IMPLEMENTING DEMPSTER-SHAFER THEORY FOR INEXACT REASONING IN EXPERT SYSTEMS

Thomas Michael Froese

B. A. Sc. (Civil Engineering) University of British Columbia, 1986

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

DEPARTMENT OF CIVIL ENGINEERING

We accept this thesis as conforming

to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

August 1988

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Civil Engineering
The University of British Columbia
1956 Main Mall
Vancouver, Canada

Date: Sept 19, 1988

Abstract

The work described in this thesis stems from the idea that expert systems should be able to accurately and appropriately handle uncertain information. The traditional approaches to dealing with uncertainty are discussed and are shown to contain many inadequacies.

The Dempster-Shafer, or D-S, theory of evidence is proposed as an appealing theoretical basis for representing uncertain knowledge and for performing inexact reasoning in expert systems. The D-S theory is reviewed in some detail; including its approaches to representing concepts, to representing belief, to combining belief and to performing inference.

The D-S implementation approaches pursued by other researchers are described and critiqued. Attempts made early in the thesis research which failed to achieve the important goal of consistency with the D-S theory are also reviewed.

Two approaches to implementing D-S theory in a completely consistent manner are discussed in detail. It is shown that the second of these systems, a frame network approach, has led to the development of a fully functional prototype expert system shell called FRO. In this system, concepts are represented using D-S frames of discernment, belief is represented using D-S belief functions, and inference is performed using stored relationships between frames of discernment (forming the frame network) and D-S belief combination rules. System control is accomplished using a discrete rule-based control component and uncertain input and output are performed through an interactive belief interface system called IBIS. Each of these features is reviewed.

Finally, a simple but detailed example of an application of a frame network expert system is provided. The FRO system user's documentation is provided in the appendix.

Table of Contents

List of Tables

List of Figures

Acknowledgement

The results reported in this thesis stem from a collaborative effort by Dr. William Caselton, Dr. Alan Russell, and myself. Dr. Russell and Dr. Caselton conceived, initiated, and obtained funding for the research project. I wish to thank them for their guidance and their contributions and for providing a productive and pleasant working environment.

I would also like to thank my friends, my family, and especially my wife for providing an ideal personal environment to complement my ideal professional environment.

Finally, the financial support provided by the Natural Sciences and Engineering Research Council of Canada, Strategic Grant G1417–Advanced Planning Technologies for Construction Productivity Improvement, is gratefully acknowledged.

Thomas Froese, August 1988.

Chapter 1

Introduction

This thesis reports on a research project which explored the use of the Dempster-Shafer theory for inexact reasoning in expert systems. This chapter provides introductions to the field of inexact reasoning, to the background of the research project, and to the structure of the thesis itself.

1.1 Introduction to Inexact Reasoning in Expert Systems

The field of **artificial intelligence (AI)** can be though of as the art and science of making a computer model human thinking. Two clear motivations for studying AI are first, to use computer models to achieve a better understanding of human thinking and second, to make computers more efficient, since the human brain is an extremely effective computer for most tasks. Under the latter category, the most direct application of AI approaches and techniques has been in the field of **expert systems**. Expert systems are computer programs which are used to solve problems or perform other reasoning tasks in a manner analogous to the way a human expert solves problems. The term "expert system" better describes a programming *style* or approach than a radically new and identifiable class of programs. Some characteristics of this style are that the code is written out in an English-like, modular way (rules, for example) which mimics the way a human might describe what he knows; that the program communicates with the user in an English-like and easy to understand manner; that the performance of the program depends first on shear volume of code (as much of the human's knowledge as possible

should be encoded) and only secondly on the efficient structuring of the program (which is the essence of performance in most programs); that the program can explain and justify its results to some extent; and, finally, that the program can deal with uncertain and missing information.

This last characteristic is of particular importance for expert systems in construction management, since decision making and problem solving in this field are invariably founded on uncertain information–often exclusively so. In order for a construction management expert system to approach human-like (or even useful) performance, then, it must accurately reflect uncertainty in its expert knowledge, it must accommodate uncertain input from the user, it must accurately account for these uncertainties in its reasoning, and it must rationally interpret and report the resulting uncertainty in its conclusions. The performance of all these tasks is called **inexact reasoning**.

Several approaches to inexact reasoning have been proposed for use in expert systems. In general, only the simplest of these have achieved any widespread usage. Yet the performance of these simple popular approaches falls far short of current state-of-the-art techniques. One such advanced technique is based on a theory outlined by Glen Shafer in his book, *A Mathematical Theory of Evidence* [57]. This theory has come to be known as the **Dempster-Shafer**, or **D-S**, theory. D-S theory is richly expressive and academically impressive. However the development of acceptable implementations of the theory for inexact reasoning is not straightforward and has been somewhat slow in coming. The research described by this thesis has made several significant steps towards the rectification of this situation.

1.2 Introduction to the Research Project

The research project documented in this thesis was initiated in the spring of 1986 by Dr. Alan Russell and Dr. William Caselton (both Associate Professors in the department of Civil Engineering at the University of British Columbia). The initial goal of the project was to explore the opportunities offered by the new field of expert systems to the field of construction management.

Thomas Froese was hired as the research engineer for the project.

It was found that almost all published work in expert systems for civil engineering dealt with the development of expert system prototypes for specific engineering applications. While these works generally advance the state-of-the-art in available solution approaches for the areas of application, they rarely contribute to the discipline of engineering expert systems in general. It was therefore decided that this research project should examine "lower level" issues and focus on some aspect of the expert systems themselves rather than on a specific application. In particular, it was decided that the ability of expert systems to deal with uncertainty was vital to most civil engineering and project management applications and should be examined in some detail.

This line of inquiry was pursued for a period of over two years and focused primarily on the implementation of a D-S based inexact reasoning system. More specifically, the following topics were addressed; with efforts ranging from theoretical enhancements of the D-S theory itself to purely practical implementation and programming concerns:

- It was determined that D-S theory held great potential as the basis for an expert system which managed uncertainty in such a way that the user could place utmost confidence in the accuracy and reasonableness of the results.

- It was found that it is possible to create a system in which all belief representation and inference is **fully consistent** with the D-S theory, though most previous implementation attempts have not succeeded in doing so. It was determined that full consistency is of utmost importance in creating a reliable system. The development of a fully consistent D-S system became the prime goal of the research project.

- A functional, fully consistent implementation was eventually developed. This implementation took the form of an expert system shell program which could be used to create expert systems for any specific applications.

- User interface issues for D-S belief were examined in some detail and an interactive graphical belief interface approach was developed.

- It was determined that a D-S representation is not well suited for representing *procedural knowledge* in an expert system. The shell program therefore incorporates a separate rule-based component which performs the procedural or control functions of the system.

The results of the research include the following achievements:

- At least four papers published or submitted for publication (these are listed in section 1.3).

- An Interactive Belief Interface System, or **IBIS**, which provides a simple graphical input and output of uncertain belief using a D-S representation.

- A full-featured rule-based expert system shell program which can be used either as a control module for a D-S based system or can be used as a stand-alone shell comparable with many commercially available shells. This program is called **RO** for **R**ules **O**nly. RO was used by Barry Chilibeck in the Civil Engineering Department at UBC to create **WASTE**, a significant environmental engineering expert system.

- A fully consistent D-S based expert system shell. Since the knowledge representation in the shell incorporates both D-S *frames of discernment* and control *rules*, the program is called **FRO**, for **F**rame-**R**ule **O**rganizer. FRO utilizes both IBIS and RO as well as a automated knowledge base creation system called **MKB**, for **M**ake **K**nowledge **B**ase. The FRO program represents the culmination of the research project's accomplishments.

1.3 Introduction to this Thesis

This Thesis is structured as follows:

Chapter 2 discusses the traditional approaches to handling uncertainty in expert systems. These include Bayesian probability theory and MYCIN certainty factor theory.

Chapters 3 through 6 provide a thorough examination of D-S theory. These chapters cover concept representation, belief representation, belief combination and inference respectively.

Chapter 7 reviews other researchers' efforts at interpreting and implementation D-S theory for inexact reasoning. Most of these systems are found to be non-consistent with D-S theory– the distinction between non-consistent and consistent systems arises in this chapter and the advantages of the latter approach are stressed.

Chapter 8 discusses non-consistent implementation attempts made early in this research project.

Chapter 9 describes the project's first major consistent implementation attempt. This is referred to as the **global frame** approach.

Chapter 10 provides a detailed account of the **frame network** implementation approach. The frame network approach to inexact reasoning incorporates all of the achievement of the research project. This approach forms the basis of the FRO expert system shell.

Chapter 11 focuses on several aspects of working with belief functions, including the specification of both *evidential* and *relational* belief functions and the interpretation of system conclusions expressed in terms of belief functions. The IBIS belief input system is introduced in this chapter.

Chapter 12 illustrates the use of a D-S based expert system through a detailed example taken from the field of construction management. Some issues of incorporating uncertainty in knowledge base design are also discussed.

Chapter 13 provides a brief summary and conclusion.

The appendix to this thesis contains the complete user documentation for the FRO expert system shell. This document provides a detailed examination of the entire system from a user's point of view.

Note that chapters 2 through 7 inclusive incorporate review and critique of other researchers' work (with the notable exceptions of section 4.5 on classes of support function, section 5.2 on alternative belief combination rules and, to some extent, chapter 6 in D-S inference). The material described in the remaining chapters represents the efforts of this research project.

Finally, much of this thesis is based on material first written for publication elsewhere. These include the following:

- Sections 3.2, 4.1, 4.4, 5.1, 6.1, and 11.2 are based on "Belief Input Procedures for Dempster-Shafer Based Expert Systems," by W.F. Caselton, T.M. Froese, A.D. Russell, and W. Luo, Presented at *AIENG88, the Third International Conference on Applications of Artificial Intelligence in Engineering*, Palo Alto, CA, August, 1988; and published in *Artificial Intelligence in Engineering: Robotics and Processes*, Computational Mechanics Publications, Southampton, 1988, pp.351–370

- Chapter 12 is derived from "A Dempster-Shafer Based Construction Expert System" by A.D. Russell, T.M. Froese and W.F. Caselton, presented at the *Third International Conference on Computing in Civil Engineering*, Vancouver, B.C., August 1988, and published in the accompanying proceedings.

- Section 5.2 and chapter 10 stem from "Implementation Strategies for Dempster-Shafer Based Inexact Reasoning" by T.M. Froese, A.D. Russell, and W.F. Caselton. Submitted for publication to the *International Journal of Intelligent Systems* in July, 1988.

- Section 4.3 and 11.1 are from "An Interactive Belief Interface System" by T.M. Froese, W.F. Caselton, and A.D. Russell. To be submitted for publication.

Chapter 2

Uncertainty in Expert Systems

This chapter briefly outlines the techniques which were first developed for handling uncertainty in expert systems and which are still most commonly found in practice. These include **probability theory** (which is described in section 2.1) and **MYCIN certainty factors** (described in section 2.2). A brief comparison of D-S theory with these other approaches is provided in section 2.1.

2.1 Probability Theory

The broad field of probability theory is, perhaps, the most commonly used general approach to dealing with any type of uncertainty in a systematic mathematical way. Not surprisingly then, the first attempts to incorporate uncertainty into the reasoning processes of expert systems were based on probability theory. The most significant of these early probability-based attempts was the **PROSPECTOR** system created by Duda, Hart and others (see Duda et al [17]). Rather than focussing on the specific approach adopted in PROSPECTOR, this section deals with some of the fundamental problems of probability theory for use in expert systems.

Probability theory (in the sense that it is used here) provides for the specification of the **probability**, $P(h)$, of some hypothesis h, where $P(h)$ is a real number between zero and one which represents a measure of belief in the hypothesis. Since the acquisition of new evidence, e, can change our belief about some hypothesis, a **prior** probability, $P(h)$, can be modified to become the **conditional** or **posterior** probability, $P(h|e)$, that is, the probability of the hypothesis h

in light of the evidence e.

For expert systems based on rules of the form "conclusion c IF premise p", applying probability theory to a rules involves the calculation of the probability of the conclusion c given that the premise p is true. Applying a rule, therefore, calls for the calculation of $P(c|p)$. $P(c|p)$ is calculated using Bayes' theorem. Given the prior probability $P(c_i)$ of each possible conclusion c_i and the probability of a particular premise p when c_i is true, $P(p|c_i)$, then the probability $P(c|p)$ for some particular conclusion c is given by:

$$P(c|p) = \frac{P(c)\,P(p|c)}{\sum_i P(c_i)\,P(p|c_i)} \tag{2.1}$$

One problem with this technique is that it requires a considerable amount of probabilistic information. For n possible conclusions and m possible premises, we need $n(m+1)$ probabilities to be able to use Bayes' theorem as above. When there are two premises p_1 and p_2 which both confirm the same conclusion, the Bayesian combination is given by:

$$P(c|p_1 \,\&\, p_2) = \frac{P(p_2|c \,\&\, p_1)\,P(c|p_1)}{\sum_i P(p_2|c_i \,\&\, p_1)\,P(c_i|p_1)} \tag{2.2}$$

In this case, $n(m^2 + m + 1)$ probabilities are required. These overwhelming information requirements effectively prevent probability from being directly applied for expert systems as shown here. In the PROSPECTOR system, the problem was avoided by making the following assumptions:

$$P(p_1|p_2 \,\&\, c) = P(p_1|c) \tag{2.3}$$

$$P(p_1|p_2 \,\&\, \text{not } c) = P(p_1|\text{not } c) \tag{2.4}$$

These assumptions, while they seem to comprise reasonable approximations, stem from an assumption of conditional independence between the two premises. This assumption has been

shown to be clearly false for general expert systems applications (see Quinlan [54, p.257]). Thus one of the main advantages of using probability theory for inexact reasoning–the fact that it is fully consistent with a sound and applicable underlying theory–is lost.

Apart from problems of finding a feasible and applicable implementation approach, the use of probability numbers themselves have shown to be unsatisfactory for representing uncertain belief in expert systems, as outlined in the following objections:

1. Probabilities don't convey the precision of the uncertainty measurement. We can't discern between the probability $P(a) = 0.500 \pm 0.001$ and the probability $P(a) = 0.5 \pm 0.3$.

2. Probabilities don't discern the difference between evidence for and against a proposition. The probability $P(a) = 0.5$ does not discern between "*moderate evidence for* (a) *and no evidence against* (a)" and "*strong evidence for* (a) *and moderate evidence against* (a)".

3. A lack of evidence for a proposition implies evidence against that proposition. In other words, the probability $P(a) = 0.4$, implies that there is a probability $P(\text{not } a) = 0.6$. The disadvantage of such a result is illustrated by Carl Hempel's *Paradox of the Ravens*:

   > "Let h_1 be the statement that '*all ravens are black*' and h_2 the statement that '*all nonblack things are nonravens.*' Clearly h_1 is logically equivalent to h_2. If one were to draw an analogy with conditional probability, it might at first seem valid, therefore, to assert that $C[h_1, e]$ (a measure of confirmation of the hypothesis h given some body of evidence e) $= C[h_2, e]$ for all e. However, it appears counter-intuitive to state that the observation of a green vase supports h_1, even though the observation does seem to support h_2. $C[h, e]$ is therefore different from $P(h|e)$ for it seems wrong that an observation of a vase could logically support an assertion about ravens." (from Buchanan and Shortliffe [4, p.244]).

Thus a direct application of probability theory, while providing an obvious and useful starting point, clearly does not provide a satisfactory scheme for dealing with uncertainty in expert systems. This is not to say that probability theory *cannot* lead to a useful technique for inexact reasoning. A great deal of work continues on altering the knowledge representation approach so that Bayesian probabilities are both feasible and appropriate. Indeed D-S theory which provides the subject for most of this thesis is closely related to probability theory (although the exact nature of the relationship is debatable). The most significant development path to date, however, has not involved finding a more complex and consistent implementation–rather it has been the simplification and approximation of probability theory leading to the certainty factor approach (which is described in the next section).

For introductory material on probability approaches and comparison with other schemes, see Buchanan and Shortliffe [4, pp.233–237], Lee et al [40], and Quinlan [54]. For discussion of state-of-the-art probability approaches, see Charniak [7], Nilsson [47], and almost anything by Pearl ([50],[51], and [52]).

2.2 Certainty Factors

Like PROSPECTOR, the **MYCIN** system was a "landmark" expert system developed in the mid-seventies by a group at Stanford University (see Buchanan and Shortliffe [4]). Also like PROSPECTOR, the group working on MYCIN considered the problem of incorporating uncertainty and looked first to probability theory. Unlike the PROSPECTOR project, however, probability theory was used only as an approximate guideline from which the new and simple **theory of certainty factors** was developed. The certainty factor approach was loosely created from a series of intuitive concepts rather than from a formal cohesive underlying theory.

In the theory of certainty factors, every proposition (a) is assigned two values: a **MEASURE OF BELIEF** – $MB(a)$, which is a numerical value between zero and one proportional to the

amount of evidence to support the proposition (a), and a **MEASURE OF DISBELIEF** – $MD(a)$, which measures the evidence against (a). For each proposition, belief can be summed up by one value, the **CERTAINTY FACTOR** – $CF(a)$, which is a number between negative one and one and is calculated from the difference between the measure of belief and the measure of disbelief, that is:

$$CF(a) = MB(a) - MD(a) \tag{2.5}$$

For a simple expert system rule of the form "conclusion c IF premise p", the belief in the conclusion can be obtained directly from the belief in the premise by setting:

$$MB(c) = MB(p) \tag{2.6}$$

$$MD(c) = MD(p) \tag{2.7}$$

However, rules are typically of a more complex form and the MYCIN system includes formulae for accommodating these:

1. Conjunctive Premises.

   For a rule of the form:

   conclusion c IF premise p_1 AND premise p_2

   We can calculate the belief in the combined premise from the following equations:

$$MB(p_1 \text{ AND } p_2) = \min\left(MB(p_1), MB(p_2)\right) \tag{2.8}$$

$$MD(p_1 \text{ AND } p_2) = \max\left(MD(p_1), MD(p_2)\right) \tag{2.9}$$

2. Disjunctive Premises.

For a rule of the form:

conclusion c IF premise p_1 OR premise p_2

Belief in the combined premise is calculated from the following equations:

$$MB(p_1 \text{ OR } p_2) = \max\Big(MB(p_1), MB(p_2)\Big) \tag{2.10}$$

$$MD(p_1 \text{ OR } p_2) = \min\Big(MD(p_1), MD(p_2)\Big) \tag{2.11}$$

3. Weak Inference.

If there is any uncertainty about the strength of the rule itself, we can assign two measures of belief to the rule: MB' and MD' – corresponding to our belief and disbelief in the conclusion assuming complete belief in the premise. These measures of belief can then be combined with the actual belief in the premise to determine the resulting belief in the conclusion, according to the following equations:

$$MB(c) = MB' \times \max\Big(0, CF(p)\Big) \tag{2.12}$$

$$MD(c) = MD' \times \max\Big(0, CF(p)\Big) \tag{2.13}$$

4. Combining Evidence.

If one rule reaches a conclusion c with belief $MB_1(c)$ and $MD_1(c)$ and a second rule reaches the same conclusion with belief $MB_2(c)$ and $MD_2(c)$, then the resulting combined belief can be derived from the following equations:

$$MB(c) = MB_1(c) + \Big(MB_2(c) \times \Big(1 - MB_1(c)\Big)\Big) \tag{2.14}$$

$$MD(c) = MD_1(c) + \Big(MD_2(c) \times \Big(1 - MD_1(c)\Big)\Big) \tag{2.15}$$

The certainty factor approach alleviates the problem found in probability theory of distinguishing between belief and disbelief, However the theory does cause the following problems:

1. Like probability numbers, certainty factors cannot distinguish the precision of the measures of belief.

2. The use of maximum and minimum operators leads to discontinuities and non-intuitive results for general cases where the premise constituents are not fully dependant. For example, suppose we have the following rule:

   ```
   IF   the temperature is below freezing
   AND  there is precipitation
   THEN there will be snow.
   ```

   Then the CF for snow will be *identical* for the following two cases:

   $$CF(\texttt{the temperature is below freezing}) = 1.0$$
   $$CF(\texttt{there is precipitation}) = 0.5$$

   and:

   $$CF(\texttt{the temperature is below freezing}) = 0.5$$
   $$CF(\texttt{there is precipitation}) = 0.5$$

   This is a counter-intuitive result.

3. The following two rules:

   ```
   IF   there is rain
   THEN there is precipitation.
   ```

   ```
   IF   there is snow
   THEN there is precipitation.
   ```

Should be identical to the following single rule:

```
IF   there is rain
OR   there is snow
THEN there is precipitation.
```

Yet these two representations of the same information will lead to two completely different combination rules being used in determining the CF for precipitation based on the CF's for rain and snow.

4. Perhaps the major criticism against this system is that it has no adequate theoretical basis (in fact, the theory of certainty factors has been shown to be essentially the same as probability theory with the assumption of conditional independence, see Buchanan and Shortliffe [4, Ch.12], although this is of academic interest since we have already shown the assumption of conditional independence to be generally invalid).

Regardless of these difficulties, MYCIN did seemed to perform well in real world medical situations. MYCIN was eventually distributed without it's knowledge base as **EMYCIN** (for Empty MYCIN) to become one of the first expert system shells. Not only the basic format of rules but also the uncertainty handling of most present expert system shells are derived directly from MYCIN and its certainty factor approach.

2.3 Comparison of D-S with Other Theories

In comparison with the uncertainty handling theories presented in this chapter, the D-S theory fares well. D-S achieves flexibility by using a less restrictive representation of uncertainty than does, for example, Bayesian analysis. D-S belief is normally expressed in terms of two belief parameters and belief can be assigned to various levels of granularity or precision. This gives D-S the expressive power to represent nuances of uncertain belief which cannot be captured by

MYCIN-like certainty factors, for instance. Shenoy and Shafer [68] suggest that D-S provides a useful compromise between the lack of structure offered by the use of certainty factors in production rules and the unrealistic demands for structure often made by Bayesian schemes. Finally, D-S provides a very rigorous theoretic foundation for both representing knowledge and for processing that knowledge. Dubois and Prade [15] have suggested that an approach such as fuzzy set theory which offers "multiple points of view on combination" more closely models human reasoning; however we believe that for use in engineering problems at least, an approach which is rigorously derivable from a cohesive underlying theory is required to give confidence in the accuracy of the uncertain output (fuzzy set theory is not discussed in this thesis, although section 7.6 deals with comments made by L. Zadeh, the originator of fuzzy set theory). Table 2.3 provides a summary comparison of D-S with three other major approaches for handling uncertainty in expert systems. See Lee et al [40] for an introduction to and further comparison with these other systems.

Table 2.1: Comparison of Uncertainty Theories

	Certainty Factors	Bayesian Probability	Fuzzy Set Theory	Dempster-Shafer Theory
Flexibility	high	low	high	high
Expressive power	low	low	high	high
Theoretical Background	weak	strong	moderate	strong

(based on Lee et al [40])

2.4 Conclusion

This chapter has dealt with two approaches to performing inexact reasoning in expert systems. It has been shown that although there are many problems associated with both approaches, both have been successfully applied and, in the case of certainty factors particularly, have become the "industry standard".

Perhaps a fitting summary to the current state of inexact reasoning in practical expert systems is illustrated by considering a feature offered by the ARITY expert system shell. This shell offers three different approaches to handling uncertainty, referred to in the documentation as a standard MYCIN approach, a fuzzy set theory approach, and a probability approach. Each approach uses simple CF's in conjunction with normal rules. Table 2.2 summarizes how each approach combines the CF's of conjunctive and disjunctive premise terms, how each includes a CF assigned to the rule itself, and how each combines positive and negative CF's (or MB's and MB's) to get a single combined CF.

Table 2.2: Inexact Reasoning Approaches used in ARITY Expert System Shell

Method	Conjunctive Premise	Disjunctive Premise	Combining CF of Rule	Combining Pos & Neg
Standard	Minimum	Maximum	$CF_{\text{RULE}} \times CF_{\text{PREM}}$	Pos + Neg
Fuzzy	Minimum	Maximum	CF_{RULE}	Largest abs. value
Prob.	$A \times B$	$A + B \times (1 - A)$	$CF_{\text{RULE}} \times CF_{\text{PREM}}$	Pos + Neg

(from the Arity Expert Systems Development Package Documentation [1, p.201])

Each of the three approaches outlined in table 2.2 uses identically defined sets of certainty numbers, but each method performs different operations on them. While it is claimed that having several alternative calculations makes for a powerful system, it could be interpreted more critically as proof that these numbers provide an extremely ad hock approach to inexact reasoning at best, and that there is no reliable formal method for using them.

Chapter 3

D-S Theory I: Concept Representation

In the 1960's, Arthur Dempster at Harvard originated a body of work which extended classical probability theory so that it could be brought to bear on a set representation rather then a point representation [12]. Glen Shafer at Princeton continued this work and, in 1975, published a book entitled *A Mathematical Theory of Evidence* [57], which describes in some detail a comprehensive, powerful, intuitively satisfying and mathematically rigorous theory of evidence. The work of these two mathematicians has lead to a body of knowledge commonly referred to as the **Dempster- Shafer** or **D-S** theory (sometimes also referred to as the theory of belief functions). This thesis deals primarily with the application of D-S theory for performing inexact reasoning in expert systems.

The following four chapters provide a thorough outline of D-S theory. These chapters deal with the representation of concepts, the representation of belief, the combination of belief, and D-S based inference respectively.

3.1 Frames of Discernment and Propositions

Shafer defines a **frame of discernment, Θ,** as a set of possible answers to some question, or a set of possible values for some variable. In any situation, one and only one of the frame of discernment's **elements, θ,** may represent the "truth" – that is, the elements must be exhaustive and mutually exclusive. We can then define a **proposition, A,** as a subset of Θ and we say that A is true if the true value of Θ lies within A. The set of all propositions derived from a

frame of discernment Θ is denoted 2^{Θ}, we call this set of propositions the **discernment space.** Among the propositions in 2^{Θ} are the frame itself, the **singletons** (or sets containing a single element only), and the **empty set,** $\emptyset$ (which can be thought of as the set with no elements). A discernment space, then, is a structured collection of propositions.

For the purposes of this thesis, we will denote the names of elements within a proposition with lower case italic letters and denote the names of propositions themselves with upper case italic letters. In some cases, we may want to give a single proposition several names or aliases corresponding to several logically equivalent concepts. The following, for example, are logically equivalent names for some proposition whose elements are a list of North American countries:

$$\{canada, usa, mexico\} = NORTH\ AMERICAN\ COUNTRIES$$

The logical relationships between propositions can be found using formal set theory. For example, the logical notions of **conjunction, disjunction, implication** and **negation** translate into the set-theoretic notions of **intersection, union, inclusion** and **complementation** ($\cap$, $\cup$, $\subset$ and $\overline{\ }$ in our notation). Note that in the context of D-S theory, two propositions are said to **conflict** if they have no common elements, that is if their intersection equals $\emptyset$. These basic building blocks of concept representation are illustrated in the following example.

Example 3.1 Suppose that we are interested in the type of ground material that we might encounter during excavation on a construction project. We might decide that the possible ground types are rock, soil, or sand. For simplicity, we will assume that these alternatives are mutually exclusive and exhaustive (e.g. there can be no mixture of rock and soil nor any other materials such as clay). Our frame of discernment in this case would then be:

$$\Theta_{MATERIAL} = \{rock, soil, sand\}$$

where *rock*, *soil* and *sand* are the elements of the frame. Any subset of $\Theta_{MATERIAL}$

is a proposition representing some concept. For example, the subset $\{rock\}$ is a singleton proposition symbolizing the concept that the ground material is rock. The subset $\{soil, sand\}$ is a proposition denoting the idea that the ground is either soil or sand. We might want to assign the name or alias "$EARTH$" to this last set. The set of all possible propositions for this problem is the discernment space $2^{\Theta_{MATERIAL}}$, shown graphically as a line diagram in figure 3.1.

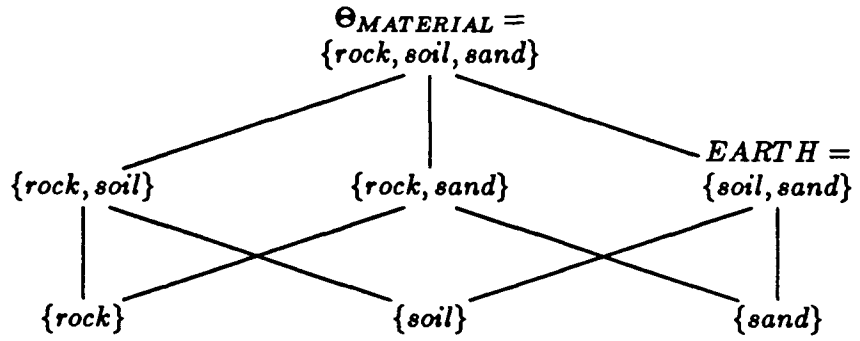


Figure 3.1: Graphical representation of the discernment space $2^{\Theta_{MATERIAL}}$

The logical relationships between propositions can be illustrated by observing that the logical notion of "not sand" is represented by the set $\overline{\{sand\}}$, the set-complement of the proposition $\{sand\}$, which is equivalent to $\{rock, soil\}$. The logical "soil or sand" becomes the union $\{soil\} \cup \{sand\}$, the value of which is the proposition $\{soil, sand\}$ or $EARTH$. Similarly, "earth and not sand" becomes the intersection $\{soil, sand\} \cap \overline{\{sand\}}$, or $\{soil, sand\} \cap \{rock, soil\}$ which yields the proposition $\{soil\}$. Finally, the logical relationship "sand implies earth" is represented by the fact that the proposition $\{sand\}$ is included in the proposition $EARTH$ or $\{soil, sand\}$ (i.e. $\{sand\} \subset \{soil, sand\}$).

3.2 Frame Refining and Frame Coarsening

We have sectioned our knowledge about the problem domain into a set of propositions. However we may find that our assignment of propositions is not precise or exact enough to discern some difference between two concepts (in the sense that the proposition "car" is not precise enough to distinguish between a Volkswagon and a Rolls Royce). We can therefore divide some of the elements from a frame Θ up into sets of smaller elements and use these to create a second, expanded and more precise frame Ω called a **refinement.**

We can specify this process by assigning, for each element $\theta \in \Theta$, a **refinement statement** of the following form:

$$\omega(\{\theta\}) = X \tag{3.1}$$

In such statements, the proposition $X \subset \Theta$ is comprised of all the new elements into which θ has been partitioned. The statements (one for every singleton) can then be used to define a refinement statement for *every* proposition $A \subset \Theta$, according to the formula:

$$\omega(A) = \bigcup_{\theta \in A} \omega(\{\theta\}) \tag{3.2}$$

This gives the new proposition $\omega(A) \subset \Omega$ obtained by the partitioning of the elements in proposition A (these two propositions, then, are logically equivalent and the proposition name A can be assigned as an alias to the proposition $\omega(A)$). The use of equation 3.2 to refine the full set Θ yields the set corresponding to the new frame Ω:

$$\omega(\Theta) = \Omega = \bigcup_{\theta \in \Theta} \omega(\{\theta\}) \tag{3.3}$$

The collection of the refinement statements for all of the propositions in the discernment space

makes up a mapping, called a **refining,** $\omega : 2^{\Theta} \rightarrow 2^{\Omega}$, which thoroughly describes the partitioning. In refining some frame of discernment, then, we use new knowledge to create a new frame which can not only discern every proposition from the original frame, but can also discern the difference between more precise concepts which were indistinguishable in the original frame.

In addition to making the elements of a single frame more precise, frame refinement arises in a more general case when we are interested in exploring the relationships between propositions which are contained in two totally separate frames of discernment. To accomplish this, we require one new frame which is a refinement of two original frames and as such will discern all of the propositions from both initial discernment spaces as well as the correct set relationships between them. This frame will not be just a cross product of the two original frames, for it is created using new knowledge which indicates how the various propositions are related. Although there could be many frames which meet this criteria, there will be one unique frame called a **minimal refinement,** denoted $\Theta_P \otimes \Theta_Q$ for two original frames Θ_P and Θ_Q, which has fewer elements than all the rest (that is, all other such frames are a refinement of the minimal refinement). A convenient notation for the elements of a minimal refinement $\Theta_P \otimes \Theta_Q$ is to use ordered pairs (p, q). An ordered pair is included in the frame only if it corresponds to a possible conjunctive combination of the elements $p \in \Theta_P$ and $q \in \Theta_Q$: that is, if it is logically possible for the concepts represented by p and q to exist simultaneously:

$$\Theta_P \otimes \Theta_Q = \{(p,q) | \, p \in \Theta_P, \; q \in \Theta_Q, \; p \text{ and } q \text{ can coexist}\} \tag{3.4}$$

Any number of frames can be combined into one minimal refinement in a similar manner.

The inverse of the refinement process is to modify a larger frame to produce a smaller, less precise one. This is called frame **coarsening** or **reduction,** $\theta : 2^{\Omega} \rightarrow 2^{\Theta}$ and it is defined for any proposition $A \subset \Omega$ by the equation:

$$\theta(A) = \{\theta \in \Theta | \, \omega(\{\theta\}) \cap A \neq \emptyset\} \tag{3.5}$$

Example 3.2 Referring to example 3.1, we may now decide that the element "*rock*" is not precise enough but that we should be able to distinguish between "*bed rock*" and "*aggregate*". We can refine our frame of discernment by specifying the refinement statements:

$$\begin{aligned} \omega(\{rock\}) &= \{bed\,rock, aggregate\} \\ \omega(\{soil\}) &= \{soil\} \\ \omega(\{sand\}) &= \{sand\} \end{aligned}$$

We can now create the complete refining $\omega : 2^{\Theta_{MATERIAL}} \rightarrow 2^{\Omega_{MATERIAL}}$ using equation 3.2, thereby yielding the new frame:

$$\Omega_{MATERIAL} = \{bed\,rock, aggregate, soil, sand\}$$

Finally, we may identify a separate frame which discerns all the possible excavation methods:

$$\Theta_{METHOD} = \{dig, blast, rip\}$$

The minimal refinement of these two frames would discern all of the possible material types, the possible excavation methods, and the proper relationships between them. If, for example, bed rock could be excavated by blasting or ripping, soil by ripping or digging, and aggregate or sand by digging only, then the minimal refinement frame would consist of the following ordered pairs:

$$\Omega_{MATERIAL} \otimes \Theta_{METHOD} = \{ \begin{array}{l} (bed\ rock, blast), \\ (bed\ rock, rip), \\ (aggregate, dig), \\ (soil, rip), \\ (soil, dig), \\ (sand, dig) \ \} \end{array}$$

Chapter 4

D-S Theory II: Belief Representation

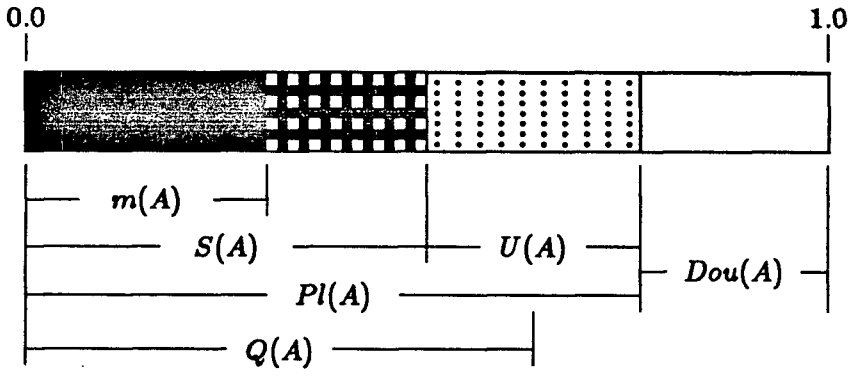
The previous chapter introduced an approach for representing concepts or variables and their possible values. This chapter discusses how to represent uncertain belief about which values are true for some particular situation.

4.1 Measures of Belief

The D-S theory utilizes several distinct parameters for describing the belief associated with a proposition. These parameters are all real numbers on the interval [0,1]. Figure 4.1 introduces some of these parameters and illustrates the relationships between them.

The **support,** $S(A)$, in some proposition A is perhaps the simplest measure of belief. It is defined formally as a number on the interval [0,1] which measures the degree to which we believe that the true value for the frame lies within set A. We can notice that $S(\emptyset) = 0.0$ and $S(\Theta) = 1.0$ since $\emptyset$ is false and Θ is true by their definitions. The **uncertainty,** $U(A)$, in a proposition A, measures the degree to which we believe that the true value could be within A, but could also be outside of A (i.e. the degree to which we don't know if A is true or not). The **doubt,** $Dou(A)$, in proposition A is the degree to which we are certain that the true value is not within set A. Because the elements in the frame of discernment are exhaustive, the doubt in A is equal to the support in not A. As shown above, the support, the uncertainty, and the doubt for any proposition must sum to 1.0.

For any proposition A, the belief parameters assigned to A can be illustrated by:



Where:

$m(A)$:
- the **basic probability** of A,
- $m(A) \leq S(A)$,
- indicates the belief which is assigned *exactly* to A; that is, belief in the statement "I believe that one of the elements of A is true, but I have no evidence to support one of these elements over another".

$S(A)$:
- the **support** for A,
- indicates the total belief that A is true.
- equals the belief exactly for A or for any subset of A (see equation 4.1).

$U(A)$:
- the **uncertainty** in A,
- indicates belief which is uncertain as to whether A is true or not.

$Dou(A)$:
- the **doubt** in A,
- $Dou(A) = S(\overline{A})$,
- indicates belief that A is not true.

$Pl(A)$:
- the **plausibility** of A,
- $Pl(A) = S(A) + U(A)$,
- $Pl(A) = 1 - Dou(A)$,
- indicates belief that A could be true.

$Q(A)$:
- the **commonality** of A,
- $m(A) \leq Q(A) \leq Pl(A)$,
- equals the belief exactly for A or for any superset of A (see equation 4.3).

Figure 4.1: Dempster-Shafer Belief Parameters

In comparison to $S(A)$ which describes our belief that A is definitely true, the **plausibility**, $Pl(A)$, describes our belief that A could be true. We can see from figure 4.1 that $Pl(A)$ is equal to $S(A) + U(A)$, which shows that plausibility represents the maximum degree of belief that could be expected if all the uncertainty in A were resolved in favour of A. $Pl(A)$ is also equal to $1 - Dou(A)$ or the degree to which we don't doubt A. It is not unusual for several propositions to have zero support but to have different, non-zero plausibilities. Thus $Pl(A)$ can be just as important as $S(A)$. Because $S(A)$ and $Pl(A)$ can be regarded as a lower and an upper bound on our belief in A, we often refer to $[S(A), Pl(A)]$ as a **support interval.**

Belief that the true value lies within a proposition implies that it also lies within every superset of that proposition. Therefore we would like some way of reflecting the level of precision, or the **granularity**, to which the belief is most accurately assigned. This information is provided by the **basic probability number,** $m(A)$, of a proposition A which is defined as the support which is assigned exactly to A, and cannot be assigned to any more precise subset of A. According to this definition, it can be shown that $m(A) = S(A)$ for all singletons. Support values and basic probability numbers are related to each other according to the equations:

$$S(A) = \sum_{B \subset A} m(B) \tag{4.1}$$

$$m(A) = \sum_{B \subset A} (-1)^{|A-B|} \, Bel(B) \tag{4.2}$$

where the term $|A - B|$ is the cardinality of the difference between sets A and B.

Finally, the **commonality, Q(A)** of a proposition A is the support assigned exactly to the proposition A or to some superset of A. This belief parameter does not correspond to any intuitive concept of belief, but it proves to be useful for certain computational processes (see section 5.2, for example). The following equations relate commonality numbers to basic probability numbers and to support values:

$$Q(A) = \sum_{\substack{B \subset \Theta \\ A \subset B}} m(B) \tag{4.3}$$

$$S(A) = \sum_{B \subset \overline{A}} (-1)^{|B|} Q(B) \tag{4.4}$$

$$Q(A) = \sum_{B \subset A} (-1)^{|B|} S(\overline{B}) \tag{4.5}$$

All of the above belief parameters are associated with some function which expresses their value for every proposition in a frame of discernment. For example, a **basic probability assignment** or **BPA**, defined over the frame Θ, is denoted $m : 2^{\Theta} \rightarrow [0,1]$. A BPA gives the basic probability number for each proposition A in a frame Θ in accordance with the following equations:

$$m(\emptyset) = 0 \tag{4.6}$$

$$0 \leq m(A) \leq 1 \tag{4.7}$$

$$\sum_{A \subset \Theta} m(A) = 1 \tag{4.8}$$

A **support function**, $S : 2^{\Theta} \rightarrow [0,1]$, gives the support for every proposition A in Θ. Support functions must obey the following formula:

$$S(\emptyset) = 0. \tag{4.9}$$

$$S(\Theta) = 1. \tag{4.10}$$

and, for every positive integer n and every collection $A_1, \ldots, A_n$ of subsets of θ:

$$S(A_1 \cup \ldots \cup A_n) \geq \sum_{\substack{I \subset \{1,\ldots,n\} \\ I \neq \emptyset}} (-1)^{|I|+1} S\left(\bigcap_{i \in I} A_i\right) \tag{4.11}$$

Plausibility functions, commonality functions, etc. can be similarly defined. It should be noted that any one of these functions carries the entire information associated with some body of belief and can be used to calculate all other parameters for each proposition. We also note that Shafer introduces support and plausibility numbers [57, p.144] as restrictions of more general **degree of belief,** $Bel(A)$, and **upper probability,** $P^*(A)$ values. However the restriction of degrees of belief to support is applicable to all information which might reasonably stem from real bodies of evidence, and in this thesis we will generally refer only to the later. Further Shafer defines a **belief function** as the set of degrees of belief for every proposition in a frame of discernment. In this thesis we will refer to a belief function in slightly more general terms as any D-S expression of some body of evidence, regardless of which D-S parameter is used for the representation.

We call all propositions with non-zero basic probability numbers in a belief function the **focal propositions.**

Different formulations of a body of belief can be used in different situations. For example, belief is usually solicited from the user or reported by system in the form of support functions or support intervals. However we have found BPA's to be the most useful form for internal processing, so the degree of belief or plausibility values are converted to basic probability after being entered into the system. We can illustrate a BPA diagrammatically in a manner similar to a Venn diagram by:

1. showing each element in a frame of discernment as a point in space,
2. representing the basic probability numbers by drawing boundaries or closed lines around the elements included in the focal propositions,

3. labeling these boundaries with the corresponding basic probability numbers.

