparison result.

1.1.4.4. If (X = X1), then

1.1.4.4.1. Subtitute value 1 of row(n2) with value 2 of row(n1) correspondingly.

1.1.4.4.2. Remove row(n1).

(6) Building Multiple Conclusions

Mathematical description:

- Find rows that have the same condition part:

$$Bi = \left\{ p \mid Ri1 = Rp1, i < p, p \in \mathrm{N} \right\}$$

- A new rule Rq that has multiple conclusions can be constructed as follows:

$$Rq2 = \left\{ Ri2 \cup Rp2 \mid p \in Bi, q \in N \right\} \text{ and}$$

$$Rq1 = Ri1$$

- After creating new rule, the previous rules can be deleted as follow:

$$R = \left\{ Ri \cup Rp \mid p \in Bi \right\}$$

Derived Algorithm:

1. For each number n1 in row, starting with the first row and ending with the second last row.
   1.1. Let X1 be number of value 1 of row(n1)
   1.2. For each number n2 in row, starting with the second row and ending with the last row.

1.2.1. Let X2 be number of value 1 of row(n2).

1.2.2. If X1 is equal X2, then compare every value 1 of row(n1) with row(n2).

1.2.3. If row(n1) is identic with row(n2), then copy all value 2 of row(n1) into row(n2); Remove row(n2).

(7) Investigation of Rotating Chain

Mathematical description:

- Find columns, in which only value 2 appears:

$$Bj = \left\{ b \;\middle|\; b \in Fj2, \left|Fj2\right| \geq 1, \left|Fj1\right| = 0 \right\}$$

If $Bj = 0$ then the chain of rules rotates.

Derived algorithm:

1. Let Y be 0.
2. For each number in column, starting with the first and ending with the last column.
   2.1. For each number in row, starting with the first and ending with the last row.
      2.1.1. Look at the grid value; Find value 1; Let X1 be number of found value.
      2.1.2. Look at the grid value; Find value 2; Let X2 be number of found value.
   2.2. If (X1 = 0) and (X2 > 0), then increment Y by 1.
3. If Y = 0, then Rotating Chain is true
4. If Y > 0, then the chain of rules is working.

Chapter 7

Transferring Knowledge into Knowledge Base

The last process of knowledge acquisition is to transfer the knowledge into database as knowledge base after elicitation process. It concerns with the transforming, storing and extracting the knowledge. The system should be able to transfer knowledge into Ternary Grid knowledge base in database and other way round.

This transferring process refers to some ideas explored in report [KER88], which mentioned four ways in which knowledge base system (KBS) and databases can be integrated: firstly, data can be extracted from databases and imported into knowledge base for processing, and equally, data generated during the execution of a knowledge base may be passed to a database for storage; secondly, there is convergence of the types of facility provided by knowledge base system tools and database management system (DBMS); thirdly, a database can be used to hold a knowledge base; and fourthly, rules, or knowledge, can be generated from data held in a database.

7.1. The Differences among Data, Information and Knowledge

Knowledge base is not database, but they can be integrated. Understanding about a problem across knowledge base and database has been important task for a knowledge engineer. It is considered an important aspect for building the integration between knowledge base and databases. Understanding about differences among data, information and knowledge give view from the problem across knowledge base and database.

A datum is the value of an observable, measurable or calculable attribute. A datum is a statement accepted at face value. Data is plural of datum. Data is more than one such

attribute value. Data is not information, but information is provided by data, because data is always specified in some conceptual context. Data alone and in the abstract therefore, does not provide information. Data on its own has no meaning, only when interpreted by some kind of data processing system does it take on meaning and become information.

Information, in general terms, is data plus conceptual commitments and interpretations. Information is data extracted, filtered or formatted in some way. Information is a term with many meanings depending on context, but is as a rule closely related to such concepts as meaning, knowledge, instruction, communication, representation, and mental stimulus.

Knowledge is a subset of information. But it is a subset that has been extracted, filtered, or formatted in a very special way. More specifically, the information we call knowledge is information that has been subjected to, and passed tests of validation. Common sense knowledge is information that has been validated by common sense experience. Scientific knowledge is information (hypotheses and theories) validated by the rules and tests applied to it by some scientific community. Knowledge is the awareness and understanding of facts, truths or information gained in the form of experience or learning. Knowledge is an appreciation of the possession of interconnected details which, in isolation, are of lesser value [WIK04].

The definition relies on a comparative analysis of data, information and knowledge as follows [AKR04]:

- Data is unstructured values, symbols, numbers etc.
- Information is data with structure and context added
- Knowledge is a conceptual higher level abstraction of information

Example:

- Data: 25
- Information: 25 ^{0}C
- Knowledge: 25 ^{0}C is comfortable temperature for most people. It is called room temperature.

From descriptions above, the relation between data, information and knowledge can be illustrated in the following egg diagram (figure 7.1):

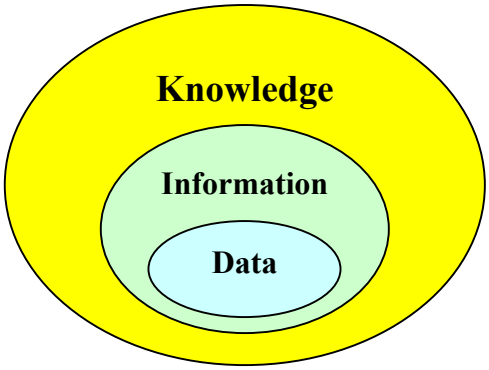


Figure 7.1 Data, information and knowledge in egg diagram

7.2. Database Considerations

There are numerous definition of a database, including the following: Database is any set of information may be called a database. Nevertheless, the term was invented to refer to computerised data, and is used almost exclusively in computing. Sometimes it is used to refer to not yet computerised data, but usually in the process of planning its possible computerisation [WIK04].

In computing, databases are sometimes classified according to their organizational approach. The most prevalent approach is the relational database, a tabular database in which data is defined so that it can be reorganized and accessed in a number of different ways. Computer databases typically contain aggregations of data records or files, such as sales transactions, product catalogues and inventories, and customer profiles. Typically, a database manager provides users the capabilities of controlling read/write access, specifying report generation, and analyzing usage [WHA04].

In the early day of expert systems, expert systems development and database development were considered two independent technologies. Expert systems used inference engine and emphasised making judgment explicit. Database package used

query matching and emphasised performance and shares access to data. There has been a growing realisation that these two technologies are really complementary. They can be used together to solve different parts of some problems.

There are several reasons why people want to connect database and expert systems [HAB90]:

- *Maintenance*. A database can be used externalised the part of the knowledge base that change frequently. This allows a clerk or other non expert to perform some kind or limited maintenance on the expert system using a database management system rather than requiring a knowledge engineer to edit the knowledge base. In general, separating data from knowledge makes the knowledge base easier to modify because it is easier to read. The system's rules clearer since they are not cluttered by large amounts of declarative data.
- *Performance*. When a knowledge base becomes filled with excessive amounts of tabular data, you should consider weather that data can be externalised in database file. This can improve the performance of the knowledge base in both its speed and memory and consumption.
- *Sharing data with other application*. Numerous application can access information and stored in a database file.
- *Sharing data with other users and maintaining security*. Systems that are networked to other PCs can allow database to be accessed by multiple users.

There are number of ways in which a database can work with an expert system, each appropriate under certain circumstances [HAB90]:

- An expert system can be used as a front end for a database, asking question and then initiating a database query.
- An expert system can be used as back end for database, taking the result of a query and analysing it with rules in order to make recommendation or decision.
- The attributes and values of rules can be stored in a database.
- A database can be used to store the cases that have been run through an expert system. In order to keep a record of its reasoning.

There is currently considerable interest in using middleware technology to integrate sources of data and knowledge. Some of these sources are legacy systems (pre-existing databases and expert systems), while others are custom-built services specifically designed to operate as components within distributed information architectures. Some of the common scenarios served by such architectures are [PBF98]:

- Extracting data from databases and providing it as the input to knowledge based systems, which in turn derive new information.
- Extracting data from databases and knowledge from knowledge bases and combining both to compose a new information source.
- Extracting and transforming data and knowledge into constraint programs, the solution of which yields new information.

As mentioned above, transferring process of knowledge is a process of transformation, storage and extraction of knowledge. The knowledge must be first transformed into data and then stored into database as knowledge base. To gain the knowledge from database, the knowledge must be extracted and transformed again into acceptable format of knowledge. Figure 7.2 illustrates the mentioned transferring process of knowledge.

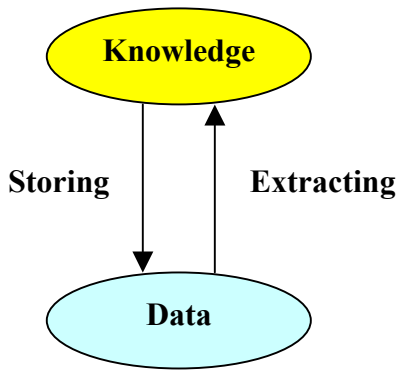


Figure 7.2 Transferring process of knowledge

Designing appropriate structure of database has important role in the integration of knowledge base and database. It will to avoid information loss, which is caused through transferring process.

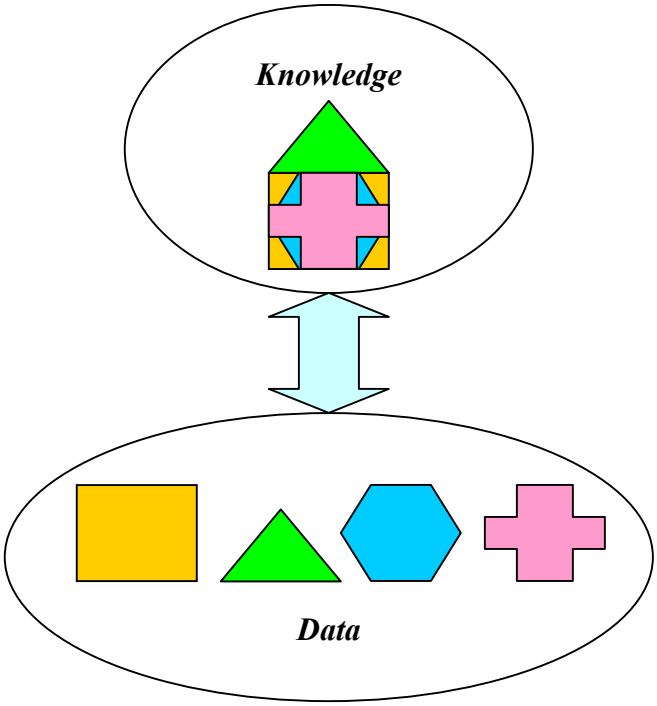


Figure 7.3 Transformation of knowledge and data

Transformation of knowledge into data must consider appropriate data format and structure that will be used for back transformation from data to knowledge.

7.3. Transferring Factual Knowledge

After obtaining process, the factual knowledge will be stored in two forms, i.e. prospective facts and definitive facts. In the prospective facts there are no operator relation between fact-name and fact-value. The transferring process can be explained in the following example by given facts as shown in table 7.1 and figure 7.4.

Fact-name	Possible fact-value
Cloth-colour	Red
	Green
	Yellow
	Blue

Table 7.1 given prospective facts as example

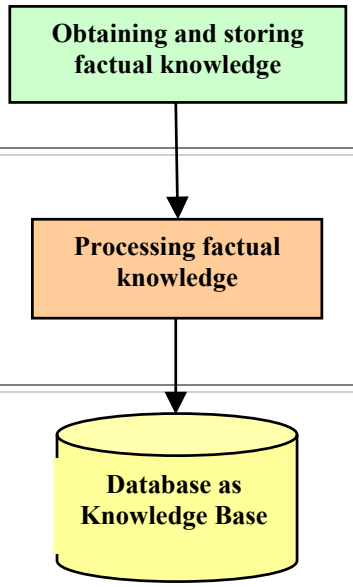
Flow Diagram	Example
	The following definitive fact are given: • Fact 1: Cloth-colour = green • Fact 2: Cloth-colour IN [red, green, blue] Fact Processor will • Eliminate repeating fact • Assign number into every definitive fact, and • Interconnect some facts that have value relationship The following facts will be stored in knowledge base in the form of database: • F1 (represent Fact 1) • F2 (represent Fact 2) • F1 → F2 (if F1 is applied, then F2 will be applied as well)

Figure 7.4 Transferring process of factual knowledge

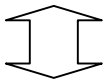
Before the obtained factual knowledge is sent to the knowledge base, it will be first processed by the Factual Knowledge Processor in order to avoid repeating fact and to determine weather a fact has any value relationship or interconnection with other facts. Finally every definitive fact will refer to a unique number that will be used for further processing using Ternary Grid method. This fact number represents a logical term or fact statement. Determining the fact value relation and referring to definite fact number will avoid the collision of facts. Figure 7.1 shows transferring process of factual knowledge.

The definitive facts will be stored in knowledge base in the following structure. This structure follows the database concept (table 7.2).

The last step is to connect those facts into Ternary Grid table. All facts will be placed in the grid column. The number of the facts might be arbitrary but the number of column should be consecutive due to the convenience of the eliciting process and the economy of the number of memory that is proportional to the number of column. To realise that

concept the look up table that correspond column number to and fact number is implemented in the memory. Table 7.3 shows the look up table that relate the column number of the grid and the fact-number.

Fact-number	Fact-name
F1	Cloth-colour
F2	Cloth-colour



Fact-number	Operator	Fact-value
F1	=	green
F2	IN	red
	IN	green
	IN	blue

Table 7.2 Definitive facts in database

Fact Number	F1	F3	F9	F6	F17	F23
Column Number	1	2	3	4	5	6

Table 7.3 Look up table for column-number and fact-number

7.4. Transferring Judgmental Knowledge

Transferring the judgmental knowledge concerns the transformation of rules from classical format into DNF-format and finally DNF-format into Ternary Grid format and storing them into database as knowledge base.

After the rules have been obtained in the classical format, they will be converted into DNF-form. The following example shows a rule in the classical format or arbitrary logical expression:

Rule Subtask 1

IF (*F1* AND (*F2* OR *F3* AND *F4*) OR *F5* AND *F6*) **THEN** *F9*

After the rule has been converted into DNF, the rule becomes:

Rule Subtask 1:

IF (*F1* AND *F2*) OR (*F1* AND *F3* AND *F4*) OR (*F5* AND *F6*) **THEN** F9

The rule can be derived into small rules, in which only DNF-format appears. It represents all logical conjunction above.

Rule R1: **IF** (*F1* AND *F2*) **THEN** *F9*
Rule R2: **IF** (*F1* AND *F3* and *F4*) **THEN** *F9*
Rule R3: **IF** (*F5* AND *F6*) **THEN** *F9*

The rule can also be written in another notation if logic operator AND is represented by the operator *. This looks like as follows:

Rule R1: **IF** (*F1 * F2*) **THEN** *F9*
Rule R2: **IF** (*F1 * F3 * F4*) **THEN** *F9*
Rule R3: **IF** (*F5 * F6*) **THEN** *F9*

The list of rules above can be displayed in the Ternary Grid as follow (table 7.4):

	F1	**F2**	**F3**	**F4**	**F5**	**F6**	**F9**
R1	1	1					2
R2	1		1	1			2
R3					1	1	2

Table 7.4 Rules in Ternary Grid

The knowledge elicitor transforms those rules into system memory in the Ternary Grid format. Due to the space consumption and process convenience the rule and fact will be stored in the following format:

Rule format (Row Representation):

The rule consists of two single dimension arrays. First one is for condition part and other one is for conclusion part. The rule notation can be described as follows:

Rn: [list of facts in condition part] [list of facts in conclusion part]

It can be also described as follow:

Rn: [List 1] [List 2]

List 1 = list of column number, where the grid value equals 1
List 2 = list of column number, where the grid value equals 2

Rules from table 7.4 can be represented as follows:

R1: [F1, F2] [F9]
R2: [F1, F3, F4] [F9]
R3: [F5, F6] [F9]

In the memory, it is stored in array format as follow:

Row 1: [1, 2] [7]
Row 2: [1, 3, 4] [7]
Row 3: [5, 6] [7]

Remember that fact F9 is represented by column number 7.

Fact format (Column Representation):

Same like for rule, the fact consists of two single dimension arrays. First one is for condition part and other one is for conclusion part. The fact notation can be described as follows:

Fm: [list of rules due to condition part] [list of rules due to conclusion part]

It can be also described as follow:

Fm: [List 1] [List 2]

List 1 = List of rules, where the grid value equals 1
List 2 = List of rules, where the grid value equals 2

Rules from table 7.4 can be represented as follows:

F1: [R1, R2] []
F2: [R1] [ ]
F3: [R2] [ ]
F4: [R2] [ ]
F5: [R3] [ ]
F6: [R3] [ ]
F9: [ ] [R1, R2, R3]

In the memory, it is stored in array format as follow:

Column 1: [1, 2] [ ]
Column 2: [1] [ ]
Column 3: [2] [ ]
Column 4: [2] [ ]
Column 5: [3] [ ]
Column 6: [3] [ ]
Column 7: [ ] [1, 2, 3]

Column 7 represents fact F9.