If we wish, we can use these diagrams to obtain the support for a proposition by totaling all of the basic probability numbers assigned exactly to that proposition or to some subset of it. Alternatively, we can derive the plausibility of a proposition by totaling all of the basic probability numbers which have any shared elements with that proposition

Example 4.1 Suppose that after a cursory examination of the construction site introduced in example 3.1, we suspected (support of 0.5) that the ground type was rock and we were almost certain (support of 0.9) that it was either rock or soil. This belief could be expressed by the belief function:

$S(\{rock\}) = 0.5$, reflecting some belief specifically in rock

$S(\{soil\}) = 0.0$, reflecting no belief specifically for soil

$S(\{sand\}) = 0.0$, reflecting no belief for sand

$S(\{rock, soil\}) = 0.9$, reflecting strong belief in either rock or soil

$S(\{rock, sand\}) = 0.5$, reflecting belief specifically in rock which implies belief in the superset $\{rock, sand\}$

$S(\{soil, sand\}) = 0.0$, reflecting no specific belief for either soil or sand

$S(\{rock, soil, sand\}) = 1.0$, reflecting full belief that the ground type is either rock, soil or sand

$= S(\Theta_{MATERIAL})$

This belief could also be summarized by the support intervals for the singletons $[S(\{x_i\}), Pl(\{x_i\})]$ (we show in section 11.1 that most bodies of belief can be fully

expressed by giving the belief intervals for the singletons only):

$$\{rock\} : [0.5, 1.0]$$

$$\{soil\} : [0.0, 0.5]$$

$$\{sand\} : [0.0, 0.1]$$

Note that there is no support for the material being either soil or sand; yet we can still use the plausibilities to show that the former is much more likely. We could further convert the information into a basic probability assignment (which we will call $m_1 : 2^{\Theta_{MATERIAL}}$) with the following focal propositions:

$$m_1(\{rock\}) = 0.5,$$

$$m_1(\{rock, soil\}) = 0.4,$$

$$m_1(\{rock, soil, sand\}) = 0.1.$$

We can then show this BPA in diagrammatic form as follows:

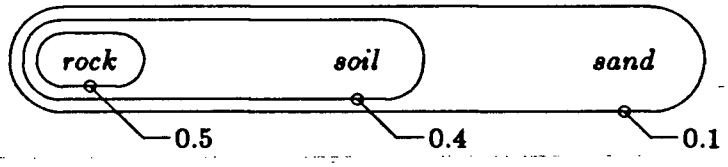


Figure 4.2: BPA #1 in Diagrammatic Form

4.2 Simple and Separable Support Functions

The simplest type of belief function is one in which some belief is assigned to a single proposition (or **focus**) with all of the remaining belief being assigned to the frame Θ itself. Such a

belief function is called a **simple support function** and is defined by the basic probability assignment:

$$0 \leq m(A) \leq 1 \tag{4.12}$$

$$m(\Theta) = 1 - m(A) \tag{4.13}$$

$$m(B) = 0 \tag{4.14}$$

For some frame Θ, some focus $A \subset \Theta$, and all other propositions $B \subset \Theta$.

Since a simple support function with focus A can essentially be described by the one basic probability number $m(A)$, we will call this value the **simple support number,** $ss(A)$.

In the combination of two or more simple support functions both with the same focus A and the basic probability assignments $m_1(A), m_2(A), \ldots$; Dempster's rule of combination (which is discussed in section 5.1) yields the special case:

$$m(A) = 1 - (1 - m_1(A)) \times (1 - m_2(A)) \ldots \tag{4.15}$$

$$m(\Theta) = (1 - m_1(A)) \times (1 - m_2(A)) \ldots \tag{4.16}$$

The result is itself a simple support function with the same focus A. It is interesting to note that this formula matches the rule of evidence combination identified by James Bernoulii in the 16th century.

A simple support function or the combination of two or more simple support functions (not necessarily containing the same focal element) is called a **separable support function**. While not all belief functions that are appropriate for the representation of evidence fall into the class of separable support function, any body of evidence that is obtained by the combination of simple support functions–a situation corresponding to the incremental gathering of single pieces evidence–will yield a separable support function.

In addition to the combination of any group of simple support functions to create one separable support function, it is possible to **decompose** any separable support function into a unique set of simple support functions. While the general formula is slightly more complex and will not be given here, it is important for us to note that there exists a one-to-one correlation between any separable support function and the set of simple support functions into which it would decompose. If we examine the relationship between the separable support function's $m(A)$ value for some focus A and the simple support number $ss(A)$, then we find that both values represent the total evidence in support of exactly A. However $m(A)$ is adjusted to reflect the magnitude of evidence relative to the amount of evidence for all other propositions while $ss(A)$ carries no indication of its relative support–we need to consider the simple support functions for all propositions in order to observe the impact of relative degrees of support.

4.3 Consistent Belief

In the case of a frame refinement $\omega : 2^{\Theta} \rightarrow 2^{\Omega}$, we can say that the two support functions $S_o : 2^{\Theta}$ and $S : 2^{\Omega}$ are **consistent** (i.e. they do not represent conflicting bodies of knowledge) if $S_o(A) = S(\omega(A))$, for all $A \subset \Theta$. When this equality holds, we refer to S_o as a **restriction** of S to Θ, or $S|2^{\Theta}$; likewise the corresponding basic probability assignment $m_o : 2^{\Theta}$ is a restriction of $m : 2^{\Omega}$ to Θ or $m|2^{\Theta}$. Given such a basic probability assignment m_o, we can calculate a corresponding basic probability assignment m which meets the restriction criteria ($m|2^{\Theta}$) by setting:

$$m(\omega(A)) = m_o(A),\ \forall A \subset \Theta \tag{4.17}$$

Thus $m : 2^{\Omega}$ will represent the same body of belief as $m_o : 2^{\Theta}$ but it will be discerned on the new frame of discernment.

In contrast to frame refinement, we may have performed the frame coarsening procedure given in equation 3.5; in which case we can derive a coarser basic probability assignment $m_o : 2^{\Theta}$ which is consistent with some finer basic probability assignment $m : 2^{\Omega}$ from the formula:

$$m_o = \sum_{\substack{B \subset \Omega \\ A = \omega(B)}} m(B) \tag{4.18}$$

Example 4.2 Suppose that we refine the frame described in the previous examples to distinguish between the bed rock and aggregate (as in example 3.2). We can use equation 4.17 to calculate a basic probability assignment $m_2 : 2^{\Omega_{MATERIAL}}$ which conveys the identical belief as $m_1 : 2^{\Theta_{MATERIAL}}$ (i.e. it is a restriction $m_2|\Theta_{MATERIAL}$):

$$m_2(\{bed\ rock, aggregate\}) = 0.5,$$

$$m_2(\{bed\ rock, aggregate, soil\}) = 0.4,$$

$$m_2(\{bed\ rock, aggregate, soil, sand\}) = 0.1.$$

4.4 Specifying Support Functions

We would like to find a way of specifying support functions which allows the user to enter functions into an expert system in a way which is simple, expressive and theoretically correct. Any general support function can be expressed by providing the degree of belief for every proposition in a frame of discernment. While this approach enables any support function to be

entered into the system, the specification of functions in this manner is typically an onerous task. This is because of the large number of propositions in a frame of discernment (for a frame with n elements, there are 2^n propositions). We have determined that specifying the support for the focal propositions only is sufficient to completely specify any support function, as shown in the following proof:

Theorem 4.1 Proof that specifying only the support values for the focal propositions is sufficient to fully specify a support function:

A basic probability assignment can be specified by providing:

$$m(A), \ \forall A \subseteq \Theta \tag{4.19}$$

where $m(A)$ is the basic probability number for any proposition A in a frame of discernment Θ. (note that for this theorem only we distinguish between proper subsets, $\subset$, and improper subsets, $\subseteq$). More succinctly, we need only provide:

$$m(F), \ \forall F \subseteq \Theta \tag{4.20}$$

where F is any focal proposition in Θ. This is because the basic probability numbers for any propositions not specifically provided could be assumed to be zero. An input system could confirm when a complete set of values had been received by checking that the specified values summed to one.

Now equation 4.1 *gives:*

$$S(A) = \sum_{B \subseteq A} m(B)$$

but $m(A) = 0$ for all non-focal propositions, so we can say:

$$S(A) = \sum_{F \subseteq A} m(F) \tag{4.21}$$

where F is any focal proposition. Furthermore, we can say:

$$S(A) = m(A) + \sum_{F \subset A} m(F) \tag{4.22}$$

or:

$$m(A) = S(A) - \sum_{F \subset A} m(F) \tag{4.23}$$

or, for any focal proposition G:

$$m(G) = S(G) - \sum_{F \subset A} m(F) \tag{4.24}$$

Thus for any focal proposition G, we can determine $m(G)$ from $S(G)$ and the basic probability numbers of true focal subsets of G. Furthermore, the lowest cardinality propositions in any belief function have no focal subsets and their basic probability numbers can thus be derived directly from $S(G)$. It is therefore evident that equation 4.24 can be used to evaluate $m(G)$ for every focal proposition G from only the focal degrees of belief by employing a "bottom-up" approach. That is, evaluate $m(G)$ for the lowest propositions first and then progress to the higher ones.

Once we have all of the $m(G)$ values, we can obtain all $m(A)$ values where A is any proposition in Θ, because $m(A) = 0$ for all non-focal A's. Further, the set of all $m(A)$'s can be used to derive all $S(A)$'s from equation 4.1. Thus the information contained in the support values for focal propositions only is sufficient to define the entire corresponding support function.

An improvement to the input approach of specifying support values for *every* proposition in a frame, then, is to supply the support for the focal propositions only. Section 11.1 will introduce

a scheme which allows belief input to be simpler still.

4.5 Classes of Belief

Some classes of belief functions which appear to be pertinent to conveying belief in many practical contexts are described below. The minimum inputs, necessary constraints, and the support and plausibility implications are described for each class.

4.5.1 Ignorance

Ignorance can arise in response to a request for belief on some specific question when the user has no information to offer. This corresponds to a response of "unknown" in a typical expert system. The appropriate response by the user is to select the vacuous support function in which the frame of discernment Θ receives full support ($S(\Theta)$ and $Pl(\Theta) = 1.0$) while all other propositions receive zero support and full plausibility ($S(A) = 0.0$, $Pl(A) = 1.0$). The BPA corresponding to a vacuous support function consists of a single basic probability number of 1.0 assigned to Θ.

This response would be constrained so that:

$$m(\Theta) = 1.0,$$
$$m(A) = 0 \text{ for all other } A \subset \Theta$$

In Bayesian methods an assignment of equal positive probabilities to all elements, i.e. a uniform prior distribution, is often used as an expression of ignorance, but this does not necessarily constitute an informationless input. The vacuous support function avoids an assignment of support to any specific elements but it does give full support to the idea that the true event lies within the frame Θ.

For example, a construction engineer may be asked to express his belief in the delivery performance of a supplier. If the frame Θ consists of the three elements:

$$\Theta_{PERFORMANCE} = \{On\ time,\ Within\ 1\ week,\ Within\ 1\ to\ 2\ weeks\}$$

then, in expressing ignorance of the supplier's performance and returning a vacuous support function, he is tacitly agreeing that this supplier can deliver neither early nor later than 2 weeks. The plausibility of each of the three delivery possibilities is thus 1.0. A more complete expression of ignorance would be possible if the frame of discernment had also included the element *"early, or later than 2 weeks"* and the vacuous support function was again entered.

4.5.2 Certain Belief

If the user wishes to express perfect knowledge or an absolute conviction, for example in response to a question concerning the number of floors in a high rise building, then this can be reflected by a support of 1.0 for the one singleton representing the truth.

If the true element is x_i then this certain response would be:

$$S(\{x_i\}) = 1.0$$

This belief response would imply that the support and plausibility for the certain element x_i are both 1.0 and that the supports and plausibilities for all other singletons are zero.

The input constraints would be:

$$m(\{x_i\}) = 1.0,$$
$$m(A) = 0.0 \text{ for all } A \neq \{x_i\}.$$

A similar situation is when the user has perfect knowledge that the truth lies within a known subset of elements, but has no knowledge as to which of the elements within that subset are true. Thus, if he wished to express the fact that a crew size for a particular job must include

either 8, 9 or 10 workers while the frame of discernment invites belief response for possible crew sizes of 5, 6, 7, 8, 9, 10, 11 or 12 then, by assigning a support of 1.0 to the proposition *{8,9,10}*, he provides certain support for a crew size in the range of 8 to 10 workers.

Once the proposition A_i for which we have certain belief is established, we would set the support and the plausibility of A_i to 1.0. Then we would have $S(\{x_i\}) = 0$, $Pl(\{x_i\}) = 1.0$ for all $x_i \in A_i$, while $S(\{x_j\})$ and $Pl(\{x_j\}) = 0$ for all $x_j \notin A_i$.

The input constraints would be:

$$m(A_i) = 1.0,$$
$$m(A) = 0.0 \text{ for all other } A.$$

4.5.3 Bayesian Belief

Bayesian belief parallels the one support function which can be represented with a conventional discrete probability distribution. The support values assigned to the singletons must sum to 1.0. In this support function, the support equals the plausibility for each singleton, i.e.:

$$m(\{x_i\}) = S(\{x_i\}) = Pl(\{x_i\}), \text{ for all } x_i \in \Theta$$

The constraint set for this class of support function is:

$$m(\{x_i\}) \geq 0.0, \text{ for all } x_i \in \Theta$$
$$\sum_{x_i \in \Theta} m(\{x_i\}) = 1.0$$
$$m(A) = 0.0, \text{ for all non-singleton propositions } A$$

4.5.4 Consonant Belief

The Bayesian support function partitions belief into parcels which are assigned to separate elements. This implies that the evidence upon which the belief is based is able to simultaneously

support a number of disparate and possibly conflicting conclusions. The quality of evidence which leads to conflicting conclusions might well be suspect. Much greater credibility might be attached to evidence which, although not precise enough to support just a single conclusion, was able to support a number of conclusions which were in general agreement with each other. The belief derived from evidence of this type is described by the consonant support structure. A consonant support structure is one in which the elements can be ordered so as to obtain:

$$S(\{x_1\}) \leq S(\{x_1, x_2\}) \leq \ldots \leq S(\{x_1, x_2, \ldots x_n\})$$

The input response required to enter a consonant support function requires specification of the support for the single element x_1 together with the plausibility values for all elements.

While some have suggested that this is the only kind of evidence and support structure which should be entertained (see Cohen [9] and Shackle [56]), others have expressed the view that such a position is unrealistic. Regardless of the relative merits of these arguments, the consonant support structure does have considerable appeal in engineering situations, and it is also attractive from the user input standpoint as it is simple to specify and highly constrained.

In engineering applications, belief must often be expressed in the value of a variable which exists on, and is normally measured on, a continuum. In such a case the elements of the frame of discernment would represent a discretization of this variable. If consonant belief applied, it would be reasonable in most situations to expected that it would reflect the ordering of the elements along the scale. This would be achieved by constraining the distribution of plausibilities to the ordered elements to be unimodal with the mode occurring at the single element to which support is assigned (i.e. in the above notation, the mode would occur at x_1, and the labelling of the x's on the continuum is not necessarily contiguous).

4.5.5 Belief Arising from Experiments

In a conventional application of Bayes equation, a prior discrete probability distribution is specified for some uncertain parameter. This is combined with information concerning an investigation or experiment, expressed in the form of sample likelihoods, and the outcome of the investigation or experiment. The sample likelihoods are expressed as probabilities of obtaining the outcome conditional upon being given each of the possible values for the unknown parameter. The combining operation, using Bayes' equation, produces a revised, or posterior, probability distribution for the uncertain parameter.

In the D-S scheme, sample likelihoods can be emulated by adopting a frame of discernment with elements corresponding to the possible discretized parameter values. A consonant support function is adopted. In an engineering context we feel that it is appropriate to refer to this as an experimental support function.

Support for the nested subsets are related to the likelihoods by Shafer ([57]) according to:

$$S(A) = 1 - \frac{\max\limits_{x_i \in \overline{A}} p[e|x_i]}{\max\limits_{x_i \in \Theta} p[e|x_i]} \qquad (4.25)$$

where $p[e|x_i]$ represents the likelihood of the experimental result e given the parameter value represented by the element x_i.

The corresponding BPA values for the focal propositions are then derived by successive subtraction.

Once an experimental support function had been selected the input would be requested in the form of a set of sample likelihoods. These would not be subject to any input constraints. All other appropriate constraining effects would occur automatically when the above equation is

applied.

When an experimental support function is combined with a prior belief, which is a Bayesian support function, then the resulting posterior belief values for the elements of Θ will be numerically identical to the conventional Bayesian posterior probabilities. It is not necessary in the D-S scheme, however, to restrict the prior belief in this fashion and it is likely in most situations that other than a Bayesian support function would more realistically describe the prior knowledge.

Chapter 5

D-S Theory III: Belief Combination

This chapter deals with the combination of two or more belief functions. Dempster's rule of combination is presented first, followed by a discussion of alternative belief combination routines which have been developed for cases in which Dempster's rule is not applicable.

5.1 Dempster's Rule

Dempster's rule of combination provides a method of combining two independent bodies of belief, (e.g. belief functions from two independent, equally credible experts) to get an aggregated basic probability assignment or **orthogonal sum**. The orthogonal sum of the two BPA's, $m_1 : 2^\Theta$ and $m_2 : 2^\Theta$, for example, is $m_1 \oplus m_2 : 2^\Theta$, and is defined by:

$$m_1 \oplus m_2(\emptyset) = 0 \tag{5.1}$$

$$m_1 \oplus m_2(A) = (1-k)^{-1} \sum_{A_i \cap B_j = A} m_1(A_i) \cdot m_2(B_j) \tag{5.2}$$

$$\text{where } k = \sum_{A_i \cap B_j = \emptyset} m_1(A_i) \cdot m_2(B_j) \tag{5.3}$$

The orthogonal combination procedure may try to attribute some belief to the empty set $\emptyset$. However the belief in $\emptyset$ must be zero by definition (since the set of singletons is exhaustive, the

truth must be represented by one of them and not $\emptyset$). The equation therefore sets $m(\emptyset)$ to zero and normalizes all of the other basic probability numbers using the factor k defined above. This factor also enables us to measure the degree to which the functions being combined represent conflicting evidence. For this purpose, the **weight of conflict** between m_1 and m_2 can be defined as follows:

$$\text{weight of conflict} = -\log(1-k) \tag{5.4}$$

The rule can be used to sequentially perform the combination of any number of basic probability assignments; in which case the order of combination is unimportant. The following example illustrates Dempster's combination in diagrammatic form.

Example 5.1 If the BPA shown in figure 4.2 is combined with a BPA which attributes most of its belief to a ground type of either soil or sand:

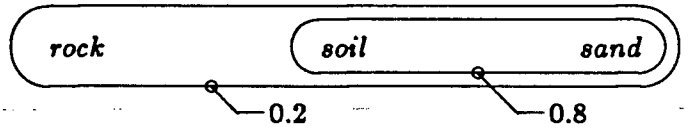


Figure 5.1: BPA #2

The following BPA is obtained:

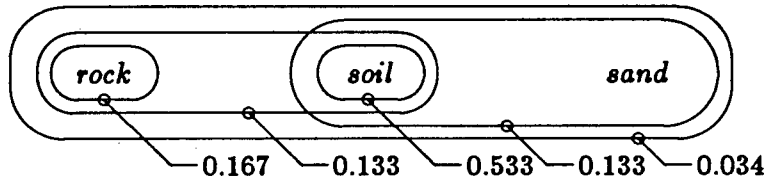


Figure 5.2: The Result of Combining BPA #1 and BPA #2

5.2 Other Combination Rules

In D-S, two belief functions can be combined using Dempster's rule of combination. Dempster's rule is intended for combining two belief functions which are independent–they arise out of completely distinct bodies of evidence–and conjunctive–both are thought to be true, even if they are partially contradictory to each other. As such, Dempster's rule amplifies belief which is common to the two input belief functions and attenuates belief which is exclusive to a single belief function. Dubois and Prade [15] have proven the unicity of Dempster's rule up to the normalization stage for the conjunctive combination of independent belief functions. However within a general D-S implementation, some combinations may not be conjunctive and some belief functions may not be independent; either condition renders Dempster's rule invalid. Therefore the identification of alternative combination rules which apply under different conditions, but which are still consistent with D-S, is highly desirable (see Dubois and Prade [15] and Yager [75] for further discussion of alternative combination schemes). Several rules have been developed during this research project which address these conditions. While these rules have been tested under a variety of combination scenarios and are believed to be valid for the applications proposed herein, no formal mathematical proofs are provided. This section introduces these rules and subsequent sections illustrate their use.

A dependant conjunctive combination deals with two different belief functions which stem from the same source of evidence and therefore provide two different representations of the same information. For the application intended here (described in section 10.4.2), both belief functions can be taken to be valid representations of the single body of evidence, thus the belief functions should not contradict each other. According to this usage of "valid," for example, one belief function cannot display full support (1.0) for singleton $\{a\}$ while the other function exhibits full support for singleton $\{b\}$. Yet the application proposed here (and the definition of a valid representation of the evidence) allows either belief function to contain more uncertainty than the evidence warrants–that is, either function may be unduly vague or imprecise. This vagueness is manifested in belief being assigned upwards to a superset of some proposition A while, according to the evidence, it is A itself which most deserves the support. For example, one belief function might show full support for singleton $\{a\}$ while the other function indicates full support for the proposition $\{a, b\}$ (in which case the former allocation of belief as being would be referred to as more precise).

Given two valid but possibly vague and dependent input belief functions, it can be observed that the level of support for any proposition will never be too high but can be too low. It can also be observed that since the two input functions are dependent, belief in some proposition should not be increased just because it is common to the two functions; this would amount to double-counting the effects of the evidence. Rather, the belief should be set to be equal to the more precise of the two input belief functions since the assumption that the input functions are completely valid but potentially vague leaves the assessment of their relative quality to be based on their precision alone; with greater precision being taken as better. Thus for two dependent belief functions S_1 and S_2 on frame Θ, the appropriate operation to produce a combined support $S(A)$ for some proposition A might be expected to be as follows:

$$S(A) = \max\left(S_1(A), S_2(A)\right), \quad \forall A \subset \Theta \tag{5.5}$$

However this operation does not necessarily produce a legitimate belief function as output. For example, when this operation is used to combine the following two belief functions which are known to have originated from the same source of evidence:

$$S_1(\{a\}) = 0, \quad S_1(\{b\}) = 0, \quad S_1(\{c\}) = 0,$$
$$S_1(\{a,b\}) = 1.0, \quad S_1(\{a,c\}) = 0, \quad S_1(\{b,c\}) = 0,$$
$$S_1(\{a,b,c\}) = 1.0$$

$$S_2(\{a\}) = 0, \quad S_2(\{b\}) = 0, \quad S_2(\{c\}) = 0,$$
$$S_2(\{a,b\}) = 0, \quad S_2(\{a,c\}) = 0, \quad S_2(\{b,c\}) = 1.0,$$
$$S_2(\{a,b,c\}) = 1.0$$

The resulting combined belief function is:

$$S(\{a\}) = 0, \quad S(\{b\}) = 0, \quad S(\{c\}) = 0,$$
$$S(\{a,b\}) = 1.0, \quad S(\{a,c\}) = 0, \quad S(\{b,c\}) = 1.0,$$
$$S(\{a,b,c\}) = 1.0$$

This is not an allowable belief function according to Shafer's formal definition [57, p.39]. The correct result must be as follows:

$$S(\{a\}) = 0, \quad S(\{b\}) = 1.0, \quad S(\{c\}) = 0,$$
$$S(\{a,b\}) = 1.0, \quad S(\{a,c\}) = 0, \quad S(\{b,c\}) = 1.0,$$
$$S(\{a,b,c\}) = 1.0$$

The error in equation 5.5 can be traced to the fact that the support value for a proposition reflects the belief which is attributed exactly to that proposition or to some subset of that proposition. That is, the support $S(A)$ for some proposition A is obtained from the basic probability assignment m according to the following formula:

$$S(A) = \sum_{B \subset A} m(B) \tag{5.6}$$

Thus when vagueness in both input belief functions causes belief which should be assigned to some proposition A to be re-assigned to supersets of A, the support values for A in both input functions contain *no evidence of that belief.* Consequently, the application of equation 5.5 cannot show that A is the more precise origin of that belief. A solution, then, is to base the operation on the commonality number, $Q(A)$, of a proposition which reflects the belief attributed exactly to that proposition or to some superset of that proposition. That is:

$$Q(A) = \sum_{\substack{B \subset \Theta \\ A \subset B}} m(B) \tag{5.7}$$

Belief which is assigned to a superset of some proposition A, then, is fully reflected in the commonality number of A. Thus when a comparison of the two input belief functions shows that some belief has been assigned to unduly vague propositions (supersets of A), the commonality numbers will lead to A as the more precise origin of that belief. It can be further noted that a conjunctive combination operation should *minimize* the commonality numbers of two dependant belief functions. This is because excessive imprecision causes belief to be re-assigned upwards to larger propositions, thereby allowing this belief to contribute to the commonality numbers of more propositions. Vagueness therefore leads to larger commonality numbers, and the more precise of two dependant input functions is obtained by minimizing the commonality numbers. Using a commonality number representation therefore, the conjunctive combination of the two dependent belief functions Q_1 and Q_2 on frame Θ is correctly expressed by the

following formula:

$$Q(A) = \min\Big(Q_1(A), Q_2(A)\Big), \quad \forall\, A \subset \Theta \tag{5.8}$$

This formulation may be used to combine the following two belief functions which are the commonality equivalent of the support functions given in the example described above:

$$\begin{aligned} Q_1(\{a\}) = 1.0, \quad & Q_1(\{b\}) = 1.0, \quad Q_1(\{c\}) = 0, \\ Q_1(\{a,b\}) = 1.0, \quad & Q_1(\{a,c\}) = 0, \quad Q_1(\{b,c\}) = 0, \\ & Q_1(\{a,b,c\}) = 0 \end{aligned}$$

$$\begin{aligned} Q_2(\{a\}) = 0, \quad & Q_2(\{b\}) = 1.0, \quad Q_2(\{c\}) = 1.0, \\ Q_2(\{a,b\}) = 0, \quad & Q_2(\{a,c\}) = 0, \quad Q_2(\{b,c\}) = 1.0, \\ & Q_2(\{a,b,c\}) = 0 \end{aligned}$$

producing the following commonality function:

$$\begin{aligned} Q(\{a\}) = 0, \quad & Q(\{b\}) = 1.0, \quad Q(\{c\}) = 0, \\ Q(\{a,b\}) = 0, \quad & Q(\{a,c\}) = 0, \quad Q(\{b,c\}) = 0, \\ & Q(\{a,b,c\}) = 0 \end{aligned}$$

which is the desired result.

In contrast to the conjunction combination case in which both input belief functions are thought to be valid, a case will be demonstrated in section 10.3.2 in which one of two belief functions is believed to bear on the situation at hand, though it is not know which one. Under these

circumstances, one can only justifiably assert a combined belief function that reflects the belief which is common to both of the input belief functions. Thus the combined support for some proposition should be taken as the minimum of the support assigned to that proposition by the two input functions. Following a similar line of reasoning to that used for the conjunctive case, it can be shown here that support numbers may be used for the operation whereas commonality numbers may not. The disjunctive combination of the two input belief functions S_1 and S_2 on frame Θ is therefore given by the following formula:

$$S(A) = \min\left(S_1(A), S_2(A)\right), \quad \forall A \subset \Theta \tag{5.9}$$

It can be further noted that the plausibility value for some proposition reflects all belief which is not assigned to the complement of that proposition or to any subset of the complement. That is, plausibility reflects the belief in both the subsets and the supersets of the proposition and therefore provides the basis for alternative combination formulations which are comparable to equations 5.8 and 5.9. It can be observed that for two dependant belief functions, the more vague function will have equal or higher plausibility values. For example, a belief function which represents complete ignorance has a plausibility of 1.0 for all propositions; while a belief function which does not reflect complete ignorance exhibits plausibilities of less than 1.0 for those propositions which have some evidence against them. For two input belief functions Pl_1 and Pl_2 on frame Θ, then, an alternative form of the **dependent conjunctive** combination (equation 5.8) which also produces legitimate results is obtained by taking the minimum of the plausibility values to obtain the more precise belief, as follows:

$$Pl(A) = \min\left(Pl_1(A), Pl_2(A)\right), \quad \forall A \subset \Theta \tag{5.10}$$

A fully equivalent alternative form of the **disjunctive** combination (equation 5.9) is defined by taking the maximum of the plausibilities to obtain the more vague belief:

$$Pl(A) = \max\left(Pl_1(A), Pl_2(A)\right), \quad \forall A \subset \Theta \tag{5.11}$$

Note that equation 5.11 is equivalent to the following equation:

$$1 - S\left(\overline{A}\right) = \max\left(1 - S_1\left(\overline{A}\right), 1 - S_2\left(\overline{A}\right)\right), \quad \forall A \subset \Theta \tag{5.12}$$

where S, S_1, and S_2 are the support function equivalents to Pl, Pl_1 and Pl_2 respectively and $\overline{A}$ is the set complement of A. Further, equation 5.12 is equivalent to the following equation:

$$S\left(\overline{A}\right) = \min\left(S_1\left(\overline{A}\right), S_2\left(\overline{A}\right)\right), \quad \forall A \subset \Theta \tag{5.13}$$

which is equivalent to equation 5.9 since both equations maximize the support for every proposition in the frame. This proves the equivalence of equations 5.9 and 5.11.

Finally, note that the combination of several dependent input belief functions can be performed through the successive application of any of the above combination formulae. For example, the conjunctive combination of the three dependent belief functions Q_1, Q_2 and Q_3 on frame Θ can be accomplished by the following operation:

$$Q(A) = \min\left(\min\left(Q_1(A), Q_2(A)\right), Q_3(A)\right), \quad \forall A \subset \Theta \tag{5.14}$$

Chapter 6

D-S Theory IV: Inference

This chapter provides a brief example of how D-S theory can be used to perform inference. Much of the ensuing chapters deals with work which was performed before this methodology of performing consistent inference was known. Yet the technique is presented here so that the early work can be compared with this technique.

6.1 Performing Inference in D-S theory

Suppose that we have a frame Θ_A with elements a_i and propositions A_i and a second, related frame Θ_B with elements b_j and propositions B_j. If a belief function which describes some body of evidence (an **evidential belief function**) is entered for the frame Θ_A, one might wish to determine the impact of this knowledge on the propositions of the second frame Θ_B. This can be done by creating the minimal refinement of the two frames, $\Theta_A \otimes \Theta_B$ or $\Theta_{(A,B)}$, which incorporates all of the elements from both initial frames. An element (a, b) in the minimal refinement frame $\Theta_{(A,B)}$ corresponds to the concept that the element a is the true value of frame Θ_A while the element b is the true value of frame Θ_B. Likewise, a proposition (A, B) in the frame $\Theta_{(A,B)}$ is true if the proposition A in frame Θ_A is true and B in frame Θ_B is true.

It was stated in section 3.2 that the minimal refinement consists of only those pairs of elements which can logically coexist. In order to simplify the process, the assumption will be made here that the two initial frames are fully independent, and that minimal refinement is simply the cross product of the two frames. Later, a **relational belief function** will be added to the

minimal refinement which states categorically that those pairs which cannot logically co-exist are impossible. Thus the minimal refinement defined in section 3.2 and the fully independent minimal refinement with the added relational belief function described here are functionally equivalent.

The next step is to extend the evidential belief function associated with frame Θ_A to the frame $\Theta_{(A,B)}$. That is, every basic probability number attributed to a proposition A is assigned to the corresponding proposition in the frame $\Theta_{(A,B)}$ (recall that the elements of Θ_A are fully discerned by $\Theta_{(A,B)}$). The extended belief function is then the restriction of the initial evidential belief function to Θ_A, as defined by equation 4.17.

This extension of the evidential belief function is then combined, using Dempster's rule, with a relational belief function discerned on frame $\Theta_{(A,B)}$. The relational belief function embodies the inference relationships between the elements in Θ_A and Θ_B. These relational belief functions will be discussed in more detail in section 11.2. The belief function resulting from the combination now incorporates both the evidential belief about the Θ_A elements and the implied belief about the Θ_B elements.

Finally, the resulting belief function is coarsened or **marginalized** from the frame $\Theta_{(A,B)}$ to the frame Θ_B, producing a new evidential belief function on Θ_B (as in equation 4.18).

Thus, using only the tools of the D-S theory, belief concerning elements in the frame Θ_B has been inferred from belief about the elements in frame Θ_A and belief about the relationships between the two frames. Note that in the context of an expert system, the evidential belief in frame Θ_A would be input from the user, the relational belief would be stored by the knowledge engineer in the system's knowledge base, the propagation procedure would be performed by the system's inference engine (possibly with the guidance of a "control knowledge base" which defines which propagations to make in various situations) and the resulting evidential belief in frame Θ_B would be either output to the user, or an intermediate point for further propagation. Example 6.1 illustrates the belief propagation process.

Example 6.1 Suppose that we have a frame which discerns the type of ground found on a construction site where the ground could be either rock or soil:

$$\Theta_{MATERIAL} = \{rock, soil\}$$

A second frame discerns the excavation method that we will use on the project, either blasting or digging:

$$\Theta_{METHOD} = \{blast, dig\}$$

These two frames would yield the minimal refinement:

$$\Theta_{(MATERIAL,METHOD)} = \{(rock, blast), (rock, dig), (soil, blast), (soil, dig)\}$$

The minimal refinement completely contains both initial frames. For example the proposition $\{(rock, blast), (rock, dig)\}$ in the frame $\Theta_{(MATERIAL,METHOD)}$ is equivalent to the proposition $\{rock\}$ in the frame $\Theta_{MATERIAL}$. Likewise, the proposition $\{(rock, blast), (soil, blast)\}$ in the frame $\Theta_{(MATERIAL,METHOD)}$ corresponds to the proposition $\{blast\}$ in the frame Θ_{METHOD}.

D-S inference, or belief propagation, for these frames is illustrated in figure 6.1.

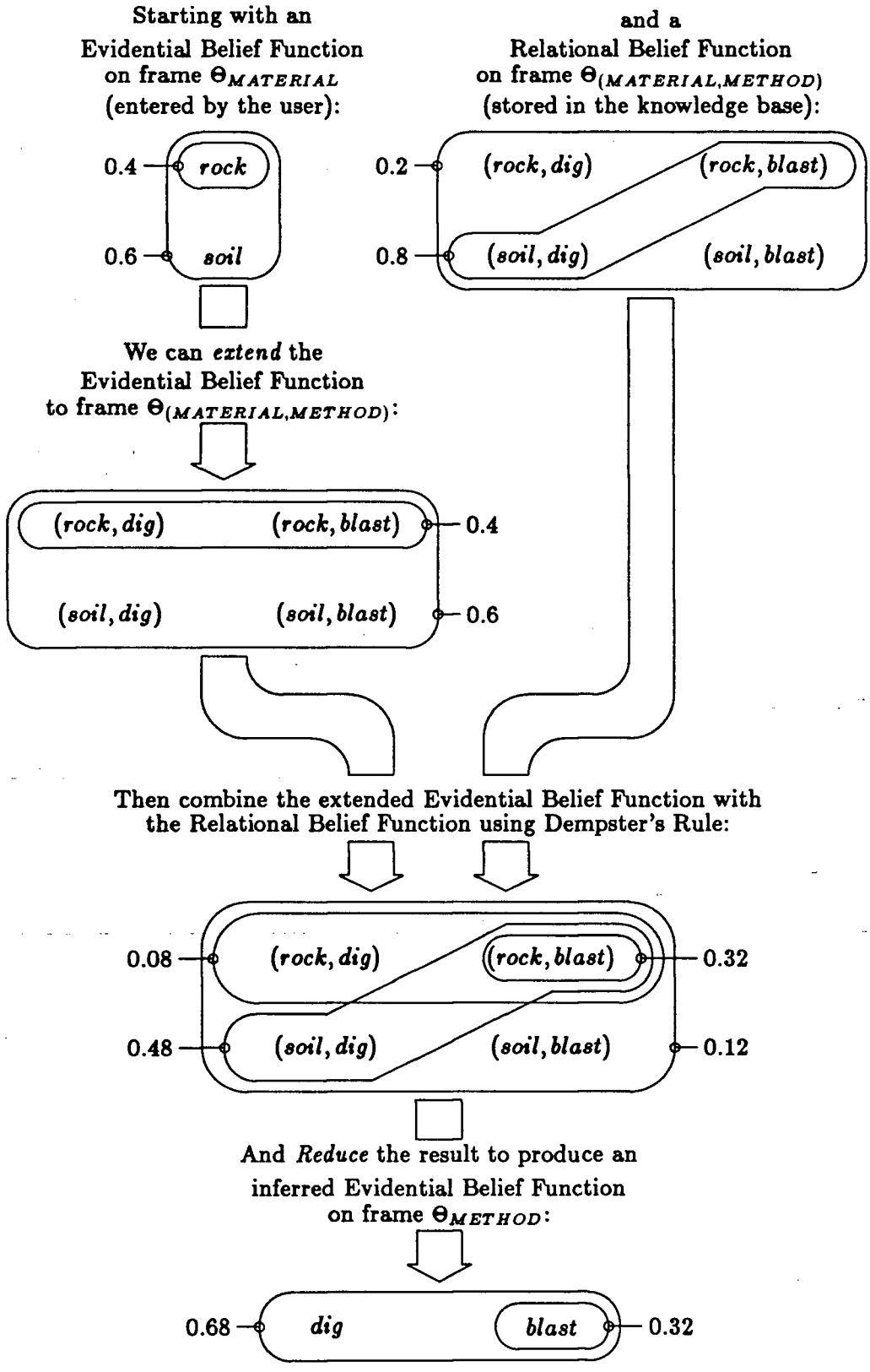


Figure 6.1: Summary of Belief Propagation for an Excavation Example

Chapter 7

Various D-S Interpretations

One of the first references to D-S theory in an artificial intelligence context was in 1981 at the Seventh Joint International Conference on AI in Vancouver, B.C. (IJCAI-81). Since that time, a good deal of work has focused on the theory and on how to implement it. This chapter briefly reviews a few of the important papers published in this field by others.

7.1 Barnett

Jeffrey Barnett presented a paper at IJCAI-81 entitled "Computational Methods for a Mathematical Theory of Evidence" which dealt exclusively with D-S Theory. There were two main points to Barnett's paper; the first was simply to present D-S theory as a possible scheme for dealing with uncertainty. The second point was to modify the process in an attempt to change the processing time for reasoning problems from a duration that is exponentially dependant on the size of the problem to one that is linear.

Barnett showed that the calculation of both the support and the plausibility numbers from a basic probability assignment as well as the combination of basic probability assignments all require processing time exponential with the size of the frame of discernment. He produced variations which he claimed reduce these to linear, however to do this he restricts the allowable propositions to only singletons, the negation of singletons and the full frame itself.

Several criticisms can be directed against such an approach. First, the computational problem does not seem to be as bad as Barnett suggests, as discussed in sections 9.8 and 9.9. Second,

the restrictions that Barnett places on the theory severely limit the power of the system's representation scheme and almost totally remove that ability to deal with belief of various precision. Third, the restrictions seem to lead to violations of the theory's underlying axioms. For example; suppose we have a frame $\{a, b, c, d\}$, a basic probability assignment that assigns belief only to the proposition NOT $\{a\}$ or $\{b, c, d\}$ and a second basic probability assignment that assigns belief only to NOT $\{d\}$ or $\{a, b, c\}$. Then the legitimate combination of the two basic probability assignments would result in all belief being assigned to the intersection of these two sets–which turns out to be $\{b, c\}$, a set which is not allowed according to the restrictions.

7.2 Garvey, Lowrance and Fischler

Another paper presented at IJCAI-81 which dealt with D-S Theory was "An Inference Technique for Integrating Knowledge from Disparate Sources" by T. Garvey, J. Lowrance and M. Fischler. This paper, like that of Barnett, is largely devoted to a good, simple explanation of portions of the D-S theory. It then goes on to give a number of allowable inference rules; for example:

Theorem 7.1 *Given a support interval for proposition A of* $[S(A), Pl(A)]$ *and a support interval for proposition B of* $[S(B), Pl(B)]$, *we can infer a support for proposition* $(A \,\&\, B)$ *of* $[S(A \,\&\, B), Pl(A \,\&\, B)]$ *from the following equations:*

$$S(A \,\&\, B) = \max\Big(0,\ S(A) + S(B) - 1\Big) \tag{7.1}$$

$$Pl(A \,\&\, B) = \min\Big(Pl(A),\ Pl(B)\Big) \tag{7.2}$$

These equations can be shown to be clearly incorrect. For example, consider a frame of discernment $\{a, b, c\}$. Let the proposition $\{a, b\}$ be A and the proposition $\{a, c\}$ be B. Then the proposition $\{a\}$ is $A \cap B$ (or $A \,\&\, B$). Thus we can see that:

$$S(A \cap B) = S(\{a\}) = m(\{a\}) \tag{7.3}$$

However equation 7.1 states that:

$$\begin{aligned}
S(A \cap B) &= \max\Big(0,\ S(A) + S(B) - 1\Big) \\
&= \max\Big(0,\ S(\{a,b\}) + S(\{a,c\}) - 1\Big) \\
&= \max\Big(0, \quad \Big(m(\{a,b\}) + m(\{a\}) + m(\{b\})\Big) \\
&\qquad + \Big(m(\{a,c\}) + m(\{a\}) + m(\{c\})\Big) \\
&\qquad - \Big(m(\{a\}) + m(\{c\}) + m(\{b\}) \\
&\qquad\quad + m(\{a,b\}) + m(\{a,c\}) + m(\{b,c\}) + m(\{a,b,c\})\Big) \Big) \\
&= \max\Big(0,\ m(\{a\}) - m(\{b,c\}) - m(\{a,b,c\})\Big) \\
&\neq m(\{a\}) \qquad\qquad \text{(as per equation 7.3)}
\end{aligned}$$

Equation 7.1, then, is incorrect.

These equations are valid, however, when they are not taken as equalities (as stated) but rather as inequalities (e.g. $S(A \,\&\, B) \geq \max(0,\ S(A) + S(B) - 1)$). The paper does not go on to show clearly how such statements can be used in a reasoning system and furthermore, it seems that they could only be of use when complete basic probability assignments are not available in the first place (there is no apparent benefit of allowing fragments of basic probability assignments). The final portion of the paper by Garvey et al works through an example of using D-S theory to combine different types of evidence about an electronic signal in order to determine which of several possible sources it was emitted from. This example incorporates "dependency graphs" which are equivalent to the minimal refinement of all input frames. This example is one of the few cases in the literature which proceeds in a manner which is truly consistent with Shafer, rather than reaching outside of D-S theory for reasoning procedures. This paper, then, contains useful results and provides the first glimpse of a consistent D-S approach; but it makes little progress towards a general implementation technique of D-S theory for expert systems.