7.5. Database Design and Structure

The quality of knowledge depends on the quality of database structure. The structure of database must be able to store knowledge in the form of data and to be extracted when the data are transformed into knowledge without having information loss.

An adequate structure of database can be achieved by good organised database design. Database design deals with the design of the logical and physical structure of databases to accommodate the information needs of the users in an organisation for a defined set of applications. The design process contains the following steps [ITS04]:

- planning and analysis
- conceptual design
- logical design
- physical design
- implementation

One of the most important parts of database design process is the data model. The data model focuses on what data should be stored in the database. To put this in the context of the relational database, the data model is used to design the relational tables. Data modelling is the most labour intensive and time consuming part of the development process.

The goal of the data model is to make sure that the all data objects required by the database are completely and accurately represented. Because the data model uses easily understood notations and natural language, it can be reviewed and verified as correct by the end-users.

A data model is a conceptual representation of the data structures that are required by a database. The data structures include the data objects, the associations between data objects, and the rules which govern operations on the objects. As the name implies, the data model focuses on what data is required and how it should be organised. To use a common analogy, the data model is equivalent to an architect's building plans [ITS04].

The Entity-Relation Model (ER) is used during the design process of database. It is the most common method used to build data models for relational databases. To ensure the quality of the knowledge, I have drafted the following data model or database structure for Ternary Grid knowledge base. It is shown in Figure 7.5. This database stores factual and judgemental knowledge.

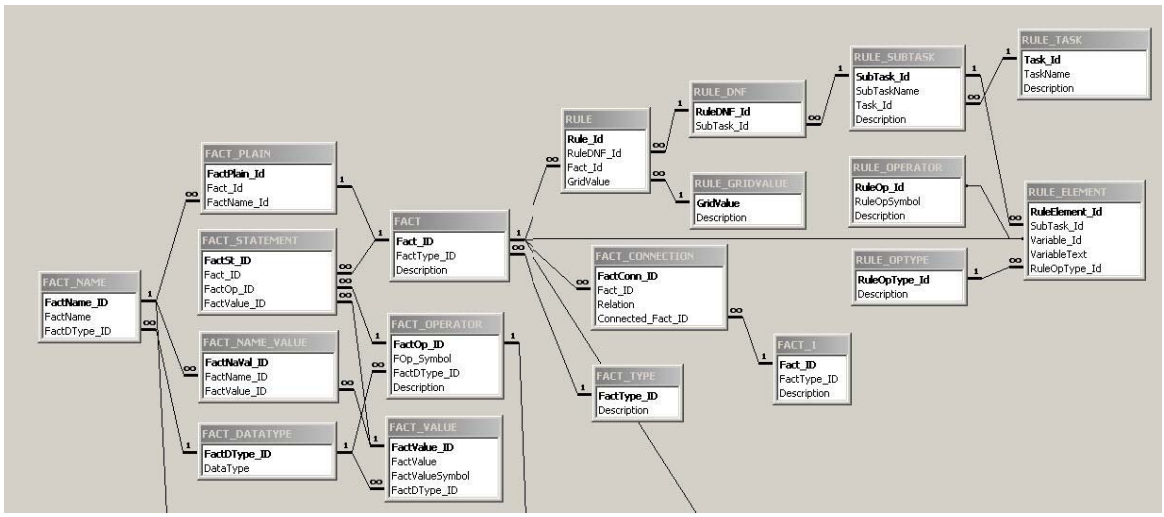


Figure 7.5 Data model or database structure of Ternary Grid knowledge base

The following figure (figure 7.6) shows a cut of whole database structure. This part represents database structure for factual knowledge.

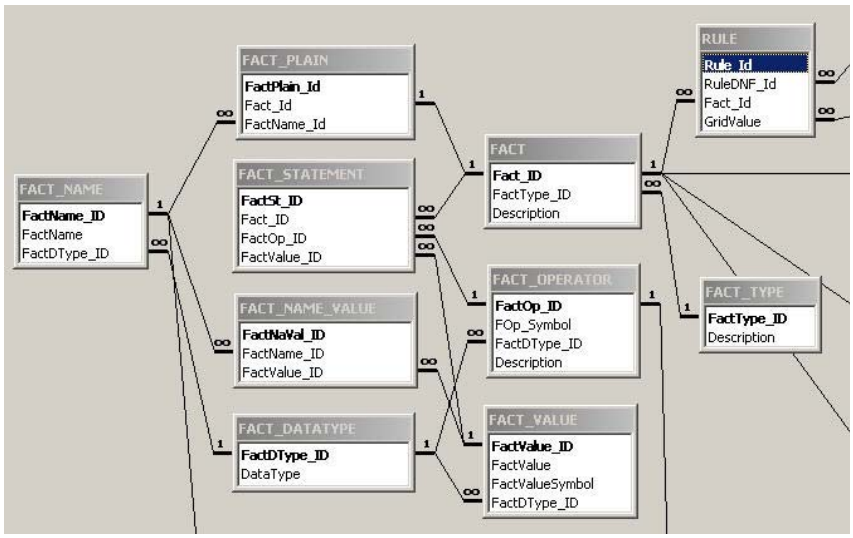


Figure 7.6 Database structure for factual knowledge

Figure 7.7 shows a cut of whole database structure for fact interconnection.

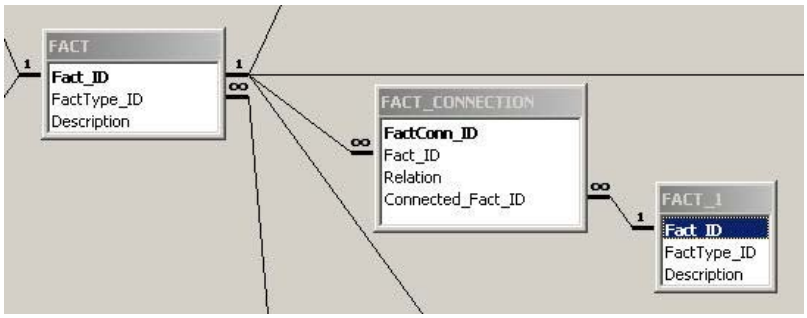


Figure 7.7 Database structure for fact Interconnection

The following figure (figure 7.8) shows a cut of whole database structure for judgemental knowledge. The structure of database represents a hierarchical structure.

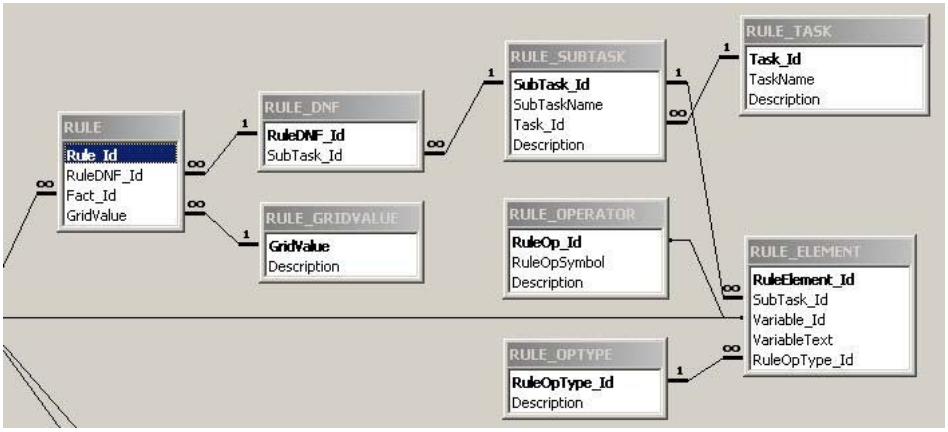


Figure 7.8 Database structure for judgemental knowledge

The hierarchical structure of database structure for judgemental knowledge can be described as follow: any task of the expert is undertaken by at least one rule or more rules that is/are called subtask rule/rules. Every subtask rule consists of at least one DNF-rule or more. Analysing and optimising all DNF-rules will be carried out task-wise by the Ternary Grid Elicitor.

Figure 7.9 illustrates this hierarchical model in rule notation. Figure 7.10 illustrates this hierarchical model in egg diagram.

Task	Subtask
Task 1	**Subtask 1:** *DNF_Rule_1* OR *DNF_Rule_2* OR *DNF_Rule_5* OR …
	Subtask 2: *DNF_Rule_4* OR *DNF_Rule_7* OR *DNF_Rule_9* OR …
	Subtask 3: *DNF_Rule_3* OR *DNF_Rule_6* OR *DNF_Rule_8* OR …

Figure 7.9 Hierarchical rule structure in rule notation

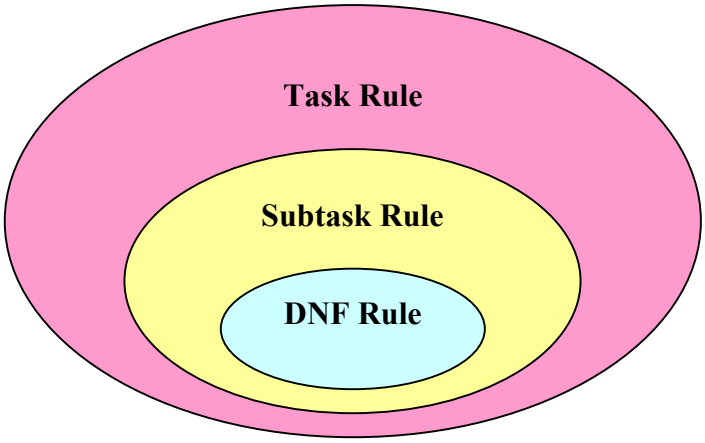


Figure 7.10 Hierarchical rule structure in egg diagram

Accessing Database

The following illustration and description explain the accesing process of database.

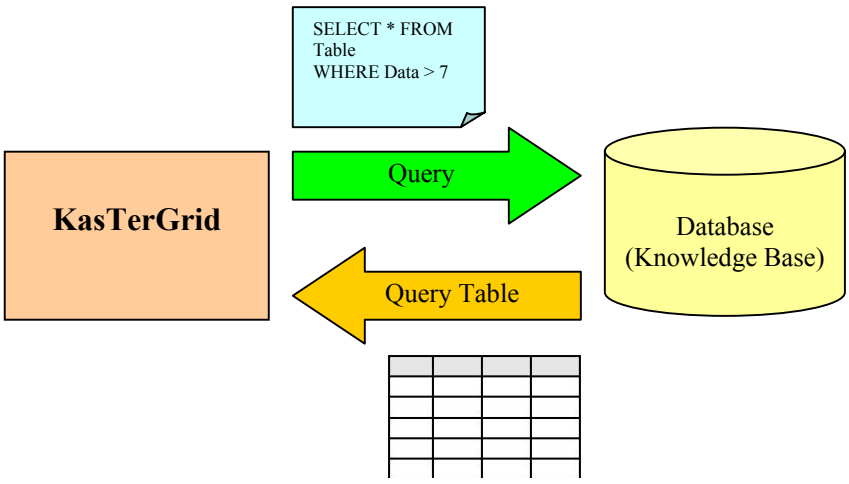


Figure 7.11 Accessing process of database

To communicate with database, the system uses Structured Query Language (SQL) command. SQL is is the most popular computer language used to create, modify and query databases. It is a standard interactive and programming language for getting information from and updating a database. Some database tools have been developed in the application. Those tools enable the system to execute SQL command.

Accessing process of database can be seen in figure 7.11. The system sends query to the database. The query is transferred to the database either via ODBC-connection or ADO-connection. The database processes the query. The result of query is a query table. This query table is sent back to the system. Furthermore the system extracts data from the query table and transforms them into knowledge.

Chapter 8

Development of Knowledge Acquisition Tool KasTerGrid

(Knowledge Acquisition System Using Ternary Grid)

8.1. Interface Considerations

One of important part of knowledge acquisition tool is interface for the expert. It is the human-computer interface. It provides interaction between human and computer (machine), so called Human-Computer Interactions (HCI). The effectiveness of knowledge acquisition tool is critically dependent on the quality of the human-computer interface, that part of a machine which the expert sees, hears and communicates with. No matter how complex the knowledge base, it is ultimately the user interface which determines how easily that knowledge can be accessed and employed.

Human-Computer Interaction is the name, adopted in the middle 1980's, used to describe the field of study in computing and computer engineering which is concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them [ACM92]. Designing a user interface from an Human-Computer Interactions viewpoint involves looking at the problem from many different perspectives and taking a range of factors into consideration before making decisions on how best to make the system usable.

From the human-computer interface perspective the most crucial component in a computer system is the user. In the past, system developers have used introspection and personal experience to shape the interface, with the result that features they assumed would be simple and obvious often proved obscure and infuriating to the less computer-oriented users. Psychology can produce a more objective picture of how human beings

in general process information. This knowledge can attune designers to some fundamental interface requirements and provides a sound starting point for interface design. Research on the perceptual/cognitive system has confirmed the counter-intuitive fact that we humans can absorb only a tiny fraction of the visual, audio and tactile information reaching us from our external environment [WAG88].

The usability of a system depends a great deal on the design of the user interface; the visual display and required actions that control user interaction with the system. Good interface design is essential for effective systems. The system must help users to achieve their goals and carry out work more effectively [SUT95]. Inadequate designed interface can result in user frustration, increased mistakes in system operation and failure to trust or even to use the system.

It is the aim of this chapter to present the application tool for Knowledge Acquisition System using Ternary Grid (KasTerGrid) with its important part concerning Human-Computer Interactions, i.e. expert interface of and to present designed concept of database for Ternary Grid knowledge base. KasTerGrid should facilitate cooperation of expert and machine through a designed user input handling with menu selection and keyboard typing.

8.2. Overview of Developed Application

KasTerGrid provides the expert as user the opportunity to interrupt processing and correct a mistake immediately he detects it. KasTerGrid provides an undo command to returns the dialogue to its previous state in Rule Editor and to change or modify knowledge in Fact Editor. In some case, the expert is possible to delete knowledge in knowledge base.

KasTerGrid provides also error detection in knowledge base and in input stream. An escape mechanism is also available. Additionally KasTerGrid enable the expert to change or modify the setting of database as knowledge base. The overview of developed application KasTerGrid is shown in figure 8.1.

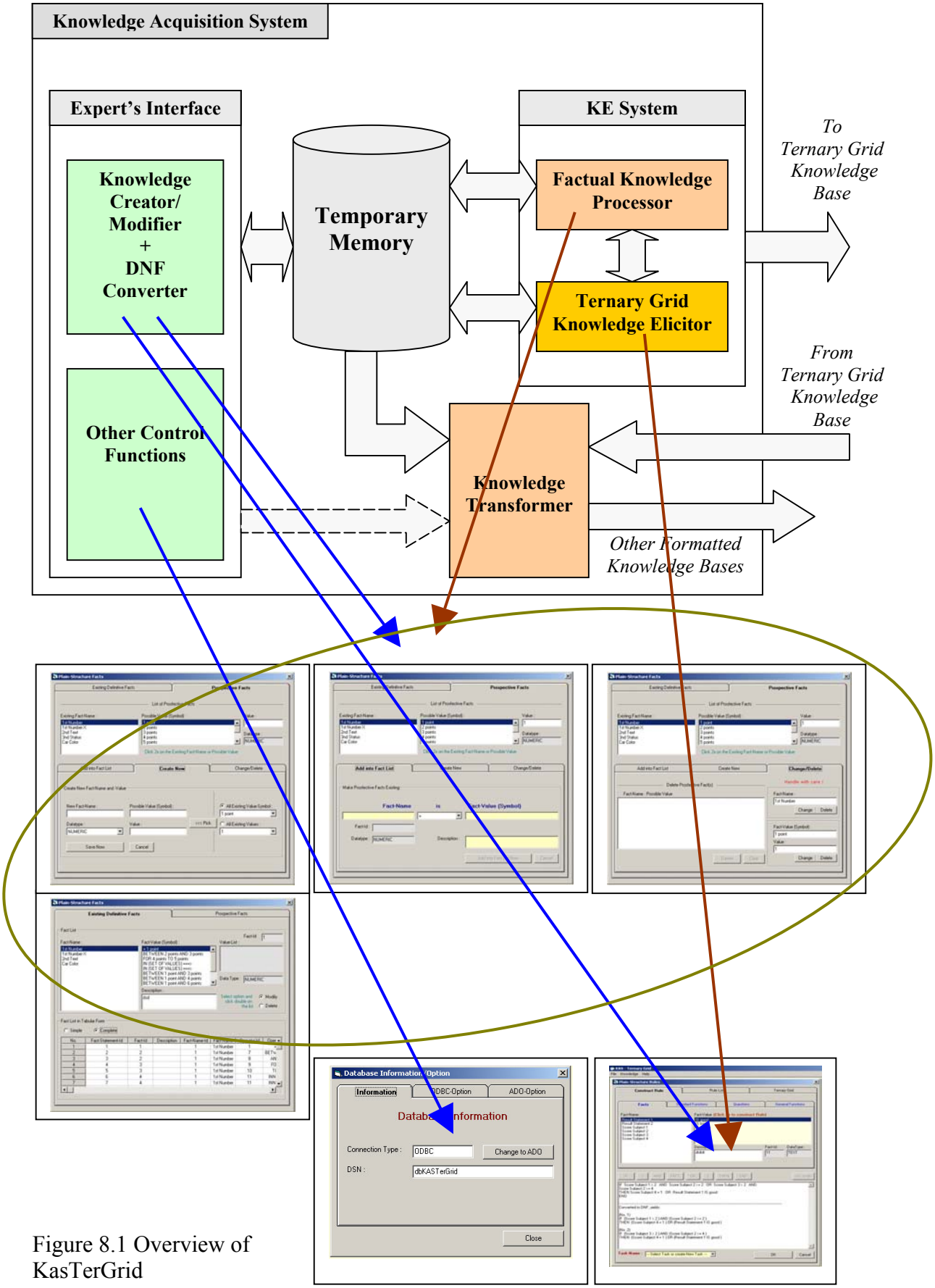


Figure 8.1 Overview of KasTerGrid

8.3. Knowledge Creator and Modifier

The major part of expert interface is Knowledge Creator and Modifier. This part consists of creator and modifier for factual and judgemental knowledge (facts and rules). They are called Fact Editor and Rule Editor. The Editor enables the expert to create or modify factual knowledge stepwise.

8.3.1. Fact Editor

The Fact Editor enables the expert to store factual knowledge. It provides editor for prospective facts and definitive facts. It is shown in figure 8.2 until figure 8.5. The Fact Editor facilitates creation of new knowledge, changing or modification and deleting existing knowledge. User input is handled with menu selection and keyboard typing. Menu selection is used to change the system menu, to create or delete definitive fact and to delete prospective fact. Keyboard typing is used create new fact and change or modify existing prospective fact. The Fact Editor displays also the content of knowledge in database in tabular form.

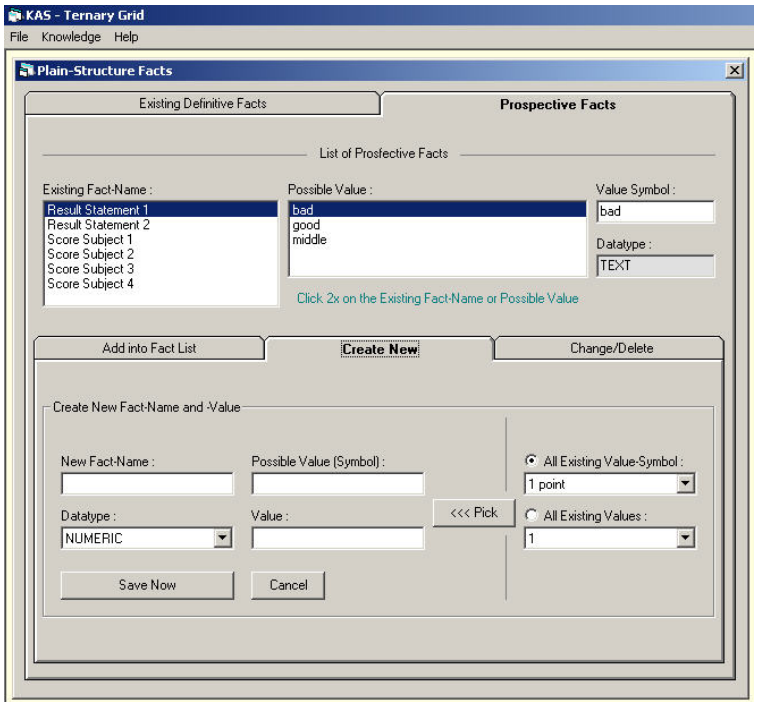


Figure 8.2 Fact Editor for creating new prospective fact

Figure 8.2 shows the Fact Editor for creating new prospective fact. To create a new prospective fact (see figure 8.2), the expert must type a fact-name. Possible fact-value in the form of symbol and value can be new created or taken over from existing fact-values in the right side. Data type can be selected from the data type list in the down left side.

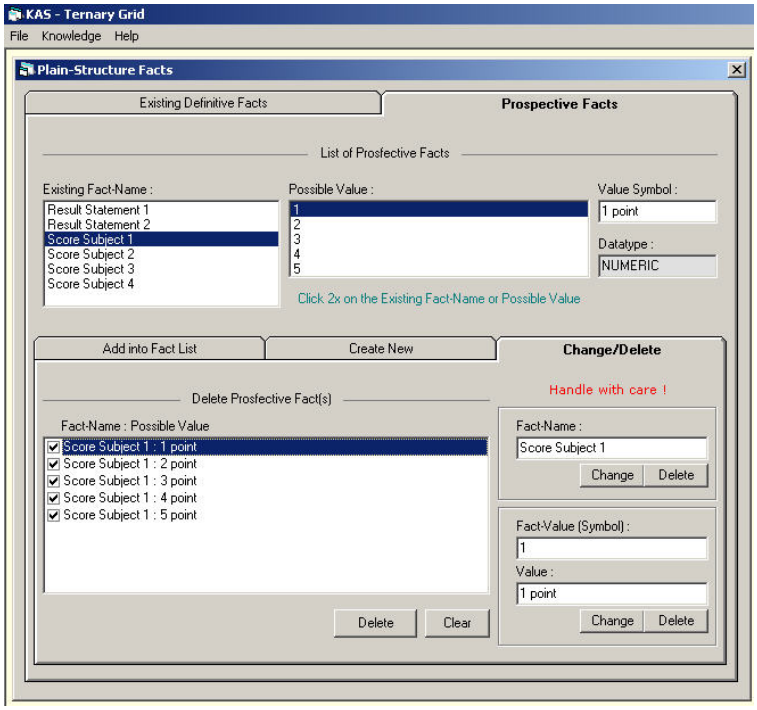


Figure 8.3 Fact Editor for changing or deleting existing prospective facts

Figure 8.3 above shows the Fact Editor for changing or deleting existing prospective facts. To delete prospective fact, the expert must select fact-name from the field named "Existing Fact-Name" or fact-value from the field named "Possible Value". If the expert selects a fact-name, that fact-name with the whole possible values will be selected. If the expert selects a possible value, only this value with corresponding fact-name will be selected.

Figure 8.4 shows the Fact Editor for adding prospective fact into list of definitive facts. This process can be viewed as creation a new definitive fact. To create a definitive fact, the expert must select a fact-name from the field "Existing Fact-Name" and a fact-value

from the field “Possible Value”. Furthermore the expert must select a fact-operator in the field named “is” and fill the description to the definitive fact.

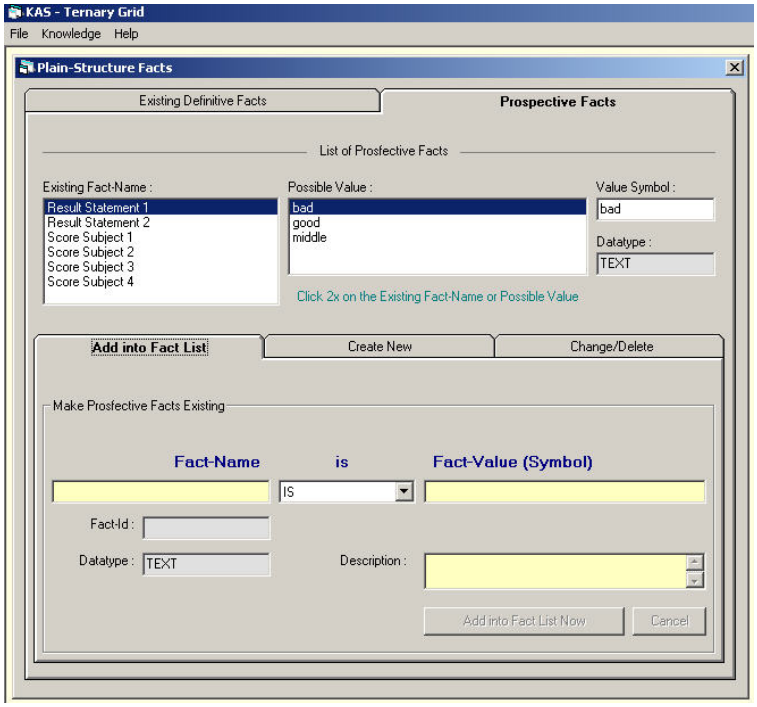


Figure 8.4 Fact Editor for creating new definitive fact

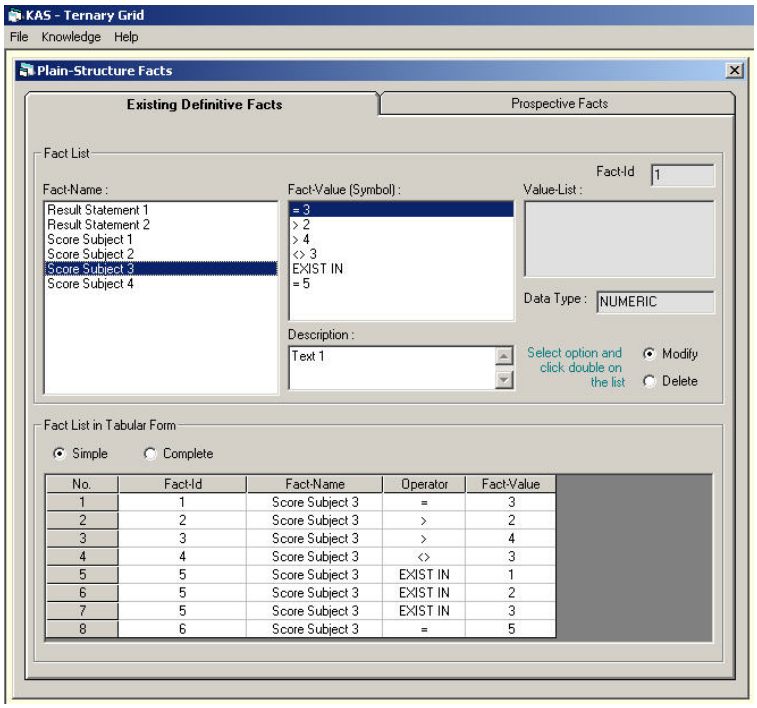


Figure 8.5 Fact Editor for displaying or deleting definitive facts

Figure 8.5 shows The Fact Editor for displaying or deleting definitive facts. This editor enables the expert to delete existing definitive fact. This editor displays also the content of knowledge in database in tabular form. It is shown in down part of the editor.

8.3.2. Rule Editor

The Rule Editor is an editor for building rules (knowledge) in classical or arbitrary format. It is show in figure 8.6. The process of collection of the input streams or instructions that are given by expert is based on syntax and state diagram. It is shown in figure 8.7 until figure 8.10. This Rule Editor is equipped with the DNF (Disjunctive Normal Form)-Converter. This converter converts that rules into another format of rule, i.e. Disjunctive Normal Form. It is a disjunction of conjunctions of literals. This form is also called *sum of products*. After rules are formatted into DNF-format, they will be sent into Ternary Grid table. This Rule Editor facilitates the creation of the new knowledge.

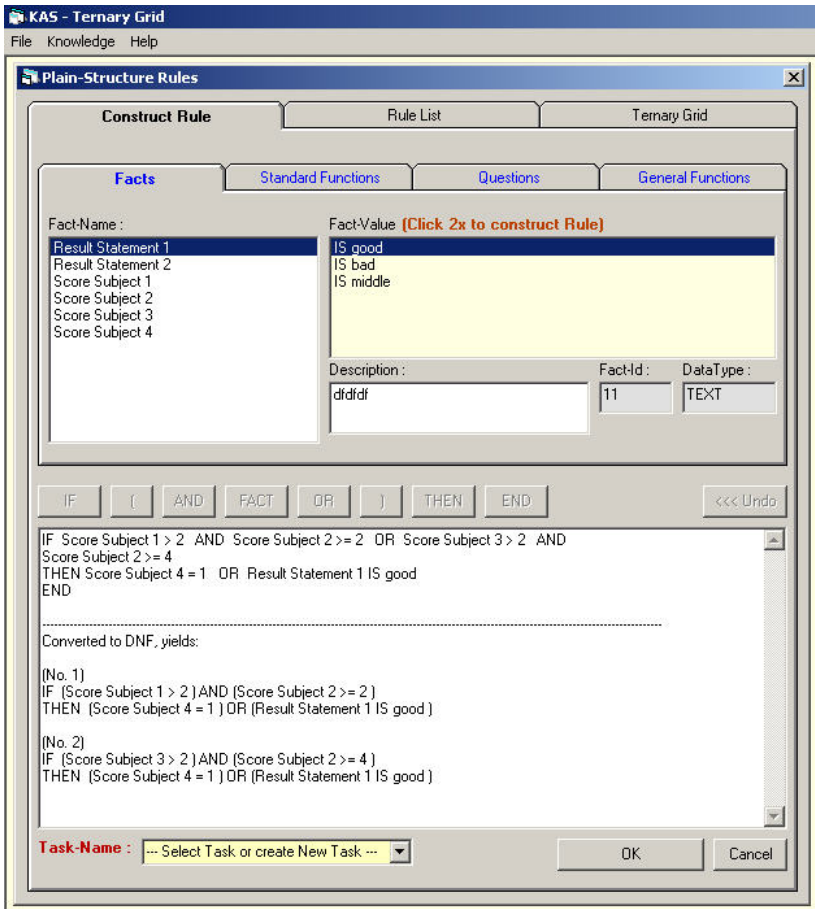


Figure 8.6 Rule Editor with rule example

The Rule Editor provides instruction buttons for building a production rule. To create a new rule, the expert must press first “IF” button to put “IF” instruction to the editor. The Rule Editor is equipped with syntax validation system. It is explained in section 8.3.3. This system will guide the expert to build a right rule or to avoid wrong rule syntax. An “Undo” button enables the expert to returns the dialogue to its previous state, so that the expert can always correct the instructions any time.

The Rule Editor takes over some tasks of knowledge elicitor. It transforms rule into DNF format, eliminates redundancy in the form of repeating fact, and investigates or detects inconsistent rule.

8.3.3. Syntax Validation

The role of Rule Editor due to the quality of input stream or instructions is very important. The Rule Editor is responsible for creating judgmental knowledge. Rule editor collects instructions from human expert and then transforms them into legal knowledge format. Wrong or illegal instruction causes wrong or illegal knowledge. To ensure right or legal instruction a syntax validation system has been implemented into the Rule Editor. The syntax validation system follows the syntax diagram, which is shown in figure 8.7.

The Syntax diagram is divided into two parts. First part is syntax diagram for condition part (IF-part). Second part is syntax diagram for conclusion part (THEN-part).

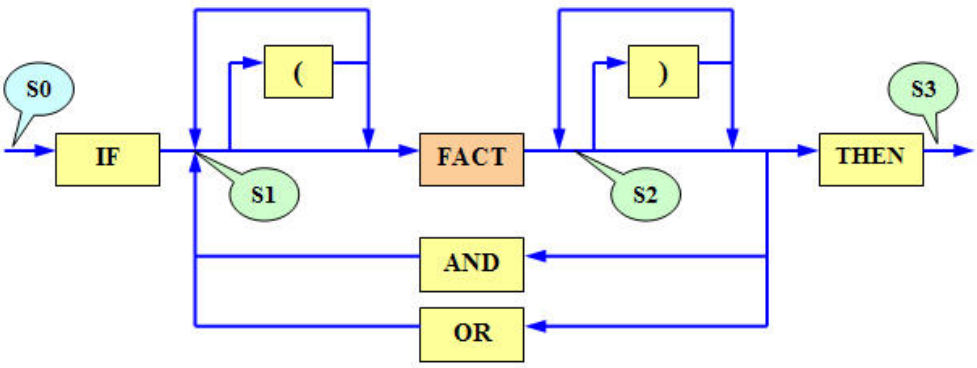


Figure 8.7 Syntax diagram for the If-part of rule

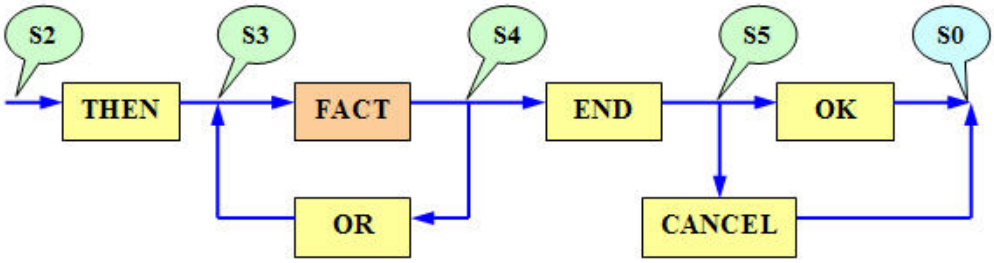


Figure 8.8 Syntax diagram for the Then-part of rule

I have also transformed those syntax diagrams into state diagrams below. These state diagrams provide a detail picture of how a specific action changes states. This state diagram is needed during the development process of the Rule Editor.