7.3 Gordon and Shortliffe

One of the first attempts to try to create a simple implementation of the D-S theory in a rule-base expert system was made by Gordon and Shortliffe who looked at the theory for improving the MYCIN system described in section 2.2 (see Buchanan and Shortliffe [4, Ch.13]). In their approach, a single frame of discernment was constructed from a list of all of the diseases known to the system. Rules were then constructed exactly as in their initial system, including the formulas for conjunctions and disjunctions in the premise and for weak inference. The difference was that the resulting certainty factors were now taken to be basic probability numbers. Dempster's Rule was then used for combining these bodies of evidence. In this system, Barnett's simplifying assumptions were adopted.

Later, Gordon and Shortliffe [27] dropped Barnett's schemes and revised the system to deal with an entire frame of discernment rather than just the singletons and their complements. However they made a different restriction that only those propositions which formed a true hierarchy or a tree instead of a lattice could be used. These restrictions alter rather than solve the problems arising from Barnett's schemes since legitimate combinations still lead to illegal propositions. This problem is dealt with by making approximations which remove the belief from any such illegal propositions but, as a result, the ability to accurately calculate the plausibility of propositions is lost. Also, Shafer and Logan [65] later pointed out that if the discernment space is, in fact, truly hierarchical, then Dempster's rule is much more efficient than in the general case anyway, so Gordon and Shortliffe's approximation technique is not necessary.

While Gordon and Shortliffe's techniques are consistent with the D-S theory (except for the restrictions mentioned above), they deal only with combining multiple sources of evidence regarding possible conclusions. In no way do they mention how these sources of evidence are arrived at in the first place. In fact, the impression is given that the sources of evidence are the results of applying rules in exactly the same manner as certainty factors. If this is the intended

approach, it marks the genesis of a very important class of D-S implementation approaches–namely **non-consistent implementations**. Such implementations use portions of the D-S knowledge representation scheme and often employ Dempster's rule for combining multiple sources of belief. But the techniques used for inferring belief (using rules) and performing weak inference (adding uncertainty to the rules themselves) are founded wholly *outside* of D-S theory. This class of D-S implementations is useful because it generally results in systems which are quite simple and practically implementible and their results are at least as good (most likely better) than those of simple certainty factors. However they miss the very important mark of providing systems which are fully consistent with a single reliable underlying theory, and thus cannot offer the reliability and confidence of truly consistent approaches. The system developed by Lu and Stehanou (described in the next section) is a typical non-consistent implementation.

7.4 Lu and Stehanou

Lu and Stehanou [44] have proposed a system which uses D-S frames to represent so-called *input* and *output* spaces. Inference is accomplished by defining *mappings* between input space propositions and related output space propositions. Whenever a non-zero level of belief is encountered for an input space proposition, that belief is multiplied by a measure of uncertainty associated with the mapping itself and then assigned to the related output space proposition in the form of an independent belief function (with any remaining belief attributed to ignorance).

This approach is intuitively sound and is adaptable to rule-based expert systems. But the mappings and their associated uncertainty values are basically ad hoc and they have no foundation in D-S. The following is an example of the problems which can result from this type of approach:

Example 7.1 Suppose that we attempt to implement a one-to-one or *identity* relationship between the input and the output propositions. For example, we might

have two input elements I_1 and I_2, two output elements O_1 and O_2, and the identity set of mappings "I_1 *implies* O_1 *with certainty* 1.0" and "I_2 *implies* O_2 *with certainty* 1.0". Then the following input belief function:

$$S(I_1) = 0.5,\ S(I_2) = 0.5,\ S(I_1, I_2) = 1.0$$

would, through the two mappings, yield the following two output belief functions:

$$S(O_1) = 0.5,\ S(O_2) = 0.0,\ S(O_1, O_2) = 1.0$$

and

$$S(O_1) = 0.0,\ S(O_2) = 0.5,\ S(O_1, O_2) = 1.0$$

Note that the support value of 1.0 for "O_1, O_2" in each of these belief functions comes from a 0.5 basic probability number assigned to one of the singletons and a 0.5 basic probability number assigned to "O_1, O_2" itself as a result of attributing remaining belief to ignorance. These belief functions would combine under Dempster's rule to yield the overall output belief function:

$$S(O_1) = 0.333,\ S(O_2) = 0.333,\ S(O_1, O_2) = 1.0$$

Here, the support value of 1.0 for "O_1, O_2" is a result of summing the 0.333 basic probability numbers assigned to each of the three propositions. Thus the set of identity mappings did not operate correctly since the output belief function is not a one-to-one mapping of the input belief function. Not only do we get different levels of belief for the two singletons, but we produce a level of uncertainty where none originally existed. Note that a different propagation approach could have been employed in which the two rule's assertions do not make up different basic probability assignments but are simply two components of a single assignment. This solution is no better, for a set of assertions about the propositions in the output

> space need not all come from the same input space frame, so we cannot simply state that all such assertions are basic probability numbers in the same basic probability assignment because they then will generally not add up to one.

These inconsistencies are not fundamental faults with systems which only partially adopt D-S theory, and solutions can be derived which deal with these particular problems. However these are typical of the types of problems that arise from the lack of a single, coherent underlying theory for the entire reasoning process.

7.5 Lowrance, Garvey and Strat

J. Lowrance, T. Garvey and T. Strat presented a paper at the National Conference on Artificial Intelligence in 1986 (AAAI-86) entitled "A Framework for Evidential-Reasoning Systems" [42]. This paper gives a detailed description of a reasoning system which, like the one presented by Garvey, Lowrance and Fischler at IJCAI-81, seems to be fully consistent with D-S theory. To solve a reasoning problem in their system, they first partition some propositional space into appropriate frames of discernment. Next, they represent their knowledge about which propositions from one frame are compatible with propositions in another frame (that is, which pairs of propositions from the two frames could be true simultaneously). This knowledge takes the form of a set called a **compatibility relation** which consists of ordered pairs corresponding to these compatible pairs of propositions. From this compatibility relationship, a set of mappings is defined which can be used to transfer belief between frames of discernment. Thus, to determine our belief in the propositions for some frame, we transfer the belief from any related frames to our frame of interest and combine all of this belief, yielding our current combined belief.

We have stated that this approach is consistent with D-S techniques for reasoning. However the system, as we've described it, uses mappings to transfer belief between frames of discernment

(as did earlier non-consistent systems such as Lu and Stehanou described above). The difference between this system and earlier systems is that the compatibility relation set used by Lowrance for transferring belief is exactly equivalent to Shafer's minimal refinement frame and the entire mapping process is mathematically equivalent to using Shafer's techniques to define a minimal refinement of two frames, transfer any belief from the two original frames to the refinement, combine the belief and then coarsen the minimal refinement frame so that we are left with a frame which is identical to the one that we were originally planning to transfer our belief to.

In chapter 10, several comparisons are offered between this D-S implementation approach and the one developed during the thesis research project.

7.6 Zadeh

Lotfi Zadeh is the father of fuzzy logic theory, one of the main "competitors" of D-S theory for inexact reasoning in expert systems. Fuzzy logic theory will not be discussed here, but Zadeh has frequently commented on D-S theory and these comments are discussed in this section. Although Zadeh's comments have been, for the most part, cautiously positive, he takes strong issue with the normalization process from Dempster's rule and he has aggressively criticizing the theory on that basis (see Zadeh [82] and [83]). However the examples that Zadeh's uses to make his point consistently show that he has failed to grasp some of the basic concepts of the theory. For example, Zadeh gives the following example:

> "Assume that Country X believes that a submarine, S, belonging to Country Y is hiding in X's territorial waters. The Minister of Defense of X summons a group of experts, $E_1, \ldots, E_n$, and asks each one to indicate the possible locations of S. Assume that the possible locations specified by the experts $E_1, \ldots, E_m$, $m < n$, are $L_1, \ldots, L_m$, respectively, where L_i, $i = 1, \ldots, m$, is a subset of the territorial waters; the remaining experts, $E_{m+1}, \ldots, E_n$, assert that there is no submarine in

> the territorial waters, or, equivalently, that $L_{m+1} = \ldots = L_n = \emptyset$, where $\emptyset$ is the empty set." [83, p.81].

Zadeh goes on to show that the because the D-S theory allows no belief in the empty set and therefore eliminates any such values and normalizes all other belief, the opinion of those experts who said that there is no submarine is completely disregarded. His criticism is summed up by his assertion that "Normalization throws out the opinion of those experts who assert that the object under consideration does not exist."[83, p.82]

If we question why D-S disallows belief in the empty set, however, we remember that one of the most basic requirements of a frame of discernment is that the elements make up a mutually exclusive and exhaustive set. If all possible solutions are represented by the elements of the frame, then any belief about what is true must be represented by some proposition and cannot be represented by the empty set unless we have belief in something which we have deemed to be impossible.

The question remains, then, of how the situation outlined in the above example could arise. Shafer gives the solution to this question in chapter twelve of his book. A frame of discernment, he shows, is based on many assumptions. For example, we have previously defined a frame which discerns whether a person is left handed or right handed. This frame assumes that the person was not ambidextrous. Assumptions are, in fact, a requirement of reasoning. For example, we also assume in this frame that a particular person exists and is not just some figment of our imagination; if we had to consider all such options then we would rarely be able to gather strong enough evidence to prove that anything were true. Never-the-less, we may find that some particular assumption is not valid and so we must redefine the frame to include that option. This is what has happened in Zadeh's example; the frame of discernment which is made up of possible locations of the submarine in the country's territorial waters is clearly based on the assumption that a submarine is in those waters. The existence of an expert opinion that says there is no submarine is not an indication of some level of belief in the empty set, it is an

indication that the frame of discernment has been incorrectly defined and it must be revised to include an element corresponding to "there is no submarine".

In the D-S theory, then, we can never assert any level of belief in the empty set. However we still need to be able to deal with such belief because it is generated by Dempster's rule when combining two bodies of evidence that contain belief which is, to some extent, conflicting. Zadeh gives a second example in the same paper which criticizes this procedure:

> "Suppose that a patient, P, is examined by two doctors, A and B. A's diagnosis is that P has either meningitis, with probability 0.99, or a brain tumor, with probability 0.01. B agrees with A that the probability of brain tumor is 0.01, but believes that it is the probability of concussion rather than meningitis that is 0.99. Applying the Dempster rule to this situation leads to the conclusion that the belief that P has a brain tumor is 1.0–a conclusion that is clearly counter-intuitive..." [83, p.82]

These results certainly demand a second look, however in so doing the problem is again found in the example, not in the theory. The belief from the first doctor states that the diagnosis could possibly be either meningitis or a brain tumor (with the former being much more probable than the latter) and that all other diagnosis are untrue with certainty (i.e. impossible). The second doctor stated that a concussion and a tumor are possible but that all other diagnosis are impossible. Since the first doctor has conclusively ruled out a concussion and the second doctor has conclusively ruled out meningitis, the only possible diagnosis is a tumor.

This example, then, does not illustrate an inherent problem with the D-S theory but rather illustrates the fact that claims of certainty are taken literally in the theory. In fact, the D-S theory would itself identify this example as being questionable in that an extremely high weight of conflict would be produced in the combination which would flag the fact that highly conflicting information was being combined. Furthermore, if the example were a more realistic one, the evidence that lead the first doctor to place a certainty of 0.99 in meningitis would very likely

have caused the second doctor to admit at least some possibility of meningitis and vise-versa. If even so slight a possibility of 0.01 were thus allowed, the results would be quite different; as illustrated in table 7.1.

Table 7.1: Modified Doctor's Example

	1st Doctor	2nd Doctor	Combined belief
m(meningitis)	0.98	0.00	0.490
m(tumor)	0.01	0.01	0.015
m(concussion)	0.00	0.98	0.490
m(meningitis, tumor, or concussion)	0.01	0.01	0.005

The results shown in table 7.1 are, of course, much more appealing. They do point out, however, that there can be a significant (although not discontinuous) difference between very small levels of belief and zero belief. Again, this is due to the fact that categorical statements are taken literally and two categorical belief statements which conflict indicate an impossible situation– one or both of the functions must be discounted or discredited to some extent.

7.7 Other Related Work

This section lists a few of the other important works relating to D-S theory without discussing their content.

- Thomas Strat presented a paper entitled "Continuous Belief Functions for Evidential Reasoning"[70] at AAAI-84. This paper describes a very interesting and useful extension of D-S frames of discernment from discrete to continuous variables.
- After writing "A Mathematical Theory of Evidence" in 1979, Glen Shafer participated very little in work relating artificial intelligence applications of his theory (his 1976 book itself was not intended for an AI audience). By the mid-1980's, however, he seems to have

been energetically drawn into the fray. Most of the works which he has since prepared or collaborated on are important references for this field. See, for example, [11], [60], [61], [63], [64], [65], [66], [67], and [68].

- Ronald R. Yager has produced a large amount of work on uncertainty in artificial intelligence. His work has focused on D-S theory in addition to fuzzy set theory and other approaches. Most of Yager's work is at a fairly theoretical level. See [72], [73], [74], [75], [76], [77], [78], and [79].

7.8 Conclusion

This chapter has reviewed some of the important work performed in the area of applying D-S theory to inexact reasoning in expert systems. A general observation of these works is that they each seem to follow one of two distinct paths. The first path borrows such components of D-S theory as proves advantages for simple, rule-like reasoning approaches: such systems are non-consistent. The second path strives to produce truly consistent D-S implementations. The common thread among these works is that they don't go very far in uncovering a general, practical application approach which remains true to the goal of consistency. A clear objective of work in this field is to bring these two paths back together to produce a simple, practical application approach which remains truly consistent with D-S theory.

Chapter 8

Non-Consistent Implementations

This chapter deals with the non-consistent implementation attempts made early in the research project. While some of the work outlined in the previous chapter was available when the project began, most of it was not. Based on the material which was available from the outset and on a rudimentary understanding of the D-S theory, the project began with the idea that a simple MYCIN-like expert system could be created in which certainty factors were replaced with D-S belief functions. In a MYCIN-like system, the function of a rule is to determine the level of belief in its premise and assert a corresponding level of belief for its conclusion. Further, the abilities exist to handle conjunction, disjunction and negation in the premise; to incorporate some level of uncertainty in the validity of the rule itself; and to combine the effects of several rules all dealing with the same conclusion.

If, as in the approach described by Gordon and Shortliffe (section 7.3), the measure of belief resulting from the application of a rule is taken to be a D-S belief function (focused exactly for or against some singleton) rather than a certainty factor, then the combination of several rules' results can be performed using Dempster's rule.

Using this approach for combining the results of multiple rules and using the normal MYCIN formulae for all other operations, a simple rule-based system can be constructed. An attempt at such an approach is described in section 8.1. However this technique has two main problems. First, the application of Dempster's rule requires two or more *belief functions* as input, not simply two single measures of belief (i.e. two support numbers). The assumption made by Lu and Stehanou (section 7.4) that the support value for any premise proposition can be

assigned to a conclusion proposition with remaining belief assigned to the frame itself will result in a belief function as the rule's output, but the belief function is created according to an arbitrary convention. It is this arbitrary process that leads to the problem previously described in example 7.1. Second, all of the other processes in the system have no connection with D-S theory, thus the system is non-consistent regardless of the first problem.

While the first problem was overcome by using an approach described in section 8.2, it was eventually realized that any approach which closely followed the MYCIN-like rule format could never be fully consistent with D-S theory. The resulting consistent work is described in the subsequent chapters.

8.1 A D-S Inexact Reasoning Module

If an expert system is created using PROLOG predicates as rules, an additional set of predicates can be written which collaborates with the rule predicates to manage uncertainty. This set of uncertainty handling predicates forms an independent module which can be modified to accommodate a variety of inexact reasoning techniques. This is the approach suggested in a paper by Lecot and Parker entitled "Control over Inexact Reasoning" (published in *AI Expert* magazine in 1986 [39]). Lecot and Parker refer to the set of inexact reasoning predicates as an **inexact reasoning module**, or **IRM**. The predicates in an IRM operate on measures of uncertainty, U, the structure of which depends on the inexact reasoning approach being used. If a MYCIN-like approach is adopted, for example, the structure of U will be a certainty factor (a number between negative one and one). As described by Lecot and Parker, an IRM, must contain at least the following main predicates:

1. *certainty_and*(U_1, U_2, U_3). Produces the uncertainty measure, U_3, resulting from a premise of the form "term-1 AND term-2", where U_1 and U_2 are the measures of uncertainty associated with each term respectively.

2. *certainty_or*(U_1, U_2, U_3). Produces the uncertainty measure, U_3, resulting from a premise of the form "term-1 OR term-2", where U_1 and U_2 are the measures of uncertainty associated with each term respectively.

3. *certainty_not*(U_1, U_2). Produces the uncertainty measure, U_2, resulting from a premise of the form "NOT term", where U_1 is the measures of uncertainty associated with the term.

4. *certainty_rule*(U_1, U_2, U_3). Produces the uncertainty measure, U_3, resulting from a rule where U_1 is the uncertainty associated with the premise and U_2 is the uncertainty associated with the rule itself.

5. *certainty_combine*(U_1, U_2, U_3). Produces the combined uncertainty measure, U_3, resulting from the application of two rules with their associated measures of uncertainty U_1 and U_2 respectively (Lecot and Parker included both the *certainty_rule* and the *certainty_combine* operations into a single predicate).

For example, in a MYCIN certainty factor IRM, the prolog predicates for the *certainty_and* and *certainty_or* operations are as follows:

```
certainty_and(cf(A),cf(B),cf(C)):-
    number_min(A,B,C).
certainty_or(cf(A),cf(B),cf(C)):-
    number_max(A,B,C).
```

where "number_min" and "number_max" simply assign the minimum and maximum of A and B to C respectively. Note that this is not quite equivalent to the actual MYCIN system which keeps measures of belief and measures of disbelief separately.

This general approach is easily implementable and the first forays into D-S implementation made by during the thesis research were to adapt D-S concepts to this format. The resulting

D-S IRM successfully implemented Gordon and Shortliffe's combination approach described in section 7.3 (which is, recall, consistent as far as it goes) as well as MYCIN's formulae for conjunctive and disjunctive premises and for weak inference as described in section 2.2 (these, of course, are non-consistent with D-S theory). The details of this implementation here will not be provided in this thesis.

The resulting D-S IRM produced results which seemed to be richer, or better able to describe the uncertainty, than did the results from the normal certainty factor IRM. However using the D-S version was slightly more complicated than the certainty factor version. Of greater interest to us, however, was the observation that the system would produce apparently inconsistent results at times, like those described in example 7.1 for instance. The investigation of these results lead to a new implementation approach which came closer to (but did not fully achieve) the goal of a consistent implementation. This system is described in the following section.

8.2 A Simple Support Function Approach

As previously mentioned, the approach outline in the preceding section produced incongruous results because of the arbitrary manner in which belief in a premise proposition was translated into a belief function for the conclusion. The approach outlined in this chapter corrects this problem by basing the inference process on simple support functions (as defined in section 4.2). We therefore call this approach the simple support function, or SSF, approach.

8.2.1 Restrictions

The SSF approach places two main restrictions on the use of D-S theory. Not only do these restrictions enable the approach to solve some of the problems associated with the D-S IRM approach, but they enable Barnett's simplified calculations (section 7.1) to be used. These restrictions are as follows.

The first of these restrictions consists of confining the allowable subsets of a frame Θ to the singletons, the compliments of the singletons and Θ itself. This restriction admittedly invalidates the D-S theory advantage of being able to assign belief to *any* level of precision (i.e. *any* possible subset of Θ). However the advantages of accommodating ignorance and of resulting in uncertainty intervals remain intact.

The second restriction is that all pieces of evidence being added to a body of knowledge must be in the form of simple support functions. As mentioned in section 4.2, this reflects the process of incrementally acquiring single pieces of evidence and is appropriate for modeling expert behavior. If the computational and representational burden were ignored, both of these restrictions could be discarded and the following theory should hold: we have not, however, attempted to identify or implement the general formulae.

8.2.2 SSF Approach to Inference

The SSF approach to inference can be observed by examining the behavior of the simplest type of rule–one in which the premise is a single proposition in one frame ($\Theta_{PREMISE}$), the conclusion is a proposition in a different frame ($\Theta_{CONCLUSION}$) and there is no uncertainty attached to the rule itself. We then wish to determine a degree of belief for the conclusion which reflects the extent of our belief in the premise. The conclusion's degree of belief will then act as a single new piece of evidence to be combined with the knowledge already collected for $\Theta_{CONCLUSION}$. As we have seen in section 4.2, the belief function representation of a single piece of evidence is a simple support function. Therefore, the conclusion of the rule must be in the form of a simple support function (in fact, this is one of the computational restrictions that we have placed on the system). Recalling that a single simple support function can be fully represented by a simple support number, we set the output of our rule to be a proposition with an attached measure of belief consisting of the simple support number *ss*(*conclusion*).

Knowing that the degree of belief in the conclusion is in the form of a simple support number

and that this number is (in this simple case) proportional only to the degree of belief in the premise, it becomes clear that we should set:

$$ss(conclusion) = ss(premise) \tag{8.1}$$

In order to use this rule, then, we must first determine $ss(premise)$. We could calculate the simple support number for the premise by taking all the pieces of evidence we have for every proposition in $\Theta_{PREMISE}$, combining them using Dempster's rule of combination to get a separable support function, decomposing this function into its unique set of simple support functions, and extracting the desired simple support number. However, we can perform this process in such way as to first combine all evidence (in the form of simple support functions, recall) for each allowable proposition in the frame, thereby producing one intermediate simple support function for every such proposition. These intermediate would then, in the process outlined above, be combined to create a single separable support function and then decomposed to produce a unique set of single support functions. It can be determined, however, that the final unique set of simple support functions will be identical to the intermediate set of simple support functions; the combination and decomposition is therefore unnecessary. If it is only $ss(premise)$ that we are after, then, we only need to combine any pieces of evidence (each a simple support number) for the one proposition contained in the rule's premise using the simplified special case of Dempster's rule given in equations 4.15 and 4.16.

For this simple, single proposition rule, then, the SSF approach offers a consistent (though restricted) techniques for inferring belief about the conclusion from belief about the premise. The following example shows how this technique prevails over the approaches described in sections 7.4 and 8.1:

Example 8.1 Suppose that we have two frames of discernment; $\Theta_{PREMISE} = \{a, b\}$ and $\Theta_{CONCLUSION} = \{x, y\}$. We may, as in example 7.1, wish to specify a

set of rules which imply that our belief in x is exactly our belief in a and our belief in y is exactly our belief in b. With this identity rule set, then, we expect our belief in $\Theta_{CONCLUSION}$ to exactly equal our belief in $\Theta_{PREMISE}$. Now suppose that our belief in $\Theta_{PREMISE}$ is described by the basic probability assignment:

$$m(a) = 0.5, m(b) = 0.3, m(a,b) = 0.2$$

The D-S IRM system described in section 8.1 would assert that $m(x) = 0.5$ and $m(y) = 0.3$, but these assertions would have to be made in the form of simple support functions. These functions, along with the corresponding combination from Dempster's rule, are:

$$m(x) = 0.50, m(y) = 0.00, m(x,y) = 0.50$$

$$\underline{m(x) = 0.00, m(y) = 0.30, m(x,y) = 0.70}$$

$$m(x) = 0.41, m(y) = 0.18, m(x,y) = 0.41$$

Thus our belief in the conclusion does not equal our belief in the premise.

However, if we redo the example using simple support numbers, our belief in $\Theta_{PREMISE}$ would be expressed in the form of a set of simple support numbers corresponding to the above belief function; $ss(a) = 0.714$ and $ss(b) = 0.6$. The rules then make the assertions $ss(x) = 0.714$ and $ss(y) = 0.6$ or, in the form of simple support functions with the corresponding dempster combination:

$$m(x) = 0.714, m(y) = 0.000, m(x,y) = 0.286$$

$$\underline{m(x) = 0.000, m(y) = 0.600, m(x,y) = 0.400}$$

$$m(x) = 0.500, m(y) = 0.300, m(x,y) = 0.200$$

Thus our belief in the conclusion does equal our belief in the premise in this case.

The underlying inexact reasoning mechanism in the SSF system, then, consists of obtaining knowledge about various frames of discernment and storing it in the form of simple support numbers. Rules are used to chain from frame to frame by combining all of the simple support numbers for some proposition and asserting the combined value as a simple support number for a new proposition.

8.2.3 Comparison of the SSF System and MYCIN

It is interesting to note that the inexact reasoning process of combining simple support numbers for both singletons and the compliments of singletons using the special case of Dempster's combination rule corresponds exactly to MYCIN's combination of MB and MD factors (Measure of Belief and Measure of Disbelief respectively). This similarity leads to an assumption that the success of MYCIN's inexact reasoning system would likely be at least equaled by the SSF system.

The similarity between our D-S based system and MYCIN, however, also points out a seemingly confusing aspect of our system. Although an underlying concept of the D-S theory is that belief in any proposition within some frame of discernment is normalized to reflect belief relative to the belief in all propositions in that frame, our system (like MYCIN) determines its belief in a rule's premise from evidence concerning that specific proposition only. The solution to this apparent problem is that the simple support numbers used by the rules, while alone they carry no indication their *relative support*, can be used to construct a fully normalized separable support function if and only if the set of simple support functions is complete. This means that if we relate one frame of discernment to another with a set of rules that is complete though not redundant, then we would be able to supply information about the first frame and obtain a separable support function from inferred facts about the second frame just as if our relative belief in all the propositions of the first frame were somehow transferred to the second.

The difference between the two reasoning mechanisms is that, while both systems propagate

degrees of confidence for and against propositions in the same way, MYCIN interprets these confidences by simply subtracting the level of disbelief from the level of belief to obtain a single ad hoc parameter. Our system, on the other hand, will combine our belief for and against each proposition with the belief for all related propositions in a mathematically coherent way. The result is an interpretation of the various degrees of belief that reflects our relative belief in each proposition, the precision of belief in each proposition, and the extent of our ignorance about each proposition.

Having now outlined the theoretical approach employed by the SSF system, we now supply some of the details of its implementation.

8.2.4 Representation of Uncertainty

In our implementation of a SSF system, a frame of discernment is made up of all the possible VALUE's of some OBJECT,ATTRIBUTE combination. The singletons of the frame, then, would be represented by **associative triples** of the form "the ATTRIBUTE of the OBJECT is VALUE" along with an associated uncertainty measure comprised of a simple support number. For each singleton in the frame, there is a corresponding compliment. Support for these compliments could be represented in the form of an associative triple with the same VALUE as the singleton, but with a negative sign in front of the simple support number (although the difference is slight, these should be read as support for the negation of the singleton, not negative support for the singleton). Thus all knowledge is stored as an associative triple statement with a single valued degree of belief.

In the SSF approach, uncertainty in rules themselves is represented by simply incorporating a single valued uncertainty measure U in each rule such that:

$$0 \leq U_{RULE} \leq 1$$

8.2.5 The Inexact Reasoning Process

In our SSF approach, the inexact reasoning (or uncertainty propagation) process involves the following steps:

1. Determine the belief in each premise proposition:

   All evidence in favor of each of the premise's propositions must be obtained. This entails combining the simple support numbers for each piece of evidence that supports the proposition:

$$ss(A) = 1 - (1 - ss_1(A)) \times (1 - ss_2(A)) \ldots \tag{8.2}$$

2. Determine the overall belief in compound premises:

   A compound premise is made up of several propositions related by the Boolean operators AND, OR or NOT. For each of these relationships, we need a formula for combining the simple support numbers of the associated propositions. A criticism of many systems (consistent and non-consistent alike) is that they assume a certain level of dependency between the propositions when they derive these formulae. We can avoid this problem by providing a different formula for each different level of dependency and then specifying the most appropriate formula to use in any given rule. The appropriate formulae, then, for deriving the simple support number, $ss(P)$, for the compound premise, P, from the set of premise propositions $p_1, p_2, \ldots p_n$, are:

   AND: P conjoins dependent $(p_1, p_2, \ldots p_n)$,

$$ss(P) = \max\left(0, 1 - \sum\left(1 - ss(p_i)\right)\right) \tag{8.3}$$

   P conjoins independent $(p_1, p_2, \ldots p_n)$,

$$ss(P) = \prod ss(p_i) \tag{8.4}$$

OR: P disjoins dependent $(p_1, p_2, \ldots p_n)$,

$$ss(P) = \max\left(ss(p_i)\right) \tag{8.5}$$

P disjoins independent $(p_1, p_2, \ldots p_n)$,

$$ss(P) = 1 - \prod\left(1 - ss(p_i)\right) \tag{8.6}$$

P disjoins exclusive $(p_1, p_2, \ldots p_n)$,

$$ss(P) = \max\left(1, \sum ss(p_i)\right) \tag{8.7}$$

NOT: P negates p,

$$ss(P) = ss(-p) \tag{8.8}$$

Within each operator class, the formulae are listed in order of increasing strength. For a given set of propositions, for example, the value of the premise's support will be lower if the disjoin dependent formula is used than if the disjoin independent formula is used, and in turn the disjoin independent formula will yield a lower value than the disjoin exclusive formula. This means than if the relationship between the propositions is not known, the weakest formulae (the dependent case) can safely be used as a lower bound.

3. Check premise acceptance:

Once the degree of support for the rules premise is known, it must be checked against some minimum threshold value. This is so a rule will not be used if there is not sufficient evidence that the premise is true. The value of this threshold has typically been set arbitrarily. If the acceptance threshold is set too high, possibly correct solutions would be ignored if they have low initial support. If the threshold is too low, the system will try to examine far too many incorrect solutions, leading to increased running time and irrelevant questioning of the user. In the absence of any theoretically correct value, the value of 0.2 which has proven acceptable in the MYCIN system seem reasonable:

Use rule if $ss(premise) \geq 0.2$

4. Determine the belief in the conclusion:

   We have seen that in the case of a rule known with certainty, we set the simple support number of the conclusion to be equal to the simple support number of the premise. In the more general case, we wish the support for the conclusion to also reflect the degree of belief in the rule itself. This is achieved by using the formula:

$$ss(conclusion) = ss(premise) \times U_{RULE} \tag{8.9}$$

5. Combine uncertainty from various sources:

   If we wish to determine the system's overall belief in some proposition at any given time, we must combine all the sources of evidence from all related statements using Dempster's rule of combination. The procedure involves three steps:

   (a) For each VALUE (A) in the frame of the proposition in question, combine the simple support numbers for all statements supporting that VALUE to obtain a new simple support number. Likewise, combine all of the simple support numbers from statements supporting that VALUE's compliment ($\overline{A}$):

$$ss(A) = 1 - (1 - ss_1(A)) \times (1 - ss_2(A)) \ldots \tag{8.10}$$

$$ss(\overline{A}) = 1 - (1 - ss_1(\overline{A})) \times (1 - ss_2(\overline{A})) \ldots \tag{8.11}$$

   (b) Combine the simple support number for each VALUE in the frame with the simple support number for that VALUE's compliment to get one separable support function. The separable support function can be described in terms of the basic probability numbers $m(A)$ and $m(\overline{A})$ as well as the parameters $r(A)$ and $d(A)$ as defined below:

$$m(A) = \frac{ss(A) \times (1 - ss(\overline{A}))}{1 - ss(A) \times ss(\overline{A})} \tag{8.12}$$

$$m(\overline{A}) = \frac{ss(\overline{A}) \times (1 - ss(A))}{1 - ss(A) \times ss(\overline{A})} \tag{8.13}$$

$$r(A) = 1 - m(A) - m(\overline{A}) \tag{8.14}$$

$$d(A) = 1 - m(A) \tag{8.15}$$

(c) Combine all of the separable support functions into one function, and describe it in terms of a series of support intervals:

$$\text{support interval of } (A) = [S(A), Pl(A)]$$

where:

$$S(A) = K \times \left(m(A) \times d(B) + r(A) \times \prod m(\overline{B})\right) \tag{8.16}$$

$$p(A) = 1 - K \times \left(\prod d(C) \times \sum \left(\frac{m(B)}{d(B)}\right) + m(\overline{A}) \times \prod d(B) - \prod m(\overline{C})\right) \tag{8.17}$$

$$K = \frac{1}{\prod d(C) \times \left(1 + \sum \left(\frac{m(C)}{d(C)}\right)\right) - \prod m(\overline{C})} \tag{8.18}$$

for all $B \neq A$ and all $C \subset \Theta$

8.2.6 Belief Decomposition

The decomposition of a separable support function into a set of simple support functions is a process which the system can use in two ways. First, a user may wish to enter his belief about a set of proposition as a separable support function so that he can more easily specify his relative support for the different hypotheses and for ignorance. However we have restricted the addition of new information to the system to simple support functions. The user could only enter his belief in the form of a separable support function if the system were to decompose the function before it added the evidence to its body of knowledge.

The decomposition process could also be used in a manner pointed out by Lu and Stehanou [44] as a tool for adjusting the certainty factors attached to rules. An expert could describes his

belief about the propositions in one frame of discernment and then a set of rules could infer a state of belief about a second frame. The expert who has written the set of rules could then compare the rule's separable support function for the second frame with his own belief about what the inferred belief should be. If he decides that the rules have not performed satisfactorily, however, he cannot tell what the impact of each rule has been. He must therefore decompose both the rule's and his own simple support functions to see which propositions contain the conflicting levels of belief. Now each proposition corresponds to one specific rule which can be adjusted to improve the inference.

8.3 Conclusion

This chapter has discussed two non-consistent D-S implementation attempts. Both the D-S IRM and the SSF approaches use frames of discernment to represent system variables, though both apply restrictions to which of the frame's propositions can be used. The D-S IRM system combines the results of multiple rules in a manner which is consistent with D-S theory. However the formulae for using each individual rule are non-consistent. The SSF approach goes one step further in that it performs inference for simple rules is a consistent manner. However non-consistent techniques are still used for dealing with compound premises and for incorporating uncertainty in a rule itself. Neither approach, then, reaches the goal of complete consistency with the D-S theory–thus neither approach was pursued further during the thesis research. However both approaches can be implemented in a simple and robust way and are likely to perform better then simple certainty factors. Anyone interested in developing practical expert systems with moderate uncertainty handling capabilities might find either one of these approaches to be extremely useful.

Chapter 9

Global Frame Approach

The previous chapter dealt with various non-consistent attempts to implement D-S theory for inexact reasoning. During the testing of the SSF approach system, it was realized that all of the capabilities required for inexact reasoning (namely inferring belief in a conclusion from belief in a premise, obtaining a combined belief in a premise with several terms, incorporating uncertainty in the inference rule itself, and aggregating the results of several inferences rules) can all be performed in a manner completely consistent with D-S theory. This discovery rendered the SSF and earlier non-consistent approaches essentially obsolete, and further research work on such systems ceased. Yet the discovery also created more problems than it solved since, while the theoretical feasibility of a consistent system seemed clear, the specific implementation approach remained quite complex conceptually and extremely onerous computationally. This chapter deals with an initial complete implementation prototype which is based on a "global frame of discernment" approach.

9.1 Overview of Global Frame Approach

The general approach to reasoning in a global frame system is to provide the system with enough information to define one large frame which can discern all propositions that might be of interest as well as the relationships between them. Then all known belief about the problem can be specified. A current state of belief about all of our propositions is maintained by combining all belief into a single basic probability assignments on the one large frame of

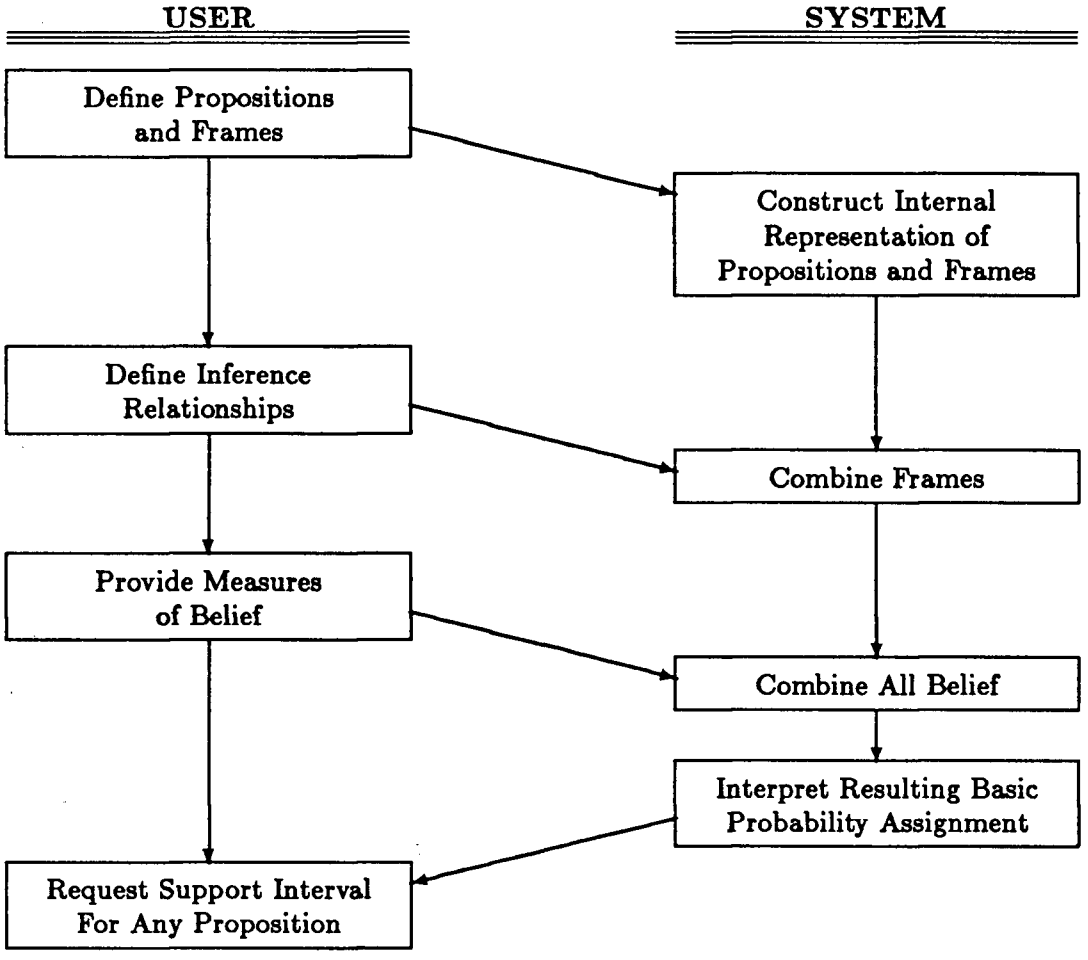


Figure 9.1: Flow chart for the Basic Reasoning Procedure in a Global Frame Approach System

discernment. We can then interpret this global belief, producing a support interval for any proposition discerned by the system.

Implementing this approach involves conveying to the system our knowledge about what concepts exist, about how concepts are related, and about the levels of belief that we have in these concepts. The system can then manipulate these three types of knowledge to perform reasoning. A flow chart of the basic reasoning process is shown in figure 9.1; each phase of the process is described in greater detail in the subsequent sections.

9.2 Concept Representation

The first step in setting up an inexact reasoning problem in a global frame system is to specify a series of frames of discernment such that every proposition that we might be interested in will be represented in some discernment space. Concepts which are of interest for some reasoning problem are identified and sorted into like groups. In cases where groups of concepts make up a set of mutually exclusive options, they can become the elements of a single frame of discernment (if the set is not exhaustive, an "all other options" element could be added). In cases where groups of concepts make up non-mutually exclusive options, each concept can form a separate frame containing two elements corresponding to that concept's truth or falsity. In addition to supplying the elements of a frame, we can also specify any aliases which we may want to assign to various propositions (aliases are used by the system to reference propositions in input/output operations).

The process of specifying the frames of discernment is not a trivial one. Each frame is based on a number of assumptions. For example, a frame containing possible locations of a ship may assume that the ship is in the water even though it could possibly have been hauled up on land; perhaps for repairs. Such assumptions seem reasonable when we first define our frame and in fact they are a requirement of reasoning. However in some cases we will find that our assumptions about a frame are wrong and we will need to change our frame (not just refine it) to suit the circumstances. The task of defining the frames of discernment, then, is both a creative and an iterative process.

9.3 Inference Knowledge Representation

The second step in resolving an inexact reasoning problem is to represent our knowledge about which concepts imply other concepts. This step is similar to the specification of inference rules in a typical rule-based expert system. The inference knowledge which we wish our system to

capture can usually be put into the form "if PREMISE is true then CONCLUSION is true," or "PREMISE $\Rightarrow$ CONCLUSION". We can interpret this as meaning that, if we have some level of belief in the premise, then we will have at least that much belief to the conclusion.

This interpretation leads to the result that inference can be represented in a D-S representation by set inclusion. If some proposition A is a subset of proposition X, then any belief attributed to A will simultaneously be attributed to X. Thus we satisfy our interpretation of logical inference by setting the proposition which represents the PREMISE concept to be a subset (not necessarily a proper subset) of the proposition which represents the CONCLUSION.

Representing inference knowledge, then, is a matter of specifying the correct set relationships between propositions. When both the premise and the conclusion propositions are within the same discernment space, we can initially construct our frame of discernment such that these relationships will be incorporated. However any two propositions are generally not initially within the same discernment space. We must therefore create a new frame which discerns both propositions and correctly incorporates the known inference relationships between them. This frame is the minimal refinement.

Although Shafer indicates that the minimal refinement is an appropriate medium for exploring the relations among different frames, he does not specifically show how knowledge about these relationships can be used to create the minimal refinement. The global frame system approach for accomplishing this is to list inference knowledge in the form of logical implication statements or inference statements wherein the knowledge that proposition $A \subset \Theta_A$ implies proposition $X \subset \Theta_X$ is represented by the statement:

$$A \Rightarrow X$$

Such statements are quite generally applicable. If we have inference knowledge in which the premise or the conclusion involve logical operators such as AND, OR or NOT, then the proposition A or X correspond simply to the sets resulting from the use of the corresponding set

operators intersection, union, or complementation. However, the inference statements are restricted by the requirements that the X proposition should be the finest proposition which is known to be implied by A without being so fine as to overstate what we can logically infer and, within a set of statements which completely describe the known relationships between two frames, all of the elements belonging to both Θ_A and Θ_X must be included at least once.