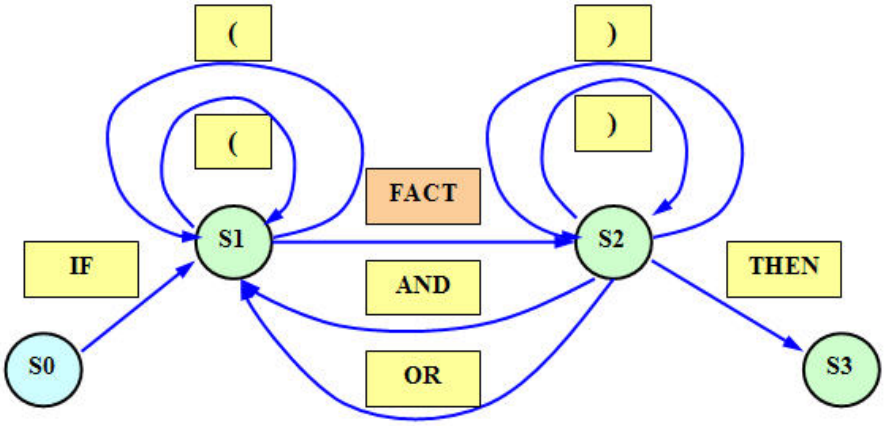


Figure 8.9 State diagrams for the If-part

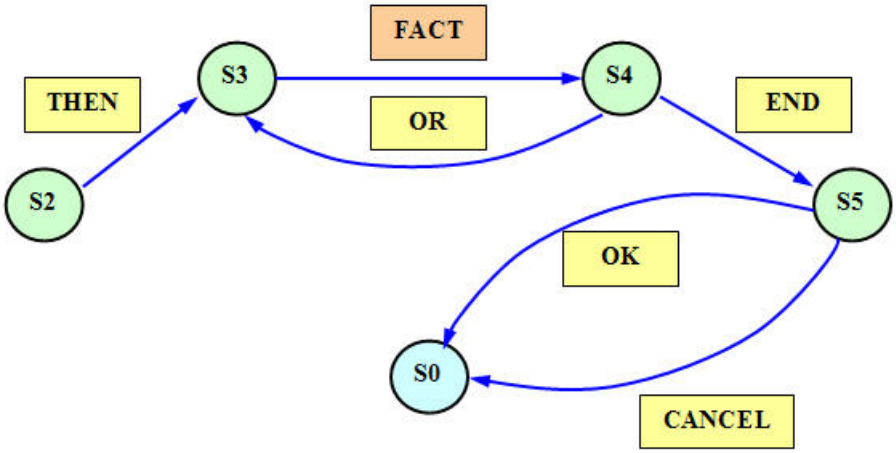


Figure 8.10 State diagrams for the Then-part

8.3.4. DNF-Converter

The task of DNF (Disjunctive Normal Form) - converter is to convert rule format from classical or arbitrary format into DNF format. This DNF format is required for further processing knowledge in Ternary Grid.

Before DNF-converter doing its job the rule will be split into two parts, i.e. IF-part and THEN-part. This job is done by so called Knowledge Splitter. It is a program utility that is integrated in and activated by the knowledge creator/modifier.

The following example shows the splitting process of rule.

Given rule (example 1):

IF *Fact1* AND *Fact2* AND (*Fact1* AND *Fact3* OR *Fact2* AND *Fact4* AND *Fact5*)
THEN *Fact6* OR *Fact7*

Knowledge splitter splits that rule into two parts in array format as follows:

IF-Part:

IF	*Fact1*	AND	*Fact2*	AND

(	*Fact1*	AND	*Fact3*	OR	*Fact2*	AND	*Fact4*	AND	*Fact5*	)

THEN-Part:

THEN	*Fact6*	OR	*Fact7*

The DNF-Converter converts rules into another format of rule, i.e. Disjunctive Normal Form. It is a disjunction of conjunctions of literals. This form is also called *sum of products*. After rules are formatted into DNF-format, they will be sent into Ternary Grid

table. This Rule Editor facilitates also the creation of the new knowledge and modification of the existing knowledge.

The following example shows the conversion process by DNF-Converter. I take an example rule from previous example (example 1). The DNF-Converter converts that rule into DNF-format in array and yields:

IF-part:

Fact1	Fact2	Fact1	Fact3	
Fact1	Fact2	Fact2	Fact4	Fact5

That means:

DNF-Rule 1: *Fact1* AND *Fact2* AND *Fact1* AND *Fact3*
DNF-Rule 2: *Fact1* AND *Fact2* AND *Fact2* AND *Fact4* AND *Fact5*

Horizontal sequence in the table represents AND-combination and vertical sequence represents OR-combination.

8.4. Factual Knowledge Processor

The task of the Factual Knowledge Processor is to define, collect, process and transform the factual knowledge into Ternary Grid format. The Factual Knowledge Processor is not a user interface but some program functions that are activated by the knowledge creator/modifier.

The Factual Knowledge Processor can avoid fact duplication, both prospective fact and definitive fact. It can also detect wrong fact-value due to corresponding data type and generates interconnection between one definitive fact and other definitive facts that have the same fact-name but different fact values. This interconnection describes the relation of fact-values and is required by the Knowledge Elicitor during analysing and optimising the knowledge in Ternary Grid.

8.5. Knowledge Elicitor

The task of Ternary Grid Knowledge Elicitor is to define, collect, validate and optimise the judgemental knowledge or rule-based knowledge. This judgemental knowledge consists of factual knowledge that has been created before by the Factual Knowledge Processor. The Ternary Grid Knowledge Elicitor is a user interface and some program functions that are integrated in and activated by the knowledge creator/modifier.

The Ternary Grid Knowledge Elicitor collects knowledge from Rule Editor, displays and optimises it in a grid format. The Ternary Grid Knowledge Elicitor is responsible to ensure the performance of knowledge due to the quality of information and reduction of error possibility. The operating principle of Ternary Grid Knowledge Elicitor follows the elicitation concept that has been mentioned in the previous chapter. Elimination of knowledge redundancy and investigation of knowledge error are the most intensive task on the Ternary Grid Knowledge Elicitor. Figure 8.11 shows the presentation of knowledge in Ternary Grid format that is processed by the Ternary Grid Knowledge Elicitor.

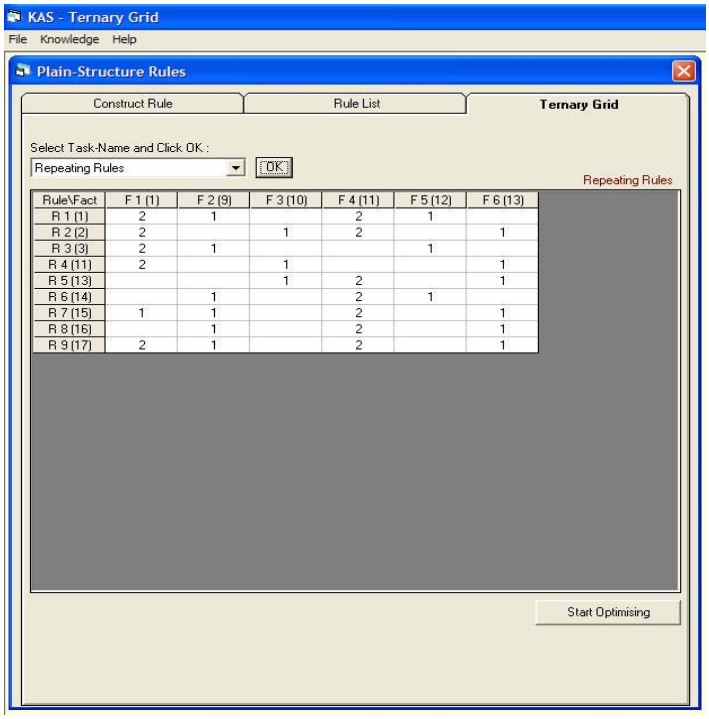


Figure 8.11 Presentation of knowledge in Ternary Grid format

8.6. Knowledge Transformer

Knowledge transformer is not user interface. It is some program functions that have task to provide the transformation of Ternary Grid format into another format of knowledge. The transformation procedure of every format must be programmed and integrated into the knowledge transformer. By the time of development the knowledge transformer is built-in in the knowledge creator/modifier and able to transform knowledge from knowledge format of CongaXpert [HUW99] [EHW03] (plain structure) to Ternary Grid format and other way round. Furthermore additional transformation procedure for other formats will be integrated into knowledge transformer.

8.7. Other Control Functions

In this application, some control functions have been developed in order to support the system function. These are user interface and functions for handling with database connection. This control function enables the expert to get information about database connection and to select the database connection.

Following figure show the user interface for database information and option. Figure 8.12 shows the current connection type between system and database. It is connection via ODBC (Open Data Base Connectivity). Figure 8.13 shows the option for ODBC-Connection. With this option, the expert can select existing system Data Source Name (DSN) or create new system DSN that is required for ODBC-connection.

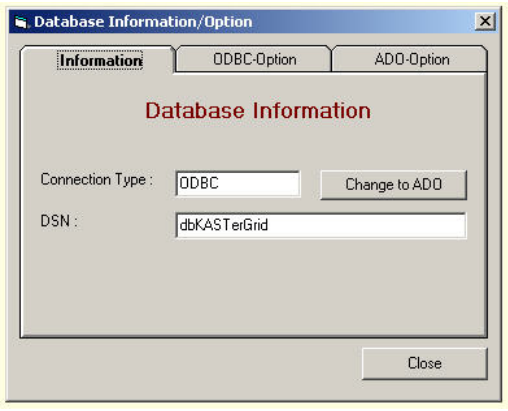


Figure 8.12 User interface for database information and option

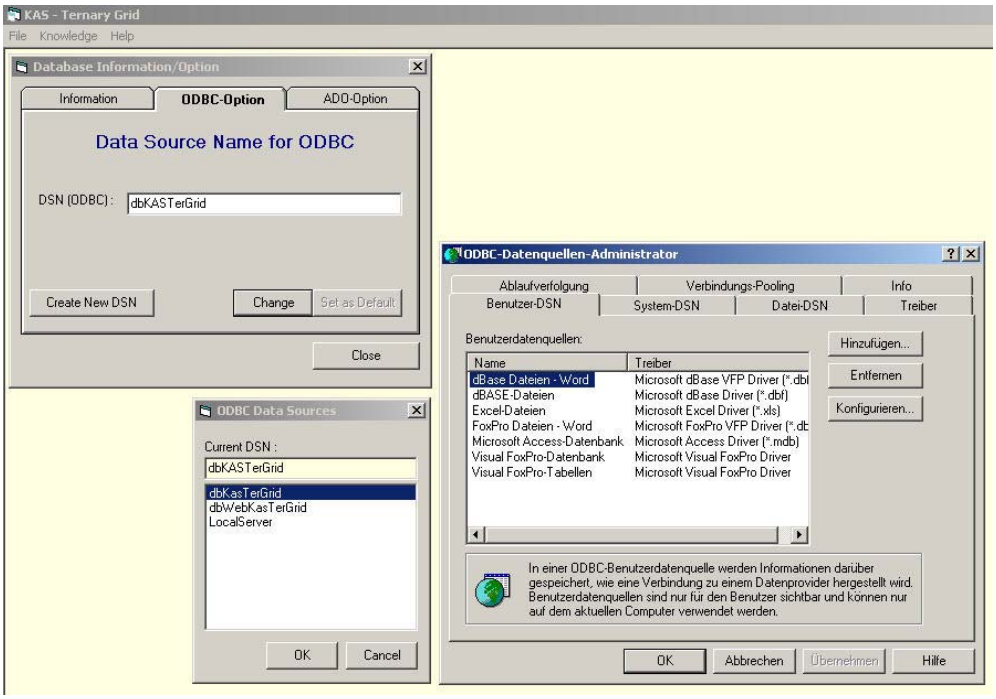


Figure 8.13 Option for ODBC-Connection

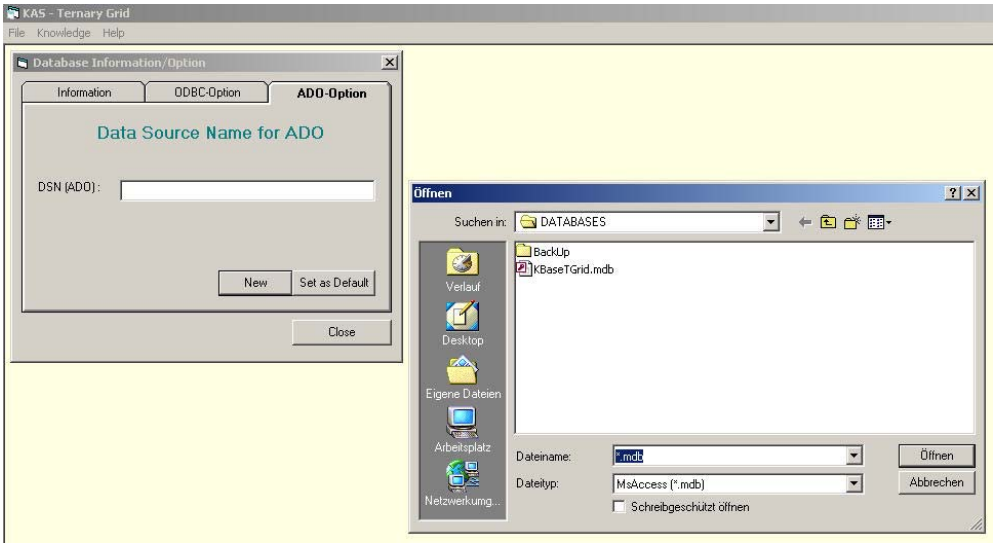


Figure 8.14 Option for ADO-Connection

Figure 8.14 shows the option for ADO-Connection. ADO is abbreviation of Active X Data Object. With this option, the expert can select existing database directly (without having to create system DSN).

All information of database connection, system status and configuration are stored in so called *registry* memory. It is a database used by Windows (operating system) for storing configuration information.

8.8. Knowledge Acquisition Process

The concept of knowledge acquisition system that is implemented in KasTerGrid should help the expert or knowledge engineer to maintain the perspective and focus of attentions, which are needed to complete a thorough and consistent knowledge base or expert system application. The approach of acquisition process involves following phases:

1) designing knowledge (top-down) and
2) implementing the designed knowledge into knowledge base (bottom-up)
3) optimising knowledge

The following diagram and description explains the mentioned approach.

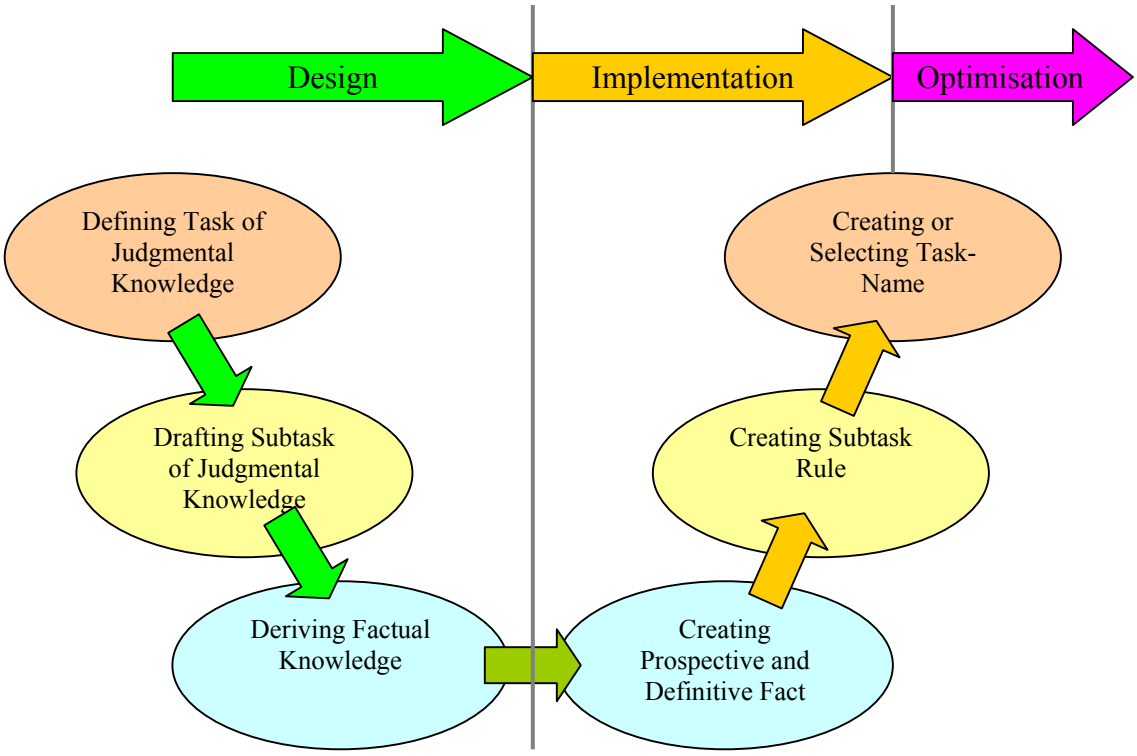


Figure 8.15 Knowledge Acquisition process

1) Design phase

a) Defining Task of judgmental knowledge

The first important thing in building knowledge base is to define task. Task in this developed system can be interpreted as a domain, in which a specific piece of work is undertaken by the expert in order to achieve any specific goal. A task contains all important judgmental knowledge and represents a term, what the expert should do. All judgmental knowledge that is created must refer to any task.

Example:

The expert should help to build an English Placement Test (EPT) system for evaluating English skill of a student. The test contains of three sections, i.e. Reading Skills, Composing Skills and Listening Skills. Every section needs 25 minutes and contains 35 multiple choice questions. The system classifies the result into three categories, i.e. Advance, Intermediate and Basic.

From this example, the name of task could be "English Placement Test" or can be called "EPT".

b) Drafting Subtask of Judgmental Knowledge

Subtask is a part of task. It contains all things what a part of task can be done. A subtask represents a term of a judgmental knowledge or a rule. Continuing the given example above, the following rules are given:

Advance category is determined if the following scores can be achieved:

- Score of every section amounts at least 30 points
- Score of one section amounts at least 25 points and other two sections amounts at least 30 points for each section and total score amounts at least 90 points.

Intermediate category is determined if the following scores can be achieved:

- Condition for advance category is not fulfilled (total score < 90 points) and score of every section amounts at least 20 points

- Condition for advance category is not fulfilled (total score < 90 points) and score of one section amounts at least 15 points and other two sections amounts at least 20 points for each section and total score amounts at least 60 points.

Basic category is determined if both conditions above are not fulfilled. It means (total score < 60 points).

c) Deriving Factual Knowledge

Composition of factual knowledge (fact) produces a subtask rule. Deriving subtask into factual knowledge involves two steps: firstly, deriving subtask into definitive facts and finally, deriving definitive facts into prospective facts. Continuing the given example above, the following facts shown in table 8.1 can be derived.

Fact-Name	Fact-Operator	Fact-Value
Score of Reading Skills	>=	30
	>=	25
	>=	20
	>=	15
Score of Composing Skills	>=	30
	>=	25
	>=	20
	>=	15
Score of Listening Skills	>=	30
	>=	25
	>=	20
	>=	15
Total Score	>=	90
	<	90
	>=	60
	<	60
Category	IS	Advance
	IS	Not Advance
	IS	Intermediate
	IS	Not Intermediate
	IS	Basic

Table 8.1 Derived definitive facts

Prospective facts can be derived from definitive facts above. Prospective facts are divided into fact-names and fact-values. It is shown in table 8.2 and 8.3.

Fact-Name
Score of Reading Skills
Score of Composing Skills
Score of Listening Skills
Total Score
Category

Table 8.2 Derived fact-names

Fact-Value	Data Type
15	Number
20	Number
25	Number
30	Number
60	Number
90	Number
Advance	Text
Not Advance	Text
Intermediate	Text
Not Intermediate	Text
Basic	Text

Table 8.3 Derived fact-values

2) Implementation phase

a) Creating Prospective and Definitive Fact

The first step to implement concept is creating fact. Creating fact involves two sub steps, i.e. creating prospective fact and creating definitive fact. Every fact-name, corresponding fact-value, fact-value-symbol and data type must be typed and entered during creating prospective facts. Creating definitive facts, the expert has only to select menus (button, list, etc) of existing prospective facts, except description field. Figure 8.16 and 8.17 illustrates these mentioned processes.

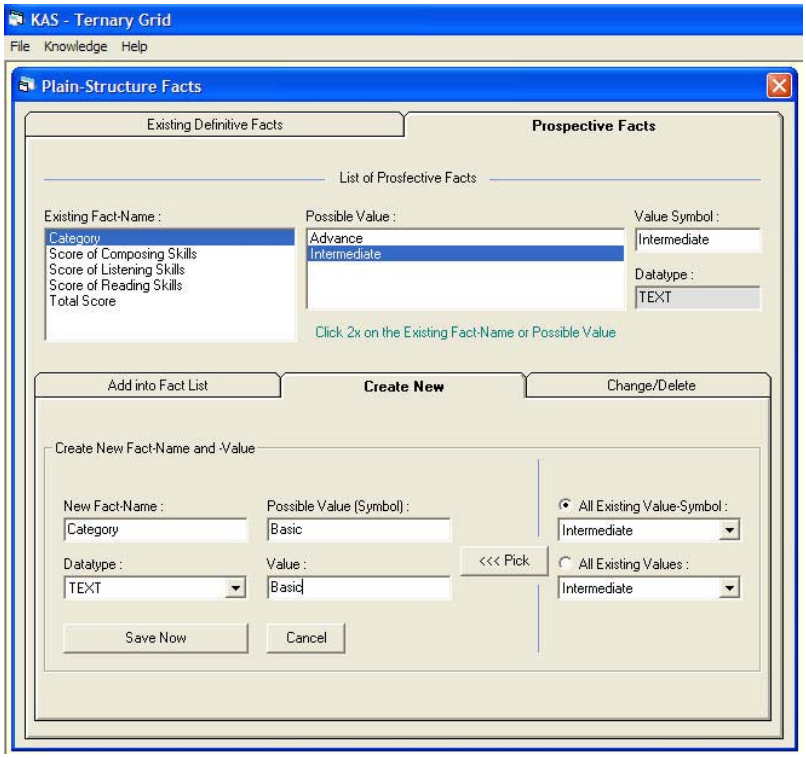


Figure 8.16 Creating prospective facts

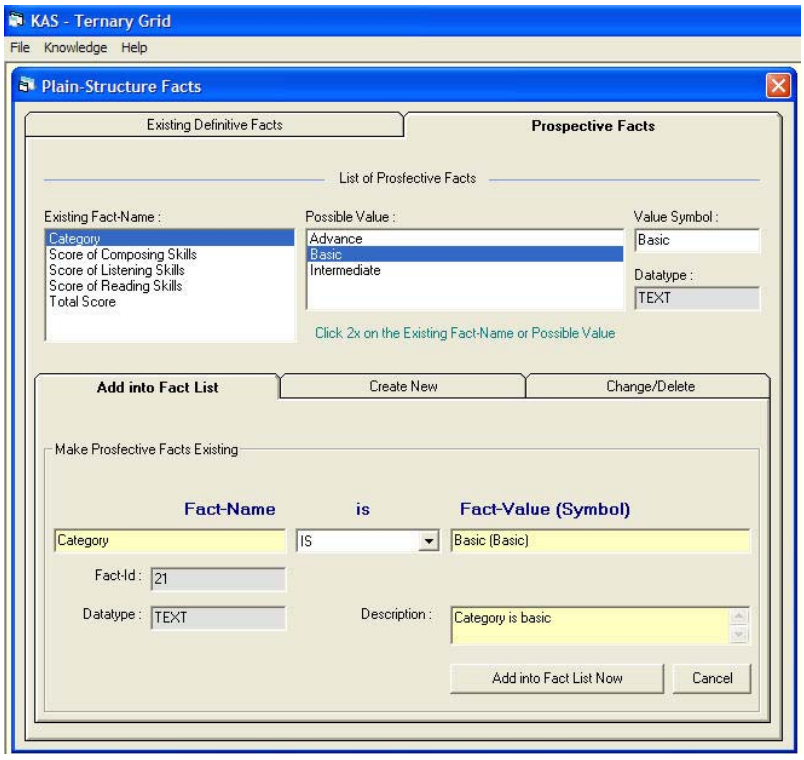


Figure 8.17 Creating definitive facts

b) Creating Subtask Rule

All definitive facts are now ready to be used. The next step what the expert has to do is to create subtask rules with Rule Editor. The expert has only to select menus of existing commands (rule instructions) and definitive facts. It is shown in figure 8.18.

c) Creating or Selecting Task-Name

Task-name can be new created or selected from existing tasks. If a new task has to be created, the expert must type and enter the task name in field "Task Name" in the Rule Editor. It is located in left down area. This step is done immediately after creating subtask. Figure 8.18 show the field "Task Name" inside the oval circle.

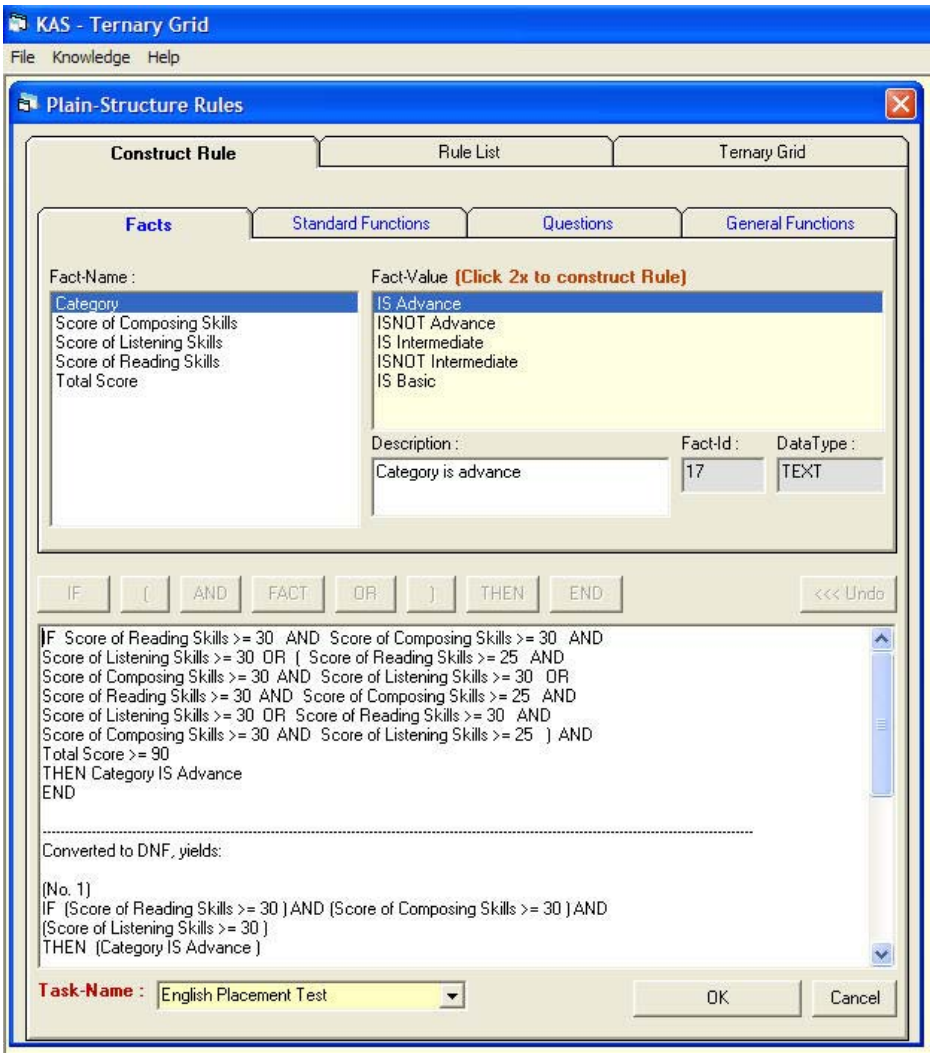
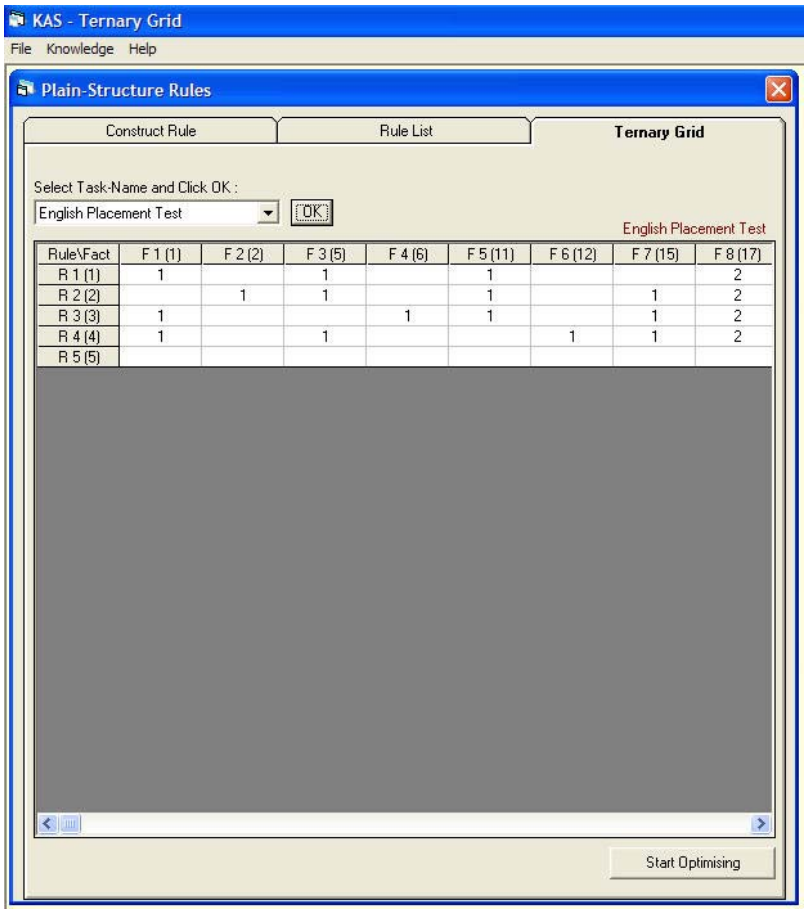


Figure 8.18 Creating subtask rule and task name

3) Optimisation phase

The final phase of acquisition process is optimisation of knowledge using Ternary Grid. Some tasks of optimisation are done in the Rule Editor. These tasks are elimination of repeating fact and investigation of inconsistent rule. Other remaining tasks of optimisation are done in the Ternary Grid Editor. Figure 8.19 shows the presentation of rules that have been optimised in Ternary Grid.



Rule\Fact	F 1 (1)	F 2 (2)	F 3 (5)	F 4 (6)	F 5 (11)	F 6 (12)	F 7 (15)	F 8 (17)
R 1 (1)	1		1		1			2
R 2 (2)		1	1		1		1	2
R 3 (3)	1			1	1		1	2
R 4 (4)	1		1			1	1	2
R 5 (5)								

Figure 8.19 Presentation of some rules in Ternary Grid

Chapter 9

Results of Experiments

Experiment can be interpreted as trial or test of a scientific hypothesis or generalisation by manipulation of environmental factors to observe whether what results agrees, or disagrees, with what the hypothesis predicts. The test is carried out under controlled conditions that are made to demonstrate a known truth, examine the validity of a hypothesis, or determine the efficacy of something previously untried.

The purpose of the experiments, which are carried out in this research work, is to examine the consistency of the concept that is discussed in chapter 4 until chapter 7 with the implementation that is discussed in chapter 8. The result of experiments will confirm the ability of the knowledge acquisition system whether the system is able to deliver reliable performance of knowledge or not.

For these experiments I have selected some example rules that can control the conditions, which are used to examine the validity of my hypothesis. Some experiments have been carried out in order execute the following tests:

- The Functionality of DNF-Converter for formatting and preparing knowledge.
- The Elimination of Redundancy due to repeating facts, repeating rules and rule with unnecessary condition.
- Investigation of error possibility due to inconsistent rules and rotating chain.