Once these relationships are known to the system they can be used to generate the refinement statements which define the minimal refinement $\Theta_A \otimes \Theta_X$ and allow the proposition names and measures of belief contained in the original frames to be mapped to the minimal refinement; thus allowing all further reference to these propositions to be made in terms of the new refined frame and allowing the original frames to be removed from the system's memory. The refinement statements for the refining $\omega : 2^{\Theta_A} \rightarrow 2 :^{\Theta_A \otimes \Theta_X}$ are given by:

$$\omega(a) = \{(a, x) | a \in A, x \in X, A \Rightarrow X\} \tag{9.1}$$

And the refinement statements for the refining $\omega : 2^{\Theta_X} \rightarrow 2 :^{\Theta_A \otimes \Theta_X}$ are given by:

$$\omega(x) = \{(a, x) | a \in A, x \in X, A \Rightarrow X\} \tag{9.2}$$

In the system proposed by Lowrance, Garvey and Strat, (see section 7.5 and ref.[42]) the minimal refinement is also used to represent knowledge about the relationships between frames (they have called it a **compatibility relation**). However they do not propose a scheme for generating the minimal refinement frame from inference knowledge, rather they require the user to provide the system with the entire frame directly. This is done, it is suggested, by considering each of the elements in a cross product of the initial frames and rejecting those which represent an impossible combination of concepts. For the minimal refinement of small frames, this method likely works well. With more sizable frames, however, the cross product from which the set of possible elements must be selected grows unmanageably large.

9.4 Belief Representation

We have now conveyed to the system our knowledge about possible concepts and their inter-relationships. The next step in resolving an inexact reasoning problem using the global frame approach is to represent the belief that we have in these various propositions. This belief is simply stated in the form of basic probability assignments, or as support functions which the system would translate directly into basic probability assignments. This task of translating all of our belief about some problem into numerical values for particular propositions will likely comprise a significant, though not complex, portion of the entire process.

9.5 Reasoning

We have now provided the system with the three types of domain knowledge which it requires to perform reasoning: the specification of all propositions in which we are interested, the relationships between these propositions and any levels of belief that we have in them. The system can proceed with the reasoning process then, by accepting the proposition definitions, using the inference statements to combine all of the initial frames into one large frame, and accepting, combining and interpreting the given basic probability assignments.

9.5.1 Basic Reasoning

First, the system reads in the frame specifications so that it will have an internal representation of all the proposition names and their relative positions in the set hierarchy of each frame.

Second, the system attempts to create the one minimal refinement of *all* of the initial frames by using the inference statements to repeatedly replace pairs of frames with their minimal refinement. If, for example, we have sets of inference statements which relate the frames $\Theta_A \Rightarrow \Theta_B$, $\Theta_A \Rightarrow \Theta_C$ and $\Theta_B \Rightarrow \Theta_C$, then using the first set of statements would remove the

frames Θ_A and Θ_B and would create the frame $\Theta_{(A,B)}$. The remaining two sets of statements could then be grouped together as both being equivalent to $\Theta_{(A,B)} \Rightarrow \Theta_C$ and this group could be used to replace the frames $\Theta_{(A,B)}$ and Θ_C with the frame $\Theta_{(A,B,C)}$. The order of combination is not important in such an operation.

Third, the system reads in all known basic probability assignments and combines them, resulting in one assignment which fully describes our belief about the problem domain. At any point, the system can be asked for an interpretation of this belief. The system, having once calculated the support and plausibility numbers for the single all-encompassing frame of discernment (only the basic probability numbers are required during the process of reading and combining belief) can then provide the user with the support interval for any proposition discerned anywhere in the system without further calculations.

9.5.2 Extended Reasoning

The process outlined above illustrates the system's most basic form of reasoning. Many variations and extensions to this process could be used to make reasoning more efficient and flexible. A few such examples are listed below:

- In addition to using inference statements to manipulate frames of discernment, statements may be given to refine propositions (thus increasing their precision without relating them to other frames) or to coarsen a frame (thereby reducing its size and precision). These options allow greater control over the makeup of the frames.

- We can introduce a method for dealing with weak implication. Suppose, for example, that we state "$A \Rightarrow X$ with certainty 0.7" This can be interpreted to mean that 70% of our belief in A implies X and 30% implies ignorance, Θ. In order to derive new belief from this known relationship, then, we first refine the proposition, $x(A) = \{A_X, A_\Theta\}$. We then use a **belief refinement procedure** outline in section 9.6 to transfer our belief in A

to these refinements. Finally we make the inferences $A_X \Rightarrow X$ and $A_\Theta \Rightarrow \Theta$. The result will be a frame of discernment in which 70% of our belief in A will imply X and 30% will imply Θ or ignorance.

- Question definition statements can be used (in addition to the initial basic probability assignments) to provide the system with the capability of asking the user to input belief information. This is just one of any number of operations that could be implemented to improve the user interface of an inexact reasoning system.

- Frame manipulation statements, basic probability assignments and question definition statements can be made conditional so that they will not take effect unless certain conditions, such as the achievement of some requisite belief threshold, are met (thereby keeping the system from asking irrelevant questions or from refining those propositions which are known to be false, for example). Such statements effectively provide meta-knowledge or knowledge about how to reason in specific cases.

9.6 Belief Refinement

This section described a **belief refinement** procedure which we have developed to allow weak inference with the global frame approach (as mentioned in the previous section). We illustrated in section 4.3 how a belief function could be transferred from some frame to a finer minimal refinement of that and some other frame. However, this direct transfer of belief from a coarse frame to a finer one cannot take advantage of the new frame's extra precision unless additional, more precise belief is added to the system. We have identified a procedure whereby this additional belief can be assigned to a new frame at the time of refinement, causing all future belief transferred to that frame to be shifted downwards towards more precise propositions. We call such a procedure a **belief refinement**.

Suppose, for example, that a frame $\Theta = \{a, b\}$ is refined to a frame $\Omega = \{a_1, a_2, b\}$. Recall that

the support for the proposition $A = \{a_1, a_2\}$ in frame Ω is the sum of the basic probability numbers $m(\{a_1\}), m(\{a_2\})$ and $m(\{a_1, a_2\})$. However the direct transfer of belief from Θ to Ω results in belief that can be no more precise than $m(\{a_1, a_2\})$. We may have some evidence to suggest that whenever we have belief in A, that at least a certain portion of that belief will be due to belief in $\{a_1\}$. Such belief is captured by a proposition which states that belief may be assigned to $\{a_1\}$ or to any non-A proposition but not to $\{a_2\}$: that is, we assign this belief to the proposition $\{a_1, b\}$ which can be given the more intuitive alias "a_1 IF A". This belief can now be combined (using the Dempster's belief combination procedure outlined in section 5.1) with any belief which is transferred to Ω, yielding the appropriately precise results.

In general, for any new element θ_i in some refinement of proposition $\{\theta\}$ to $\{\theta_1, \theta_2, \theta_3, \ldots\}$, we can assign both the alias "θ_i IF θ" and our associated level of belief to the new proposition corresponding to $\{\theta_i\} \cup \overline{\{\theta\}}$.

Example 9.1 In example 4.2 we calculated the restriction $m_2|\Theta_{MATERIAL}$. However, we may want to incorporate the additional knowledge that ground material is more likely to be bed rock than aggregate. We can therefore attach aliases to some of the propositions in the refined frame as shown in figure 9.2.

We could then define the basic probability assignment $m_3 : 2^{\Omega_{MATERIAL}}$ which would, for example, convey the belief that 40% of all future belief in $ROCK$ will be allocated more specifically to *bed rock* while the rest will remain at $ROCK$:

$$m_3(\Omega_{MATERIAL}) = 0.6,$$

$$m_3(bed\ rock\ IF\ ROCK) = 0.4.$$

Now if we again transferred $m_1 : 2^{\Theta_{MATERIAL}}$ to frame Ω as in example 4.2, and combined the result with $m_3 : 2^{\Omega_{MATERIAL}}$ (using Dempster's rule) we would get a

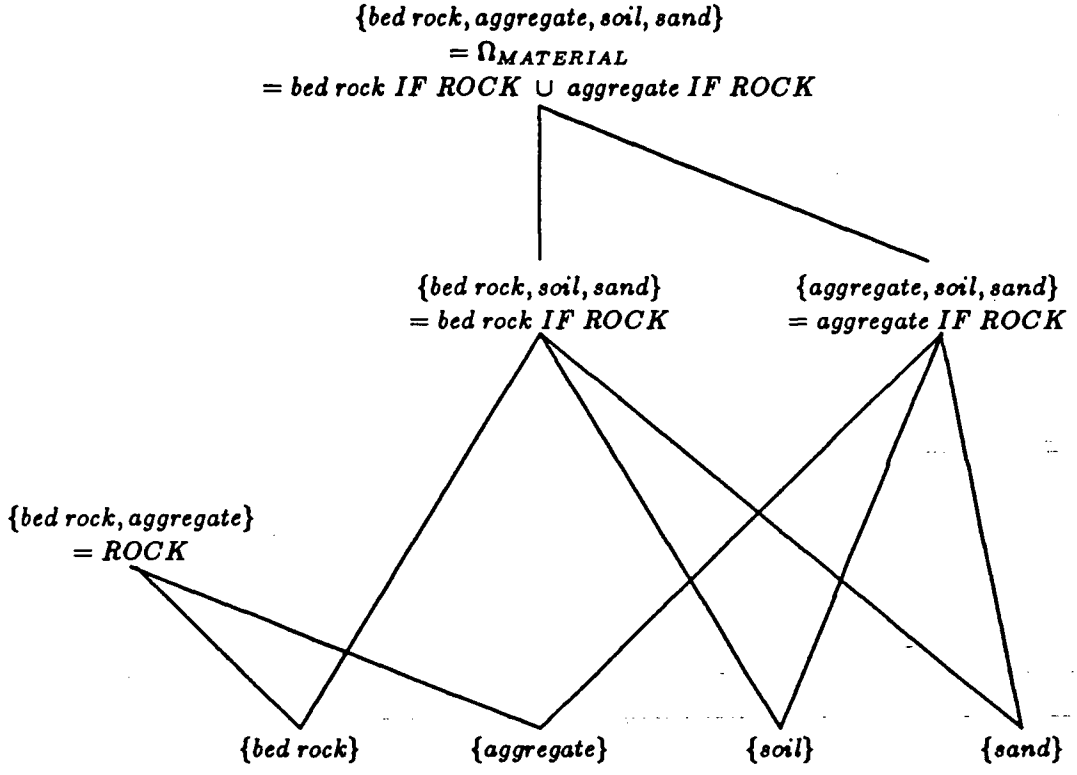


Figure 9.2: Some propositions with aliases for the discernment space $2^{\Omega_{MATERIAL}}$

more precise basic probability assignment:

$$\begin{aligned} m_4(\Omega_{MATERIAL}) &= 0.06, \\ m_4(\{bed\ rock, aggregate, soil\}) &= 0.24, \\ m_4(ROCK) &= 0.30, \\ m_4(\{bed\ rock, soil, sand\}) &= 0.04, \\ m_4(\{bed\ rock, soil\}) &= 0.16, \\ m_4(\{bed\ rock\}) &= 0.20. \end{aligned}$$

9.7 Comparison with Other Systems

We have mentioned in section 9.3 that the system presented by Lowrance, Garvey and Strat accepts a minimal refinement directly from the user rather than generating it from given relationships, as done in our global frame system. However a more significant difference is that, rather than using the minimal refinements to group all propositions together under one large frame, their system maintains all propositions in terms of the originally specified frames and uses the minimal refinement to define a series of mapping relationships which are stored and later used in transferring belief between these original frames. This difference has several consequences; including the fact that, by not utilizing a minimal refinement as a frame of discernment, the system is effectually throwing away more precise knowledge. This leads to the system's inability to return the same level of precision which had been made available to it and, in some instances, it can be shown to give erroneous results (Shafer [57, p.175] shows that, "Dempster's rule of Combination may give inaccurate results when it is applied in too coarse a frame of discernment."). Another problem arising from this treatment of frames is presented in the Lowrance paper itself [42, p.898]. The results obtained from relating some frame Θ_A to a frame Θ_B and then relating frame Θ_B to a frame Θ_C may not match the results obtained from relating frame Θ_A to frame Θ_C directly. This problem does not arise in the global frame system.

However there is another consequence of this difference between the systems which is perhaps even more fundamental. In our system, belief is read into the system as a basic probability assignment on the single large frame. As soon as it enters the system and is combined with the existing basic probability assignment, its impact on *every* proposition in the system is *immediately felt* and all future interaction with the system will include the contribution of this belief. In their system, the identical basic probability assignment is read into the system in terms of the original frame and its effect on the belief in any other frame is only discerned as the user tells the system what frames to transfer it to and what other basic probability assignments to combine it with. Thus their system requires the input of a fourth body of knowledge–the knowledge about how to proceed with the given reasoning problem at hand (this knowledge makes up an **analysis** in their notation)–whereas this knowledge is inherent in our system.

9.8 System Speed and Efficiency

Although our goal in pursuing the D-S theory concentrated on finding a sound and consistent theoretical basis for inexact reasoning, a few observations regarding the computational speed and efficiency of such a system may be made. The use of one large frame of discernment for the entire problem domain suggests that the time required for the reasoning process grows exponentially with the number of propositions in the problem. However, our concept of the use of such a system is that this large frame could be initially defined and assembled for some domain and then stored intact. This frame need not be regenerated for each problem within the domain (although it could still be easily modified and enlarged in the face of new information). Furthermore, only singletons and those propositions which possess either aliases or some basic probability number need be stored. Once this frame is stored, the computational time required for the reasoning process is proportional to the number of propositions in which we have some belief rather than in the total number of propositions in the system. Finally, meta-knowledge can be used were necessary to coarsen frames and thereby reduce their size.

9.9 Exponential Growth Problems

The following example illustrates the creation of a minimal refinement frame from two initial frames:

Example 9.2 Suppose some frame Θ_A consists of singletons $\{a_1, a_2, a_3\}$ and frame Θ_B consists of singletons $\{b_1, b_2, b_3\}$. If we tell the system that a_1 implies b_1, b_1 implies a_1, and so on for a_2, b_2 and a_3, b_3; then the minimal refinement frame $\Theta_C = \Theta_{(A,B)}$ would consist of singletons:

$$\{c_1, c_2, c_3\},$$

$$\text{where } c_1 = a_1 = b_1, c_2 = a_2 = b_2, \text{ and } c_3 = a_3 = b_3.$$

However if we stated that none of the a's implied anything about the b's and vise-versa, the singletons of Θ_C would be:

$$\{c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9\},$$

$$\text{where } c_1 = a_1 \cap b_1, c_2 = a_1 \cap b_2, \ldots, c_4 = a_2 \cap b_1, \ldots \text{etc.}$$

$$\text{also } c_1 \cup c_2 \cup c_3 = a_1, \ldots, c_1 \cup c_4 \cup c_7 = b_1, \ldots, \ldots \text{etc.}$$

This example shows that for one frame of size (or cardinality) n_1 and a second frame of size n_2, the combined frame has a size of between $\max(n_1, n_2)$ and $(n_1 \times n_2)$. That is the combined frame has a size of order $O(n_1 \times n_2)$. More generally, for m frames of size n_i, (for $i = 1$ to m), the size of the resultant full frame is of the order $O(n_1 \times n_2 \times \ldots \times n_m)$. The size of the full frame, then, grows exponentially.

However the size of the resulting frame is not of *direct* consequence because we need not store every proposition in the frame nor even every singleton. We need only store those propositions which have some "name" or some basic probability number assigned to them: neither of these

are functions of the size of the frame. Never-the-less, there are *indirect* reasons for us to be interested in the size of the frame. Specifically, the size of propositions themselves grows. In the example above, the proposition $\{a_1\}$ was represented in the minimal refinement frame by either $\{c_1\}$ or $\{c_1, c_2, c_3\}$, depending on the relationships between Θ_A and Θ_B. In general, adding a frame Θ_A to an existing global frame of size n results in an increase in the size of Θ_A's singletons to the order $O(n)$. This growth has a direct impact on the set theory operations which are used for the calculation of support and plausibility and for Dempster's rule. This exponential growth problem, then, causes inevitable size "explosion" in systems which directly implement a global frame approach.

Although the details will not be provided here, an alternative global frame representation scheme was developed which eliminated growth problems arising from combining frames into one large minimal refinement. Using this approach, however, the application of Dempster's rule itself lead to problems. Specifically, if one basic probability assignment with n focal propositions is combined with a second containing m focal propositions, the resulting basic probability assignment has a number of focal propositions of the order $O(n \times m)$. This exponential growth problem could not be overcome.

9.10 Conclusion

Although the global frame approach is fully consistent with D-S theory and is theoretically feasible, exponential growth problem could not be fully overcome and the entire approach was eventually abandoned in favor of the frame network approach described in the next chapter.

Chapter 10

A Frame Network Approach

In the global frame approach described in the previous chapter, all frames of discernment involved in some problem are combined into one large global frame. It has been shown that although this produces a consistent implementation, it exhibits growth characteristics which severely limit its practicality. A refinement to the global frame approach which overcomes these problematic characteristics is the **frame network approach**. The frame network approach is the basis for the D-S based FRO expert system shell. This chapter described this expert system shell and the theory behind the frame network approach.

10.1 System Overview

The format of our D-S implementation is an expert system shell. With our shell program, the designer of an expert system application (or *knowledge engineer*) works in conjunction with the domain expert to create a knowledge base which describes the expert's solution approach for the problem at hand. A user can then direct the shell program to read and execute this knowledge base file, thus invoking the specific expert system. Our system is written in PROLOG running on an IBM-AT type machine which has proved to be adequate for prototyping.

Concepts used in the knowledge base are represented by D-S frames of discernment; in this chapter we will refer to these simply as frames.

In D-S, belief about the truth of propositions can be represented by several different types of belief expressions. Our system utilizes several of these expressions including basic probability

assignments or BPA's and commonality functions, as well as support and plausibility functions which are used for most input and output. Each of these expressions embody exactly the same information, but the different representations offer different perspectives and advantages. As mentioned previously, we will refer to a body of belief as a ***belief function***, regardless of the belief expression being used to represent it (this contrasts slightly with Shafer's more specific definition of a belief function, [57, p.39]). A belief function can be translated from any one representation expression to any other.

Our implementation approach also employs a variety of techniques for combining belief functions. These include Dempster's rule of combination as well as various dependent combination rules (see section 5.2) which we have developed for use in situations where Dempster's rule is not valid.

The relationships between propositions in different frames are stored in structures which we call ***links***. The system uses links to infer belief about one frame from given belief about another frame. We will show that inference accomplished using links is completely compatible with D-S.

Using the frame and link representations, the knowledge engineer specifies all of the variables required to solve a problem and the appropriate relationships between them; thus establishing a graph structure which we call the ***frame network***. During the execution of the resulting expert system, belief is added to certain frames and propagated throughout the frame network. We refer to the development of a correct and efficient frame network for an expert system as the system ***design***.

We have found that the D-S representation works well for the relational information which makes up the expert system design. However, there is an additional body of knowledge which must go into the creation of an expert system—the *control* knowledge. Control knowledge involves the determination of which questions should be asked, which conclusions should be drawn, and when these actions should be performed. Control knowledge embodies procedural rather than

relational knowledge and is not well suited for D-S representation. We have therefore chosen to represent the control knowledge in a separate, rule-based component of our shell. Since our system combines frame-based design with rule-based control, we have called the program **FRO**, for **F**rame–**R**ule **O**rganizer.

Finally, we have found that the user interface is extremely important in a D-S based system. Our work in this area has lead to useful and relatively simple belief interfaces. We address this issue in chapter 11.

The specifications of our system are outlined in table 10.1. We will now proceed to deal with each of these topics in greater detail.

Table 10.1: Specifications of Our D-S Implementation

Format:	Expert System Shell
Concept Representation:	D-S Frames of Discernment
Belief Representation:	D-S Belief Functions
Inference Operation:	"Links"
Design Representation:	Frame Network
Control Representation:	Separate Rule-Based System

10.2 Frames of Discernment and Belief Functions

The basic form of concept representation in D-S, and correspondingly in our system, is the frame of discernment. The knowledge engineer defines frames by interacting with the system and supplying a name for each frame and for each of the frame's elements or alternatives. These definitions are written out to a file which can later be edited. This knowledge base input scheme, or *system development interface*, represents an attempt to provide a simple, knowledge-based component which queries the knowledge engineer or expert in order to help construct a new expert system (see chapter 15 of the appendix).

Within our shell program, frames are recorded by storing the knowledge engineer's definitions using PROLOG atoms to represent names and PROLOG list structures to represent sets. The frame definitions are also cross referenced with any associated belief functions or links. Throughout the system, propositions are represented by subsets of these initially defined frames—that is, by lists of the element names which were originally entered for each frame. Using this representation for propositions, the set theory operations of union, intersection, exclusion and inclusion which are frequently employed in D-S have been implemented in a straightforward manner.

Any legitimate D-S frame of discernment can be used in our system. Furthermore, propositions of any size or cardinality may be utilized throughout the system. Thus the concept representation scheme used in our system is consistent with D-S and places no limiting restrictions on it. This cannot be said of some other implementations, as in the system proposed by Gordon and Shortliffe [27] which permits the use of only those propositions which form a strict hierarchy (see section 7.3).

Like concept representation, belief representation is performed by our system in strict accordance with D-S. During both the creation of the knowledge base by the knowledge engineer and the execution of the completed expert system by the user, uncertain belief is entered via an interactive graphic interface which employs support and plausibility values (or support intervals) for specific propositions. Our interface techniques are discussed in chapter 11. Immediately after a belief function is entered into our system, it is usually converted to its equivalent BPA form; the format used internally for most belief. BPA's are stored as a list of propositions, each with an accompanying basic probability number. When belief is output to the user, it is usually converted back into a series of support intervals (depending upon the instructions of the knowledge engineer). Thus it is possible for any legitimate D-S belief function to be entered and stored in our system, again providing consistent and un-restricted conformance with D-S.

10.3 Inference and Links

10.3.1 Performing Inference

While frames of discernment and belief functions are explicitly defined by Shafer in reference [57], a method for applying these to a general inexact reasoning procedure applicable to expert systems is not. At the root of this process is simple inference: given some level of belief in proposition A, what should we believe about proposition B? When A and B are both propositions in the same frame, inference is derived from set theory. Belief in A implies belief in B if A is a subset of B, belief in A implies belief in *not* B if A is a subset of the set complement of B. These relationships are inherent to the definition of belief functions and require no specific implementation. However in the general case, A and B may be propositions in different frames and so a general procedure for transferring belief from A to B is required.

We have found that the resolution of this inference process is the primary obstacle to implementing D-S for an expert system. In this section we discuss some of the different methods explored by ourselves and others and we give a detailed account of the technique now used in our system.

We first note that the inference procedure must be based on D-S theory. This is often neglected. Problems with the system proposed by Lu and Stehanou [44], for example, have already been discussed in section 7.4. This is but one illustration of how the use of D-S representations without a D-S based inference procedure can lead to inconsistent results and does not provide the theoretic rigor which we seek.

Performing inference between frames, then, should be derived from formal D-S, even though the theory does not explicitly show how. The theory does outline one process which does not involve the transfer of belief among general frames at all, but rather requires the combination of the frames into one larger one first, and then performs all inference operations upon that single frame. This is the global frame of discernment approach described in the previous chapter.

As shown, global frames suffer from an overwhelming problem of exponential growth in the number of propositions and, although we were able to overcome many aspects of this problem, exponential growth finally caused us to abandon the global frame approach.

10.3.2 Links

A refinement of the global frame method which avoids the exponential growth problem involves producing the minimal refinement of two frames, transferring an input belief function from the first frame to the minimal refinement, and then reducing the belief function from the minimal refinement to the second frame, thereby producing an output belief function containing all belief about the second frame which can legitimately be inferred from the input belief. The minimal refinement frame can then be removed from memory, leaving only the initial frames along with the newly inferred belief. We can accommodate weak inference by adding a belief function to the minimal refinement which contains uncertain belief about the relationships between the frames. Shafer [11] has described an approach to propagation which is similar to this.

We have further modified this process for computational efficiency by deriving a single procedure which produces the identical numerical results without actually creating the minimal refinement frame (The system created by Lowrance et al, section 7.5, employs in an approach which is mathematically equivalent to this). In our system, a belief function defined on one frame is combined with a previously stored set of uncertain relationships to produce an inferred belief function on a second frame. Inference performed in this manner is completely consistent with D-S since its results exactly match those of the minimal refinement approach.

We call the stored set of relationships between two frames a ***link***. Conceptually, a link between two frames contains the same information as a belief function defined on the minimal refinement which describes the relationships between the frames. In section 11.2 we discuss in some detail how such ***relational*** belief functions defined on the minimal refinement frame can be generated and used to perform inference. However, since the minimal refinement is not actually generated

by the system described here, the link must be represented in terms of the initial frames only. In order to represent links in our system, then, we list the names of the two frames involved and, for each element in the first frame, we provide a belief function which describes what we would believe about the second frame if that element were known to be the true value of the first frame. A link, in other words, has the following form:

$$\begin{aligned} \text{link:} \quad & \Theta_A \longrightarrow \Theta_B : \\ & a_1 \longrightarrow S_{\text{link}}^{a_1} \\ & a_2 \longrightarrow S_{\text{link}}^{a_2} \\ & \quad \vdots \\ & a_n \longrightarrow S_{\text{link}}^{a_n} \end{aligned}$$

where:

Θ_A	is the input frame
Θ_B	is the output frame
a_i	is an element of Θ_A
$S_{\text{link}}^{a_i}$	is a "link support function", a belief function on Θ_B represented in the form of a support function which is implied by complete certainty that a_i is the true value for Θ_A

This technique of supplying a link support function for each element in the first frame greatly simplifies the process of creating links. In our system, links are created through interactive sessions with the knowledge engineer and the expert who enter the uncertain link belief through a graphical interface. More importantly, this format is sufficient to represent any possible set of relationships between the propositions in the two frames.

Using this representation for links, the system can transfer belief from one frame to another in the following way:

1. The system recalls the known belief function (in the form of a BPA) for the first frame.

2. The system considers each basic probability number in this BPA. Only the focal propositions (those with non-zero basic probability numbers) need be considered.

3. The system refers to the link to obtain the link support function corresponding to each element in the focal proposition.

4. All of the resulting link support functions for the proposition are combined using the disjunctive combination rule shown in equation 5.9. This combination must be disjunctive because belief exactly for that proposition implies belief that one of the elements in the proposition will be true, though it is not known which one.

5. The single resulting support function on the second frame is weighted according to the basic probability number for the focal proposition on the first frame. This, then, yields the contribution of that focal proposition to the inferred belief about the second frame.

6. These normalized support functions are summed, leaving a single, formally correct and logically implied support function on the second frame.

This process is formalized in the following theorem:

Theorem 10.1 *Suppose that $S_{link}^{a_i}$ is the link support function over an output frame Θ_B for any element a_i in the input frame Θ_A. That is, $S_{link}^{a_i}$ represents what we would believe about frame Θ_B if we knew with certainty that the proposition $\{a_i\}$ was the true value of the input frame Θ_A.*

Further, for any focal proposition A_k in an input BPA m_{in} on Θ_A, where A_k is some disjunctive set $\{a_1 \cup a_2 \cup \cdots \cup a_j \cup \cdots \cup a_n\}$, the link support function for A_k is $S_{link}^{A_k}$ representing the belief about the output frame Θ_B which is implied by certain belief that A_k is the true value of Θ_A. $S_{link}^{A_k}$ can be obtained by combining all $S_{link}^{a_j}$, for $j = 1$ to n, using the dependent disjunctive combination of equation 5.9 and the multiple combination principle of equation 5.14.

Then, the output belief function S_{out} on Θ_B which is implied by the input BPA m_{in} on Θ_A is defined by:

$$S_{out}(B) = \sum_{A_k \in \Theta_A} m_{in}(A_k) \;\; S_{link}^{A_k}(B) \tag{10.1}$$

for all propositions B in Θ_B and all focal propositions A_k in Θ_A.

Note that links, as they are defined here, are only valid for inference in a single direction; from Θ_A to Θ_B but not vice-versa, for example. A pair of links (one in each direction) must be defined if the expert believes that inference in either direction is valid.

Again, it should be emphasized that through the use of the links to represent inference knowledge and the use of the process outlined above to transfer belief, the system can perform inference in accordance with D-S.

10.4 Frame Networks

10.4.1 Belief propagation in frame networks

A D-S based expert system will encompass many frames with numerous links between them. These make up a graph with frames as the nodes and links as the arcs—we call this graph the *frame network* (figure 10.1 illustrates a hypothetical frame network). The frame network is the complete representation of all variables or concepts in the system, and of the logical relationships between them. The network is represented in the system only through its constituent frames and links, it is not treated as a separate entity. Operators act on the network to enter belief functions, to propagate belief around the network, and to retrieve the resulting belief about conclusions. The conceptual aspects of frame networks are further discussed in section 10.4.4, but we first examine the mechanics of propagating belief around the network.

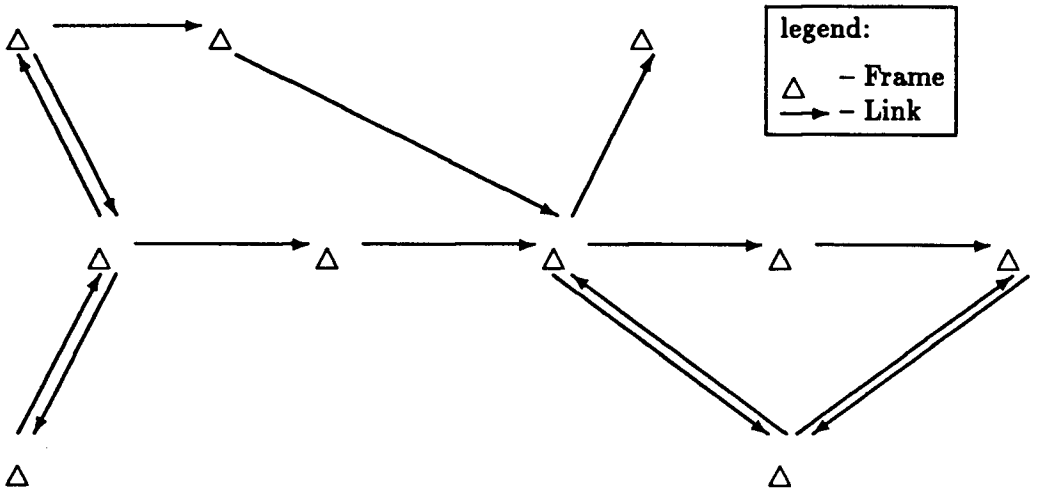


Figure 10.1: Schematic of a Hypothetical Frame Network

When a belief function is added to a frame in the network, the system must combine the new belief with what is already known about the frame. Also, the system must use the links emanating from that frame to perform any legitimate inferences, thereby adding new belief to other frames and causing the cycle to be repeated. The repetition of this process for all appropriate frames is what we refer to as the propagation of belief throughout the network.

In our system, newly added belief is propagated to all relevant frames before it is combined with the previously existing belief. To this end, the system stores two belief functions for each frame. One, the *current belief*, represents belief arising from the most recently added belief function only. The current belief for every frame is reset to represent complete ignorance before a new belief function is added to the system. When belief is added and propagated around the network, its effect on each frame is temporarily stored as the current belief function for that frame. After propagation has been completed, the current belief for each frame is combined with the second belief function, the *total belief*, which represents the resultant of all belief previously added to the system. The resulting belief function is then stored as the new total belief.

When propagating belief throughout the network in this fashion, the following two major problems arise:

1. A belief function entered on one frame by the user can be propagated to another frame via more than one path in the frame network. How should this situation be handled?

2. Belief propagated from frame to frame in the network can become so imprecise as to add no new information to subsequent frames. How can we detect and avoid this unnecessary propagation?

We have developed a belief propagation algorithm which resolves both problems. This algorithm is described in section 10.4.2 and a propagation example is given in section 10.4.3.

10.4.2 Belief propagation algorithm

An outline of our propagation algorithm is provided in PROLOG-like pseudo-code in figure 10.2. This algorithm accomplishes the propagation of a newly added belief function throughout the network.

Referring to figure 10.2, it can be seen that a belief function is obtained from the user (line 2) and combined with the frame's current belief (line 9) using a dependent conjunctive combination scheme (equation 5.8). When the new belief is first propagated to some frame, that frame's current belief function will represent total ignorance and the dependent combination reduces to the trivial case in which the ignorance has no effect on the propagated belief. However if the newly added belief has previously been propagated to the frame by some other path, then that frame's current belief will already represent the effects of the new belief. With two belief functions originating from the same source, the appropriate operation must be a fully dependent conjunctive combination. Thus we have a solution to problem number 1.

The belief function resulting from the dependent combination is compared with the frame's

Line #: (bf = belief function, Capital letters = dummy variables)

```
(1)  enter new belief into network:
(2)    get bf A for frame B from user,
(3)    add bf A to frame B, (line 7)
(4)    combine belief on all frames, (line 27)
(5)    stop.
(6)
(7)  add bf C to frame D:
(8)    get current bf E for frame D,
(9)    dependent combine bf C with bf E to get bf F,
(10)   if bf F equals bf E,
(11)     then terminate propagation and return,
(12)   otherwise,
(13)     store bf F as the new current bf,
(14)     propagate bf F from frame D, (line 17)
(15)     return.
(16)
(17) propagate bf G from frame H:
(18)   if there are no more links from frame H,
(19)     then return,
(20)   otherwise,
(21)     get the next link from frame H which is to frame I,
(22)     transfer bf G from frame H to get new bf J on frame I,
(23)     add bf J to frame I, (line 7)
(24)     propagate bf G from frame H, (line 17)
(25)     return.
(26)
(27) combine belief on all frames:
(28)   for every frame which has been affected,
(29)     combine current belief with total belief,
(30)     store the result as new total belief,
(31)     reset the current belief to ignorance,
(32)     return.
```

Figure 10.2: Belief Propagation Algorithm

current belief function (line 10). If they are the same, then the propagate belief has added no new information to the current belief, and any further propagation would be redundant (line 11). However, if the belief functions differ, then new information has been added and propagation continues outwards from that frame (line 14). This approach is applicable for both the initial propagation of the new belief to any frame and for subsequent propagations to the same frame by way of different network paths. We have, then, an indication of when a particular propagation path has become redundant and can be terminated, thereby solving problem number 2.

We note that both of these propagation problems occur during the special case of a propagation path looping back onto itself. Not only must we accommodate the propagation of a belief function onto the same frame twice, but failure to terminate redundant propagation would result in a continuous processing loop. Fortunately, a propagated belief function which has looped back onto some frame can contain no more precise information than it did on its initial pass. This is because propagation can only result in a loss of precision, it cannot acquire added precision. Thus any loops in the propagation path are immediately terminated by the general propagation algorithm, they need not be given special treatment.

Finally, we note that our propagation algorithm involves the recursion necessary to propagate the belief along every applicable path, regardless of the configuration of the network (lines 23 and 24). In the algorithm, the transfer of belief from one frame to another (line 22) uses the approach described in section 10.3.2 and the combination of the current belief with the total belief (line 29) uses Dempster's rule .

10.4.3 A Propagation Example

The following qualitative example illustrates the propagation process. Suppose that a system contains three frames pertaining to the correction of problems with an ordinary table lamp (see table 10.2). Suppose also that there are appropriate links between these frames. A possible

propagation session might involve the following steps:

Table 10.2: Frames for table lamp example

Frame	Element
A: Light bulb condition	a_1: bulb burned out a_2: bulb O.K.
B: Response to turning switch on	b_1: light goes on b_2: nothing happens
C: Required action for bulb	c_1: replace bulb c_2: no action required

1. The current belief function for each of the three frames is reset to complete ignorance.

2. The user of the system declares with certainty that the light bulb is burned out causing the corresponding belief function to be added to frame A.

3. The newly added belief function is dependently combined with frame A's current belief. Since the current belief represents complete ignorance, the newly added belief function remains unchanged and becomes the new current belief function.

4. This new current belief function for frame A is propagated to frame B resulting in a belief function which shows certainty for b_2 (that is, certainty that nothing happens when the switch is turned on).

5. The newly added belief function on frame B becomes the new current belief function after being unchanged by the dependent combination process.

6. This new belief about the response to the light switch is, in turn, transferred to frame C creating a belief function which shows some level of belief for c_1. However the links do not allow the system to infer with certainty that the bulb should be replaced since, based only on the response to the switch, we are not sure that the problem lies with the bulb.

7. Again, the newly added belief becomes the new current belief function for frame C.

8. The belief function is further propagated back to frame A. During this transfer, the belief which is already less than certain becomes even less precise. This is because knowledge that the bulb should be replaced does not categorically imply that the bulb is burned out (it could, for example, require replacing because it's the wrong wattage).

9. This time, dependent combination of the newly propagated belief function with frame A's current belief function results in no change to the current belief. Thus the propagated belief has added no new information and the propagation path can terminate. This is an occurrence of the circular propagation problem and its solution as described in section 10.4.2.

10. Next, the system returns to step 4 but this time it propagates the belief along the second path to frame C. In this case, the knowledge of a burned out bulb results in certain support for the bulb's replacement.

11. The newly propagated belief function is combined with frame C's current belief function which, although it no longer represents complete ignorance, is still sufficiently imprecise so as to cause no change in the propagated belief, which therefore becomes the new current belief function. This is an occurrence of propagation problem number 1 and its solution through the use of dependent combination.

12. Next, the belief is further propagated to frame B. Since knowledge that the bulb should be replaced does not necessarily imply that nothing will happen when the switch is turned on (again the bulb might be the wrong wattage, for example), the belief function transferred to frame B will convey less than certain support for b_2.

13. The dependent combination of this newly propagated belief with the current belief function for frame B will not alter the current belief (which already shows certain support for b_2). Therefore this propagation path will terminate, as will the entire propagation process since this was the last possible propagation path. This illustrates propagation problem number 2 and its solution.

14. Finally, the current belief functions are combined with the total belief on each frame using Dempster's rule. The total belief functions will then represent the resultant of all belief in the system and will show with certainty, for example, that the bulb is burned out, that it should be replaced, and that nothing will happen when the switch is turned on.

10.4.4 Network Design

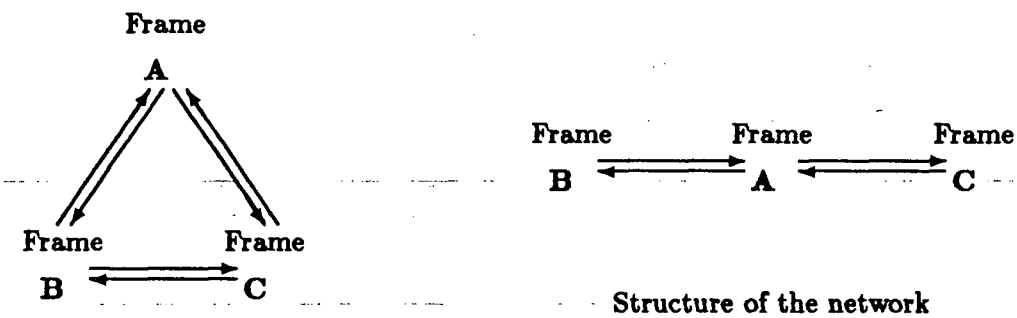
In creating an expert system, the knowledge engineer and the expert must decide what the specific goal of the system is, what the important variables of the problem are, how these variables are related to each other, and how knowledge about these variables can lead to useful conclusions. We refer to the solution of these collective issues as the *design* of the expert system. The frame network can be said to be the representation of the system's design. It is important to recognize that the design of a system is not merely a listing of all possible frames and links; it also involves ensuring that the network is complete but not redundant, that it is organized in such a way that the belief can be propagated efficiently, and that issues of dependency are addressed.

Of course, a complete and non-redundant network must be achieved through a detailed study of the particular problem domain. Shafer points out that "In Dempster-Shafer theory, the design comes from an analysis of our evidence" [11, p.122]. A possible interpretation of this concept is that we must wait for a specific domain situation to arise before we can begin to create our design. The system described by Lowrance et al [42] adopts just such an approach. However we contend that in an engineering expert system context, the knowledge engineer and the expert can anticipate what bodies of evidence might reasonably be brought to bear on the problem, and can therefore base the design on these potential sources of information.

Perhaps the most important factor in creating an efficient design is to take advantage of every opportunity to reduce the frame network from a general graph structure to a tree structure. While neither structure poses a theoretical problem for the system, trees provide significantly

less possible propagation paths and therefore they require much less time to process. The elimination of any redundant frames and links will contribute to a tree-like structure, as will the exploitation of conditional independence.

While the process of creating a link between two frames is not constrained by issues of dependency, it can be observed that complete independence between two frames is properly represented by the absence of a link. Of coarse, this provides the most computationally efficient situation and this efficiency is delivered from conditional as well as absolute independence. For example, the scenario described in section 10.4.3 involves three frames. If a link existed in both directions between each of the frames, there would be a total of six links making up a graph structure. However, the required action for the light bulb (frame C) is completely independent of the response to turning on the switch (frame B) if we already know the condition of the bulb (frame A). Thus C is independent of B conditional on A: we need not provide the links between C and B and the structure of the network is reduced to that of a chain (see figure 10.3). The appropriate exploitation of conditional independence, then, can lead to significant increases in efficiency.



General graph structure for frame network.

Structure of the network simplified though the use of conditional independence.

Figure 10.3: Simplification of the Network Structure through the use of Conditional Independence

Dependance plays a second role in the expert system design. Recall that when belief is added to the system, it is first propagated and then combined with the previously known belief using Dempster's rule. Dempster's rule assumes independence of the input belief functions, and therefore our system is valid only if each belief function added to the system is completely independent of every other input. One way to help ensure that this condition is met is to ensure that the frames which will be used for input from the user are independent of each other. For example, designing the system to accept input for both of the two dependent variables "month of the year" and "season" would likely lead to the input of dependent belief functions; designing for input about both "month of the year" and "day of the week" would not. The application of this principle will also precipitate a more tree-like network with the accompanying benefits in efficiency.

10.5 Control

The design of the expert system knowledge base encompasses the specification of the frames and links in the frame network. It can be said that the design captures the ***relational knowledge.*** However the relational knowledge is only a portion of the total knowledge required. We must also include information about what specific questions to ask and when to ask them. We must specify when an acceptable conclusion has been reached and how to report the conclusion. In short, we must provide the ***procedural knowledge.*** Whereas relational knowledge is captured by the system design, procedural knowledge is captured by the system *control.*

A frame network representation is well suited for the design knowledge, but it does not seem to be a useful representation for the control information. Rather than trying to force the control knowledge into an inappropriate representation, we have chosen to handle control in an entirely separate system which interfaces with the frame network. For simplicity, we have chosen a rule-based representation for control.

The knowledge engineer and the expert, then, incorporate control knowledge into the expert

system by creating a second knowledge base file which is made up of rule and question definition statements. The function of these statements is to ask the user for belief input at the appropriate time and in the appropriate manner; to pass this information on to the frame network; to obtain belief about conclusions from the network; and to use this information to guide subsequent processing or to inform the user of the appropriate conclusions and recommendations. In order to achieve these functions, our system provides a set of interface routines between the control rules and the design frame network which are used in conjunction with the features normally found in backwards-chaining rule-based expert system shells. It is also possible for the control component to interact with external processing routines to obtain additional certain or uncertain information. This is particularly useful for engineering situations where the results of traditional computer programs are a normal component of the expert decision making process. We have found that the separate rule-based component provides a simple yet powerful approach to the management of system control. Figure 10.4 depicts the various components of our system.

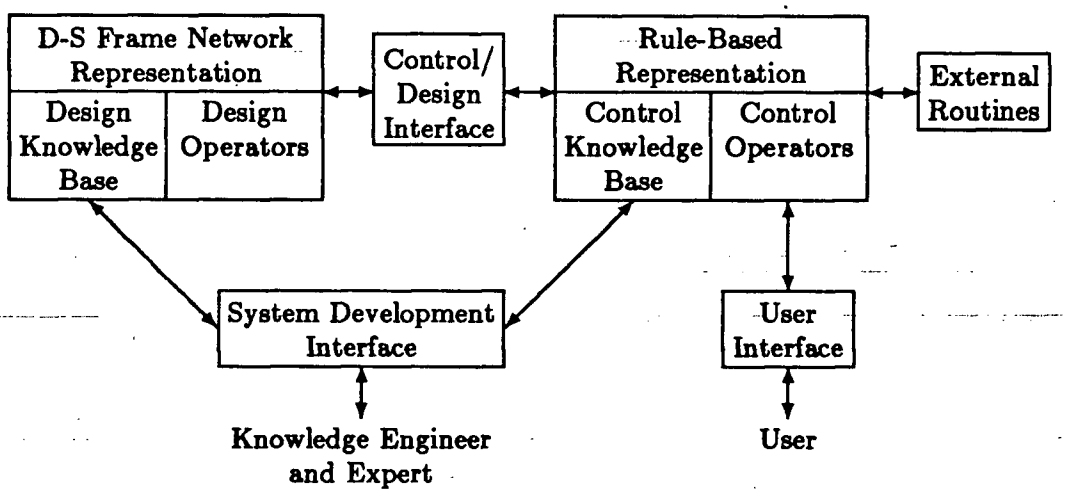


Figure 10.4: Components of Our Expert System Shell

10.6 Further Enhancements

We have produced a prototype expert system shell based on the implementation approach described above. This system has operated successfully for some small initial problems and we are encouraged by the results to date. We see the next steps along this line of inquiry to be as follows:

At present, our system uses approximations to equations 5.8 and 5.9. These approximations are implemented in terms of basic probability numbers for the focal propositions only. As such, they greatly simplify the calculations. In future work, we hope to reformulate the exact equations 5.8 and 5.9 in such a way as to allow their efficient implementation in the system.

The dependent conjunctive combination (equation 5.8) used in the propagation algorithm could be further utilized by enabling the creation of pre-specified *dependent links* in the frame network. A dependent link could be used to relate frames which are expected to receive dependent belief inputs. By employing dependent combination rather than Dempster's rule to merge the current and the total belief functions, a dependent link could eliminate the requirement that all belief functions entered into the system be independent.

Another feature which would promote efficient knowledge base design is a technique for handling conjunctive inference components. That is, in order to represent the assertion that "A and B imply C," where A, B and C are propositions in different frames, the present system requires the knowledge engineer to define a minimal refinement of the frames containing A and B, to link these initial frames to the refinement, and then to link the refinement to the frame containing C. While this is not a theoretical problem with the scheme proposed here, the process could be streamlined to provide much more efficiency in defining and processing such relationships.

Further, a technique for tracking the reasoning performed by the system and reporting the reasoning to the user would be very useful. Such a capability would assist in debugging the system, in explaining to the user how conclusions where reached, and in providing the user with

some feeling for the sensitivity of the reasoning to various belief inputs.

Finally, the entire approach must be tested with problems of increasing size and complexity in order to determine what type of constraints are placed on the system by practical processing limits. In turn, work will continue on enabling the system to operate efficiently with large knowledge bases.

10.7 Conclusion

This chapter has provided an overview of a frame network D-S implementation approach and has examined details of strategies for overcoming certain obstacles. Namely, descriptions have been provided of the representation of the various types of knowledge, of the approach to D-S based inference and belief propagation, and of both system design and control issues. The D-S implementation approach outlined in this chapter has been implemented in the FRO expert system shell. Finally, several issues which have yet to be explored were outlined.

Chapter 11

Working with Belief Functions

The previous chapter outlined the theoretical approach and the implementation details for representing knowledge and for performing reasoning in a frame network D-S system. It has been shown that this type of a system offers extremely advanced uncertainty handling capabilities. However it must be realized that advanced expert system features will find very little usage if the are notably more complicated to use than existing systems. Thus D-S uncertainty handling capabilities must be accompanied by interface techniques which are simple and straight-forward to use without restricting the system's power. This chapter provides an examination of high level interface procedures. Section 11.1 presents an graphical interactive belief interface system called IBIS which enables users to communicate their uncertain belief to the system in a simple manner. Section 11.2 described how knowledge about the relationships between frames can be specified and entered into the system. Finally, section 11.3 describes how conclusions expressed in terms of belief functions can be evaluated by the system.

11.1 Entering Belief Functions

11.1.1 Theoretical Basis for a Simple Belief Input System

It was shown in section 4.4 that specifying the support for the focal propositions only is sufficient to completely specify any support function. Yet asking a user to express support values for each focal proposition is not a sufficiently simple way of entering belief for two main reasons. First, when belief is expressed in terms of support rather then basic probability numbers, it

is not intuitively obvious which propositions are focal. Second, this approach involves the specification of support for joint propositions. We have found joint propositions to be awkward to deal with; both for implementing an interface design and for expressing one's belief. In terms of implementation, difficulties arise because the names of joint propositions can consist of lengthy lists of element names and they must be entered by the user rather then selected the from a list or menu (since there are generally too many propositions to offer in a list). With regards to expressing belief, we have found that asserting belief for a singleton (i.e. belief that a particular alternative is the correct answer) is far more natural for users then asserting belief for a joint proposition (i.e. belief that one of a set of alternatives is the correct answer). It would be preferable, then, to enter belief only at the singleton level. It is vital to realize that this does not mean we wish to revert back to a conventional discrete probability assignment for representing belief. Rather we simply wish to imply our joint proposition belief by asserting our singleton belief.

In order to achieve the desired simplicity, then, we would like to devise an input scheme which consists only of the user entering support intervals for singletons. This section examines whether such an approach is theoretically consistent–or if not, what restrictions must be imposed to make it so.

Singleton support intervals can be calculated for any support function. Therefore, if we can show that any particular set of singleton support intervals can arise only from one unique and derivable support function, we will know that our simple scheme is consistent and sufficient to specify and enter any general support function. Belief which leads to support for some singleton must be assigned exactly to that singleton. Thus singleton support is associated with a unique belief assignment. However, some or all of the user's belief may not be assigned to singletons. Such belief will manifest itself as uncertainty for the singletons (that is, as a difference between their plausibility and support values). Singleton uncertainty cannot be traced to a unique belief assignment. For example, suppose that we have two quite different support functions on the

frame $\{a, b, c\}$. The first assigns all belief to the proposition $\{a, b, c\}$. The second assigns equal belief to the three propositions $\{a, b\}$, $\{a, c\}$, and $\{b, c\}$. Both functions will lead to support intervals of $[0, 1]$ for each of the three singletons. We therefore need to impose some additional requirements in order to obtain a unique solution.

Suppose, then, that out of all the possible support functions which could produce these support intervals, we select the one which expresses the least internal conflict. We will argue that this is both a reasonable and appropriate choice.

This one support function with minimum uncertainty conflict will, in fact, exhibit no conflict at all; and the belief leading to singleton uncertainty can thus be termed consonant according to the definition in section 4.5.4. We therefore refer to this conflict minimization assumption as the assumption of consonant uncertainty. We believe that this assumption is not at all unrealistic. In fact, it has been suggested (see Cohen [9] and Shackle [56]) that all belief should be consonant, since some body of evidence should not simultaneously lead to two mutually exclusive conclusions. While Shafer disagrees with this restriction, it does seem that it will be the exception which will violate consonance to any large degree. Furthermore, consonant uncertainty is less restrictive than full consonance since belief assigned to singletons may be disconsonant under the former category.

Through the application of a single reasonable restriction, then, the consonant uncertainty principle achieves the outstanding practical result of always producing a unique support function from a set of legitimate singleton support intervals. This entire approach is analogous to statements by Shafer [57, p.221] in which he shows that a truly consonant belief function can be determined by the plausibilities awarded to the singletons, and that a consonant belief function is essentially a point function rather than a set function. Note that the approach described here is somewhat less constrained than fully consonant and that singleton support intervals are required to define a belief function rather than just singleton plausibility values.

Our approach to belief input, then, is to first allow the user to enter support intervals for every

singleton in a frame of discernment, and then convert this input into the corresponding unique support function by invoking consonant uncertainty in a manner which can be formalized as follows:

The elements of a frame are first relabelled to reflect the ranking of the singleton uncertainties. Thus, the elements are relabelled $\theta_1, \theta_2, \theta_3, \ldots, \theta_n$ such that:

$$\begin{aligned} U(\{\theta_1\}) &= U(\{\theta_2\}) \\ U(\{\theta_2\}) &> U(\{\theta_3\}) \\ U(\{\theta_3\}) &> U(\{\theta_3\}) \\ \vdots & \quad \vdots \\ U(\{\theta_{n-1}\}) &> U(\{\theta_n\}) \end{aligned} \tag{11.1}$$

Then the BPA values for the joint propositions are:

$$\begin{aligned} m(\{\theta_1, \theta_2, \theta_3, \ldots, \theta_n\}) &= U(\{\theta_n\}) \\ m(\{\theta_1, \theta_2, \theta_3, \ldots, \theta_{n-1}\}) &= U(\{\theta_{n-1}\}) - U(\{\theta_n\}) \\ \vdots & \quad \vdots \\ m(\{\theta_1, \theta_2\}) &= U(\{\theta_2\}) - U(\{\theta_3\}) \end{aligned} \tag{11.2}$$

These basic probability numbers, then, will always form nested propositions as required by consonance. (Note that the next equation in this sequence would be $m(\{\theta_1\}) = S(\{\theta_1\}) = U(\{\theta_1\}) - U(\{\theta_2\})$ which would result in the assignment of singleton support greater than that specified by the user. In such a case the derived support function would not reflect the user's belief. This situation is therefore prevented by the restriction that $U(\{\theta_1\}) = U(\{\theta_2\})$, as shown above).

The basic probability assignment is completed by including the basic probability numbers arising from the singleton support values, that is:

$$m(\{\theta_i\}) = S(\{\theta_i\}), \ \forall i \tag{11.3}$$

Example 11.1 Suppose that the user has entered the following singleton support intervals for a frame containing the four elements $\{a, b, c, d\}$:

$$\begin{aligned} S(\{a\}) &= 0.1 & Pl(\{a\}) &= 0.7 \\ S(\{b\}) &= 0.3 & Pl(\{b\}) &= 0.9 \\ S(\{c\}) &= 0.0 & Pl(\{c\}) &= 0.5 \\ S(\{d\}) &= 0.0 & Pl(\{d\}) &= 0.2 \end{aligned}$$

The singleton support values lead to the following basic probability numbers:

$$\begin{aligned} m(\{a\}) &= S(\{a\}) = 0.1 \\ m(\{b\}) &= S(\{b\}) = 0.3 \\ m(\{c\}) &= S(\{c\}) = 0.0 \quad (ignore) \\ m(\{d\}) &= S(\{d\}) = 0.0 \quad (ignore) \end{aligned}$$

Further, we can calculate the following singleton uncertainties:

$$\begin{aligned} U(\{a\}) &= 0.6 \\ U(\{b\}) &= 0.6 \\ U(\{c\}) &= 0.5 \\ U(\{d\}) &= 0.2 \end{aligned}$$

Ranking these uncertainties leads to the relabelling of the elements $\theta_1 = a$, $\theta_2 = b$, $\theta_3 = c$, and $\theta_4 = d$. Equation 11.2 then gives:

$$\begin{aligned} m(\{\theta_1, \theta_2, \theta_3, \theta_4\}) &= U(\{\theta_4\}) &&= 0.2 \\ m(\{\theta_1, \theta_2, \theta_3\}) &= U(\{\theta_3\}) - U(\{\theta_4\}) &&= 0.5 - 0.2 = 0.3 \\ m(\{\theta_1, \theta_2\}) &= U(\{\theta_2\}) - U(\{\theta_3\}) &&= 0.6 - 0.5 = 0.1 \end{aligned}$$

Thus the complete basic probability assignment derived from the user's input of singleton supports intervals is:

$$\begin{aligned} m(\{a\}) &= 0.1 \\ m(\{b\}) &= 0.3 \\ m(\{a,b\}) &= 0.1 \\ m(\{a,b,c\}) &= 0.3 \\ m(\{a,b,c,d\}) &= 0.2 \end{aligned}$$

with all other propositions having zero basic probability.

11.1.2 Attributes for the Implementation of a Belief Input Scheme

We have determined that a simple and consistent belief input scheme can be achieved through the application of consonant uncertainty. This section describes additional attributes which should be incorporated into a belief interface scheme. The implementation of these ideas into an actual belief entry scheme are described in section 11.1.3. The first attributes deal with the efficient management of constraints:

1. Exploit applicable classes of belief:

   The belief input scheme proposed in the previous section relies for its effectiveness on the constraints imposed by consonant uncertainty. In general, each additional constraint imposed results in a simpler input scheme, since constraints limit the user's options. Whenever the belief to be entered into a system is known or expected to belong to a certain class of support functions (classes of belief were discussed in section 4.5), then, the constraints which can be ascribed to those classes should be exploited in the input scheme. If belief is to be entered for which no uncertainty should exist, for example, the response need only consist of selecting one of the alternatives and no support or plausibility values are required.

2. Avoid overconstraint:

   It is very important, however, not to unnecessarily overconstrain the input. For example, imposing the class of certain support when the user is, in fact, quite uncertain about the answer to some question would result in a very inappropriate support function being entered into the system. Classes of belief should only be exploited when there is a reasonable expectation that they will apply (and even then it should be possible to override such constraints).

The following attributes deal with the actual processes of entering values into the computer program:

1. Allow user to select from options:

   A user interface will be generally be much simpler to use if the user is allowed to select options offered to him rather than being required to remember the correct format, syntax, order, and so on for the values being entered. This will only be possible when there are not too many options from which to choose, but the restriction to singleton support intervals only should ensure that this condition is met. Since belief input is specified in terms of real number values, allowing the user to select a value is best achieved through the use of a graphical interface. We propose that support and plausibility values be entered by allowing the user to adjust bars on a bar chart to the desired magnitude.

2. Establish a valid support function in response to every incremental belief entry:

   To encourage convergence upon a satisfactory support function it is preferable that, during the entire process of providing inputs of belief values for a single support function, the consequences of the each entry are presented in the context of a valid support function. This requirement also applies to the initialized state of a support function before the first user belief input is entered. In most cases this requirement is met with a belief function

corresponding to complete ignorance–a belief assignment of 1.0 to the full frame, and a consequent plausibility of 1.0 for all singletons.

Furthermore, if a class of belief has been selected, the support function should be automatically constrained accordingly. As well, the system should automatically reject, with an explanatory warning, any attempt to enter values which violate the general requirements of a support function or any constraint applicable to a chosen belief class. For example, if a consonant belief structure had been selected then, any attempt to enter belief in a second singleton, or any attempt to deviate from a plausibility value of 1.0 for the focal singleton should be resisted.

3. Permit support and plausibility to be increased or decreased:

   The entry of support or plausibility implies the act of adding belief or plausibility to propositions, usually singletons. It is essential that it also be possible to deduct support or plausibility from any singleton or proposition. This provides an opportunity to correct entries which are sensed to be inappropriate. It also allows, in conjunction with the immediate adjustment of the support function to constraints as suggested above, some indication of the sensitivity of the complete support function to current belief value entry.

11.1.3 The Prototype System

In the previous section we have discussed what we feel to be the important attributes of a support function interface system. In this section we give a description of the system developed for use in our prototype D-S based expert system shell. We show how our system attempts to fulfill the ideal attributes given above and present our plans for some future modifications to our system. This system is called the Interactive Belief Interface System, or **IBIS**.

Questions which allow uncertain responses return a complete support function rather than just

a single value. Internally they might be defined as:

```
question 13:
  weather_tomorrow
    ask_uncertainty "What will tomorrow's weather be like?"
       alternatives (sunny, cloudy, raining, snowing).
```

In this case, the alternatives are a set of mutually exclusive and exhaustive propositions for a previously stored frame "weather_tomorrow". When the expert system needs to know what the expected weather will be for the next day, the question is presented graphically to the user in the initial form shown in figure 11.1:

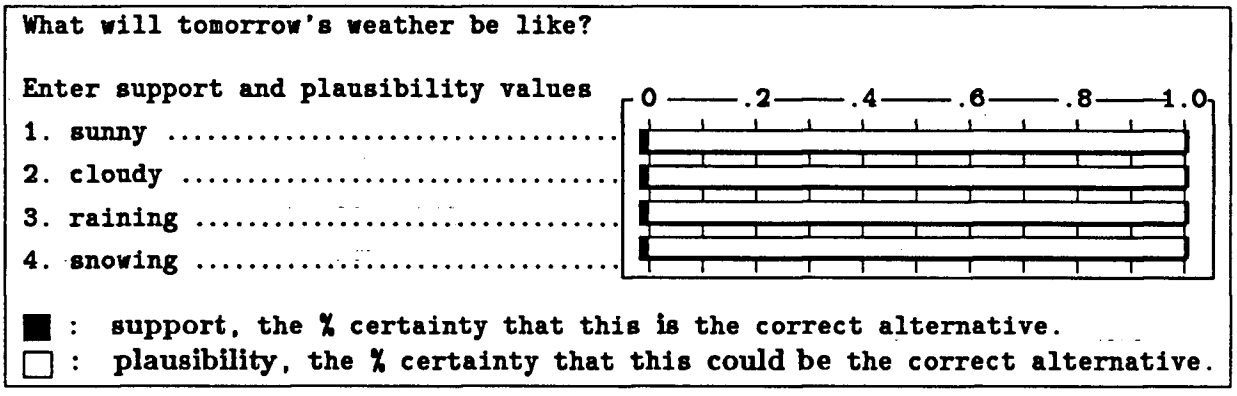


Figure 11.1: Initial belief input screen

This screen states the question, lists the possible alternatives, and displays the support and plausibility values associated with each alternative in the form of a bar-chart. The support and plausibility values presented initially reflect a belief of absolute ignorance regarding the question. The user can alter the values on the bar chart by first selecting the corresponding support or plausibility bar with a mouse cursor. The support or plausibility represented by the bar can then be decreased or increased by moving the cursor to the left or right. The system employs the previously discussed simplifying technique of confining entries of supports and plausibilities to singletons.

Visualization of the support function is encouraged by providing a fully interactive graphical interface. The system automatically and immediately applies the appropriate constraints and warns the user of attempted violations. For example, the requirement that the sum of all implied BPN's must equal one is automatically maintained. If the first entry made was to increase the support for tomorrow's weather being cloudy up to 0.4, the remaining uncommitted belief must be 0.6 and plausibility in all other alternatives must equal 0.6. Thus the plausibilities for all other weather alternatives will be reduced to 0.6 and the input scheme automatically responds with the display shown in figure 11.2:

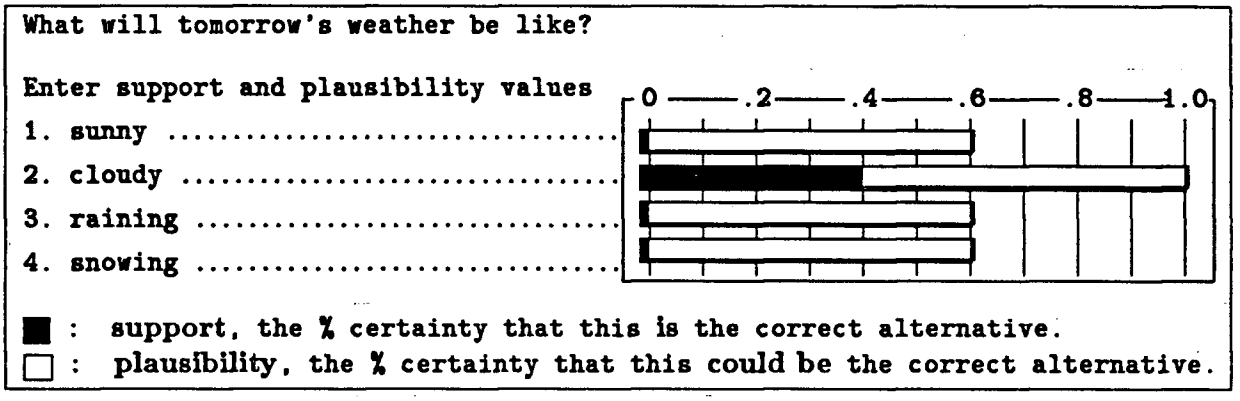


Figure 11.2: Belief input screen expressing some support for "cloudy"

Other support function constraints, such as the belief cannot be greater than the plausibility for that proposition, are automatically checked and enforced. If the user attempts to violate these constraints, a warning is given and he is not allowed to make the incorrect entry. For example, if the user attempts to reduce the plausibility of cloudy weather to less than 0.4, the following warning would appear, as shown in figure 11.3:

The above mentioned constraints are fundamental to support function theory–all legitimate support functions must obey them. After the user is satisfied with the values displayed and actuates the "Click here when finished" box, the system takes the users input, applies the consonant uncertainty constraint and derives the complete support function. This is entered

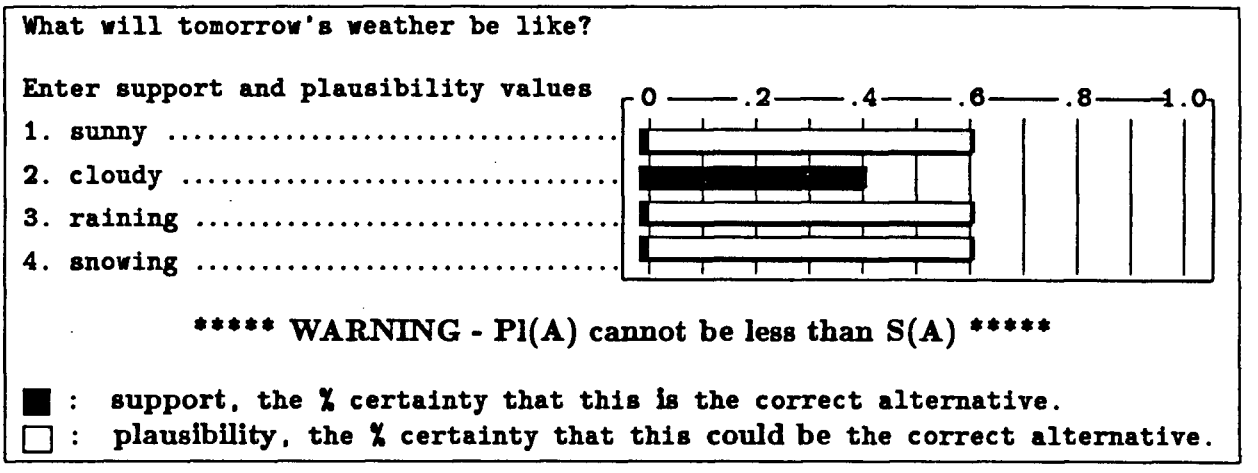


Figure 11.3: Belief input screen warning of constraint

internally to the expert system but, if requested, the focal joint and singleton propositions with their corresponding basic probability numbers can also be displayed on screen. This reinforces the fact that neither consonant uncertainty, nor the input system itself, confine the focal propositions to singletons. For example, if the user is quite confident that tomorrow's weather will be raining or snowing but he has no idea which, he would enter this belief by reducing the plausibility of both sunny and cloudy while leaving full plausibility for raining or snowing. Such an entry would be as shown in figure 11.4:

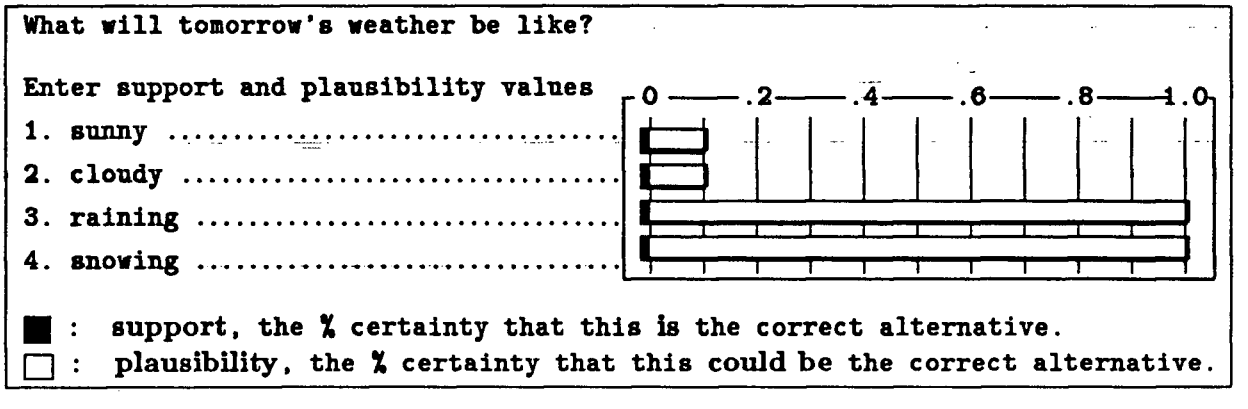


Figure 11.4: An example belief input screen

Such an entry would cause the following support function to be entered into the system:

$$m(\{raining, snowing\}) = 0.9$$
$$m(\{sunny, cloudy, raining, snowing\}) = 0.1$$

The IBIS interface is illustrated and described further in chapter 12 and in chapter 14 of the appendix.

11.1.4 Additional Features

This section discusses some features which are not currently implemented in the IBIS system, but which could be added to enhance its performance.

Additional belief classes could be investigated, and possibly subclasses governed by belief and plausibility profile shapes, and the circumstances which would prompt their use. For example, in the case of propositions which are formed by discretization along a continuum, a subclass might be specified which requires the belief and plausibility distributions to be unimodal. When the knowledge engineer programs questions into the knowledge base, he would have the option of stating that the response should fit into one of a group of belief classes or possibly subclasses. In turn the user would respond with his choice of belief class or subclass before entering beliefs. Once a belief class has been selected then the appropriate additional constraints would be applied during input and when calculating the complete support function once input is completed. An option for the user to enter a completely general support function is not precluded but, because of the high dimensionality of the inputs involved, this would probably necessitate a non-graphics based entry scheme and would require the user to enter basic probability numbers for all focal singleton and joint propositions.

It is possible that even greater feedback can be provided to the user during belief entry. A measure of the degree of conflict within the support function as it is being entered could easily be provided. Similarly, a measure of the quality of information contained in the support function

(as described in section 11.3) might be useful under some circumstances. Once the user became familiar with these feedback parameters, he could use them as a check of whether his input corresponded well with his feelings about the question, and he could modify his response if this was not the case.

It has been very apparent through using the input scheme described that it rapidly improves familiarity with the D-S expression of belief. An effective tutorial program could be built around the graphical input scheme and would demonstrate the concept of D-S belief, the consequences of belief combination, and so on. This would be invaluable for building the users confidence in expressing his beliefs.

Finally, all of these additional features could be made optional and passive (i.e. they would only be provided if requested). The system has the virtue of being functional with the simplest of inputs and only an elementary understanding of belief by the user. Refinement of the interface to capture more subtle nuances of belief and provide greater feedback for the more sophisticated user should not be at the expense of this simplicity when it is appropriate.

11.2 Expressing Relational Belief

Belief which must be entered into a D-S based expert system can be classified as either **evidential belief** or **relational belief**. Evidential belief expresses some body of evidence pertaining to the true value of some frame of discernment. This type of belief is most commonly supplied by the user in response to some question. In the FRO expert system shell, evidential belief is entered using the IBIS system described in the previous section. Relational belief, on the other hand, is belief about how propositions in one frame are related to propositions in a different frame. This type of belief is usually entered by the expert and the knowledge engineer during the creation of the knowledge base. This section provides some insight into the meaning of relation belief functions by showing how they can be expressed in the form of simple "IF...THEN..." rules.

Formally, a relational belief function is defined on a minimal refinement frame. Therefore the function assigns belief to elements of the form (x, y), or "x is the true value of frame Θ_X and y is the true value of the frame Θ_Y." It can be shown that this is a suitable representation for relational information. However, it is not an intuitive way of describing our relational belief. We would like to find an equivalent form which would allow us to give a more natural expression of our belief about relationships.

We have found that the most common form of representing relationships in expert systems, namely "IF...THEN..." type rules, provides a representation form which is acceptable to both the expert's intuition and the belief function theory. Specifically, suppose we have the rule "if A then B" where A is a proposition in the frame Θ_A and B is a proposition from the frame Θ_B. This rule is equivalent to the conditional statement "if A is true then B is true", which is equivalent to the unconditional statement "A is true and B is true, or not A is true." In this last statement, the "A is true and B is true" component is directly equivalent to the proposition (A, B) in the minimal refinement frame $\Theta_{(A,B)}$. The "not A is true" component corresponds exactly to the set complement of proposition A in the initial frame Θ_A, and must therefore exist in the minimal refinement which completely discerns the initial frames. This conversion from rules to minimal refinement propositions is shown in example 11.2.

Example 11.2 For the frames given in example 3.2, we could define a rule:

"If rock, then blast."

This rule is equivalent to the statement:

"If rock is true, then blast is true."

Which is equivalent to the statement:

"rock is true and blast is true, or not rock is true."

Which is equivalent to the statement:

"rock is true and blast is true, or soil is true."

Which is equivalent to the statement:

"rock is true and blast is true,
or soil is true and blast is true,
or soil is true and dig is true."

Which is equivalent to the proposition:

$$\{(rock, blast), (soil, blast), (soil, dig)\}$$

Thus the rule "If rock, then blast" corresponds directly to the proposition $\{(rock, blast), (soil, blast), (soil, dig)\}$ in the frame $\Theta_{(MATERIAL,METHOD)}$.

Using this conversion from rules to minimal refinement propositions, we can see that a rule statement exhibits the same basic set theory characteristics as propositions. For example, an "A_1 or A_2" expression in a rule corresponds to the union of the propositions A_1 and A_2, while an "A_1 and A_2" expression in a rule corresponds to the intersection of A_1 and A_2. It should be noted that we can refer to an "A and B" expression where the propositions A and B are from different initial frames. For example, we could use a rule "if A and B then C" (where A is a proposition in frame Θ_A, B in frame Θ_B and C in frame Θ_C) by relating the minimal refinement of frames Θ_A and Θ_B to frame Θ_C; i.e. the rule would be equivalent to a proposition on the frame $\Theta_{((A,B,C)}$.

We can also see that one rule can imply another rule in a manner analogous to the set inclusion property of propositions, as shown in the following example. This concept becomes important when we begin to assign belief to rules (as shown below) and we must observe the same constraints as for evidential belief functions.

Example 11.3 Suppose we have three rules:

Rule 1: if A then B_1 and B_2

Rule 2: if A then B_1

Rule 3: if A then B_1 or B_3

Where A is any proposition in frame Θ_A and B_1, B_2 and B_3 are any propositions in frame Θ_B. These rules can be converted into propositions on the frame $\Theta_{(A,B)}$. Thus if Rule 1 yielded some proposition labeled $(A,B)_1$, Rule 2 yielded $(A,B)_2$ and Rule 3 yielded $(A,B)_3$, it can then be shown that:

$$(A,B)_1 \subset (A,B)_2 \subset (A,B)_3$$

or alternatively:

$$(A,B)_1 \text{ implies } (A,B)_2 \text{ which implies } (A,B)_3$$

Therefore, we can say that:

Rule 1 implies Rule 2 which implies Rule 3

This result can also be derived directly from the logical relationships between the rules. For example, if the rule "if A then B_1 and B_2" is true, then the less precise rule "if A then B_1" must also be true; that is:

"If A then B_1 and B_2"

implies "If A then B_1"

which similarly implies "If A then B_1 or B_3"

The same argument can be used to show that:

"If A_1 or A_2 then B"

implies "If A_1 then B"

which implies "If A_1 and A_2 then B"

Using this approach, we can define an approach to specifying relation belief which consists of supplying a series of rules pertaining to two related frames of discernment, with a single support number assigned to each rule (each rule would represent a focal proposition in the relational belief function, so specifying the support for each rule would fully specify the belief function). A rule with support of 1.0, for example, would apply categorically, while a rule with near zero support would result in only negligible inferences. Example 11.4 shows a set of rules with associated support numbers and the relational belief function (in the form of a BPA).

Example 11.4 Suppose the possible types of ground on a construction site are rock, soil, or sand:

$$\text{frame } \Theta_{MATERIAL} = \{rock, soil, sand\}$$

and the possible excavation methods are digging, blasting, or ripping:

$$\text{frame } \Theta_{METHOD} = \{dig, blast, rip\}$$

Then the rules:

If rock then blast, with support 0.5,

If rock then blast or rip, with support 0.8

if soil or sand then dig, with support 0.9.

could be transformed into a relational belief function on the minimal refinement frame $\Theta_{(MATERIAL,METHOD)}$ which, when converted to a BPA, would give us the following belief:

Support associated with rules carry the normal constraints for belief functions. For example, the support must be a number on the range 0 to 1.0; the support for one rule can not exceed the support for a second rule if the first rule implies the second; and the sum of support numbers

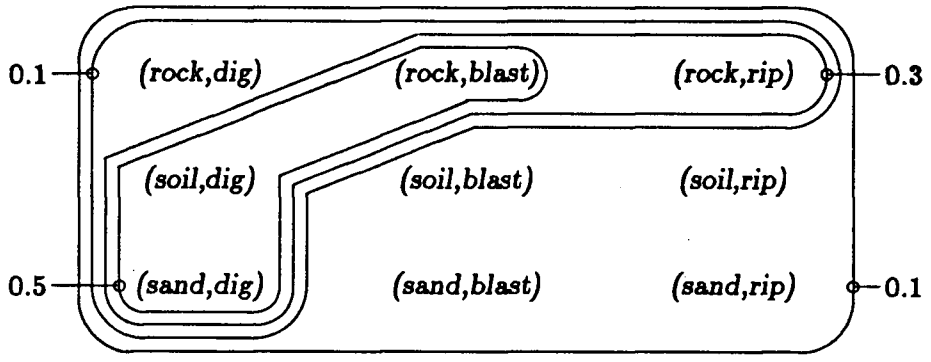


Figure 11.5: A Relational BPA

for complementary rules (i.e. two rules for which the intersection equals $\emptyset$) must not exceed 1.0. Furthermore, a vacuous relational belief function (i.e. a complete lack of information about a set of relationships) corresponds to the case of completely independent frames of discernment.

Finally, we should note that a relational belief function provides knowledge which can allow belief propagation between two frames ***in either direction.*** This can be a problem if we wish to be able to infer belief about a frame Θ_B from the elements of a frame Θ_A, but we do not believe that reverse inferences should be made. Suppose, for example, that in the scenario outlined in example 11.4, we had no information as to the ground type, but we decided to try a ripping method of excavation because we owned the appropriate equipment. Then, knowing the excavation method, the given relational belief function would yield some unfounded belief in the ground type.

This problem stems from the fact that a rule of the type "if rock then rip" is a subset of a rule which would *fully* describe the known relationships, for example:

"if rock
or appropriate equipment is available
or other factors exist which imply rip
then rip"

It can be shown that the rule "if rock then rip" is a completely valid and sufficient subset for belief propagation from frame Θ_A to frame Θ_B, but that the rule gives incorrect relationships when propagating belief in the opposite direction. We have found that a feasible solution to this problem is to provide a different relational belief function for propagating belief in each direction. In this system, then, belief about frame Θ_A would be combined with the stored Θ_A-Θ_B relational belief function to derive belief about Θ_B. Conversely, belief about frame Θ_B would be combined with a *completely different* Θ_B-Θ_A relational belief function to derive belief about frame Θ_A.

In summary, relational belief can be expressed in terms of simple rules. This form provides the most insight into the meaning of relation belief functions expressed on a minimal refinement frame. However the FRO frame network system employs a more simple form of relation belief input using IBIS screens in a similar manner to the technique used for entering evidential belief. The theoretical basis for this approach was described in section 10.3.2 and the resulting implementation is illustrated in chapter 12 and in chapter 15 of the appendix.

11.3 Evaluating Output Belief Functions

The role of belief functions for belief input has been discussed. In a frame network system, belief functions are also produced as output. Given belief output, the main question to be resolved is usually to determine the "best" alternative. It is also important to know how reliable this selection of best is. It is not obvious how the information contained in support and plausibility numbers should be used to make these determinations. The following provides some possible approaches when the output is provided in the previously described form of singleton support intervals.

11.3.1 Determining the BEST Alternative

If there were no uncertainty in a belief function, all alternatives would have a single support number. The alternative with the highest support could then easily be chosen as best (this is analogous to a Bayesian probability distribution). Similarly, if each alternative has only plausibility numbers and no direct support, we could simply take the alternative with the highest plausibility as the best. In general, however, we need a single measure of belief which is a function of *both* the support and plausibility. Suppose we call this parameter the **nominal belief** of the proposition, $NB(A)$:

$$\text{Best } A = A \text{ with } \max(NB(A)) \tag{11.4}$$

$$\text{where } NB(A) = F_{(S(A),Pl(A))} \tag{11.5}$$

Suppose we take the nominal belief to be the simple linear average of the support and plausibility numbers:

$$NB(A) = F_{(S(A),Pl(A))} = \frac{S(A) + Pl(A)}{2} \tag{11.6}$$

Figure 11.6 shows some examples of nominal belief.

This equation for nominal belief is completely arbitrary but it seems to be suitable for a first attempt. For example, given two propositions A and B, there are five possible types of belief relationships between them. We can check the results of using a linear averaging equation for each of these general classes, as shown in table 11.1.

(X)	$S(X)$	$NB(X)$	$Pl(X)$
A	0.0	0.0	0.0
B	0.0	0.4	0.8
C	0.4	0.6	0.8
D	0.7	0.7	0.7

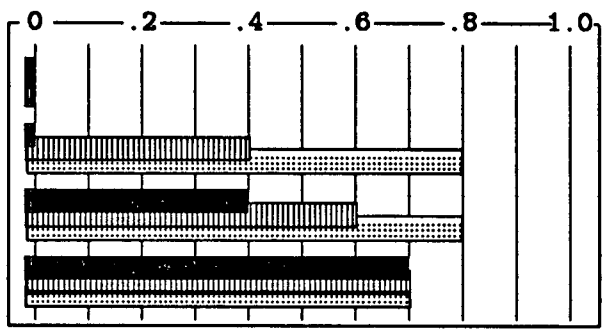


Figure 11.6: Examples of Nominal Belief

Table 11.1: Possible belief relationships

Relationships between belief for A and B	Relative values using: $NB(X) = \frac{S(X)+PL(X)}{2}$
$S(A) = S(B)$, $PL(A) = PL(B)$	A equal to B
$S(A) = S(B)$, $PL(A) > PL(B)$	A better than B
$S(A) > S(B)$, $PL(A) = PL(B)$	A better than B
$S(A) > S(B)$, $PL(A) > PL(B)$	A better than B
$S(A) > S(B)$, $PL(A) < PL(B)$	depends
(and opposites which are symmetric)	

The first four cases shown in table 11.1 are correct by inspection. The last case (which is probably atypical in practice) does not have an obvious result. However we cannot simply say that the proposition is better whenever the support is higher because if $S(A) > S(B)$ and $Pl(A) < Pl(B)$ then there is both more support for A and more support against A. The use of a linear average for the nominal belief will cause the support for and the support against a proposition to carry equal weight in resolving these cases. Other combining functions could place greater emphasis on one or the other of these supports (which is not likely to be valid). In any event, the determination of the nominal belief upon which we base our selection of best alternative is not particularly critical if we have a reasonably accurate method of determining our confidence in (or reliability of) the "best" selection, as shown in the next section.

11.3.2 Determining the Reliability of the Best Alternative Selection

For some belief functions it may be very easy to determine which alternative is best, but we may have a very low confidence about the selection. There are two sources of such doubt. First, we can lack confidence if there is much uncertainty in the belief function. Figure 11.7 shows such a situation. Second, we can be doubtful if the categorical belief for the best alternative is only marginally higher than it is for the next best, as in figure 11.8.

Ronald Yager [73] discusses both of these source of doubt. He introduces two parameters which describe how informative a belief function is.

First, he introduces the **specificity**, Sm, of a belief function as a measure of the degree to which belief is assigned to large sets of elements. That is, the specificity measures the precision expressed in the belief function, with imprecision contributing to uncertainty.