9.1. The Functionality of DNF-Converter

The experiment deals with the functionality testing of DNF-converter. The result of this experiment confirms whether the DNF-Converter can convert rules from classical or arbitrary format to DNF-format (sum of product) or not. For this testing, the following subtask rule is given:

IF *Score Subject 1 > 2* **AND** *Score Subject 2 >= 2* **AND** (*Score Subject 1 > 2* **AND** *Score Subject 3 > 2* **OR** *Score Subject 2 >= 2* **AND** *Score Subject 3 > 4* **AND** *Score Subject 4 >= 2*)

THEN *Result Statement 1 IS good* **OR** *Result Statement 2 IS good*

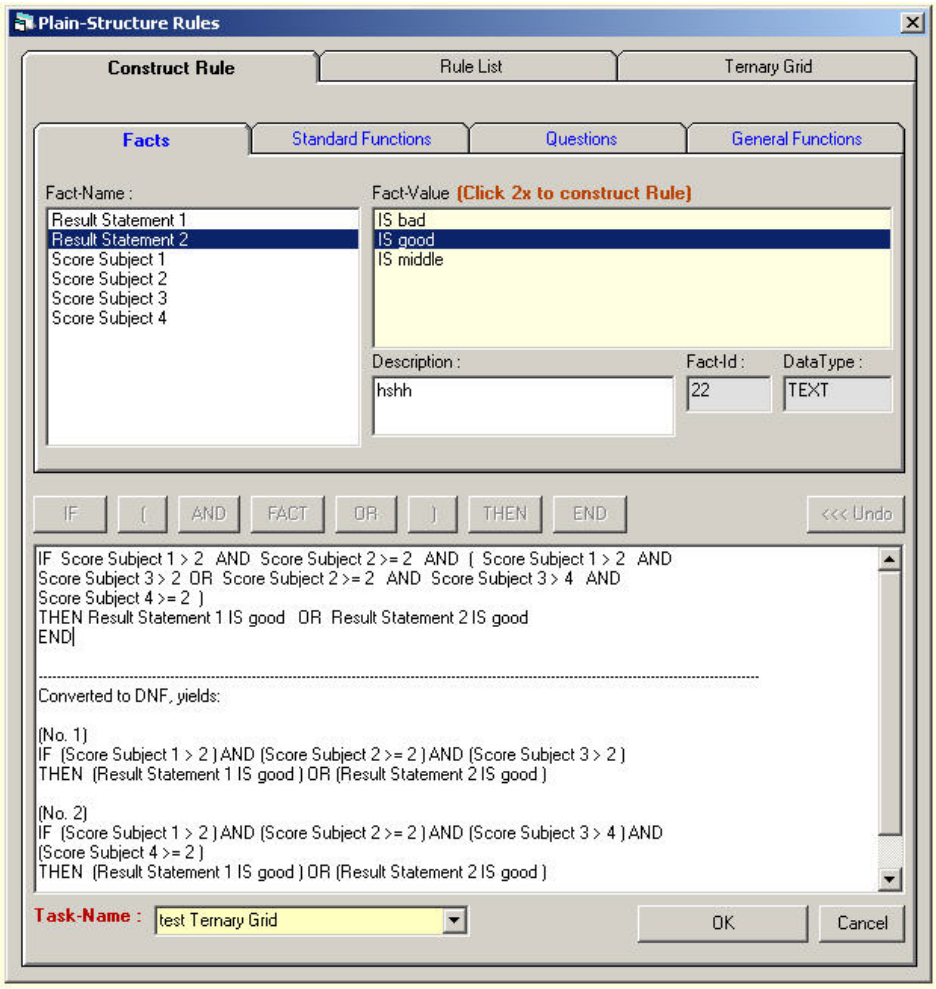


Figure 9.1 DNF-rules conversion and elimination of repeating facts

According to the function of The DNF-Converter, the DNF-Converter must deliver two DNF-rules as follow:

DNF-Rule 1:

IF *Score Subject 1 > 2* **AND** *Score Subject 2 >= 2* **AND** *Score Subject 1 > 2* **AND** *Score Subject 3 > 2*

THEN *Result Statement 1 IS good* **OR** *Result Statement 2 IS good*

DNF-Rule 2:

IF *Score Subject 1 > 2* **AND** *Score Subject 2 >= 2* **AND OR** *Score Subject 2 >= 2* **AND** *Score Subject 3 > 4* **AND** *Score Subject 4 >= 2*)

THEN *Result Statement 1 IS good* **OR** *Result Statement 2 IS good*

Figure 9.1 shows the result of experiment due to DNF conversion (after eliminating repeating facts). The DNF-converter has converted the given rule and delivered two DNF-rules. Those rules agree with DNF-Rule 1 and DNF-Rule 2. *Remember that the facts, which are written in underline term in DNF-Rule 1 and DNF-Rule 2, are repeating facts. Those facts have been removed by the system.*

9.2. Elimination of Redundancy

9.2.1. Repeating Facts

A repeating fact is the same fact that appears within a condition part of a rule. Refer to the given rule for testing DNF-Converter, repeating facts occur in Rule-DNF 1 and Rule-DNF 2 (written in underline). In Rule-DNF 1 fact "*Score Subject 1 > 2"* appears twice. In Rule-DNF 2 fact "*Score Subject 2 >= 2"* appears twice as well.

Repeating facts are redundant and therefore will be removed by the system. Figure 9.1 shows the result of given rule after converting to DNF-format and eliminating repeating facts. Every fact in DNF-format appears only once.

9.2.2. Repeating Rules

The experiment deals with the investigation and elimination of repeating rules. The result of this experiment confirms whether the system can remove repeating rules or not. For this testing, the following task rules are given in the Ternary Grid. It is shown in table 9.1 as follow:

	F1	F2	F3	F4	F5	F6
R1	2	1		2	1	
R2	2		1	2		1
R3	2	1			1	
R4	2		1			1
R5			1	2		1
R6		1		2	1	
R7	1	1		2		1
R8		1		2		1
R9	2	1		2		1

Table 9.1 Given task rules that contain repeating rules

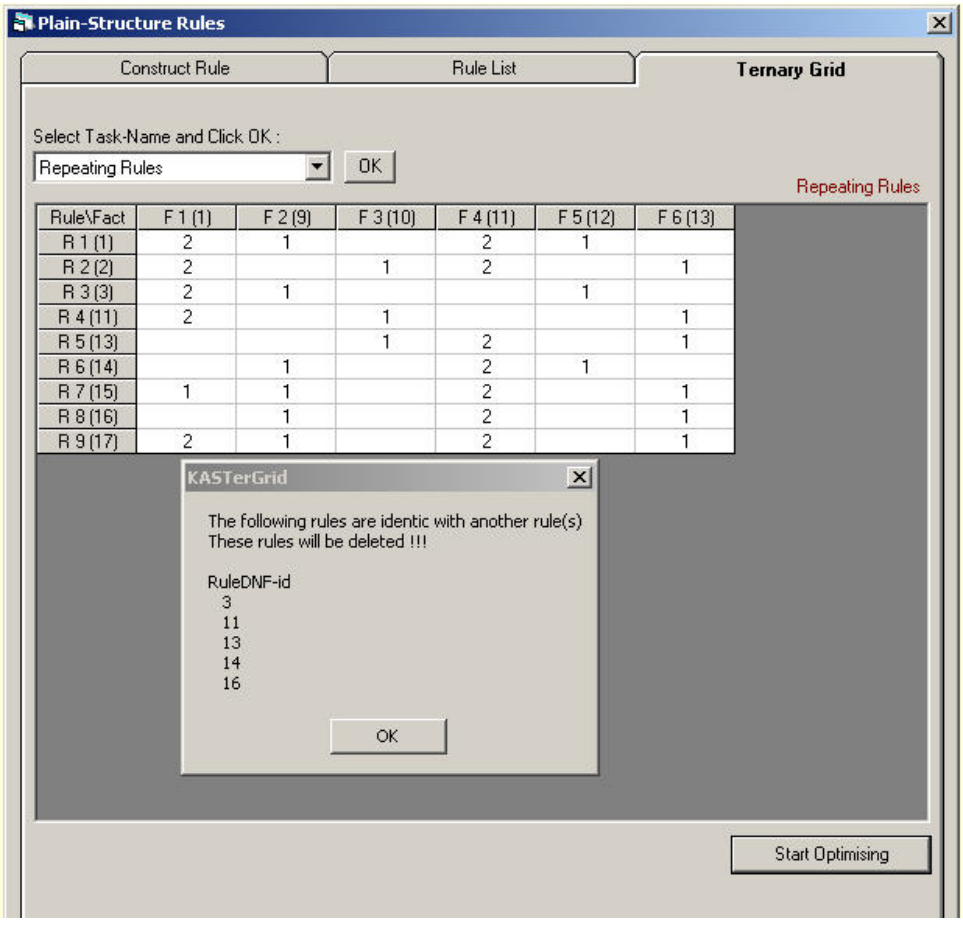


Figure 9.2 Elimination of repeating rules

From table 9.1 we can see that there are some repeating rules (rows). *Furthermore we call row instead of rule*. Those rows are identical with other rows. The system must be able to deliver following results:

Row R3	Is identical with	Row R1
Row R4	Is identical with	Row R2
Row R5	Is identical with	Row R2
Row R6	Is identical with	Row R1
Row R8	Is identical with	Row R7

Table 9.2 Expected result of repeating rule investigation

Figure 9.2 shows the result of experiment due to repeating rules. The system can remove rules that are identical with other rules. The numbers that are shows in the figure (3, 11, 13, 14, 16) are the number of rule identity number (rule-id). These numbers represent the row R3, R4, R5, R6 and R8. These rows agree with the rows in table 9.2.

9.2.3. Rule with unnecessary Condition

The experiment deals with the investigation and elimination of rule with unnecessary condition. The result of this experiment confirms whether the system can remove those rules or not. For this testing, the following task rules are given in the Ternary Grid. It is shown in table 9.3 as follow:

	F1	F2	F3	F4	F5	F6	F7	F8
R1				1	2	1		
R2				1	2		1	
R3	1			1	2	1		
R4		1		1	2	1		2
R5			1	1	2	1		2
R6			1		2		1	2

Table 9.3 given task rules that contain rule with unnecessary condition

From table 9.3, if row R3 is compared to row R1, we can see that row R3 represents a rule that has unnecessary condition.

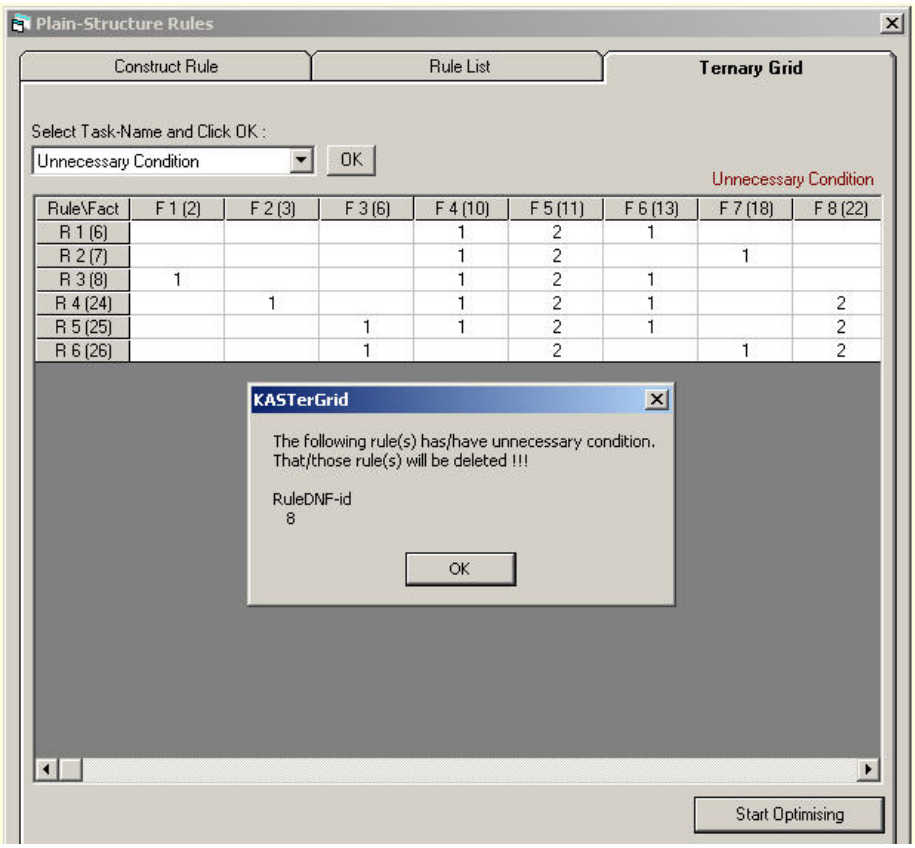


Figure 9.3 Elimination of a rule that has unnecessary condition

Figure 9.3 shows the result of experiment due to rule that has unnecessary condition. Rule 8, which is represented by row R3, is such rule. The system removes that rule. This result agrees with expected result.

9.3. Investigation of Knowledge Error

Knowledge error is a crucial problem and intolerable because this knowledge will produce wrong decision and therefore can not be used for expert system.

9.3.1. Inconsistent Rules

The experiment deals with the investigation and inconsistent rules. The result of this experiment confirms whether the system can investigate such rules or not. For this testing, the following subtask rule is given:

IF *Score Subject 1 > 2* **AND** *Score Subject 2 >= 2* **AND** (*Score Subject 1 = 3* **AND** *Score Subject 3 > 2* **OR** *Score Subject 4 >= 2*)

THEN *Result Statement 1 IS good* **OR** *Score Subject 4 = 3*

Expected result:

DNF-Rule 1:

IF *Score Subject 1 > 2* **AND** *Score Subject 2 >= 2* **AND** *Score Subject 1 = 3* **AND** *Score Subject 3 > 2*

THEN *Result Statement 1 IS good* **OR** *Score Subject 4 = 3*

DNF-Rule 2:

IF *Score Subject 1 > 2* **AND** *Score Subject 2 >= 2* **AND** *Score Subject 4 >= 2*

THEN *Result Statement 1 IS good* **OR** *Score Subject 4 = 3*

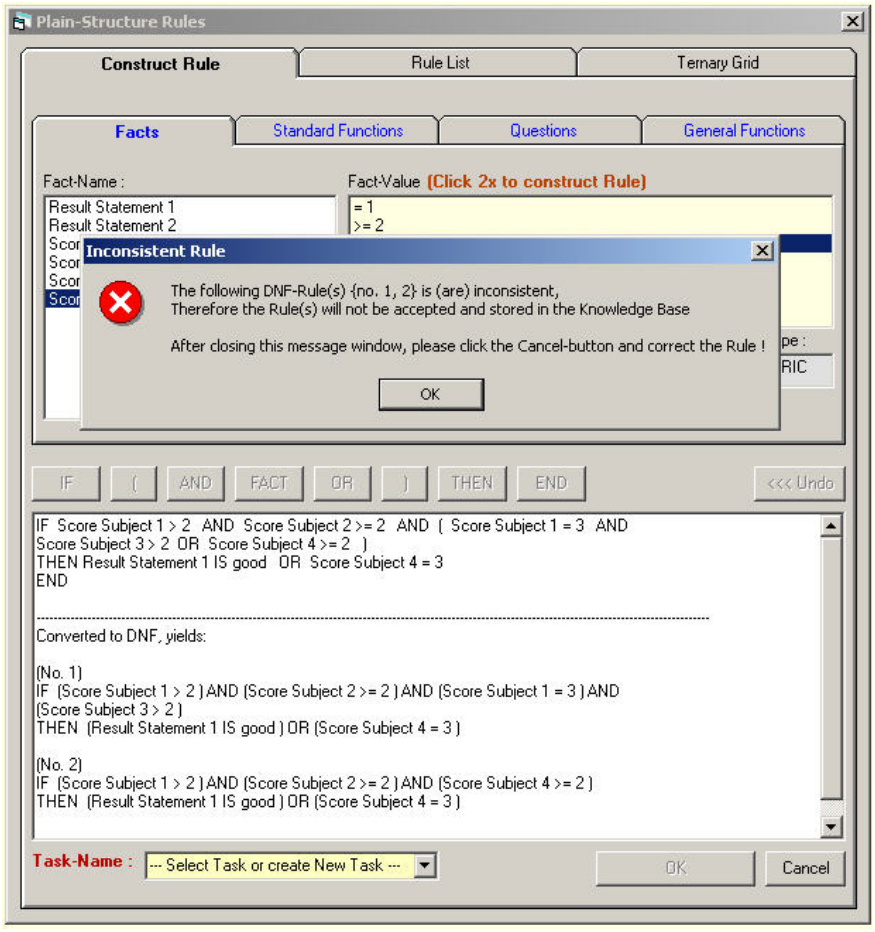


Figure 9.4 Inconsistent Rule

In DNF-Rule 1 there are two different definitive facts that have the same fact-name appear in the condition part or rule. They are written in underline term. In DNF-Rule 2 1 two different definitive facts that have the same fact-name appear in the condition and conclusion part or rule. They are written in underline term. Both DNF-rules are inconsistent and not accepted by the system. The expert must create new subtask rule again.

Figure 9.4 shows the result of experiment due to inconsistent rule. The result shows that the created rule is inconsistent. This occurs to both generated DNF-rules. Those rules will not be accepted and stored in knowledge base. This statement agrees with the statement in previous paragraph above (statement of expected result).

9.3.2. Rotating Chain

The experiment deals with the investigation of rotating chain. The result of this experiment confirms whether the system can investigate rules that have rotating chain or not. Two scenarios are carried out for this experiment. First one is for investigating rotating chain and second one is for investigating correct chain. For first scenario, the following rules are given:

	F1	F2	F3	F4	F5	F6	F7
R1	1				2	1	
R2		1			2		1
R3			2			1	1
R4			2		1		
R5				1	2		
R6			1	2			

Table 9.4 given rules that contains rotating chain

In table 9.4 we can see that we do not find any column, in which only value “2” appears. This situation is a precondition that rotating chain occurs. I expected that rotating chain can be detected in those rules. A rule task, in which rotating chain is detected, will not be accepted and stored in knowledge base by the system.

Figure 9.5 shows the result of experiment due to investigation of rotating chain. The result shows that there is not found any end-rule. This means, there is rotating chain in those rule and therefore the rule task is not working. This statement agrees with the statement in previous paragraph above (statement of expected result). Those rules will not be accepted and stored in knowledge base. The expert must start creating rule again from beginning.

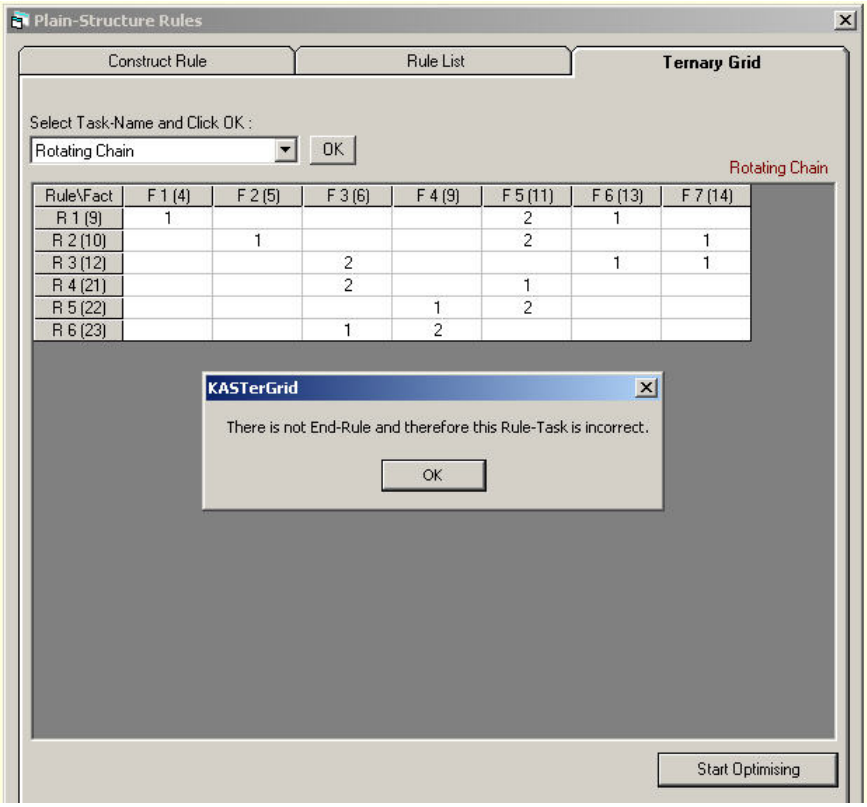


Figure 9.5 Result of rotating chain investigation

For second scenario, the following rules are given:

	F1	F2	F3	F4
R1		1	2	
R2			2	1
R3		1	2	
R4	2		2	1
R5	1	2		

Table 9.5 given rules with correct chain

In table 9.5 we can see that in third column (F3) only value “2” appears. This is a precondition of correct chain. I expected that the system can prove this situation.

Figure 9.6 shows the result of experiment due to investigation of correct chain. The result shows that some end-rules are found. This means, the chain or rules is correct and therefore the rule task is working. This statement agrees with the statement in previous paragraph above (statement of expected result). Those rules ca be accepted and stored in knowledge base.

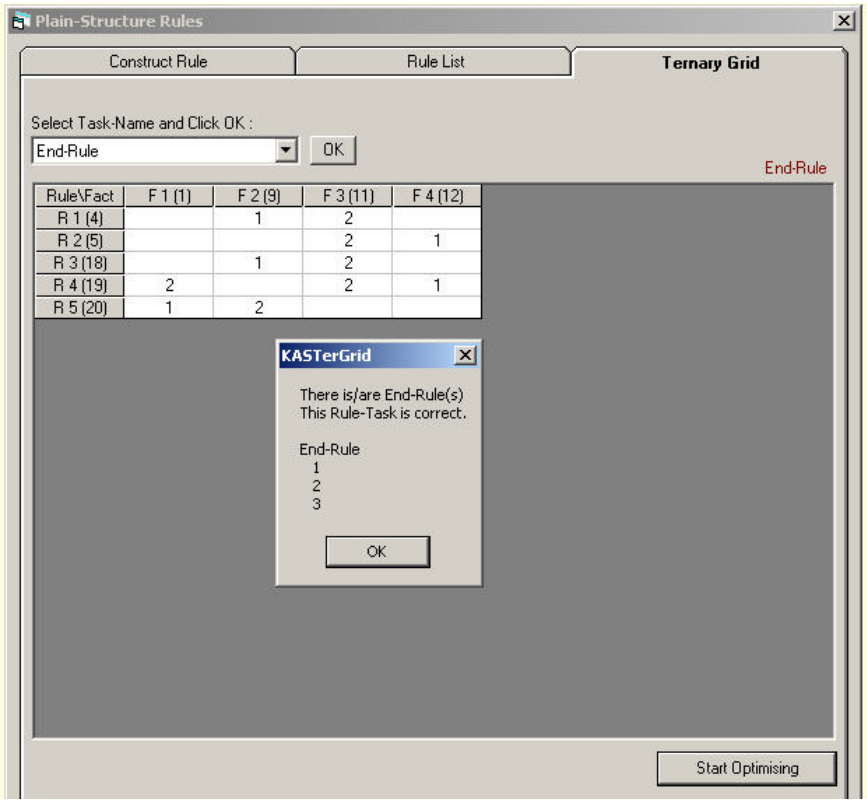


Figure 9.6 Result of correct chain investigation

Chapter 10

Contributions

10.1. Contributions of the Research

This section lists those ideas developed in this research which are believed to contribute to a general understanding of how to apply practically knowledge engineering, which deals with knowledge acquisition system from human experts. The primary idea of using Ternary Grid for knowledge acquisition system is novel, while other ideas also results in the performance improvement of existing ideas or else new applications of old ideas.

The thesis makes three primary ideas that can be considered as contributions to field of knowledge engineering. First, a novel way for acquiring knowledge from human expert is presented. The acquisition approach is decomposed into set of tasks that provide solution to problems encountered during acquisition process. Dividing factual knowledge into prospective and definitive facts avoids factual knowledge errors. Method for avoiding factual knowledge duplication optimises the content of knowledge. Elicitation method, which is followed by knowledge optimisation and validation, is the core idea for this approach. It improves the performance of knowledge concerning quality of information and reduces error possibility. The role of developed algorithms is keys to the success of concept implementation. Structured acquisition process guides the expert in creating knowledge step by step.

A second primary contribution of the work, a software application for Knowledge Acquisition System using Ternary Grid (KasTerGrid) has been developed. The user interface for obtaining factual knowledge is designed to give view of knowledge structure to the expert. Syntax validation system, which is applied in rule editor, controls the accuracy of input streams for creating judgmental knowledge. DNF-converter provides required knowledge format for Ternary Grid. Integration of

knowledge-base and database provides persistent knowledge. User interface for database connection provides flexible option for choosing type of connection.

A third primary contribution of the work, Ternary Grid presents new approach to knowledge representation. It represents a structure in which knowledge can be stored in a way that allows the system to understand the relationships among pieces of knowledge and to manipulate those relationships.

10.2. Summary of Comparison with related Work

A summary of the comparisons among the novel and the existing research is given in Table below:

	Repertory Grid	**Formal Concept Analysis**	***Ternary Grid***
Using Matrix/Grid	Yes	Yes	Yes
Matrix element	Value	Symbol	Value
Matrix value	Multi Scaling	empty cell, X (0,1)	0,1,2
Matrix construction	Attribute v. Value	Problem v. Solution	Tasks v. Resources
Matrix representation	construct, element	problem, solution	rule, fact, syntax
Knowledge composition	-	independent	independent
Rule construction	Indirect, via Repertory Grid	Indirect, via Concept Matrix	Direct, via Rule Editor
Elicitation method	direct	direct	direct
Elicitation type	Concept	Concept	Construct
Mathematical methods	Euclidean, Manhattan and Hamming distance	Ripple Down Rules and Set Operation	Logical Operation, Boolean Algebra and Set Operation
Elimination of Redundancy	No	No	Yes
Error Investigation	No	No	Yes
Acquisition process	not structured	not structured	structured

Table 10.1 Comparison with related work

As it can be seen the comparison table (table 10.1), the novel approach performs better in term of quality of knowledge, which means that the system removes redundancies and therefore reduces quantity of knowledge without having to loss of important information. Ultimately it will reduce computational time of problem-solving part of knowledge based system and the other resources required for knowledge storage.

The novel approach is robust against errors such as inconsistent rule and rotating chain of rule. As it is mentioned before, when such inconsistent rule error is encountered, the problem-solving part of knowledge-based system will not be able to produce correct result. Another case, when such rotating chain error is encountered, the problem-solving part will enter in endless loop, which would consequently result in wastage of computational resources.

The novel approach provides a structured knowledge acquisition process. This process guides the expert to avoid mistake in creating knowledge, which would result in performance improvement.

While considering rule construction methodology, in this research construction is applied directly to the rule editor, unlike the existing approaches where indirect approach is used. Now the direct approach offers the benefit that it is fast as compared with the indirect approaches and this is one of the key factors in the performance of any knowledge acquisition process.

Chapter 11

Conclusions and Future Work

11.1. Conclusions

The Ternary Grid technique and representation are convenient for processing the knowledge. They may be directly viewed as task domain and production rule structure or as intermediate stage of optimisation. This technique can optimize not only logical terms within a rule but also logical relations between rules. The grid has elements as *problem-solving domains* which can be derived into sub domains or group of rules, rows as *rules*, columns as *facts* and values as *IF-THEN syntax*.

The Ternary Grid elicitation works in a model domain using concept matrix and logical operation. The organisation and logical content of expert knowledge in Ternary Grid can be easily inspected and analysed. Completion and recognition of patterns which consists of “0 or empty”, ”1”, “2”values in the grid are facilitated by the structure and relative compactness of the matrix representation.

Ternary Grid is position independent, which means that any composition of some factual knowledge as input stream in judgmental knowledge will always be ordered in a specific way by the grid column and it can be considered as mathematical or statistical combinations.

Representation in the Ternary Grid facilitates optimising and testing for conditions of ambiguity, redundancy, completeness [CRS87] and correctness. It will contribute in achieving the performance of the knowledge due to the quality of information and the reduction of error possibility. Implication of that achieved performance is reduction of large body of knowledge and reduction of accessing time of inference engine.

Dividing factual knowledge into prospective and definitive facts supports structured knowledge acquisition process and avoids mistakes in creating definitive facts concerning the consistency of fact-value and data type to the fact-name. The expert is always guided to create a fact-name with its possible fact-values. When the expert creates a definitive fact, he has to select only existing fact-name with corresponding fact-values. The expert will never be offered any fact-name with irrelevant fact-values by the system.

Integration of knowledge-base and database ensures the persistency and integrity of knowledge in knowledge base. Conceptual design process of database can be considered as a transformation from knowledge to data and data to knowledge.

Ternary Grid knowledge acquisition system will encourage the expert to make clear the distinctions he uses in applying his expertise. This system will help him also to structure his knowledge and identify and formalise his concept.

11.2. Future Work

Knowledge-grouping is done manually by the expert. Developed hierarchical structure of knowledge-base facilitates the process of knowledge-grouping. The effectiveness of knowledge-grouping depends on the capability of the expert to do this job. This might lead to knowledge-grouping mistakes. In future the knowledge-grouping should be done automatically to avoid possible mistakes. So a method for knowledge-grouping should be developed to address this issue.

First phase of knowledge acquisition process, which consists of conceptual and designing parts, and the related calculations, for problem solving are so far done manually by the expert. The efficiency of this process is expert dependent, which might lead lower efficiency as well. However, to avoid such scenarios an additional guiding algorithm should be developed.

Problems with rotating chain arise with the number of created knowledge in knowledge-base proportionally. The possibility that an error in the form of rotating chain occurs therefore increases. New scenarios with more complicated structures can be encountered while considering higher number of knowledge-base content. Investigation for new cases, which lead to errors, should be continued followed by the algorithmic development.

Concerning the problem-solving part of knowledge base system, an inference method based on Ternary Grid, which is combined with Rete Algorithm, could be considered for future research work to achieve better performance of in term of computation time and resources.

Implementation of Ternary Grid to knowledge acquisition system for class-structured knowledge base, which has been developing by Institute for Multimedia and Software Engineering (IMSE) University of Duisburg-Essen since 2003, could be carried out as future work as well.

The proposed Ternary Grid based approach given in this research work for the development of knowledge acquisition system could give a new future direction for solving more complex Boolean equations. Based on the initial experimental results, it is expected that given approach could be able to substitute the existing Quine Mc Clusky method used for solving more complex Boolean equations.

References

[AKR04] Applied Knowledge Research Institute, What is Knowledge, 2004 http://www.akri.org/museum/what.htm

[BAF81] Barr, A. and Feigenbaum, E.A., The Handbook of Artificial Intelligence, Vol 1, Morgan Kaufmann, Los Altos CA, 1981

[BAI79] Bainbridge, L., Verbal report as evidence of the process operator's knowledge, *International Journal of Man-Machine Studies*, 11(4), p. 411-436, 1979

[BCF98] Barr, A., Cohen, Feigenbaum, E.A., The Handbook of Artificial Intelligence (Volumes I-IV), Addison Wesley Publisher, 1998

[BCZ92] Byrd, T., Cossick, K., Zmud, R., A synthesis of research on requirements analysis and knowledge acquisition techniques, *MIS Quarterly*, 16(1), p. 117-138., 1992

[BOO87] Boose, J. H., Rapid Acquisition and Combination of Knowledge from Multiple Experts in the Same Domain, Future Computing Systems 1(2) p. 191-216, 1987

[BOO89] Boose, J. H., A survey of knowledge acquisition techniques and tools, *Readings in Knowledge Acquisition and Learning*, California, Morgan Kaufmann, p. 39-58, 1989

[BUC83] Buchanan, B. G., Barstow, D., Bechtel, R., Bennet, J., Clancey, W., Kulikowski, C., Mitchell, T. M., Watermann, D. A., Constructing an Expert System, in Hayes-Roth, F., Waterman, D. A., Lenat D., (1983), *Building Expert Systems*, Teknowledge Series in Knowledge Engineering, Addison Wesley, Reading, Massachusetts, USA, 1983

[BYR92] Byrd, T., Implementation and use of expert systems in organizations: Perceptions of knowledge engineers, *Journal of Management Information Systems*, 8(4), 97, 1992

[CAW03] Cawsey, Alison, Introductory AI Course, Department of Computing and Electrical Engineering, Heriot-Watt University Edinburgh, UK, 2003

[CHK01] Chklovski, Thesis Proposal: Effective Knowledge Acquisition, 2001. Page 4, 2001

[CON01] Considine, J., Lecture Notes for CS210, 2001 http://cs-people.bu.edu/jconsidi/teaching/notes/cs210/node8.html

[COR89] Cordingley, E. S., Knowledge elicitation techniques for knowledge-based systems, In D. Diaper (Ed.), *Knowledge elicitation: Principles, techniques and applications*, Ellis Horwood Ltd., Chichester, England, p. 89-173, 1989

[CPC91] CPCS, KSS0 Manual, Centre for Person-Computer Studies, Underhill Drive, Calgary, Canada, 1991

[CRS87] Cragun, B.J., Steudel, H.J., A decision-table-based processor for checking completeness and consistency in rule-based expert systems, *International Journal of Man-Machine Studies,* 26, p. 633-648, 1987.