Second, Yager proposes the **entropy**, Em, as a measure of the degree to which the belief is distributed among disjoint sets. The following phenomena are all manifestations of this type of

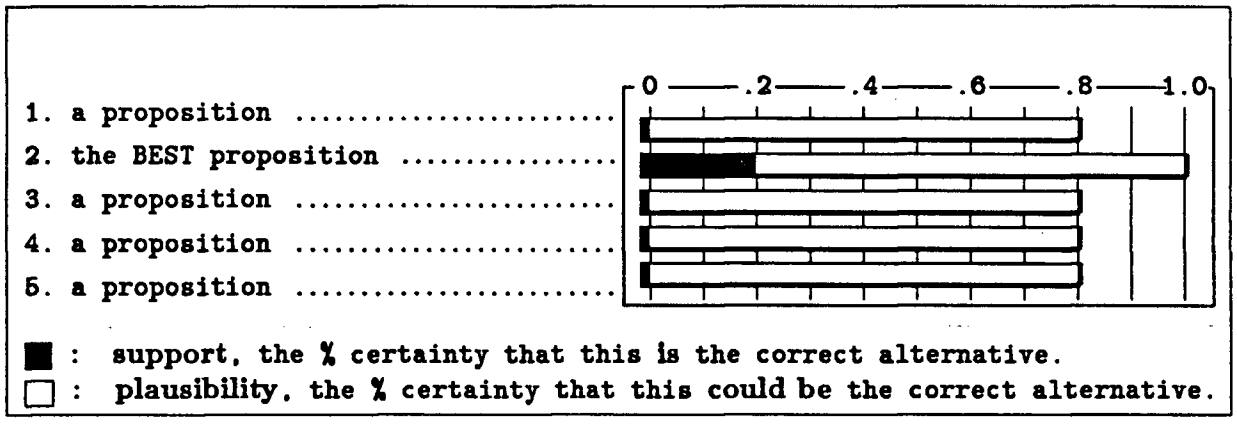


Figure 11.7: Low Confidence in the BEST alternative because of a large amount of uncertainty in the belief function

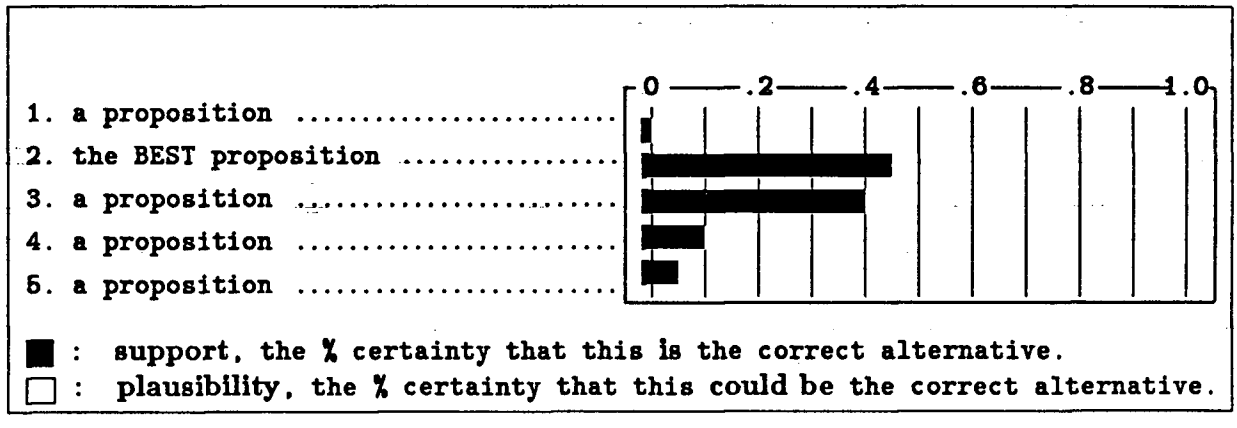


Figure 11.8: Low Confidence in the BEST alternative because of a large amount of disconsonance in the belief function

doubt: a belief function which distributes belief among disjoint sets; one which discerns significant conflict; one which is disconsonant; and one in which there is relatively small differences in support between the main focal propositions.

A measure of our confidence in the selection of best alternative, then, must combine measures of uncertainty and consonance; or entropy and specificity to use Yager's terms. Yager derived specific formulae for deriving his specificity and entropy parameters, however he was using them as measures of how informative overall belief functions are. We are interested in finding parameters which tell us how confident our assignment of the best alternative is. This goal requires different formula for specificity and entropy parameters than Yager's.

1. Measuring Specificity:

   A measure of specificity must indicate the relative amount of uncertainty assigned to our best alternative. A useful definition seems to be:

   > The specificity $Sp(A)$ of the best alternative A is the percentage of the belief function's uncertainty which could be resolved against A before A is no longer considered "best".

   However this is not a very easy definition to implement, we have rather adopted the definition:

   > The specificity $Sp(A)$ of the best alternative A is the support for A as a percentage of the plausibility for A:

$$Sp(A) = \frac{S(A)}{Pl(A)} \tag{11.7}$$

   This is an arbitrary definition, but it reasonably exhibits the expected traits of being directly proportional with the amount of uncertainty in the proposition and being zero for total ignorance and one for total certainty.

2. Measuring Entropy:

The measure of entropy must indicate the degree to which the belief in the best alternative is greater than the belief for other alternatives. A simple definition is:

> The entropy $E(A)$ of the best alternative A is the difference between the nominal belief in A and the nominal belief in the next best alternative, B, as a percentage of the nominal belief in A:

$$E(A) = \frac{NB(A) - NB(B)}{NB(A)} \tag{11.8}$$

Again, this arbitrary definition satisfies the expected traits of being directly proportional to the difference between the best and next best alternatives, being zero for two or more "equally good" propositions and one for total full support of a single proposition.

3. Combining Specificity and Entropy:

The specificity and the entropy can be combined into a single measure of the quality of the "best" selection. We want our quality parameter, $Q(A)$, to range from zero to one and to be directly proportional to both the specificity and the entropy. Figures 11.9 through 11.12 show four arbitrary formulae for this combination. For a given E and Sp, the value of Q can be obtained from the graph's "quality contours" for the following formulae:

$$Q^2 = \frac{Sp^2 + E^2}{2} \tag{11.9}$$

$$Q = \frac{Sp + E}{2} \tag{11.10}$$

$$Q = Sp \times E \tag{11.11}$$

$$Q^2 = Sp \times E \tag{11.12}$$

Again, the specific formula is not especially critical because the way we use the quality (to compare it with thresholds etc.) will vary with which ever characteristics the parameter possesses. Nevertheless, we should try to make the quality parameter intuitively appropriate. For this reason, the formulae shown in figures 11.9 and 11.10 should be rejected because a relatively high Q can be obtained when one of the E or Sp is high even if the other is zero. The formulae illustrated in figure 11.11 seems more reasonable. Notice that both E and Sp need to be high in order for a moderate Q to be obtained. The formulae illustrated in figure 11.12 is more lenient in that a moderate E and a moderate Sp can lead to a moderate Q. Testing could be performed to show which alternative more closely mimics human feeling. The approach shown in figure 11.11 has been implemented in the FRO system illustrates the best combination.

11.4 Conclusion

This chapter has outlined techniques for working with belief functions which simplify both evidential and relational belief input and which help to evaluate inferred belief functions. These simplifying procedures are required to counteract the increase in complexity caused by the use of a powerful and complex knowledge representation and reasoning scheme like D-S. Most of the approaches outlined in this chapter are implemented in the FRO expert system shell.

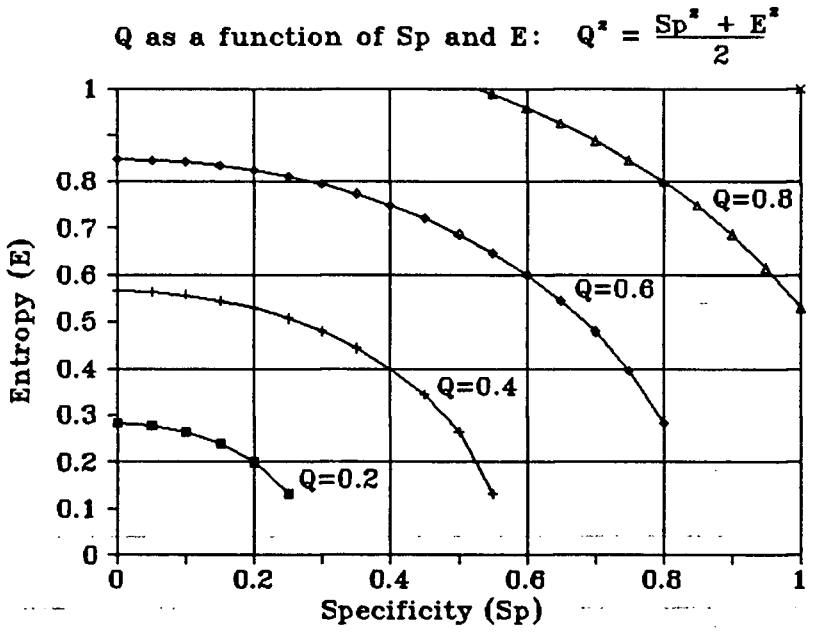


Figure 11.9: Quality factor from equation 11.9

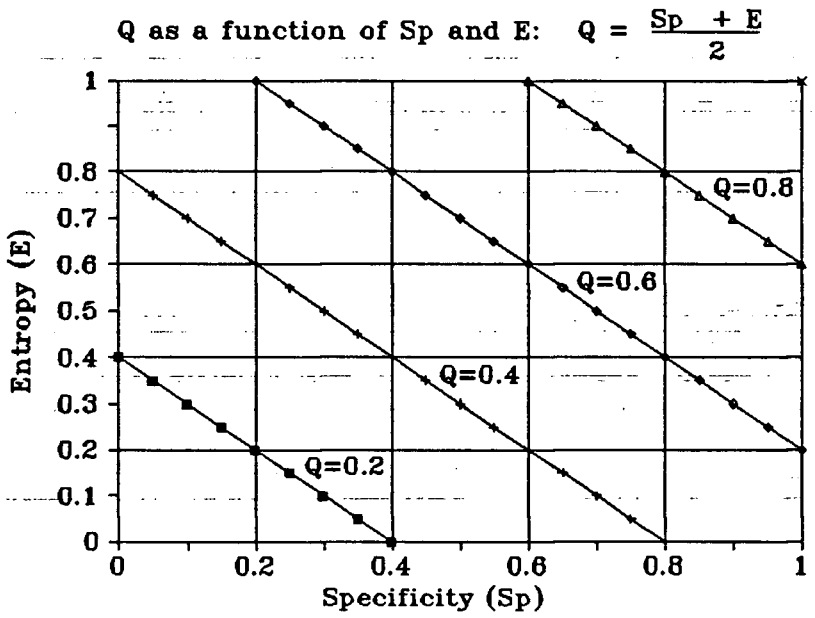


Figure 11.10: Quality factor from equation 11.10

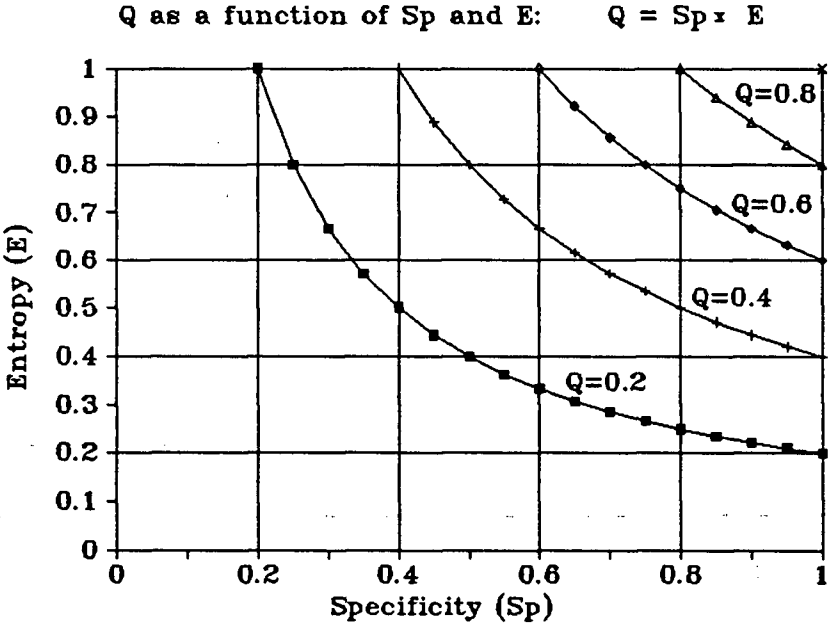


Figure 11.11: Quality factor from equation 11.11

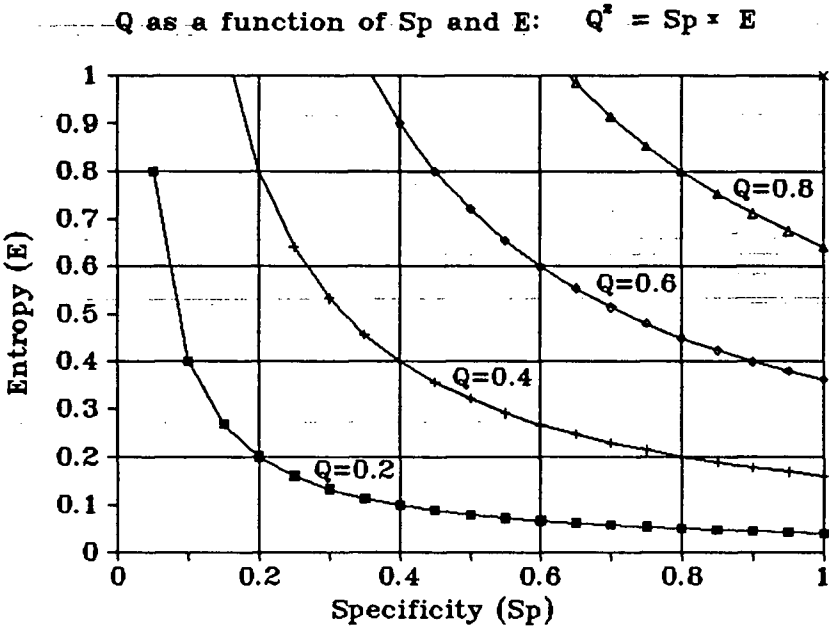


Figure 11.12: Quality factor from equation 11.12

Chapter 12

Using a D-S based System

This chapter provides an extended example of an expert system based on the D-S frame network approach such as that implemented in the FRO shell program. The point of view for this chapter is mainly that of the user. Some issues of incorporating uncertainty in knowledge base design are also discussed in section 12.5.

12.1 Illustration of D-S Implementation

Consider a scenario in which an innovative contractor is selecting his construction strategy for a high-rise project to be built in Vancouver, B.C. A preliminary schedule indicates that the concrete superstructure will be built from mid-November through to mid-February. The contract stipulates that time is of the essence and that days lost due to cold and excessively wet weather must be minimized. The contractor's expert system has been appraised of the time frame and contractual obligations and has initiated a series of questions and responses dealing with cold weather concrete practices (for the reader unfamiliar with Vancouver weather, the climate is temperate and extended periods of substantially below-freezing temperatures are unusual). The goal of the session is to identify the most suitable strategy to achieve acceptable cure times in order to maintain a four day cycle time. The system starts by querying the user regarding long range weather forecasts. Specifically, the system might first pose the following question:

```
What is the long range forecast for cold weather conditions
for the construction period Nov. 15 1988 to Feb. 15 1989 ?
```

The system also offers the user a number of possible answers to the question. In this case, the following forecast descriptions might be presented:

```
1. Mild.
2. Normal.
3. Severe.
```

"Explanation" capabilities built into the system could define the specific meteorological meanings of each of these descriptions.

In this format, the set of alternatives for a particular question makes up a D-S frame of discernment while each individual alternative is a singleton. Since frame of discernment should be exhaustive and mutually exclusive, one and only one of the alternatives must be the correct answer.

If the user was completely certain that one of these alternatives was the correct answer to the question, then he could simply select that alternative. However, as is often the case in civil engineering problems and construction ones in particular, the user may not be completely certain. Thus his response could only accurately reflect his belief if it conveyed his uncertainty about the answer.

The system, then, must be able to accept the user's uncertain input. With a numerical scheme such as D-S, this input can become a fairly complex procedure, so it is critical that the uncertainty input method of any expert system be as simple and as easily understood as possible. We believe that considerable attention must be directed to developing interfaces which are natural to engineering and construction practitioners. Our approach to this problem is to use a graphical representation and to add certain minor constraints to the general D-S theory (see

the description of the IBIS input system in section 11.1). The input format developed is shown in figure 12.1.

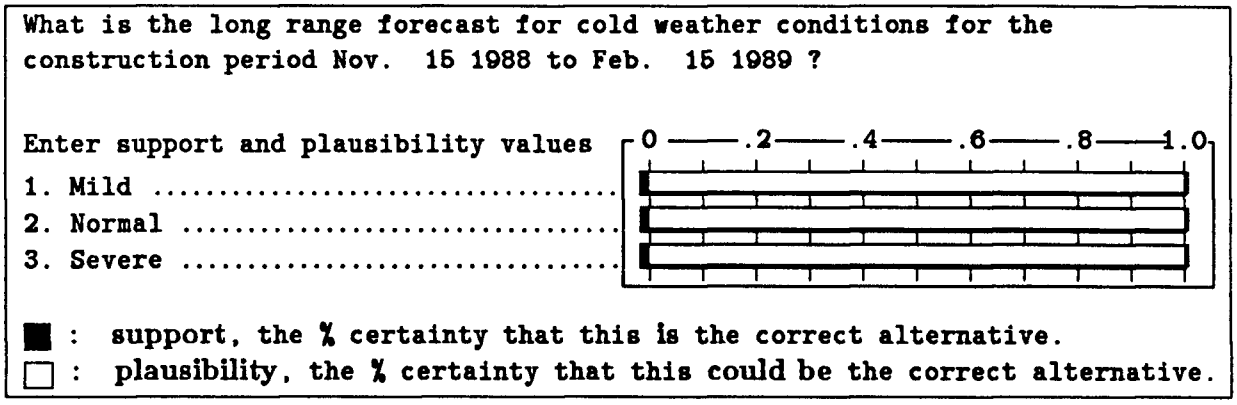


Figure 12.1: Initial uncertainty input screen

In this form, the same question is asked, the same alternatives are offered, but instead of asking the user to select the single best alternative, the system now asks the user to enter two values for each alternative. These values, represented by a bar chart in figure 12.1, are defined as follows:

Support: a number between zero and one reflecting the user's belief that the particular alternative *is* the correct alternative. The support is represented by a solid black bar in figure 12.1 which shows a value of zero for each alternative.

Plausibility: a number between zero and one reflecting the user's belief that the particular alternative *could be* the correct alternative. The plausibility is represented by a white bar in figure 12.1 which shows a value of 1.0 for each alternative.

The support and plausibility values for each alternative of a particular question make up a **belief function** for that frame of discernment.

The fact that every alternative listed in figure 12.1 has zero support and full (1.0) plausibility indicates that none of the alternatives are known to be the one correct alternative, yet each

is believed to be entirely possible. This state accurately represents total ignorance about the question (the appropriate status of the system before the user has entered any belief). In contrast, a Bayesian based system would overstate what is known by assigning an equal, *non-zero* level of belief to each alternative even though none had received any belief from the user. We will elaborate on the definitions of support and plausibility numbers and on their relationship to probabilities below, but first we will extend our example in order to illustrate the use of belief functions.

Our user may decide that based on the unusually hot weather experienced in Vancouver during the summer, he expects the winter to be a mild one. He therefore increases the support value for the "mild" alternative (in the case of our system, this is done by sliding the corresponding bar with the mouse cursor). Because the contractor realizes that this is not a very reliable estimate of the winter's weather, he may only increase the support to 0.2. Suppose that the user also decides to obtain better information by telephoning Environment Canada. He is informed by these experts that all meteorological indications point to a very normal winter. This influences him to increase the support value for the "normal" alternative of the question to a value of, say, 0.5.

The result of adjusting the support values for these two alternatives is shown in figure 12.2. Notice that by expressing some support for one particular alternative being the correct answer, the user implies the same amount of belief that all of the other alternatives must *not* be correct. This results in the lowered plausibility values shown in figure 12.2. The user does not have to consciously make this adjustment since it is performed by the system automatically.

Finally, from both his own observations and from the advice of Environment Canada, the contractor is fairly confident that the winter's weather will not be severe. He expresses this by lowering the plausibility of the "severe" alternative to a low value of 0.1. Figure 12.3 shows the resulting levels of belief. It can now be said that the values shown are an accurate reflection of the contractor's belief about the long range weather forecast; these numbers can therefore be

returned to the system as the complete response to the question.

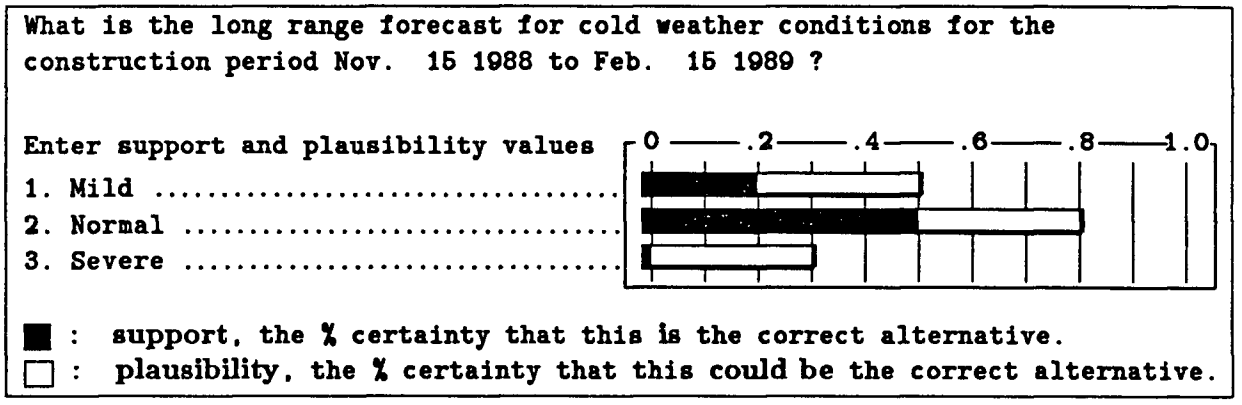


Figure 12.2: Intermediate uncertainty input screen

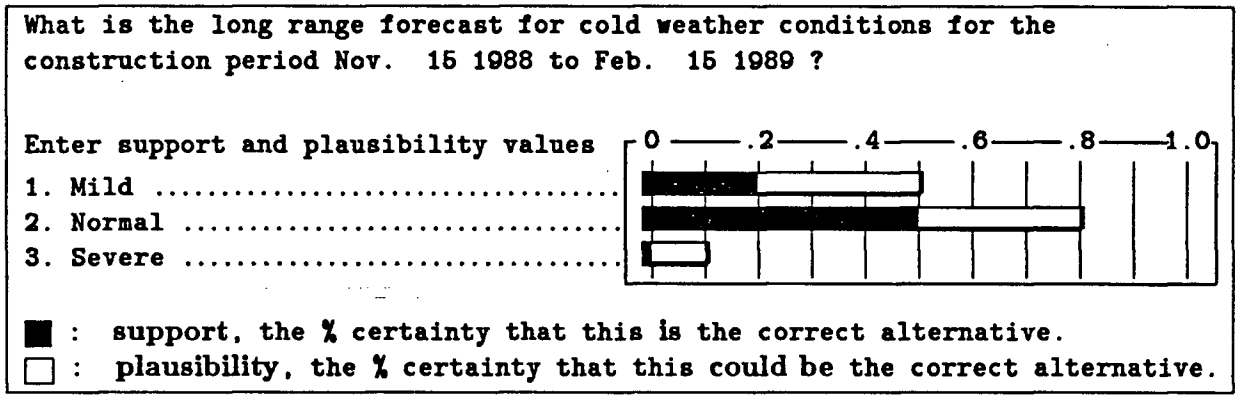


Figure 12.3: Final uncertainty input screen

12.2 A Note On Belief Functions

Belief functions expressed using the system described must conform to the D-S definitions described in earlier chapters. As such, they obey the following equations:

If:

a_i = one of the alternatives

$\overline{a_i}$ = *not* a_i = the set containing all alternatives except a_i

$S_{(a_i)}$ = the support for a_i

$Pl_{(a_i)}$ = the plausibility for a_i

Then entering belief in our system will be governed by:

$$0 \leq S_{(a_i)} \leq Pl_{(a_i)} \leq 1.0 \tag{12.1}$$

$$\sum_{\forall a_i} S_{(a_i)} \leq 1.0 \tag{12.2}$$

$$Pl_{(a_i)} = 1 - S_{(\overline{a_i})} = 1 - \sum_{\forall a_j, i \neq j} S_{(a_j)} \tag{12.3}$$

Even the simple set of relationships given here need not be learnt by a user of the system, however. This is because the system automatically adjusts the plausibility values to account for equation 12.3 (as described above) in addition to issuing warning messages if the user tries to violate the constraints imposed by equations 12.1 or 12.2.

Although this input scheme is fairly easy to use, it is extremely powerful in terms of the spectrum of belief that can be expressed. To illustrate this flexibility, consider the following classes of belief:

Ignorance: We have shown in figure 12.1 that our input scheme can be used to represent ignorance about a question. It is worth noting that in the absence of any support for some alternative, we can still differentiate between disbelief (belief that the alternative is not correct) and ignorance (a lack of belief for or against the alternative). In a belief function, disbelief is manifested as zero support and zero plausibility while ignorance is expressed by zero support and high plausibility. No such distinctions can be made by assigning a single probability value to each alternative.

Certainty: Complete certainty about a single alternative is represented by increasing the support for that alternative to 1.0. This automatically forces the plausibility of all other

alternatives to zero. We can also express certainty that the correct answer lies within some particular sub-group of alternatives, though we may have no belief about which of these alternatives is the best answer. This is represented by a plausibility of 1.0 for each alternative in the sub-group and plausibility of zero for each alternative not in the sub-group. The ability to deal with sets of alternatives is a particular strength of the D-S theory (although our representation tends to de-emphasize this strength by using the list of singletons as the basic input format, as opposed to one which allows the explicit assignment of support values to sets of alternatives).

Consonance and Disconsonance: Figure 12.3 illustrates typical belief which lies between complete ignorance and complete certainty. It can be observed that this belief function displays "conflict" in that the support for the mild weather alternative says that the weather will be mild and therefore *not normal* (or severe). This conflicts with the other support value which dictates that the weather *will be normal.* Belief functions are said to be **consonant** if they show no such conflict and **disconsonant** if they do. Consonance is normally appropriate for expressing belief which stems from a single body of evidence, since such evidence should not simultaneously point to two mutually exclusive conclusions. Disconsonance is appropriate in the case of our example, however, since the user was expressing his combined belief based on two distinct and conflicting sources on information.

A Bayesian probability distribution can be entered into the system by entering $S_{(a_i)} = Pl_{(a_i)} =$ *Probability of* a_i for each alternative. However such a belief function displays complete disconsonance. This may be appropriate for statistical information based on large numbers of observed outcomes, but it does not seem to be appropriate for the general case.

In our example, the user considers two distinct bodies of evidence in order to answer the question. He enters into the system his combined belief about the answer, based on these two

information sources. It is interesting to note that the expert system in this example could equally have been designed to ask the user two separate questions relating to these two specific bodies of evidence (e.g. "What has the summer's weather been like?"), in which case the system would have performed the process of combining the two sources of belief. There are no fixed boundaries between what information should be combined in the mind of the user and what information should be combined by the system, rather the question depends mainly on which portions of the entire reasoning process the system designer wishes to capture in the expert system. We feel that the flexibility in this boundary between the user's role and the system's role provides some indication of success in the modeling of human reasoning characteristics by the system.

To summarize our belief input methodology, the system starts by listing the mutually exclusive and exhaustive alternative answers to some question and by assigning ignorance values to each corresponding $S_{(a_i)}$ and $Pl_{(a_i)}$. The user then responds to the system by altering these values to accurately model his uncertain belief about which alternative is the correct answer.

12.3 Illustration of Reasoning with a D-S Representation

Using the representation scheme outlined in the previous section, we now give a brief overview of how the D-S theory can be used to perform reasoning in an expert system. We begin with three steps which, taken together, comprise the process of creating the system's knowledge base.

First, a frame of discernment is defined for each item in a particular problem that might be uncertain. This involves providing a name for the frame and a list of the possible alternatives. In our example, the previously described frame would have the name "`winter weather`" and the following list of alternatives:

```
mild,
normal,
severe.
```

Based on the advice of a human expert, a second frame describing all practical cold weather concreting strategies might have the name "**concrete strategy**" and the following list of possibilities:

```
use standard mix design only,
modify mix design only,
modify mix & use selective heating,
some mix modification & full heat.
```

Second, any logical relationships between different frames are defined. This step is analogous to specifying the rules in a rule-based expert system. For example, we might want to express the relationship that if the weather is mild, use a standard mix design only. This type of relationship is recorded in the form of a **link** between the two frames. In this case, the result of creating a link is that any belief in mild weather would automatically lead to belief in using standard mix design only.

Third, levels of belief can be assigned to the actual links if the relationships which they represent are themselves not certain. Like the belief values entered by the user in response to questions, these levels of belief in links make up true D-S belief functions. This is illustrated by the fact that levels of belief can be obtained from the expert in the same graphical format as shown above (figure 12.4 shows a belief input screen that could have been used to construct the expert system knowledge base described in our example).

By answering this question, the expert conveys to the system his belief about the relationship between mild winter weather and the best concreting strategy (a similar process would be

followed for normal and severe weather forecasts in order to completely define the relationships between winter weather and concreting strategy).

These three steps are provided by the expert and the knowledge engineer, or designer of the expert system, who work together to create the system's knowledge base. The final ingredient required to perform reasoning is a method for combining belief functions. This procedure is provided by Dempster's combination rule, which is incorporated into the expert system shell.

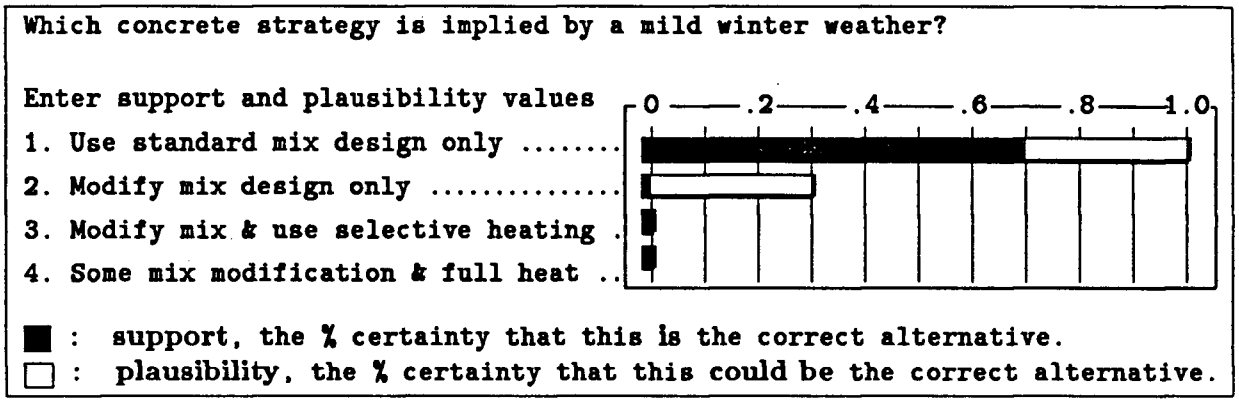


Figure 12.4: Link uncertainty input screen

All of the necessary components are now in place. The expert system can proceed to infer belief in the second frame from the user's belief about the first by performing the following procedures:

1. Obtain input from the user in the form of belief functions.

2. Use Dempster's combination rule to combine these input belief functions with those describing belief in the link relationships, resulting in a new function describing belief about the related frame.

In our simple example, this is all that is required to produce the required results about preferred concreting techniques. However the general case would involve many more frames of discernment with numerous interconnecting links. The general reasoning capability needed to

cope with such a "frame network" is constructed from the foregoing two procedures by adding the following steps:

3. After inferring a new belief function (as in step 2), combine it with any belief which may have previously been assigned to the second frame.

4. Repeat the above procedures until all possible inferences have been made.

The result of this process will be belief profiles for a number of possible conclusions. One of the strengths of this scheme is that it doesn't merely produce a single "best" conclusion by aggregating all uncertainty. Rather it simply represents the uncertainty in each conclusion, enabling the system designer to decide how best to proceed in the face of the resulting uncertainty.

12.4 Example Problem

To conclude, we will show exactly how the example scenario discussed throughout this paper would be incorporated into an actual expert system using our shell. First, the contents of the D-S knowledge base would be entered. The knowledge, as previously shown, is expressed in the form of two frame definitions and the link relationship between them. These are shown here in a form which can be interpreted by our shell:

```
1. frame: winter weather = mild, normal, or severe.

2. frame: concrete strategy =
     use standard mix design only,
     modify mix design only,
     modify mix & use selective heating, or
     some mix modification & full heat.
```

```
3. link: winter weather, concrete strategy,
     if mild    then   use standard mix design only,        s(0.7), pl(1.0),
     if mild    then   modify mix design only,              s(0.0), pl(0.3),

     if normal  then   use standard mix design only,        s(0.2), pl(0.6),
     if normal  then   modify mix design only,              s(0.4), pl(0.8),
     if normal  then   modify mix & use selective heating,  s(0.0), pl(0.2),

     if severe  then   modify mix design only,              s(0.0), pl(0.2),
     if severe  then   modify mix & use selective heating,  s(0.1), pl(0.7),
     if severe  then   some mix modification & full heat,   s(0.3), pl(0.9).
```

In addition to the D-S knowledge base file, a control knowledge base file must be created. In this simplified case, the control file would contain a statement which shows that the goal of the system is to check the concrete strategy. A second statement would be a rule asserting that the concrete strategy has been checked if the user has been asked about the weather and if he has been shown the resulting belief about strategies. Notice that the belief about concreting strategies is inferred *automatically* from belief about weather since this relationship is contained in the D-S knowledge base, the inference procedure does not have to be expressed in the control rule. A final statement defines the question to be asked in order to solicit information about the weather. In a form which could be read by the shell, this file would look like the following:

```
1. goal:  concrete strategy is checked

2. rule:  concrete strategy is checked
      if  query  winter weather
     and  show  concrete strategy,
                message "The best strategy to use for maintaining
                         acceptable cure times this winter is:".

3. question: winter weather
     ask "What is the long range forecast for cold weather conditions
          for the construction period Nov. 15 1988 to Feb. 15 1989 ?".
```

Now, in order to use the system, the shell program is run. Both knowledge bases are first loaded into the shell program. Once this is done and execution has begun, the question about the weather is asked in the manner discussed above (suppose that the response is as shown in figure 12.3). The system then proceeds to determine the belief in concreting strategies and the result is displayed. In this case, the output would be as shown in figure 12.5.

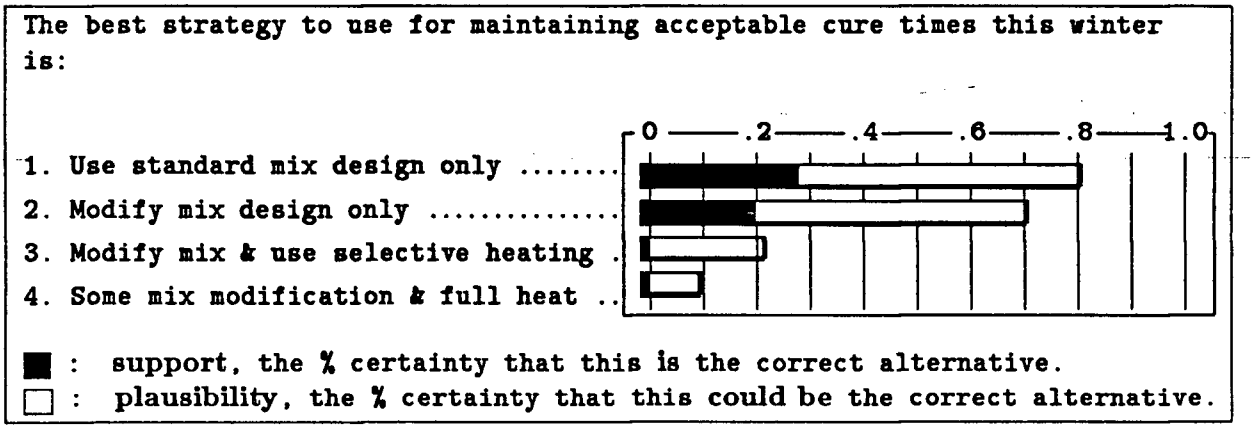


Figure 12.5: Belief output screen

The contractor's conclusion from the output shown in figure 12.5 would be confidence that he will not need to use heating equipment in order to maintain his cycle schedule. This could be an important determination if, for example, there was a very long lead time required to obtain high output propane heaters and tarps for such a process. However, since there are only marginal differences in both the support and plausibility values for the first two alternatives, the contractor could not confidently assert whether or not he should modify the mix design. This would suggest to him that he does not yet have sufficient information to resolve the mix design question. Finally, we can note that the decision of what to do in the face of different uncertain outcomes is likely to be part of the expertise of the human or humans who created the system. If this expertise is built into the system, then the user of the system will never have to face the question of how to interpret the output.

Many issues remain to be examined before the D-S theory could provide a practical, generally applicable approach to dealing with uncertainty in expert systems. However our work to-date indicates that the goal of using the D-S theory to represent and manipulate realistic amounts of uncertain information in working expert systems may not be far away.

12.5 Incorporating Uncertainty in Knowledge Base Design

It has been shown that a frame network based expert system with interactive belief input and rule-based control can be used to perform inexact reasoning. But these features alone does not ensure appropriate treatment of uncertainty in an expert system. The expert system's treatment of uncertainty also depends on the way in which the knowledge engineer and the human expert decide to employ the uncertainty handling capabilities of the system. The example discussed in the previous sections, for instance, illustrates a very simple use of inexact reasoning (there is only one input and one conclusion). This section describes a series of knowledge base design approaches which form a hierarchy of sophistication in their treatment of uncertainty:

1. No Uncertainty:

   At the simplest level, the existence of all uncertainty in an application can be ignored and all input and inference can be restricted to purely categorical processes.

2. Simple Uncertain Input and Inference:

   The first steps to incorporating uncertainty are to allow both uncertain input and uncertain inference. This is as described in the example of the preceding sections.

3. Multiple Uncertain Inputs and Inference:

   A more advanced design involves allowing multiple uncertain inputs which produce multiple uncertain inferences about one or more conclusions. Referring to the previously described example, for instance, it could be stated that the appropriate concreting strategy is based not only on the expected winter weather, but also on the local costs of the various alternatives, on the experience of the labour force, on the type of structure being built, and so on. Inputs could be accepted relating to each of these factors which would result in several different belief functions for concreting strategies. Each of these inferred belief functions would be combined to give the overall resulting belief.

4. Simple Evaluation of Uncertain Output:

   The combination of multiple uncertain inputs and inferences results in a belief function representation of the system's conclusions. A refinement to the direct output of these belief functions in the form of IBIS screens is to perform simple evaluation of the information in order to provide a best alternative. This is done in the manner described in section 11.3.

5. Assignment of Verbal Qualifiers:

   An alternative to providing numeric output of either belief functions or of the evaluations of belief functions is to translate the information into verbal qualifiers. For example, if it is concluded that some proposition A is the best alternative with a degree of confidence of 0.95 (as outlined in section 11.3), then the system could report that it is ***very certain*** that

A is the best answer. On the other hand, if the system's confidence is only 0.2 then it could report that it is has only a *small amount* of confidence that *A* is the best alternative. These verbal qualifiers can be obtained from a "look-up table" in the knowledge base.

6. Using Uncertain Results to Guide Reasoning:

   The results of the system's inexact reasoning can be used in a simple manner to guide the problem solution by adding control knowledge which checks the degree of confidence in the selection of the best alternative. If the degree of confidence in the answer is sufficiently high, the reasoning can be stopped. If the degree of confidence in the solution is too low, however, the system can continue to ask additional questions in an attempt to better resolve the issue.

7. Fully Incorporating Uncertainty into the Expert Knowledge:

   Finally, the uncertainty can be fully incorporated into the reasoning process. The way in which the solution should proceed in the face of certain types of uncertainty is as much a part of the expert's domain knowledge as is any other information; it cannot be resolved in a general sense. Since a frame network D-S system is fully capable of representing uncertainty, rules can be added to the knowledge base which state that if the system's subconclusions exhibit certain forms of uncertainty, then specific actions should be pursued in order to reach the final result.

   Taken to the extreme, this approach would require an extremely large amount of input pertaining to the expert's solution approaches. However, it is exactly through the accurate modelling of expert techniques using this form of very "knowledge intensive" programming that expert systems achieve their power. D-S based systems posses a unique capability for representing and processing uncertain information in a manner which is rich enough to allow for the detailed incorporation of uncertainty into the knowledge base.

12.6 Conclusion

This chapter has illustrated the application of a frame network expert system to a simple example problem. It has been shown that the system allows accurate expression of uncertain knowledge while remaining relatively simple to use for the user as well as for the knowledge engineer and the human expert.

Chapter 13

Summary

The work described in this thesis stems from the idea that expert systems should be able to accurately and appropriately handle uncertain information. The traditional approaches for dealing with uncertainty were discussed and are shown to contain many inadequacies.

The Dempster-Shafer, or D-S, theory of evidence was proposed as an appealing theoretical basis for representing uncertain knowledge and for performing inexact reasoning in expert systems. The D-S theory was reviewed in some detail; including its approaches to representing concepts, to representing belief, to combining belief and to performing inference.

The D-S implementation approaches pursued by other researchers were described and critiqued. Attempts made early in the thesis research which failed to achieve the important goal of consistency with the D-S theory were also reviewed.

Two approaches to implementing D-S theory in a completely consistent manner were discussed in detail. It was shown that the second of these systems, a frame network approach, led to the development of a fully functional prototype expert system shell called FRO. In this system, concepts are represented using D-S frames of discernment, belief is represented using D-S belief functions, and inference is performed using stored relationships between frames of discernment (forming the frame network) and D-S belief combination rules. System control is accomplished using a discrete rule-based control component and uncertain input and output are performed through an interactive belief interface system called IBIS. Each of these features is reviewed.

Finally, a simple but detailed example of an application of a frame network expert system is

provided. The appendix of this thesis contains the entire FRO system user's documentation.

Bibliography

[1] Arity Corporation, *The Arity Expert Systems Development Package*, Arity Corporation, Concord, MASS, 1986.

[2] Barnett, J.A. "Computational Methods For A Mathematical Theory of Evidence." *Proceedings of IJCAI-81, The Seventh International Joint Conference on Aritificial Intelligence*, Vancouver, B.C., Aug. 1981. pp.868–875.

[3] Bonissone, P.P. and Tong, R.M. "Editorial: Reasoning with Uncertainty in Expert Systems." *International Journal of Man-Machine Studies.* Vol. 22. March 1985. pp. 241–250.

[4] Buchanan, B.G. and Shortliffe, E.H. (Eds.). *Rule-Based Expert Systems: MYCIN Experiments.* Reading, M.A.: Addison Wesley, 1984.

[5] Buckley, J.J. and Tucker, D. "The Utility of Information and Risk-Taking Fuzzy Expert Systems," *International Journal of Intelligent Systems* Vol.3, No.2, Summer 1988. pp.179–197.

[6] Caselton, W.F., Froese, T.M., Russell, A.D. and Luo, W. "Belief Input Procedures for Dempster-Shafer Based Expert Systems," *Artificial Intelligence in Engineering: Robotics and Processes,* (the proceedings of AIENG-88, the Third International Conference on Applications of Artificial Intelligence in Engineering, Palo Alto, CA, August, 1988) Computational Mechanics Publications, Southampton, 1988. pp. 351–370.

[7] Charniak, E. "The Bayesian Basis of Common Sense Medical Diagnosis", *Proceedings of AAAI-83, the Third National Conference on Artificial Intelligence,* Washington, D.C., 1983. pp.70–73.

[8] Cheeseman, P. "In Defense of Probability," *Proceedings of AAAI-83, the Third National Conference on Artificial Intelligence,* Washington, D.C., 1983. pp.1002–1009.

[9] Cohen, L.J. "A Note on Inductive Logic", *The Journal of Philosophy*, LXXX, 1973. pp.27–40.

[10] Cohen, P.R. *Heuristic Reasoning about Uncertainty: An Artificial Intelligence Approach.* North-Holland, 1985.

[11] Cohen, P.R. and Shafer, G.A. *Managing Uncertainty*, AAAI-87, the Sixth National Conference on Artificial Intelligence, Seattle, WA, Conference Tutorial Program, 1987.

[12] Dempster, A.P. "Upper and lower probabilities induced by a multivariate mapping," *Annals of Mathematical Statistics*, Vol. 38, 1976. pp.325–339.

[13] Driankov, D. "Uncertainty Calculus with Verbally Defined Belief-Intervals," *International Journal of Intelligent Systems,* Vol. 1, No. 4, Winter 1986. pp.219–246.

[14] Dubois, D. and Prade, H. "Combination and Propagation with Belief Functions – A Reexamination," *Proceedings of IJCAI-85, The Ninth International Joint Conference on Aritificial Intelligence*, Los Angles, CA, Aug. 1985. pp. 111–113.

[15] Dubois, D. and Prade, H. "On the Unicity of Dempster Rule of Combination," *International Journal of Intelligent Systems* Vol.1, No.2, Summer 1986. pp.133–142.

[16] Dubois, D. and Prade, H. "The Treatment of Uncertainty in Knowledge-based Systems using Fuzzy Sets and Possibility Theory," *International Journal of Intelligent Systems* Vol.3, No.2, Summer 1988. pp.141–165.

[17] Duda, R.O., Hart, P.E., and Nilsson, N.J. "Subjective Bayesian methods for rule-based inference systems." *AFIPS Conference Proceedings*, New York City, NY, June 1976. pp.1075–1082.

[18] Esmo, L., Saitta, L. and Torasso, P. "Evidence Combination in Expert Systems," *International Journal of Man-Machine Studies* Vol. 22. March 1985. pp. 307–326.

[19] Fall, T.C., "Evidential Reasoning with Temporal Aspects," *Proceedings of AAAI-86, the Fifth National Conference on Artificial Intelligence*, Philadelphia, PA, 1986. pp.210–214.

[20] Friedman, L. "Extended Plausible Inference." *Proceedings of IJCAI-81, The Seventh International Joint Conference on Aritificial Intelligence*, Vancouver, B.C., Aug. 1981. pp.487–495.

[21] Froese, T.M., Caselton, W.F., and Russell, A.D. "An Interactive Belief Interface System," To be submitted for publication.

[22] Froese, T.M. "Applied Inexact Reasoning in Civil Engineering," *Campus Computing*, The University of British Columbia Computing Center, Vol. 2, No. 4, April 1987. pp.13–15.

[23] Froese, T.M., Russell, A.D. and Caselton, W.F. "Implementation Strategies for Dempster-Shafer Based Inexact Reasoning," Submitted for publication to the *International Journal of Intelligent Systems* in July, 1988.

[24] Froese, T.M., Caselton, W.F., and Russell, A.D. "Knowledge Representation and Inexact Reasoning consistent with the Dempster-Shafer Theory," unpublished, 1987.

[25] Garvey, T.D., Lowrance, J.D. and Fischler, M.A. "An Inference Technique for Integrating Knowledge from Disparate Sources." *Proceedings of IJCAI-81, The Seventh International Joint Conference on Aritificial Intelligence*, Vancouver, B.C., Aug. 1981. pp.319–325.

[26] Ginsberg, M.L. "Non-monotonic Reasoning using Dempster's Rule." *Proceedings of AAAI-83, the Fourth National Conference on Artificial Intelligence,* Austin, TX, Aug. 1984. pp.126–129.

[27] Gordon, J. and Shortliffe, E.H. "A Method for Managing Evidential Reasoning in a Hierarchical Hypothesis Space," *Artificial Intelligence*, Vol. 26, July, 1985. pp.323–357.

[28] Gray, C. "Intelligent Construction Time and Cost Analysis," *Construction Management and Economics,* Vol. 4, 1986. pp.135–150.

[29] Guth, M.A. "Uncertainty Analysis of Rule-Based Expert Systems with Dempster-Shafer Mass Assignments," *International Journal of Intelligent Systems* Vol.3, No.2, Summer 1988. pp.123–139.

[30] Haddawy, P. "Implementaion of and Experiments with a Variable Percision Logic Inference System," *Proceedings of AAAI-86, the Fifth National Conference on Artificial Intelligence*, Philadelphia, PA, 1986. pp.238–242.

[31] Halpern, J.Y. and Rabin, M.O. "A Logic to Reason about Likelihood," *Artificial Intelligence*, Vol. 32, 1987. pp.379–405.

[32] Hendrickson, C., Martinelli, D., and Rehak, D. "Hierarchical Rule-Based Activity Duration Estimation," *Journal of Construction Engineering and Management*, Vol. 113, No. 2, June 1987. pp.288–301.

[33] Horvitz, E.J., Heckerman, D.E., and Langlotz, C.P. "A Framework for Comparing Alternative Formalisms for Plausible Reasoning," *Proceedings of AAAI-86, the Fifth National Conference on Artificial Intelligence*, Philadelphia, PA, 1986. pp.210–214.

[34] Hudson, D.L. and Cohen, M.E. "Approaches to Management of Uncertainty in an Expert System," *International Journal of Intelligent Systems* Vol.3, No.1, Spring 1988. pp.45–58.

[35] Ishizuka, M., Fu, K.S., and Yao, J.T.P. "Inference Procedures under Uncertainty for Problem-Reduction Method," *Information Sciences,* Vol. 28, 1982. pp. 179–206.

[36] Kong, A. *Multivariate Belief Functions and Graphical Models,* Doctoral Dissertation, Department of Statistics, Harvard University.

[37] Kosko, B. "Fuzzy Knowledge Combination," *International Journal of Intelligent Systems* Vol.1, No.4, Winter 1986. pp.293–320.

[38] Lauritzen, S.L. and Spiegelhalter, D.J. "Fast manipulation of probabilities with local representations – with applications to expert systems," Institute of Electronic Systems, Aalborg University, Alborg, Denmark.

[39] Lecot, K. and Parker, D.S. "Control Over Inexact Reasoning." *AI Expert,* Premier, 1986. pp. 32–43.

[40] Lee, N.S., Grize, Y.L., and Dehnad, K. "Qualitative Models for Reasoning under Uncertainty in Knowledge-Based Expert Systems," *International Journal of Intelligent Systems* Vol.2, No.1, Spring 1987. pp.15–38.

[41] Liu, G.S. "Causual and Plausible Reasoning in Expert Systems." *Proceedings of AAAI-86, the Fifth National Conference on Artificial Intelligence*, Philadelphia, PA, 1986. pp.220–225.

[42] Lowrance, J.D., Garvey, T.D. and Strat, T.M. "A Framework For Evidential-Reasoning Systems," *Proceedings of AAAI-86, the Fifth National Conference on Artificial Intelligence*, Philadelphia, PA, 1986. pp.896–903.

[43] Lowrance, J.D. *Dependency-Graph Models of Evidential Support.* PhD thesis, Dept. of Computer and Information Science, University of Massachusetts, Amherst, MA, 1982.

[44] Lu, S.Y. and Stehanou, E. "A Set-Theoretic Framework for the Processing of Uncertain Knowledge." *Proceedings of AAAI-83, the Fourth National Conference on Artificial Intelligence,* Austin, TX, Aug. 1984. pp.216–221.

[45] Mangiaracina, S. and Beni, G. "On the Logical and Physical Combination of Evidence in Intelligent Machines," *International Journal of Intelligent Systems* Vol.1, No.2, Summer 1986. pp.143–152.

[46] Martin-Clouaire, R. and Prade, H. "On the Problems of Representation and Propagation of Uncertainty in Expert Systems," *International Journal of Man-Machine Studies,* Vol. 22. March 1985. pp.251–264.

[47] Nilsson, N.J. "Probabilistic Logic," *Artificial Intelligence*, Vol. 28, 1986. pp.71–87.

[48] Ogawa, H., Fu, K.S., and Yao, J.T.P. "An Inexact Inference For Damage Assessment Of Existing Structures," *International Journal of Man-Machine Studies,* Vol. 22. March 1985. pp.295–306.

[49] Ogawa, H., Fu, K.S., and Yao, J.T.P. "Speril-II: An Expert System for Damage Assessment of Existing Structures," *Approximate Reasoning in Expert Systems*, North-Holland, 1985. pp.731–744.

[50] Pearl, J. "Distributed Revision of Composite Beliefs," *Artificial Intelligence*, Vol. 33, 1987. pp. 173–216.

[51] Pearl, J. "Fusion, propagation, and structuring in belief networks" *Artificial Intelligence*, Vol. 29, 1986. pp. 241–288.

[52] Pearl, J. "On Evidential Reasoning in a Hierarchy of Hypotheses," *Artificial Intelligence,* Vol. 28, 1986. pp.9–15.

[53] Prade, H. "A Computational Approach to Approximate and Plausibile Reasoning with Applications to Expert Systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. PAMI-7, No. 3, May 1985. pp.260–282.

[54] Quinlan, J.R. "Inferno: A Cautious Approach To Uncertain Inference." *The Computer Journal,* Vol 26, No. 3, Aug. 1983. pp.255–269.

[55] Russell, A.D., Froese, T.M. and Caselton, W.F. "A Dempster-Shafer Based Construction Expert System," *Proceedings of the Third International Conference on Computing in Civil Engineering,* Vancouver, B.C., August 1988.

[56] Shackle, G.L.S. *Decision, Order and Time in Human Affairs,* Cambridge, Second Edition. 1969.

[57] Shafer, G.A. *A Mathematical Theory of Evidence*, Princeton University Press, Princeton, NJ, 1976.

[58] Shafer, G.A. "Belief Functions and Parametric Models," *The Journal of the Royal Statistical Society,* Series B, Vol. 44, 1982. pp.322–352.

[59] Shafer, G.A. "Belief Functions and Possibility Measures," *Analysis of Fuzzy Information,* Vol. 2, CRC Press.

[60] Shafer, G.A. "Hierarchical Evidence," *Proceddings of The Second Conference on Artificial Intelligence Applications - The Engineering of Knowledge-Based Systems,* IEEE Computer Society Press, 1985. pp. 16–25.

[61] Shafer, G.A. "Probability Judgement in AI and Expert Systems," *Statistical Science,* 1987.

[62] Shafer, G.A. "Savage revisited" *Statistical Science*, Vol 1, 1986. pp. 463–501.

[63] Shafer, G.A. "The Combination of Evidence," *International Journal of Intelligent Systems* Vol.1, No.3, Fall 1986. pp.155–179.

[64] Shafer, G.A. "The Problem of Dependent Evidence," Working paper No. 164, School of Business, University of Kansas, 1985.

[65] Shafer, G.A., and Logan, R. "Implementing Dempster's Rule For Hierarchical Evidence," *Artificial Intelligence,* Vol. 33, 1987. pp.271–298.

[66] Shafer, G.A. and Tversky, A. "Languages and Design for Probability Judgment," *Cognitive Science*, Vol.9, 1985. pp.309–339.

[67] Shafer, G.A., Shenoy, P., and Mellouli, K. "Propagating belief functions in qualitative Markov trees," Working paper No. 186, School of Business, University of Kansas.

[68] Shenoy, P.P. and Shafer, G.A. "Propagating Belief Functions with Local Computations," *IEEE Expert*, Fall, 1986. pp.43–52.

[69] Smets, P. "The Degree of Belief in a Fuzzy Event," *Information Science,* Vol. 25, 1981. pp.1–19.

[70] Strat, T.M. "Continuous Belief Functions For Evidential Reasoning." *Proceedings of AAAI-83, the Fourth National Conference on Artificial Intelligence,* Austin, TX, Aug. 1984. pp.308–313.

[71] Warszawski, A. "Decision Models and Expert Systems in Construction Management," *Building and Environment*, Vol. 20, No. 4, 1985. pp.201–210.

[72] Yager, R.R. "Aggregating Evidence Using Quantified Statements," *Information Sciences,* Vol. 36, 1985. pp.179–206.

[73] Yager, R.R. "Entropy and Specificity in a Mathematical Theory of Evidence", *International Journal of General Systems*, Vol. 9, 1983. pp.249–260.

[74] Yager, R.R. "Generalized Probabilities of Fuzzy Events from Fuzzy Belief Structures," *Information Sciences*, Vol. 28, 1982. pp.45–62.

[75] Yager, R.R. "On the Dempster-Shafer Framework and New Combination Rules," *Information Sciences*, Vol. 41, 1987. pp.93–137.

[76] Yager, R.R. "Set-based Representations of Conjunctive and Disjunctive Knowledge," *Information Sciences*, Vol. 41, 1987. pp.1–22.

[77] Yager, R.R. "The Entailment Principle for Dempster-Shafer Granules," *International Journal of Intelligent Systems* Vol.1, No.4, Winter 1986. pp.247–262.

[78] Yager, R.R. "Toward a General Theory of Reasoning with Uncertainty. I:Nonspecificity and Fuzziness," *International Journal of Intelligent Systems* Vol.1, No.1, Spring 1986. pp.45–67.

[79] Yager, R.R. "Using Approximate Reasoning to Represent Default Knowledge," Research Note, *Artificial Intelligence,* Vol. 31, 1987. pp.99-112.

[80] Yen, J. "A Reasoning Model Based on an Extended Dempster-Shafer Theory," *Proceedings of AAAI-86, the Fifth National Conference on Artificial Intelligence*, Philadelphia, PA, 1986. pp.125–131.

[81] Zadeh, L.A. '1'A Computational Theory of Dispositions," *International Journal of Intelligent Systems* Vol.2, No.1, Spring 1987. pp.39–63.

[82] Zadeh, L.A. "A Simple View of the Dempster-Shafer Theory of Evidence and its Implication for the Rule of Combination," *AI Magazine*, Summer, 1986. pp.85–90.

[83] Zadeh, L.A. "Book Review of A Mathematical Theory of Evidence," *The AI Magazine*, Fall 1984. pp.81–83.

APPENDIX A:

F · R · O

An Expert System Shell

SYSTEM DOCUMENTATION

Preface

FRO is an expert system shell written in PROLOG. It was created at the Department of Civil Engineering, UBC for use in expert systems research. FRO can be used at two levels.

First, FRO can be used as a shell program for creating rule-based expert systems. In this role, FRO behaves much like other commercially available shell programs. This use of the program is discussed in Part I, Rule-Based FRO.

Second, FRO can be used to create expert systems which provide sophisticated handling of uncertainty based on the Dempster-Shafer (D-S) theory of evidence. In these systems, the program employs both a **frame** representation for D-S components and a **rule** representation for system control—therefore the program is a **F**rame/**R**ule **O**rganizer, hence **FRO**. This use of FRO is discussed in Part II, Frame-Based FRO.

Please note the following type style conventions used in this manual. `Typewriter style type face` is used for text that appears exactly as it would be entered into the system (or displayed by the system). Items written in *SLANTED UPPER CASE TYPE FACE* are not entered into the system exactly as they appear, but rather they are replaced with the appropriate term as described in the text.

Contents

PART I

RULE-BASED FRO

The discussion of the rule-based portion of the FRO program begins with a description of the system's underlying theory and instructions for creating a problem file; this comprises chapters 1 through 10. Chapter 11 describes how to actually run the FRO program once a problem file has been created. Chapter 12 lists several sample problem files. The recommended approach to learning the system is to alternate between reading this manual, examining the sample files, and experimenting with the program itself.

Chapter 1: Overview

1.1 Introduction

FRO is a tool for building expert systems. An expert system created using FRO is made up of two major parts; the **inference engine**, which performs reasoning, and the **knowledge base**, which contains the expert knowledge.

The compiled FRO program makes up the inference engine portion of the expert system. This inference engine uses backward chaining, depth-first search to provide expert system reasoning. This means that the system is given some goal or conclusion which it tries to prove to be true using the information which is made available to it. A **problem file** which is written by the user and read as input by FRO makes up the knowledge base.

The various components of the inference engine and knowledge base are briefly outlined below and then discussed more fully in subsequent chapters.

1.2 Knowledge Representation

In FRO, most knowledge is stored using in the form:

CONCEPT `is` *VALUE*

Where *CONCEPT* is the name of some concept or variable which is of interest to us and *VALUE* is the name of some value which can be assigned to that concept. We can call this knowledge representation unit a **proposition**. Some examples of propositions are:

```
gravity is 9.81
'your name' is George
'a solution can not be found' is true
```

1.3 Solving Goals

FRO's basic task is to solve goals. A primary goal is defined in a problem file. In order to solve this primary goal, FRO may need to first solve other sub-goals which may, in turn, require the investigation of still more goals. It is through this recursive goal solving process that the system performs reasoning. The methodology is analogous to that of a decision tree, though the expert system is a more flexible and powerful problem solver.

In FRO, a goal is composed of a proposition. A goal is considered to be solved if the system can determined that the proposition is correct. There are three ways the system can determine the correctness of a proposition; it can check with its memory, it can check with the user, or it can infer the correctness from rules.

1.4 Memory

Whenever a proposition is asserted in the problem file, stated by the user in response to a question, or inferred by the system using a rule; it is stored in the system's data base. It thereby becomes part of FRO's **memory** or **knowledge**. If the system is trying to solve a goal which relates to that proposition, it will be able to refer to the stored value as a known statement of fact. ***This is the system's basic form of memory.***

1.5 Rules

If FRO has not succeeded in solving a goal by referring to its memory, it will search the problem file for an applicable **rule statement**. These statements take the form "*CONCLUSION* `if` *PREMISE* ". If the conclusion of a rule relates to the system's present goal, the system will take the premise as it's new goal. By proving that the premise is true, the system can determine that the conclusion is also true; thereby proving or disproving its initial goal. ***This is the system's basic form of reasoning.***

1.6 Questions

Finally, if FRO cannot solve some goal by referring to stored values or stored rules, it will check to see if the problem file contains a **question statement** relating to the goal proposition. If such a statement does exist, it will supply the system with a question which can be used to illicit the correct value from the user. ***This is the system's basic form of user input.***

Chapter 2: Problem Files

The knowledge base for a FRO expert system takes the form of a problem file. This problem file is a standard ASCII text file created by the user with any file editor or any word processor which produces ASCII files. The problem file can have any name.

Comments and notes can be entered into a problem file by beginning a comment line with a % character or by enclosing any number of comment lines between a /* symbol and a */ symbol.

Apart from comments, the problem file is made up of **statements**. There are four allowable types of statements: **Goal** statements, **Fact** statements, **Rule** statements, and **Question** statements. Statements can begin with a **type declaration** (i.e. a word stating which type the statement is) and a **statement number**. It is recommended that type declarations and statement numbers be used so that it will be easier for users to understand and refer to the knowledge base. However, statements can also be given without type declarations or numbers in which case the system assigns its own number and determines the type from the statement syntax. The syntax for each statement type is given in chapters 4 through 7, but first chapter 3 discusses propositions which are the basic building blocks of statements. ***Note that each statement must end with a period.***

Spaces, tabs and blank lines are ignored in the problem file, although they are very important for making the problem file legible to users. No special end-of-file marker is needed. Chapter 12 contains sample problem files.

Chapter 3: Propositions

The statements in the problem file are made up mostly of propositions (the basic unit of knowledge in FRO). As stated previously, proposition are of the form:

CONCEPT `is` *VALUE*

Where *CONCEPT* is some concept or variable of interest to us and *VALUE* is a value which can be assigned to the concept. Some examples of propositions are:

```
gravity is 9.81
'your name' is George
'a solution can not be found' is true
```

In a proposition, both the *CONCEPT* and the *VALUE* must be either a single atom or a single string. An atom is a basic symbolic unit in FRO. Atoms can be made up of any letters, numbers, or symbols. However, an atom must be enclosed inside a set of apostrophes if it contains capital letters, combinations of letters and numbers, spaces, or any non alpha-numeric symbols other than the underscore symbol "_". The following are examples of legal atoms.

```
true
the_consultation
9.81
'George'
'GEORGE'
'the number of apples'
```

An atom can be enclosed in apostrophes even if it does contain only lower case letters; in which case the system recognizes both forms as being equivalent. Atoms can be up to 255 characters long. If an apostrophe is desired within an atom, it must be represented by a pair of apostrophes. For example, the possessive form of `'George'` is:

```
'George''s'
```

Strings are similar to atoms except that they are always enclosed inside a pair of dollar signs. Some examples are:

```
$string$
$This is also a string$
$1,000,000$
```

Strings take up more memory than atoms but they can be up to 16 kilobytes long. A dollar sign contained within a string must be represented by a pair of dollar signs "$$".

Note that a string is not recognized as being the same as an atom even if they are made up of identical characters. In most cases it doesn't matter whether an atom or a string is used for a particular situation as long a string is not used to represent something which is represented elsewhere in a problem file by an atom (and vise-versa). A useful convention is to use atoms for most general terms in propositions but to use strings for longer terms which are intended to be written as messages to the screen (printing message to the screen will be discussed later).

Often the only appropriate values for a proposition are **true** and **false**. For efficiency, FRO allows any proposition of the form:

CONCEPT `is true`

To be entered simply as:

CONCEPT

The "`is true`" part is assumed. The following are examples of this special proposition form where the part in braces "{ }" need not be written by the user because it is assumed by the system:

```
the_consultation_is_finished {is true}
'birds fly' {is true}
```

As a final point about propositions, it should be noted that the words and names used in propositions don't have any *meaning* to the system; they are just arbitrary symbols. Nevertheless it is important that the propositions should be worded so as to accurately and precisely reflect the concepts that they are being used to represent. This will allow the system to come as close as possible to an accurate and precise reflection of the *captured* human knowledge and reasoning. It will also make the knowledge base more intuitive and easier for users to understand.

Chapter 4: Goal Statements

Each problem file must contain a single **goal statement** of the form:

`goal` # :: *PROPOSITION*.

where the word "**goal**" declares the statement type, the "#" is a statement number, and *PROPOSITION* is the **top-level goal** which FRO is being asked to solve.

A statement number is an arbitrary number assigned to each statement by the user. Statement numbers can be any alpha-numeric atom, with the only restrictions being that negative integers should not be used (as these are used by the system when no number is supplied by the user) and the same number should not be used for two different statements of the same type (in which case the system will ignore the second statement number and assign one itself). Statement numbers are used by the system when referring the user to a particular statement.

A goal can be given without a statement number in the following form:

`goal` *PROPOSITION*.

Unlike other statement types, a goal statement can not be given without a type declaration. Some examples of goal statements are:

```
goal 1 :: consultation is finished.
goal find_the_problem.
```

Recall that in the second example goal statement the proposition is assumed to mean "`find_the_problem is true`" and that in both examples the statement must be followed by a period.

Chapter 5: Fact Statements

The problem file may contain **fact statements** consisting solely of a proposition followed by a period:

`fact #` :: *PROPOSITION*.

Or simply:

PROPOSITION.

examples:

```
fact 13 ::  'the 1st month' is january.
number_of_items is 6.
'birds fly'.
```

These facts are stored in the system's data base and can be recalled at any time to solve a goal. These statements are usually **domain knowledge** in that they are valid for all problems covered by the knowledge base, not **context knowledge** which are statements that may only relate to the particular situation at hand (once in the system's memory, all facts are treated identically).

Chapter 6: Rule Statements

Rule statements allow the system to perform reasoning by storing **inferential knowledge** (or knowledge about how different propositions are related) and **procedural control knowledge** (or knowledge concerning how to go about solving a problem). A rule is written in the form:

`rule #` :: *CONCLUSION* `if` *PREMISE*

Or:

CONCLUSION `if` *PREMISE*

for example:

```
rule 42 ::  'typical weather' is sunny if season is summer.

rule 18 ::
  'the animal can fly'
     if 'the animal type' is bird.

consultation is finished
      if problem is identified
     and problem is solved
     and user is satisfied.
```

The *CONCLUSION* of the rule is made up of a single proposition (i.e. the *CONCLUSION* is of the form "*CONCEPT* `is` *VALUE*"). If the *CONCEPT* of the goal matches the *CONCEPT* of the rule's *CONCLUSION*, then the system will take the rule's *PREMISE* as its new goal. If the system can solve the *PREMISE*, the *CONCLUSION* proposition is first stored in the knowledge base as a known fact and then used to try to solve the goal. If the *PREMISE* can't be solved, the goal neither succeeds nor fails and the system looks for other means of solving it.

The *PREMISE* is made up of one or more propositions. A *PREMISE* consisting of a single proposition becomes the new goal directly. The *PREMISE* can also consist of several propositions joined by the word "`and`" in which case each

proposition is taken as a separate goal in turn and each one must succeed in order for the entire *PREMISE* to succeed; or it can consist of several propositions joined by the word "or" in which case the success of any proposition allows the entire *PREMISE* to succeed. Combinations of "and" and "or" can also be used in the premise; "and" has a higher precedence than "or" (so, for example, "a and b or c" is interpreted as "(a and b) or c") but parenthesis should be used to avoid confusion. Some examples of possible *PREMISES* in rules are:

```
rule 21 :: CONCLUSION if today is 'Sunday'.

rule 5 :: CONCLUSION if finished.

CONCLUSION if 'Part 1' and 'Part 2'.

rule 8 ::
  CONCLUSION
     if 'the animal type' is bird
     or 'the animal type' is insect.

CONCLUSION
     if (    today is 'Saturday'
          or today is 'Sunday')
     or (    today is 'a weekday'
         and today is 'a holiday').
```

Note that in these examples a period is placed at the end of each rule and the assumed "is true" *VALUE* is sometimes used. Again, line feeds (and/or carriage returns) and extra spaces are ignored. In order to make the problem file easy to understand and debug, rules should be well laid out with ample white space and comments. Rules should also be kept fairly short, using a hierarchy of smaller sub-rules rather than one long rule.

Chapter 7: Question Statements

7.1 Basic Question Statements

In order for the system to be able to ask the user about the value of some goal's *CONCEPT*, the problem file must contain a **Question statement** for that concept. The question statement defines the question to be asked of the user and may supply the allowable responses. A question is written in the form:

question # :: *CONCEPT* **ask** *QUESTION*.

Or:

CONCEPT **ask** *QUESTION*.

Where *CONCEPT* is identical to the *CONCEPT* of the goal being solved and *QUESTION* is a question that the system will ask of the user (usually given in the form of a character string). For example:

```
question 88 :: weather ask $What is today's weather like?$.

weather ask $What is today's weather like?$.
```

Thus, if the system is trying to solve the goal "**weather is sunny**" but it finds no facts or rules pertaining to weather, then it will find the question statement shown above (in either form) and will write on the screen:

```
What is today's weather like?
```

The user can then enter a *VALUE* which is assigned to the *CONCEPT*, creating a proposition which is stored in the knowledge base. For example, if the user responds with "sunny", then the proposition "**weather is sunny**" will be stored in the knowledge base. After the question is asked and a proposition is stored, the systems tries to solve the goal again and this time the goal will succeed or fail based on the newly stored proposition (e.g. in this case the original goal "**weather is sunny**" will succeed because this fact has just been stored).

Note that if the system finds no question pertaining to the goal (having already found no applicable facts or rules), then the goal will fail.

In addition to the basic form of question statement just discussed, there are several optional parameters that can be used with question statements. These are discussed below (here we show only the form containing statement numbers, though either form is acceptable in each case).

7.2 The Question Character String

As mentioned, the *QUESTION* term of the question statement is a character string which is printed on the screen. If desired, special formatting control commands can be included in these strings. Section 8.7 describes these commands in detail.

7.3 Alternative Lists

Rather than allowing the user to give any response to a question, it will frequently be more convenient to supply the user with a list of alternative responses from which he can select the best choice. This can be done by using a question statement of the form:

`question #` :: *CONCEPT* `ask` *QUESTION* `alt` (*ALT 1,ALT 2,...*).

where *CONCEPT* and *QUESTION* are the same as in the basic form discussed in section 7.1. and (*ALT 1,ALT 2,...*) is a list of the possible alternatives enclosed in parentheses and separated by commas. For example, the statement:

```
question 6 ::
  weather
      ask $What is today's weather like?$
      alt (sunny, rainy, foggy, snowing).
```

would allow the system to ask the user the following question:

```
What is today's weather like?
1.  sunny
2.  rainy
3.  foggy
4.  snowing
>>>>
```

The user would enter the number of the best alternative (or select that alternative with the mouse). The system would then assign the corresponding

alternative to the question's *CONCEPT* in the same way as described above.

7.4 Return Alternative Lists

The terms in the alternative list need to be sufficiently descriptive to allow the user to understand their exact meaning. However, these terms are returned as a value in a proposition and long descriptive values are awkward to work with in the knowledge base. To circumvent this conflict, it is possible to supply two lists of terms; one descriptive list of alternatives to appear on the screen for the user and a second, terser list of corresponding "return alternatives" that are assigned to the *CONCEPT* in the proposition. The syntax is:

```
question # ::
  CONCEPT
      ask QUESTION
      alt (ALT 1, ALT 2, ...)
     ralt (RALT 1,RALT 2,...).
```

Where (*ALT 1, ALT 2, ...*) is the list of alternatives to appear on the screen and (*RALT 1, RALT 2,...*) is the list of return alternatives. An typical question using this feature might look like the following:

```
question 99 ::
  weather
      ask $What is today's weather like?$
      alt ($Clear and Sunny$,
           $Overcast or rainy$,
           $Foggy$,
           $Hailing or Snowing$)
    ralt (sunny, rainy, foggy, snowing).
```

7.5 Yes/No Questions

Because simple yes/no questions are common, they can be treated as a special case and can be defined in the form:

```
question # ::  CONCEPT ask QUESTION alt yn.
```

In this case the two alternatives asked of the user will be "yes" and "no" while the return alternative values assigned to the *CONCEPT* will be "true" or "false".

7.6 Question Explanations

It is possible to assign an *explanation* to a question statement. If a question statement contains an explanation (which is a character string) then when the question is asked of the user, he has the option of selecting "EXPLAIN" from the menu rather than answering the question. If he does so, a pop-up window appears with the explanation term in it. The creator of the knowledge base can use this explanation term to give a more detailed description of what the question means, to explain each alternative more fully, to describe why the question is being asked, etc. When the user finishes reading the explanation, he is asked the question again. If the user never selects "EXPLAIN", there is no difference from normal questions. The syntax for explanations is:

`question #` :: *CONCEPT* `ask` *QUESTION* `expl` *EXPLANATION.*

in which the *EXPLANATION* term is a character string that will be printed on the screen if requested. This term may contain the formatting commands described in section 8.7.

7.7 Question Options Sequence

While the additional question features described above are all optional, they must appear in the correct order if the are used. This order is:

```
question # ::
  CONCEPT
     ask QUESTION
     alt (ALT 1, ALT 2, ...)
    ralt (RALT 1,RALT 2,...)
    expl EXPLANATION.
```

Chapter 8: Advanced Features

FRO provides many advanced features in addition to the basic abilities described previously. These advanced features are described in this chapter.

8.1 Proposition Attributes

Propositions, as they've been described previously, are knowledge *couples* in that they consist of two pieces of information (the concept and the value). However knowledge is not inherently two part; in general, knowledge can be described as an *n-tuple*, having *n* parts to each knowledge proposition. For example, we might want to refer to a three part proposition of the form "the *ATTRIBUTE* of the *OBJECT* is *VALUE* " as in "the color of the dog is brown". In FRO, n-tuple propositions can be used by assigning *attributes* to the *CONCEPT* portion of the standard knowledge couple proposition. The syntax is:

CONCEPT(ATTRIB-1,ATTRIB-2,...) is *VALUE*

where "*ATTRIB-1,ATTRIB-2,...*" are an arbitrary number of attributes or parameters of the proposition. For example, rather than using the propositions:

```
'the shape of block 1' is 'square'
'the shape of block 2' is 'round'
'the shape of block 3' is 'triangular'
```

the following propositions which have a single attribute could be used:

```
'the shape of'('block 1') is 'square'
'the shape of'('block 2') is 'round'
'the shape of'('block 3') is 'triangular'
```

The advantages of this form become more clear in the next section when we introduce variables, but to preview these benefits we can point out that by using the attribute format, a single rule could be written to calculate the area of any block based on its shape (and dimensions) whereas the former format would

have required a separate area calculation rule for each block.

The use of multiple attributes also has the advantage of simple brevity. For example the fact statements:

```
'the shape of block 1' is rectangular.
'the height of block 1' is 12.
'the width of block 1' is 6.
'the colour of block 1' is blue.
```

could be replaced by the single statement:

```
description('block 1', 'shape is rectangular',
        'height is 12', 'width is 6', 'colour is blue').
```

Note that a proposition with attributes will only match with another proposition if they have an identical number of attributes, each of equal value. Propositions with concept attributes can be used to replace normal propositions anywhere in the system.

Finally, it is possible to assign attributes to the *VALUE* of the proposition in the same way as the *CONCEPT*, should this prove desirable.

8.2 Variables

The flexibility and power of rules can be greatly increased by using variables. A variable must start with a capital letter or an underscore symbol "_" (with no $ or ' symbols). Some examples of variables are:

```
X
VARIABLE
Block_no
```

A variable can be used as a value or an attribute in a premise statement. If this premise statement becomes the current goal of the system, it will succeed if any fact, rule or question can be found that matches all non-variable portions of the statement. In the process of succeeding, the variables will adopt the values of the proposition which matched with the goal. For the remainder of the premise and for the value returned in the conclusion, these variables have fixed values and they act in every way as normal atoms. For example, consider the following statements:

```
rule 1 ::
  'number of sides of block 1' is N
     if 'shape of block 1' is SHAPE
    and 'number of sides of a'(SHAPE) is N.

fact 1 ::  'shape of block 1' is rectangle.

fact 2 ::  'number of sides of a'(triangle) is 3.

fact 3 ::  'number of sides of a'(rectangle) is 4.

fact 4 ::  'number of sides of a'(pentagon) is 5.
```

The rule could be called upon if the system has reached a goal pertaining to the number of sides of block 1. In evaluating the rule, the first sub-goal, "`'shape of block 1' is S`", will succeed by matching with the first fact causing the variable "SHAPE" to adopt the value "`rectangle`". The next sub-goal, therefore, will effectively become "`'number of sides of a'(rectangle) is N`". This, of course, will succeed by matching with fact number 3, resulting in the value "4" being assigned to the variable "`N`". Thus the use of this rule will result in the fact "`'number of sides of block 1' is 4`" being stored in the memory.

An alternative way of using variables is for value passing. This is when variables are used as attributes in the conclusions of rules. In this way, a single rule can be used in more than one situation and the variables in the conclusion are assigned values by the goal which has called the rule, thus allowing values to be passed from the goal to the rule. To illustrate, we can recast the example above so that it can be used for any block, not just for block 1:

```
rule 1 ::
  'number of sides of block'(B) is N
     if 'shape of block'(B) is SHAPE
    and 'number of sides of a'(SHAPE) is N.

fact 1 ::  'shape of block'(1) is rectangle.

fact 2 ::  'shape of block'(2) is triangle.

etc.
```

Now this rule would match with goals pertaining to the number of sides of block 1, block 2, or so on.

Sometimes, variables can be used as an attribute or value in a premise proposi-

tion simply because we are not interested in that value or attribute and we don't want to be required to find an exact match in order to succeed. The underscore character alone "_" can be used for this purpose as an *un-named* variable, or one which will match with anything but will not retain the matching value. This is illustrated in the following example:

```
rule 1 ::
  'object shape' is SHAPE
     if object(SHAPE,_,_,_).

fact 1 ::  object(triangle,large,blue,1.0).
```

It is important to remember that once a particular variable has been assigned a value, it will no longer act as a variable but will act as a constant valued term for the remainder of the current rule. However the variable does not retain that value for any other rule in the system. Therefore there is no relationships between some variable name in one rule and the same name used for a variable in another rule.

Although this explanation of variables may seem confusing, the concept behind their use is fairly simple. An examination of the examples given above and of the sample files in chapter 12 should help to clarify the use of variables.

8.3 Not

syntax: `not` *CONCEPT* `is` *VALUE*

A proposition of this form may be used in a premise. When such a proposition becomes a goal, the system first tries to solve the proposition "*CONCEPT* `is X`" in the normal way and stores the result as a fact in the knowledge base. Then the goal succeeds if the *VALUE* given in the goal is not the same as the value stored. The goal fails if the values are the same or if the proposition "*CONCEPT* `is X`" cannot be solved.

8.4 Known

syntax: *CONCEPT* `is known`

"`is known`" propositions may be used in any premise. When used as a goal, the system first tries to solve the goal "*CONCEPT* `is X`". If the system is able to solve this goal for any value of `X`, the proposition "*CONCEPT* `is known`" succeeds, otherwise it fails.

8.5 Command Propositions

Although they are of minimal importance to the theory of expert systems, much of the practical power of systems created with FRO stems from the use of command propositions. Command propositions can be used in the premise of any rule. If, through the normal reasoning process, a command proposition becomes the system's current goal, it normally succeeds. However in the process of being evaluated as a goal, some useful ***by-product*** is performed, such as printing a message on the screen. The command propositions which are recognized by FRO are listed here in alphabetical order.

8.5.1 DOS Shell

syntax: **dos** *DOS_COMMAND*

The DOS shell command proposition provides access to other programs and utilities through DOS from within an operating expert system. The *DOS_COMMAND* is a character string representing the exact characters that would normally be typed in at the dos prompt. When this command is executed, the screen is cleared and the dos program is run. When the external program is finished, the screen returns to the state it was in before the dos shell command was encountered. For example, it may be possible to use an external graphics program to display pictures to the user during the expert system execution. If some graphics display program is run from dos by the command "`display datafile.pic`", then this program would be called from within a FRO system by the rule:

```
'show graphics' if dos $display datafile.pic$.
```

There is no guarantee that any particular external program will be compatible with this technique. In particular, there will be much less system memory available to the external program than there would be if it where executed from dos normally.

8.5.2 Explanations

syntax: **explanation** *EXPLANATION*

The explanation command is similar to the explain option in question statements. It allows the user to access additional, optional information from the knowledge base. When the explanation command becomes a current goal, the system pauses (the message "`Press any key to continue...`" appears on the screen). In addition, the "`EXPLAIN`" option appears on the menu. If the user simply presses any key or any mouse button, the goal succeeds and system con-

tinues. However if the user selects "EXPLAIN" from the menu, a pop-up window appears and the *EXPLANATION* character string term is printed in it. After the user has finished reading the explanation, he presses any key to continue, the goal succeeds, and the system continues.

Typically, the explanation command will be used right after a print statement which has printed some conclusion, recommendation, etc. on the screen. The knowledge base designer will use the *EXPLANATION* term to include some additional detail about the conclusion or some explanation of how the conclusion has been justified. The *EXPLANATION* term may contain all of the formatting commands described in section 8.7.

8.5.3 Fail

syntax: `fail`

The fail command always fails when it becomes the current goal. This might be useful, for example, in forcing the reasoning to abandon the current attempt to prove some goal and to try to find alternative paths of proving the goal true.

8.5.4 Garbage Collection

syntax: `gc`

Writing information to the screen causes FRO's memory to temporarily fill up with information which it subsequently does not need. This is normally of no concern to the user because every time the system asks a question it clears out this un-needed memory. However if a problem file causes the system to perform a large amount of printing with no questions in between, a **garbage collection** command should occasionally be placed between print statements to force this corrective action.

8.5.5 Loading Additional Problem Files

syntax: `load` *FILE_NAME*

When the FRO shell program is first called from DOS, one of the first things the user does is to load in the main problem file using the main menu (as explained in Chapter 11). However it is possible to have the expert system load in more facts, rules or questions during execution. The *FILE_NAME* is a character string consisting of the name of the additional problem file to be loaded in to the system. The format of these additional problem files is identical to the initial one, except that a top level goal will not normally be included. This feature can

be useful, for example, if the knowledge base is quite large but if most of it is only used under certain circumstances. In this case the knowledge base can be partitioned into smaller files; the main file will be loaded and executed initially and it will make initial queries of the user to determine what additional portions of the knowledge base will be required. The main problem file will then go on to load these additional files and continue execution. The following example illustrates such a situation:

File MAIN.KB:

```
goal 1 :: 'solve problem'.

rule 1 ::
  'solve problem'
     if (     'problem type' is 'type A'
          and load $TYPE-A.KB$
          and 'solve type A problem' )
     or (     'problem type' is 'type B'
          and load $TYPE-B.KB$
          and 'solve type B problem' ).

question 1 ::
  'problem type'
     ask $What problem type do you want solved?$
     alt ('type A','type B').
```

File TYPE-A.KB:

```
rule 2 :: 'solve type A problem' if ... etc.
```

File TYPE-B.KB:

```
rule 3 :: 'solve type B problem' if ... etc.
```

8.5.6 Print to the Screen

syntax: print *PRINT*

The print command is the most common command proposition. The *PRINT* term is a character string or a list of character strings to be printed to the screen. When the command proposition becomes a goal, the *PRINT* term is written to the screen and the goal succeeds. Special command characters can

be embedded in the character strings to provide line feeds, indents, and so on; these are described in section 8.7 which deals with print strings.

Normally, the *PRINT* term is printed on a new line of the screen. However if desired, the system can be forced to print the string right after the previous screen output by including a "+" in front of the print string, for example:

```
print +$a continuation of the previous string$.
```

8.5.7 Print to File

syntax: `printf (`*FILE_NAME,PRINT*`)`

The printf command is similar to the print command except that it is for printing information to a file or to the printer rather than to the screen. The *FILE_NAME* term must either be set to the name of a file (if the file exists, the *PRINT* term will be appended to the end of it, if the file doesn't exist, a new file of that name will be created) or it can be set to "prn" to direct output to the printer. The *PRINT* term ***may not*** contain the same control characters used by the print command, the term is copied exactly as it is given with the following exceptions: First, lists may be used in which case each of the items in the list will be printed out in turn (in this context, a list is a series of terms to be printed, each separated by commas; the entire list should have square brackets on either side). Second, the atom "nl" (for new line) can be included as an item in the print list or as the sole item in the *PRINT* term. This will produce a carriage return and line feed. Note that if output is sent to the printer, it may remain in the printer's buffer and not be printed out until a "nl" is sent.