[DAR00] Darpa, Rapid Knowledge Formation (RFK) initiative, 2000
http://reliant.teknowledge.com/RKF/publication/index.html

[EHW03] Erdani, Y., Hunger, A., Werner, Stefan., Mertens, S. Web-Based Consultation System with Expert System, *IASTED CST 2003 – International conference (International Association of Science and Technology for Development – Computer Science and Technology)*, May 19-21, 2003, Cancun, Mexico, 2003
ISBN – 0-88986-349-0, page 61-64

[EHW04] Erdani, Y., Hunger, A., Werner, S., Mertens, S., Ternary Grid as a Potentially New Technique for Knowledge Elicitation/Acquisition, 2nd IEEE Conference on Intelligent System, vol I: pp. 312-315. ISBN 0-7803-8278-1, Varna - Bulgaria, June 22-24, 2004.
This paper has been accepted also by SEKE 04 (Software Engineering and Knowledge Engineering) conference in Banff, Canada.

[EHW05a] Erdani, Y., Hunger, A., Werner, S., Improving the Knowledge Performance using Ternary Grid Knowledge Acquisition and Model, *Proc. 4th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases (AIKED 2005)*, Salzburg - Austria, 13-15 February 2005. ISBN 960-8457-09-2

[EHW05b] Erdani, Y., Hunger, A., Werner, S., Improving the Knowledge Performance using Ternary Grid Knowledge Acquisition and Model, *WSEAS Transactions (Journals) on Information Science and Application, Issue 2, Volume 2*, February 2005. ISSN 1790-0832

[EHW05c] Erdani, Y., Hunger, A., Werner, S., Improving the Knowledge Performance using Ternary Grid Knowledge Acquisition and Model, *Workshop WSB 2005 - Wissensbasierte Studienberatung im Internet zur Förderung der internationalen Wettbewerbsfähigkeit des Studienstandortes Deutschland*, 14-15 Februar 2005, Duisburg – Germany.

[EHW05d] Erdani, Y., Hunger, A., Werner, S., Mathematical Approach for Solving Knowledge Optimisation Problem in Ternary Grid Knowledge Acquisition, **to appear in** *Proceedings 19th The Society for Modelling and Simulation*

International - European Simulation Multi-Conference (SCS-ESM) 2005, June 1-4, 2005, Riga – Latvia.

[ERD93] Edwards, G., Compton, P., Malor, R., Srinivasan, A., Lazarus, L., (1993) PEIRS: a Pathologist Maintained Expert System for the Interpretation of Chemical Pathology Reports *Pathology 25*: 27-34, 1993

[EPI04] Epistemics, 2004. http://www.epistemics.co.uk/

[FEE93] Feigenbaum, E. A, Engelmore, R. S., Expert System and Artificial Intelligence, Japanese Technology Evaluation Center panel's report about Knowledge-Based Systems, Japan, 1993

[FEI84] Feigenbaum, E.A., Knowledge Engineering: The Applied of Artificial Intelligence, *Annals of the New York Academy of Science*. Page 91-107, 1984

[FEM83] Feigenbaum, E. A., McCorduck, P., The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World, Reading, MA: Addison Wesley, 1983

[FEM84] Feigenbaum, E. A., McCorduck, P., The Fifth Generation, Pan Books, London, 1984

[FEL87] Fellers, J., Key factors in knowledge acquisition, *ACM SIGCPR Computer Personnel*, v.11 n.1, p.10-24, May 1987

[FID91] Firlej, M., Hellens, D., Knowledge Elicitation, A Practical Handbook, Prentice Hall, ISBN 0-13-517145-8, UK, 1991

[FRE14] Freud, S., Psychopathology of everyday life, London, 1914

[GAB88] Gaines, B.R. & Boose, J.H., Knowledge Acquisition for Knowledge-Based Systems, London, Academic Press, 1988

[GAI87] Gaines, B.R., An overview of knowledge acquisition and transfer. International Journal of Man-Machine Studies 26 (4), p. 453-472, 1987

[GAW96] Ganther, B, Wille, R., Formale Begriffsanalyse - Mathematische Grundlagen, Springer Verlag, 1996

[GAW99] Ganther, B, Wille, R., Applied Lattice Theory: Formal Concept Analysis, Dresden – Germany, 1999

[GAS89] Gaines, B.R., Shaw, M.L.G., Comparing the conceptual systems of experts, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence,* pp.633-638, Morgan Kaufmann, San Mateo, California, 1989

[GAS93] Gaines, B.R., Shaw, M.L.G., Eliciting Knowledge and Transferring it effectively to a Knowledge Base System, IEEE Transactions on Knowledge and Data Engineering 5(1), p. 4-14, 1993

[GAS95] Gaines, Brian R., Shaw, Mildred L. G., Knowledge Acquisition Tools based on
Personal Construct Psychology, Knowledge Science Institute University of Calgary, Alberta, Canada, 1995
http://ksi.cpsc.ucalgary.ca/articles/KBS/KER/

[GAS96] Gaines, B.R., Shaw, M.L.G., A networked, open architecture knowledge management system. Gaines, B.R. and Musen, M.A., Ed. Proceedings of Tenth Knowledge Acquisition Workshop, 1996

[GRC87] Gruber, T.R., Cohen, P. R., Design for Acquisition: Principle of Knowledge-System Design to Facilitate Knowledge Acquisition, *International Journal of Man-Machine Studies*, 26: p.143-159, 1987

[HAB90] Harmon, P., Sawyer B., Creating Expert System for Business and Industry, John Wiley & Sons Inc, ISBN 0-471-61495-5, Canada, 1990

[HAR86] Hart, Anna., Knowledge Acquisition for Expert Systems, Kogan Page Ltd, London, 1986, ISBN 1-85091-091-X

[HAS02] Hartmann, K., Schlechtweg, S., Helbing, R., Strothotte, Th., Knowledge Supported Graphical Illustration of Texts, *Advanced Visual Interfaces (AVI 2002),* May 22–24, 2002, Trento, Italy, 2002

[HAY83] Hayes-Roth, F., Waterman, D. A., Lenat, D. B., Building Expert Systems, reading, Mass.: Addison-Wesley, 1983

[HOF87] Hoffman, R., The problem of extracting knowledge of experts from the prospective of experimental psychology, *AI Magazine, 8*, 53-64, 1987

[HUD97] Hudlicka, E., Summary of Knowledge Elicitation Techniques for Requirements Analysis, Course Material for Human Computer Interaction, Worcester Polytechnic Institute, 1997

[HUW99] Hunger, A., Werner, S., CONGA: A Course Online/ Offline Information and Guidance System to support an International Degree Course, *Proceeding of ICCE 99.* Chiba-Japan, 1999, ISBN 1 58603 027 2, Page 577-583

[HWA94] Hwang, G., Knowledge elicitation and integration from multiple experts, *Journal of Information Science and Engineering*, 10, 99-109, 1994

[IGN91] Ignizio J.P., The Development and Implementation of Rule-Based Expert Systems, *Book: Introduction to Expert Systems*, University of Houston, Copyright McGraw Hill Publications, Pages pp 111-133 , 1991

[INM97] Inman, D., Some questions and answers on Knowledge Acquisition, South Bank University, 1997
http://www.scism.sbu.ac.uk/inmandw/tutorials/

[ITS04] Information Technology Service, Window Service: Introduction into Data Modelling, the University of Texas at Austin, 2004.
http://www.utexas.edu/its/windows/database/datamodeling/index.html

[LIC93] Lichte, K., Knowledge capture model for expert systems, Proceedings First New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems, p. 163-164, 1993

[KAN96] Kang, B., Validating Knowledge Acquisition: Multiple Classification Ripple Down Rules, PhD Thesis, School of Computer Science and Engineering, University of NSW, Australia, 1996

[KCP95] Kang, B., Compton, P., Preston, P., Multiple Classification Ripple Down Rules - Evaluation and Possibilities, *Proceedings 9th Banff Knowledge Acquisition for Knowledge Based Systems Workshop***,** vol. 1: 17.1-17.20., Banff –Canada, 1995

[KEL55] Kelly, G., The Psychology of Personal Constructs. New York, 1955

[KER88] Kerry, R. E., Integrating Expert System and Databases, CCTA ISE Report Number 29, 1988

[KEY89] Keyes, J., Why expert systems fail. *AI Expert*, 4(11), 50-53, 1989

[KRI96] Krishnamoorthy, C. S., Rajeev, S., Artificial Intelligence and Expert Systems for Engineer, CRC Press, Boca Raton – Florid, 1996, ISBN 0-8493-9125-3

[LIE93] Liebowitz, J., Educating knowledge engineers on knowledge acquisition, *IEEE International Conference on Developing and Managing Intelligent Systems Projects*, 110-117, 1993

[LUS98] Luger, G.F., Stubblefield, W.A., Artificial Intelligence (3rd ed.), Addison Wesley Publisher, 1998

[MCH89] Mcgraw K.L., Harbison-Briggs K., Knowledge Acquisition for Expert System. Principles and Guidelines Prentice-Hall International Editions By: Karan.L. Mcgraw and Karan Harbison-Briggs, Page 1- 27, 1989

[MIN68] Minsky, M., Semantic Information Processing, MIT Press, Cambridge MA, 1968

[MOR93] Morik, K., Knowledge Acquisition and Machine Learning: Theory, Methods and Application, Universität Dortmund, Germany, Academic Press, Page 5, 1993

[NEG02] Negnevitsyk, M., Artificial Intelligence - A Guide to Intelligent Systems, Addison Wesley, 2002

[NEW72] Newell, A., Simon H. A., Human Problem Solving, Prentice Hall, Englewood Cliffs, NJ, 1972

[NIE96] Niederman, F., Acquiring knowledge about group facilitation: research propositions, *Proceedings of the 1996 conference on ACM SIGCPR/SIGMIS,* p.58-67, 1996

[NIL71] Nilson, N.J., Problem Solving Method in Artificial Intelligence, Tioga, Palo Alto CA, 1971

[ONM89] O'Neil, M. & Morris, A., Expert systems in the United Kingdom and evaluation of development methodologies, *Expert Systems, 6*, p. 90-99, 1989

[PBF98] Preece, A. D., Borrowman A. J., Francis, T. J., Reusable Components for Knowledge Base and Database Integration, Technical Report AUCS/TR9803, 1998

[PSU02] PSU, Lecture Note: Knowledge Acquisition INFSY 565, The Pennsylvania State University, 2002
http://www.courses.psu.edu/Materials/_off_/infsy565_lmm121/Knowacq.ppt

[REE96] Rees, P. L., User participation in expert systems. *Industrial Management & Data Systems, 93*(6), p. 3-7, 1996

[RHE01] Rhem, A.J. & Associates Inc., A Framework for Knowledge Acquisition, White Paper, Page 3, 2001

[RIK91] Rich, E. and Knight, K., Artificial Intelligence (2nd. ed.) by, McGraw Hill Publisher, 1991

[RIC98] Richard, D., Ripple Down Rules with Formal Concept Analysis: A Comparison to Personal Construct Psychology, Eleventh Workshop on Knowledge Acquisition, Modelling and Management, Banff - Canada, 1998

[RCO97] Richards, D., Compton, P., Uncovering the Conceptual Models in RDR, *Proceedings of the International Conference on Conceptual Structures ICCS'97*, Springer Verlag , Seattle, 1997.

[ROG67] Rogers, C. R., On becoming a person: A therapist's view of psychotherapy, London, 1967

[ROO89] Rook, F., Croghan, J., The knowledge acquisition activity matrix: a systems engineering conceptual framework, *IEEE Transaction on Systems, Man & Cybernetics*, 19(3), 586-597, 1989

[SIA97] Siau, K., Lecture Note #2: Building an Expert System - Decision Support and Expert Systems, Department of Management, University of Nebraska-Lincoln, 1997
http://www.ait.unl.edu/siau/1997/mgmt457/lecture12.rtf

[SHA96] Sharples, M., Hogg, D., Hutchinson, C., Torrance, S. and Young, D., Computers and Thought: A Practical Introduction to Artificial Intelligence, University of Sussex, UK, 1996

[SHG87] Shaw, M.L.G., Gaines, B.R., KITTEN: Knowledge initiation and transfer tools for experts and novices, International Journal of Man-Machine Studies 27(3) p. 251-280, 1987

[SHG88] Shaw, M.L.G., Gaines, B.R., A methodology for recognizing consensus, correspondence, conflict and contrast in a knowledge acquisition system, Boose, J.H. & Gaines, B.R. (Eds) *Proceedings of the Third AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop*, pp.30-1-30-19. Banff – Canada, 1988

[SHW87] Shaw, M.L.G., Woodward, J.B., Validation of a knowledge support system, Boose, J.H. & Gaines, B.R. (Eds) *Proceedings of the Second AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop*, pp.18-0-18-15. Banff - Canada, 1987

[ACM92] ACM SIGCHI, *Curricula for Human-computer Interaction*. New York: ACM Special Interest Group on Computer-Human Interaction Curriculum Development Group, 1992

[SMI96] Smith, P., *An introduction to knowledge engineering*. London: International Thompson Computer Press, 1996

[STE93] Stein, E. W., A method to identify candidates for knowledge acquisition, *Journal of Management Information Systems, 9*(2), p. 161-178, 1993

[STU00] Studer, Rudi., Decker, Stefan., Fensel, Fensel., and Staab, Steffen., Situation and Perspective of Knowledge Engineering, Knowledge Engineering and Agent Technologies. IOS Press, Amsterdam, 2000

[SUT95] Sutcliffe, A. G., Human-Computer Interface Design, 2nd ed. Houndsmills, Basingstoke, Hampshire: Macmillan Press Ltd, 1995

[SWG95] Swartout, B., Gil, Y., "EXPECT: Explicit Representations for Flexible Acquisition", *In Proceedings of the Ninth Knowledge Acquisition for Knowledge-Based Systems Workshop*, February 26-March 3, Banff, Alberta, Canada, 1995

[TSA94] Tsai, N, Necco, C., Wei, G., An assessment of current expert systems: Are your expectations realistic? *Journal of Systems Management*, November 1994, 28-32, 1994

[TUR98] Turban, E., Aronson, J.E., Decision Support Systems and Intelligent Systems, Prentice Hall, Upper Saddle River, NJ, 1998

[VIL99] Villegas, Difficulties in Knowledge Acquisition, *Chapter 13: Knowledge Acquisition and Validation*, 1999
http://www.usfca.edu/~villegas/classes/992-6275/6275ch13/

[WAG88] Wagner, E. The Computer Display Designer's Handbook, Chartwell-Bratt, Lund, 1988

[WAP81] Waterman, D. A., Peterson, M., Model of Legal Decision-Making, Rand Repost, Santa Monica, Canada, 1981

[WAT85] Waterman, D.A., *A guide to expert systems*, Reading, Mass: Addison-Wesley Publishing, 1985

[WEL83] Welbank, M. A., A Review of Knowledge Acquisition Techniques for Expert Systems, British Telecommunications Research Laboratories Technical Report, England, 1983

[WHI97] White, S., Java-based Repertory Knowledge Elicitation Tool, Department of Computing Science, University of Aberdeen, Scotland, 1997

[WIK03] Wikipedia, The free encyclopedia, 2003
http://www.wikipedia.org/wiki/Transitive_property_of_equality

[WIK04] Wikipedia, The free encyclopedia, 2004
http://en.wikipedia.org/wiki/

[WIL82] Wille, R., Restructuring Lattice Theory: An Approach Based on Hierarchies of Concepts in *Ordered Sets* (Ed. Rival), pp. 445-470, Reidel, Dordrecht, Boston, 1982

[WIL92] Wille, R., Concept Lattices and Conceptual Knowledge Systems, Computers Mathematic Application *(23)* 6-9, p. 493-515, 1992

[WIM86] Williamson, M., Artificial Intelligence for Microcomputer, The Guide for Business Decision Makers, Brady/Simon & Schuster, p.2-3, 1986.

[WIN72] Winograd, T., Understanding Natural Language, Cognitive Psychology, 1972

[WHA04] WhatIs.com, Database – Definition, Feb 22, 2004
http://searchdatabase.techtarget.com/sDefinition

[WIS92] Winston, P.H., Artificial Intelligence, Addison Wesley Publisher, 1992

[WRA87] Wright G., Ayton P., Eliciting and Modelling Expert Knowledge, Decision Support Systems Volume 3, p. 13-26, 1987