8.5.8 Reading a term from a file

syntax: `read_term (` *FILE_NAME, TERM* `)`

The "read_term" command can be used to read a single statement from a file. *FILE_NAME* is the name of the file containing the term to be read and *TERM* will return the term. The term must be a legitimate statement as defined by the statement syntax described in the preceding chapters.

8.5.9 Removing a Term from the Database

syntax: `remove (` *TERM* `)`

The remove command can be used to remove a previously stored goal, rule, question, or fact term from the database. The *TERM* term is given in exactly

the same form as is used for defining a statement in the problem file except that the ending period is not required. For example, suppose that the following rule succeeded based on some information which was assumed rather than known with certainty:

```
rule 42 :: weather is raining if ....
```

If it is later determined that the assumption was incorrect, the fact "`weather is raining`", which will have been placed in the database by the success of the rule, could be removed by including the following term in the premise of a later rule:

```
remove ( weather is raining )
```

8.5.10 Restoring the Knowledge Base

syntax: `restore_kb` *NAME*

If a knowledge base has been stored in binary form (using the "`save_kb`" command discussed in section 8.5.11) it can be restored using the "`restore_kb`" command. The *NAME* term is the name which was used for storing the knowledge base. Once a previously stored knowledge base is restored, the entire contents of the current knowledge base is lost with the exception of immediate set of goals being solved. See the example under section 8.5.11.

8.5.11 Saving the Knowledge Base in Binary Form

syntax: `save_kb` *NAME*

The entire contents of the system's knowledge base can be saved in binary form using the "`save_kb`" command. The *NAME* term provides a name used for storing the data; it can be up to eight letters using the same allowable characters as regular file names (do not use the names "`fro`" or "`fro_save`"). This procedure produces files called *NAME.IDB* and *NAME.P00* in the current directory. The advantage of this form of saving files is that they can be reloaded into the system very quickly using the "`restore_kb`" command. The disadvantage is that they cannot be viewed or edited in any way. The following is an example of a problem file which contains an expert system knowledge base and a top level goal which saves the entire problem in binary form, as well as a second file which reloads the knowledge base and executes it:

Problem file #1: (saves the knowledge base after it is loaded)

```
goal 1a :: save_kb test.

rule 1 :: 'solve problem' if ...
                (typical problem file).
```

Problem file #2: (restores the knowledge base and then executes it)

```
goal 1b ::     restore_kb test
           and 'solve problem'.
```

8.5.12 Storing Terms in the Database

syntax: `store ( TERM )`

When a problem file is loaded into the system, the goal, rule, question, and fact statements are stored in the knowledge base. Also, every time a rule succeeds or a question is answered a new fact is generated and is stored in the database. However it may occasionally be beneficial to explicitly tell the system to store a term in the database. This can be done with the store command. The *TERM* term is given in exactly the same format as is used for defining statements in the knowledge base except that an ending period is not needed. For example, if a certain response from the user can generate several conclusions, all of which the system designer would like to store as facts without writing a separate rule for each, the following rule could be used:

```
rule 1 ::
  'check for only child'
     if 'subject is an only child' is true
    and store ( 'subject has sisters'  is false )
    and store ( 'subject has brothers' is false )
    and store ( 'subject has twin'     is false ).
```

Note that goals, rules, and questions can be stored in the same manner as the facts shown here.

8.5.13 Succeed

syntax: `succeed`

The succeed command always succeeds as a goal. For example, if a rule involves terms which the knowledge base designer thinks may fail, yet he doesn't want the rule to fail, he can add "or succeed" as the last premise term of the rule.

8.5.14 Write Term to File

syntax: `write_term (`*FILE_NAME,TERM*`)`

The write_term command is similar to the printf command in that it writes information to a file. However unlike the printf command, write_term is designed for writing knowledge base terms into a file which can later be loaded as a problem file itself. This could be used, for example, to add *configuration* information specific to each user to the main problem file or possibly even to create a system which automatically generates new systems.

The *FILE_NAME* term is a string or atom corresponding to the name of a file. If the file exists, the term will be added to the end of the file, otherwise the file will be created. The *TERM* term should be in exactly the same form as is normally used for a problem file, for example:

```
write_term ($TEST.KB$, goal test is completed)
```

A period and a line feed are automatically added to the end of the term in to file.

8.6 Custom Help Information

During the execution of the FRO program, the user may select the "HELP" alternative from the menu. This will be explained more fully in Chapter 11, however this section explains briefly how the help information shown to the user can be modified from within the problem file. When the FRO system first starts up, a fact is stored in the knowledge base of the form:

`help is` *HELP_STRING*

Where *HELP_STRING* is a character string containing the help information. When the user selects help from the menu, the program retrieves this term and prints the *HELP_STRING* to the screen. It is therefore possible to change the information printed to the screen from within some rule by removing this term using the "remove" command and adding a new term using the "store" command. For example, the following rule will achieve this:

```
rule 1 ::
  'change help'
     if remove ( help is _ )
    and store  ( help is $New HELP information$ ).
```

The character string stored can use all of the formatting commands described in the next section on print strings (including the "pause" if more than one screen

full of information is desired). This technique can be used once at the beginning of a system to create help that better suits a particular application or it can be used frequently throughout the system to create context sensitive help.

8.7 Print Strings

As mentioned in several sections above, print strings (or character strings which are to be printed to the screen) may have special control characters imbedded in them. Print strings are given these special capabilities largely because it is not convenient to write out the strings in the knowledge base in exactly the same way that they should appear on the screen during program execution. For example, terms in the knowledge base should be written out making liberal use of indentation to make the knowledge base clear, yet this should not cause the print strings to be similarly indented on the screen during execution. Furthermore, it may not be clear while writing the knowledge base what width of window certain print strings will appear in.

When printing out character strings to the screen, FRO gets around these problems by using spaces, tabs and carriage returns only to determine the ends of words. Once it has a list of the words in the print string, it ignores all spaces and so on. FRO then prints this list with a single space between each word and with automatic word-wrap (end-of-line identification). However, this process results in the user losing the power to put in intentional spaces, tabs, and carriage returns. FRO therefore gives the user back this power by allowing him to insert special control characters in print strings to achieve these and other actions. Most control characters begin with a ^ symbol followed by a lower case letter and sometimes an integer number. For example, if the command "^t" (which stands for a 5 space tab) is included in a print string, the system will print out five spaces in its place. The complete list of the allowable print string commands is as follows (the # symbol stands for some integer number):

- ^c# – causes the next word to be printed starting in column #
- ^l# – sets a left indent of # spaces (indent begins to take affect after the first line feed subsequent to the command)
- ^n – new line
- ^p – pause
- ^r# – sets a right indent of # spaces
- ^s# – prints # fixed spaces
- ^t – tab (five spaces)
- ^ ^ – prints a ^ symbol

^ | – prints a | symbol
| – (ASCII 124) alternative symbol for a new line
· – (ASCII 250) alternative symbol for fixed space

Also, where ever a print string is allowed, a list of print strings may also be given, according to the form:

[*PRINT STRING 1,PRINT STRING 2,....*]

In which case each of the print strings will be printed one after another in turn. The main use for such lists is when printing a mixture of constant print strings and variables, as in the following example portion of a rule:

```
and 'the shape of object'(OBJ_NUMBER) is SHAPE
and print [$The shape of object number $,
      OBJ_NUMBER, $ is $, SHAPE]
```

8.8 Math Operators

Mathematical operations can be incorporated into a FRO expert systems to perform calculations. This is done by using a mathematical comparison as a proposition in the premise of some rule. A mathematical comparison has the simple form:

EXPRESSION COMPARISON_OPERATOR EXPRESSION

Allowable *EXPRESSIONS* and *COMPARISON_OPERATORS* will be described below, but some example are:

```
AGE > 65
ENERGY = MASS * SPEED_OF LIGHT ^ 2
```

When a mathematical comparison becomes a goal, both expressions are evaluated and are then compared according to the comparison operator. If the comparison is true the goal succeeds, if not the goal fails. In some cases (described below) the left expression may be a variable, in which case it is assigned a value which makes the comparison true and the goal succeeds. The allowable comparison operators are:

= – left equal to right
\= – left not equal to right
> – left greater than right
>= – left greater than or equal to right
< – left less than right

`=<` – left less than or equal to right

For the "=" operator, the left expression may be a variable which will be assigned the value of the right expression when the proposition becomes a goal. Also, with the "=" operator only, any non-numeric expressions may be used, for example the goal will succeed if two identical character strings are compared. In every other case, non-variable numbers must be used on both right and left sides (recall that variables which have been used previously in the same premise and have thus been assigned values may be used since they are now effectively constant valued terms). Numbers may be either integers or real numbers. With real numbers, the decimal place must have at least one digit on either side of it (e.g. 0.1 or 1.0 rather than .1 or 1.). The allowable numeric operators are:

`X + Y`	– Addition
`X - Y`	– Subtraction
`X * Y`	– Multiplication
`X / Y`	– Ordinary division – returns a floating point number
`X // Y`	– Integer division – returns an integer. If a floating point number is supplied as an argument, only the integer portion will be used for the division
`X ^ Y`	– Exponentiation
`- X`	– Unary minus (negative of X)
`X mod Y`	– Returns the remainder of X divided by Y (integers only)
`abs(X)`	– Absolute value of X
`acos(X)`	– Arc cosine of X
`asin(X)`	– Arc sine of X
`atan(X)`	– Arc tangent of X
`cos(X)`	– Cosine of X
`exp(X)`	– e raised to the power of X
`ln(X)`	– Logarithm base e
`log(X)`	– Logarithm base 10
`sin(X)`	– Sine of X
`sqrt(X)`	– Square root of X
`tan(X)`	– Tangent of X
`round(X,N)`	– X rounded to N decimal places (N between 0 and 15).

The following example uses a few of the mathematical features in a rule:

```
rule 42 ::
  'the size of the object' is SIZE
      if 'the shape of the object' is SHAPE
     and (   (     SHAPE = square
               and 'length of a side' is L
               and AREA = L ^ 2 )
          or (     SHAPE = triangle
               and 'length of base' is BASE
               and 'length of height' is HEIGHT
               and AREA = (BASE * HEIGHT)/2 ) )
     and (    (     AREA < 1.0
                and SIZE = small )
           or (      AREA >= 1.0
                and SIZE = large ) ).
```

8.9 Hidden Statements

It has been shown that the use of a rule or question statement to solve a goal results in a new fact being stored in the knowledge base. This is so that the system will not have to infer or ask about the same information twice, should it be needed a second time. However it is sometimes desirable to prevent such facts from being stored. This can be done by using **hidden statements**. The syntax for hidden rules is:

rule # :: **hide** *CONCLUSION* **if** *PREMISE*

and for hidden questions:

question # :: **hide** *CONCEPT* **ask** *QUESTION*.

In both cases, the statement type and statement number can be left off; in the case of the question statements, lists of alternatives and explanations can be added as in chapter 7.

Hidden statements are useful when some rule or question can be used repeatedly for different situations within the expert system. For example, the following rule and question statements could be used to ask the user whether or not he wishes to continue. If these statements were not declared as hidden, the could only be used once since subsequently the values obtained from their first use would be stored in the system's memory.

```
rule 12 :: hide
  'what to do next' is VALUE
     if 'users wish' is VALUE
    and (    (     VALUE = continue
                 and print $Very well, let's continue....$ )
          or (     VALUE = stop
                 and print $O.K., we'll stop now.$ ) ).

question 12 :: hide
  'users wish'
     ask $What would you like to do now?$
     alt (continue, stop).
```

Chapter 9: Excluded Features

For the sake of comparison with other expert system shells it is worthwhile mentioning some typical expert system features which are not included in FRO. The reason for most of these exclusions is simply that this system is not intended to be all encompassing and, in fact, is not even intended primarily as a stand-alone program. Rather it is intended as an experimental control system for the more advanced Dempster-Shafer based uncertain reasoning system described in Part II. Never-the-less these features would be useful in FRO and may be added at some later date.

9.1 View Knowledge Base

This feature involves the ability to stop the reasoning process at any point and scan or view the contents of the knowledge base.

9.2 Uncertainty

Many expert systems include simple forms of handling uncertainty using *certainty factors*. Since FRO is actually designed for use in conjunction with the Dempster-Shafer based uncertainty features, there is no point in including certainty factors in the rule-based portion.

9.3 Multi-Valued Propositions

One side effect of the fact that rule-based FRO is fundamentally intended as a categorical control system for Dempster-Shafer based reasoning is that each proposition *CONCEPT* is assumed to have only one possible *VALUE*. This means, for example, that if a goal matches with the *CONCEPT* of some rule but the *VALUE* does not match, then the goal to fail rather than search for some other rule which might allow the goal to succeed. This can also be troublesome where

some *CONCEPT* is not inherently single-valued, but could have several different values simultaneously. For example, the *VALUE* of the *CONCEPT* "`today's weather`" could be both "`raining`" and "`cold`" at the same time. Multi-valued propositions may be incorporated at a later date.

Chapter 10: Problem Design

As a final comment on problem files, a few brief notes should be made about good problem file style and about application development techniques.

- Create a very small and simple knowledge base at the very beginning of the application development process. Having a working system, even though it is simplified to the extreme, provides a vital focus for the process of obtaining and organizing knowledge.
- Proceed using an iterative process of expanding the prototype system in both knowledge breath and knowledge depth. That is, early prototypes should contain a little knowledge about all aspects of the problem and a large amount of knowledge about a few specific aspects of the problem.
- The first goal of the system should be to try and make the knowledge base model the way that a human expert would actually try to solve the problem. Only secondly should "programming tricks" and so on be used to try to make the knowledge base more efficient.
- Look for repetition in the solution process and exploit it using variables and the other tools described in this documentation.
- Provide abundant information back to the user to inform him of the status of the solution, of any sub-conclusions, and so on.
- Experimenting with different approaches is the best way to find the best way of solving problems.

Chapter 11: Running FRO

Thus far, we have discussed the underlying theory of the FRO system and the techniques for developing a problem file. In this chapter we discuss the actual process of running FRO after the problem file exists.

11.1 Starting FRO

The FRO system is written for an IBM PC, IBM AT or compatible computer. It should operate correctly with either floppy diskettes or with a hard disk and with any type of monitor. It is designed for 512K or more of system memory but it may function with as little as 256K (see section 11.5). FRO is designed to work with a Microsoft compatible mouse, although a mouse is not required. The system is written in ARITY PROLOG version 4.0. The program consists of three mandatory files, FRO.EXE, FRO.IDB, and FRO.P00. All three files must be in the current directory for the program to operate correctly. In addition, there may be a configuration file called FRO.ENV which must be in the same directory to be effective. To start up FRO, simply go to the directory containing the FRO program files and at the DOS prompt type:

```
FRO <enter>
```

If a mouse is not being used, the following should be typed:

```
FRO /n <enter>
```

The main screen should then appear.

It is also possible to use the command line (or the DOS command which calls FRO) to supply the name of the problem file which should be used. The syntax is:

FRO *FILE_NAME* <enter>

where *FILE_NAME* is the name of the problem file. For example:

```
FRO TEST.KB <enter>
```

or, if a mouse is not being used:

```
FRO TEST.KB /n <enter>
```

When a problem file is named in this way, the system does not go to the main menu, but rather loads the file and begins execution directly as described in section 11.3.

11.2 The Main Screen

The main screen consists of the main menu across the top, a large dialog area underneath, and a border all around. The main menu contains the following options:

F1: HELP
F2: FILES
F3: EXECUTE
F4: SAVE
F5: TRACE
F6: QUIT

A main menu option can be selected with the keyboard by pressing the function key corresponding to the desired command or with the mouse by pointing to the option with the mouse cursor and pressing any button. Each option is explained below.

11.2.1 F1 - HELP

"HELP" will cause a pop-up window to appear which contains information describing what the user's options are.

11.2.2 F2 - FILES

This option is used for loading problem files into the system. When "FILES" is selected, the system asks the user for the name of a problem file. The user must enter the name of a problem file, including the file name extension and the path if the file is not in the current directory. When the user presses enter, the problem file is loaded into the system's memory. The message "Loaded!" appears when the loading is complete. More than one file can be loaded in sequentially if several files are used for one problem (and if the first file does not itself contain instructions to load the other files). See section 11.5. for possible

errors that could occur at this point.

11.2.3 F3 - EXECUTE

Once a problem file has been loaded into the system, selecting "EXECUTE" will begin the problem execution. See Section 11.5 for an explanation of possible error messages that can occur at this point. The execution of a problem is described more fully in Section 11.3 below.

11.2.4 F4 - SAVE

The "SAVE" option is a special feature of FRO that allows systems to be created which load quickly and are self-starting; this can be thought of as compiling a problem file. To use this feature, a normal problem file is written and debugged. The FRO program is then started and the problem file is loaded into the system as described above. The "SAVE" option is then selected from the main menu. This causes the entire knowledge base to be saved in the files FRO_SAVE.P00 and FRO_SAVE.IDB. When FRO is started thereafter, it finds that these files exist and goes on to restore the saved knowledge base and begin execution, rather than going to the main menu.

The advantage of this procedure is that after the initial saving, the knowledge base loads much faster and begins execution automatically. The disadvantage is that the knowledge base saved in this way cannot be viewed or edited with an ASCII editor (although the original problem file still exists and can be re-loaded at any time).

To cancel the automatic restoration and execution of a previously stored knowledge base, simple delete the FRO_SAVE.P00 and FRO_SAVE.IDB files from the current directory and FRO will revert to normal operation.

11.2.5 F5 - TRACE

Selecting "TRACE" from the main menu turns on the tracing facilities. No observable changes appear at this point other than a confirmation message. The tracing features of FRO are explained in Section 11.4. Selecting "TRACE" a second time toggles the tracing off.

11.2.6 F6 - QUIT

If "QUIT" is selected from the main menu (or from anywhere else in the system)

execution will halt and control will return to DOS. It is preferable to exit FRO in this manner rather than by pressing "CONTROL-BREAK" or "CONTROL-ALT-DEL".

11.2.7 F9 - STATISTICS

Although it does not appear on the main menu, pressing F9 will bring up the memory statistics window. This is explained in Section 11.5.

11.3 The Execution Screen

Once "EXECUTE" is selected from the main menu, the main screen will be replaced with the execution screen which looks very similar but has a slightly different menu across the top. At this point the program will begin execution and messages or questions will begin to appear on the screen.

To answer questions, the user must simply type the number which appears next to the best alternative and press enter or alternatively he must point to the best alternative with the mouse cursor and press any mouse button. If an incorrect answer is given, the question is asked again. If the system asks a question which does not have a list of alternatives, the user types in his response and presses enter.

In addition to answering questions, the user may select commands from the execution menu, as explain below.

11.3.1 F1 - HELP

"HELP" will cause a pop-up window to appear which contains help information.

11.3.2 F2 - EXPLAIN

As described above, it is possible to attach explanations to questions and propositions in the knowledge base which are displayed only upon the user's request. When the system encounters such an explanation term, the "EXPLAIN" alternative appears in the execution menu. The user may select this alternative with the mouse or the keyboard. The explanation will then appear in a pop-up window.

11.3.3 F4 - WHY

Whenever the system asks the user a questions, the option "WHY?" appears on

the menu. If the user wishes to find out more about why the system is asking this particular question, he may select this option from the menu. If he does so, a window will appear which displays the goal which the system is trying to solve and the number of the rule which is currently being used to try to solve the goal. The system then gives the user the option of viewing the rule, of asking why the system wants to prove that goal to be true (thus allowing the user to view the entire chain of goals up to the top-level goal), or of continuing with the question.

11.3.4 F5 - TRACE

Selecting "TRACE" from the execution menu turns on the tracing facilities. The main window will then split in half, creating a second tracing window. The tracing features of FRO are explained in Section 11.4. Selecting "TRACE" a second time toggles the tracing off and closes the tracing window.

11.3.5 F6 - QUIT

If "QUIT" is selected from the execution menu execution will halt and control will return to DOS. It is preferable to exit FRO in this manner rather than by pressing "CONTROL-BREAK" or "CONTROL-ALT-DEL".

11.3.6 F9 - STATISTICS

Although it does not appear on the execution menu, pressing F9 will bring up the memory statistics window. This is explained in Section 11.5.

11.4 Tracing

FRO includes tracing capabilities which are useful (if not essential) for debugging problem files. By tracing the reasoning during the execution of a problem file, the user can see which goals are failing unexpectedly (often an indication of incorrect syntax), can observe the values adopted by variables, can check the precedence of "and" and "or" terms, and so on.

Once tracing has been turned on and execution has begun, the screen splits into a dialog screen and a tracing screen. Every time the system finds a goal which it must solve, the goal is printed out in the tracing window and the system pauses. The user allows the system to proceed by pressing any keyboard key (except "s") or by pressing the LEFT mouse button. The system will then go ahead and attempt to solve the goal. It will display a message stating whether it found

a rule, a fact, a question, or nothing that successfully matches with the goal. After resolving the matching term (possibly causing more sub-goals to first be evaluated and traced) a message appears in the tracing window stating whether the goal succeeded or failed. Whenever the system comes across terms that require user interaction, this occurs in the dialog window in the normal fashion. In this way, the user can follow the step-by-step reasoning of the system.

One additional useful feature is **skipping**. When a goal is found and displayed in the tracing window, the user may press "**s**" or the RIGHT mouse button which causes the system to proceed without tracing until that goal succeeds or fails. This allows the user to move quickly through branches of the reasoning tree which he is not interested in tracing. With this skipping feature and with the ability to turn the tracing on and off at any time, the user may trace the reasoning quite efficiently even in very large systems .

It should be noted that when terms are written out in the tracing window, variables do not retain the names given to them in the problem file but rather appear as a hexadecimal number preceded by an underscore (e.g. _00A39). Once variables have adopted some value, that value appears in the tracing window in place of the variable.

11.5 Errors and Memory Management

There are two main areas which can lead to problems with the FRO program. The first and most frequent is errors in the problem file. Many syntactical errors will cause PROLOG error messages to appear when the problem file is first loaded. The message will likely say "`error using operators`" and will list the offending statement. This points the user to the term that needs correcting, although it does not indicate the exact error. The user should look for spelling mistakes in key words, incorrect use of parenthesis, and so on. Occasionally, if no apparent errors can be found, additional parentheses should be added to clarify precedence intentions, for example the term:

```
remove goal run
```

should be replaced with:

```
remove ( goal run )
```

Once syntax is correct, the problem file should load correctly. There still could be errors in the problem file. For example, proposition concepts or values could be spelt differently in different places. The use of tracing should help to clarify these errors. Finally, there could be some problems with memory management. This topic will be discussed at a later date.

The FRO program requires two adjustable memory areas, one for ***stack*** space and one for database ***worlds***. The amount of memory assigned to the system for each of these uses is displayed when the user presses F9 will running the system.

Stack space relates to the basic operational memory requirements of the system. The system must be allotted enough stack memory to operate, but if more stack memory is assigned than is needed, less memory will be available for database memory, leading to slow system operation.

Chapter 12: Sample Files

12.1 Sample file 1: Wine Selection

The following is a simple system which gives advice on which colour of wine should accompany a meal. It uses only FRO's basic capabilities.

```
%%%% Problem File: SAMPL-01.KB
%%%% Sample Expert System Number 1
%%%%
%%%% System to suggest the best colour of wine to accompany a meal.
%%%%
%%%% This system uses only basic goal, rule, and question statements,
%%%%   along with the "print" command proposition.
%%%%

%%%% The Goal Statement
goal 1 :: 'the session' is complete.

%%%% The Rule Statements.
rule 1 ::
  'the session' is complete
     if print $This expert system will suggest the best colour
              of wine to go with your meal.$
    and 'the best colour' is suggested.

rule 2 ::
  'the best colour' is suggested
     if 'the main component' is meat
    and print $A red wine is generally best with meat.$.
```

```
rule 3 ::
  'the best colour' is suggested
     if 'the main component' is fish
    and print $The best wine with fish is white.$.

rule 4 ::
  'the best colour' is suggested
     if 'the main component' is poultry
    and (    (     'meal has turkey'
               and print $A red wine is preferable with turkey.$ )
          or (     'meal has turkey' is false
               and print $A white wine would best suit this meal.$ )).

%%%% The Question Statements.
question 1 ::
  'the main component'
    ask $What is the main component on this meal?$
    alt (meat, fish, poultry).

question 2 ::
  'meal has turkey'
    ask $Will the meal contain turkey?$
    alt yn.
```

12.2 Sample file 2: Golf Club Selection

This simple problem file deals with selecting the best golf club for a particular shot. The knowledge which is to be incorporated into the system is summarized below. The knowledge base, which consists mainly of FRO's most simple features, is given on the following pages.

Location:		Best Club:
on the green		putter
in a sand trap		sand wedge
on the fairway	Distance to hole:	
	less than 100 yards	pitching wedge
	between 100 and 130 yards	#8 iron
	between 130 and 155 yards	#5 iron
	between 155 and 180 yards	#3 iron
	more than 180 yards	#3 wood
at the tee	Par:	
	par 3	#5 iron
	par 4	driver
	par 5	driver

```
%%%% Problem File: SAMPL-02.KB
%%%% Sample Expert System Number 2
%%%%
%%%% System to suggest the best golf club to use
%%%%  for a particular shot.
%%%%
%%%% This system uses explanations and
%%%%     variables in the 'Fairway shot' rules.
%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% The Goal Statement
goal 1 :: 'best club' is known.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% The Rule Statements.
%%%% PUTTING %%%%%%%%%%%%%%%%%
rule p1 ::
  'best club' is putter
     if location is 'on the green'
    and print $The best club for this shot is a putter$
    and explanation $Once on the green, you should be
                     putting. Therefore use a putter$.

%%%% WEDGE SHOTS %%%%%%%%%%%%%%
rule w1 ::
  'best club' is 'sand wedge'
     if location is 'in a sand trap'
    and print $The best club for this shot is a sand wedge$
    and explanation $The loft provided by the wedge is needed
                    to lift the ball out of the sand trap$.

%%%% TEE SHOTS %%%%%%%%%%%%%%%%
rule t1 ::
  'best club' is '#5 iron'
     if location is 'the tee'
    and par is 'par 3'
    and print $The best club for this shot is a #5 iron$
    and explanation $For short tee shots (par 3) a #5 iron
                     will provide the correct distance$.
```

```
rule t2 ::
  'best club' is driver
     if location is 'the tee'
    and (    par is 'par 5'
          or par is 'par 4' )
    and print $The best club for this shot is a driver$
    and explanation $For long tee shots (par 4 or 5) a
                     driver will provide the required
                     distance$.

%%%% FAIRWAY SHOTS %%%%%%%%%%%
rule f1 ::
  'best club' is CLUB
     if location is 'on the fairway'
    and 'best fairway club' is CLUB
    and print [$The best club for this shot is a $,CLUB]
    and explanation $This club is most likely to provide
                     the correct distance for the
                     fairway shot$.

rule f2 ::
  'best fairway club' is 'pitching wedge'
    if distance is 'less than 100 yards'.

rule f3 ::
  'best fairway club' is '#8 iron'
    if distance is 'between 100 and 130 yards'.

rule f4 ::
  'best fairway club' is '#5 iron'
    if distance is 'between 130 and 155 yards'.

rule f5 ::
  'best fairway club' is '#3 iron'
    if distance is 'between 155 and 180 yards'.

rule f6 ::
  'best fairway club' is '#3 wood'
    if distance is 'more than 180 yards'.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
%%%% The Question Statements.
question 1 ::
  distance ask $How far is it to the hole?$
         alt ('less than 100 yards',
              'between 100 and 130 yards',
              'between 130 and 155 yards',
              'between 155 and 180 yards',
              'more than 180 yards')
         expl $Distance is one of the primary factors
               in selecting the best club$.

question 2 ::
  par ask $What is the par for this hole?$
    alt ('par 3','par 4','par 5')
    expl $The par of the hole gives an indication of
          the distance and hence the club required$.

question 3 ::
  location ask $Where are you making this shot from?$
         alt ('the tee',
              'on the fairway',
              'in a sand trap',
              'on the green')
        expl $The different locations have characteristics
              which suit some clubs better than others$.
```

PART II

FRAME-BASED FRO

The frame-based portion of the FRO expert system shell can be used in conjunction with the rule-based portion (described in Part I of this documentation) to facilitate inexact reasoning based on the Dempster-Shafer (or D-S) theory of evidence. This document will not provide a review of this theory, rather it will describe how to use frame-based FRO, including a brief introduction (chapter 13), a description of the interactive belief interface system used by FRO (chapter 14), an example of the MKB program which assists in the creation of problem files for frame-based FRO applications (chapter 15), an explanation of both design problem files (chapter 16) and control problem files (chapter 17), a brief comment on running frame-based applications (chapter 18) and a listing of some sample problem files (chapter 19). The recommended approach to learning the system is to learn the rule-based portion of FRO first (see Part I of this manual) and then to alternate between reading this part of the manual, examining the sample files, and experimenting with the program itself.

Chapter 13: Overview of frame-based FRO

This section provides a brief overview of how FRO uses D-S theory to represent uncertain information and perform inexact reasoning.

The basic knowledge representation scheme for the frame-based portion of FRO is quite different from the rule-based portion. In the frame-based portion of FRO, each major concept or variable is represented by a **frame of discernment** (which we commonly refer to as simply a **frame**). Each frame is represented in the system by a name and a list of possible alternatives or values. These alternatives must be exhaustive and mutually exclusive (that is, one and only one of them may be true). Knowledge about how the alternatives for one frame imply knowledge about the alternatives in another frame is stored in structures called **links**. The knowledge contained in links may be certain or uncertain. The collection of all the frames defined for some problem and the links connecting the frames to each other makes up a graph structure called a **frame network**. The frame network with its constituent frames and links, then, makes up the representation of all the variables involved for some problem, all the possible values for these variables, and all of the inferential relationships between these values.

Belief about which value for some variable is though to be true or which values are known to be false is represented by a **belief function**. When the system asks a user for information about some variable, the response is provided in the form of a belief function. This belief function is added to the frame which is being asked about. At the same time, the links connected to that frame are used to infer belief about any related frames. Whenever the user responds to a frame-based question, then, both his answer and any information which can be inferred from his answer is added to the system. In this way inexact reasoning is accomplished.

The frame network approach is well suited for representing concepts, the relationships between them, and the belief about them. We call this knowledge the **design knowledge**. However it is not well suited for representing procedural or

control knowledge which makes up information about when to ask questions, when to draw conclusions, etc. In FRO, this function is performed by the rule-based portion of the program (in fact, this is why the rule-based portion was created in the first place). The rule-based portion and the frame-based portion can interact with each other through a series of interface command propositions which can be embedded in the rule-based problem file.

In order to use FRO to create an expert system with inexact reasoning, then, the expert and the knowledge engineer must create at least two problem files. The first, the **design problem file** defines each frame that relates to the problem, the links that relate the frames, and an initial belief that should be assigned to the frames. The second file, the **control problem file**, contains the procedural rules about which questions to ask, when to ask them, which frames to consult for conclusions, how to interpret the conclusions for particular situations, etc. This dichotomy between the two types of knowledge is fairly transparent to the user of the system who operates the finished product in almost exactly the same way that he operates a rule-only system, with the exception of the uncertain belief input/output interface which is discussed in the following chapter.

Chapter 14: The IBIS Belief Interface

All uncertain communication between the FRO system and the user is accomplished with the aid of FRO's unique belief interface. This interface allows the relatively simple input and output of complicated D-S belief functions. The interface is called **IBIS**, for Interactive Belief Interface System. This chapter explains how to use the IBIS interface.

14.1 Belief Input

When FRO needs to obtain uncertain input from the user, it first asks the user a question in exactly the same form as is used for categorical questions. If the system is trying to find the user's preferred vacation destination, for example, it might ask the user the following question:

```
Where would you prefer to go?
1.  Paris
2.  New York
3.  Hawaii
4.  Australia
>>>>
```

Note that the alternatives for these questions are exhaustive and mutually exclusive. That is, one and only one of the alternatives represents the best or correct answer to the question. If the user has a categorical response, he would be allowed to answer this question in exactly the same manner as for rule-based FRO. However an important difference is that an "UNCERTAINTY" option will appear on the menu while the question is being asked. If the user is not completely certain about his response, he may select this option. When this occurs, a pop-up window appears and the question is re-asked in a different format shown in figure 14.1.

As shown in figure 14.1, the same question is asked and the same alternatives are offered. However a bar-chart style graph appears beside the list of alternatives;

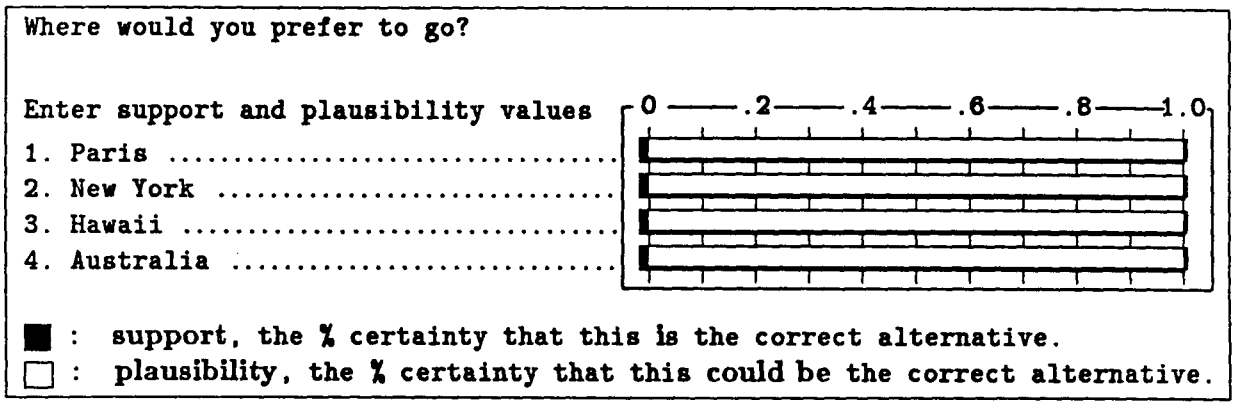


Figure 14.1: Initial belief input screen

two bars are shown for each alternative. The bars on the graph can be moved up or down by selecting the bar with the mouse and *sliding* the bar up or down. In this way the bars allow the user to enter two *belief parameters* for each alternative. One bar (the white bar in figure 14.1) represents the **plausibility** of the alternative which is the user's percent certainty that the alternative *could be* the correct answer to the question. The plausibility is initially set to 1.0 for each alternative indicating that each of the alternatives is initially considered to be equally possible by the system. The other bar (the black bar in figure 14.1) represents the **support** of the alternative which is the user's percent certainty that the alternative *is* the correct answer to the question. The support is initially set to 0 for each alternative representing the fact that the system initially has no specific belief for any one of the alternatives over any other. The initial state of the system, then, is one of complete ignorance about the answer to the question.

If the user has some level of belief that one of the alternatives is the best answer to the question, then he may point to the "support bar" on the graph adjacent to that alternative with the mouse cursor. He then presses and holds down any of the mouse buttons (the mouse cursor will change shape if he has successfully selected a bar). He may then move the cursor in one direction or the other in order to change the support value for that alternative. For the example given above, the user may feel somewhat sure that she would like to go to Hawaii (with a percent certainty of 0.4, say). Figure 14.2 shows the result of the user entering this level of belief.

Notice that in figure 14.2 the plausibility values for all alternatives other than Hawaii are lowered to 0.6. This has been done automatically by the system after the user entered a support of 0.4 for Hawaii because belief indicating that one particular alternative *is* the correct answer implies the same amount of belief

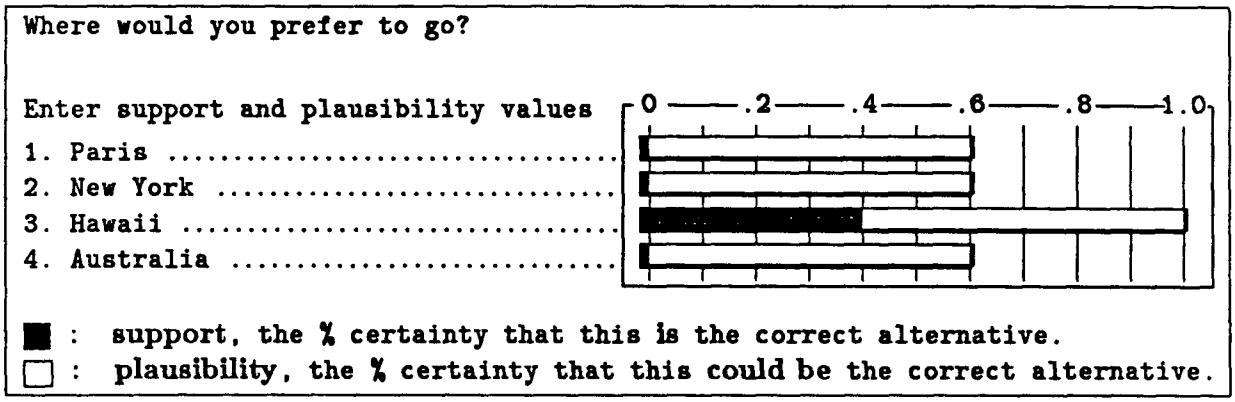


Figure 14.2: Belief input screen showing some support for "Hawaii"

that all other alternatives are *not* possible, and thus their plausibilities should be lowered correspondingly.

Suppose also, for this example, that the user's husband would prefer to go to Australia but that he is not vary committed to this location because it would be quite expensive. This would lead to a level of support of, say, 0.2 for the alternative "Australia", as shown in figure 14.3.

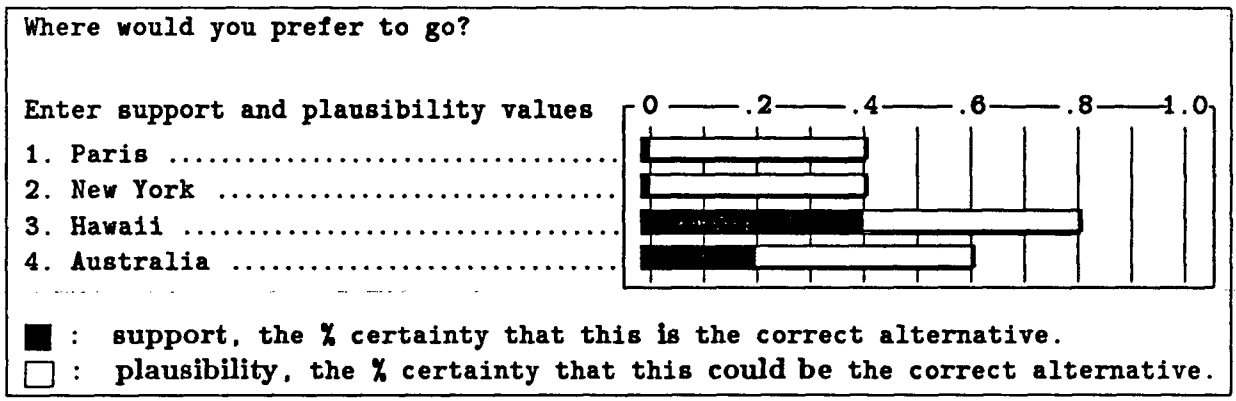


Figure 14.3: Belief input screen showing support for both "Hawaii" and "Australia"

Figure 14.3 shows how the plausibility values will be lowered further by this additional assignment of support. Finally, suppose that the user is completely sure that she does not want to go to Paris since she doesn't speak french. She could therefore lower the plausibility of the alternative "Paris" to 0, indicating that this is not a possible best location. This is shown in figure 14.4.

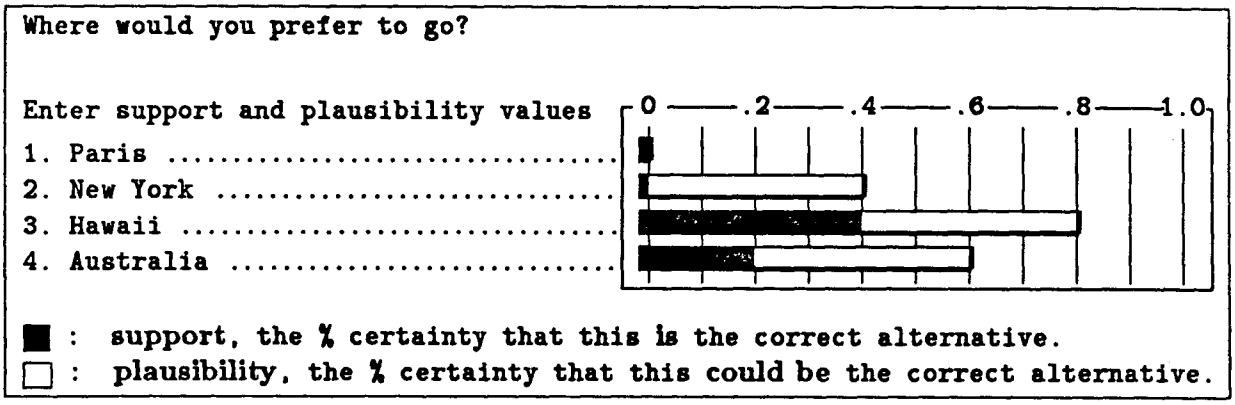


Figure 14.4: Final belief input screen

Figure 14.4, then, represents the final belief input screen for this example. The user would then point with the mouse cursor to a place on the screen which displays "Click here when finished" and press any of the mouse buttons. The belief input window would then disappear and the system would continue.

It should be noted that there are several things which the user is not allowed to do when entering belief. He cannot, for example, move any of the values below 0.0 nor above 1.0. Nor can he increase the support value greater than the support value. If the user attempts to perform any of these or other un-allowed operations, the system will prevent the move and a warning message will appear.

There are also a great many things which the user may do when entering belief; that is, the scheme provides a very flexible tool for the representing the user's belief. The best way for a user to understand the possibilities of the input scheme is to experiment with entering belief for a simple sample system.

14.2 Belief Output

The FRO system can output uncertain belief using the same format which is used for entering uncertain belief. Figure 14.5 shows such an output screen.

For belief output, the user may not alter the values shown. However, two options will appear in a pop-up menu. One will allow the user to view the output in the form of a numerical rather than a graphical output. The second option allows the user to continue with the session. Both options are selected by pointing to the appropriate box with the mouse cursor and by pressing any of the mouse buttons.

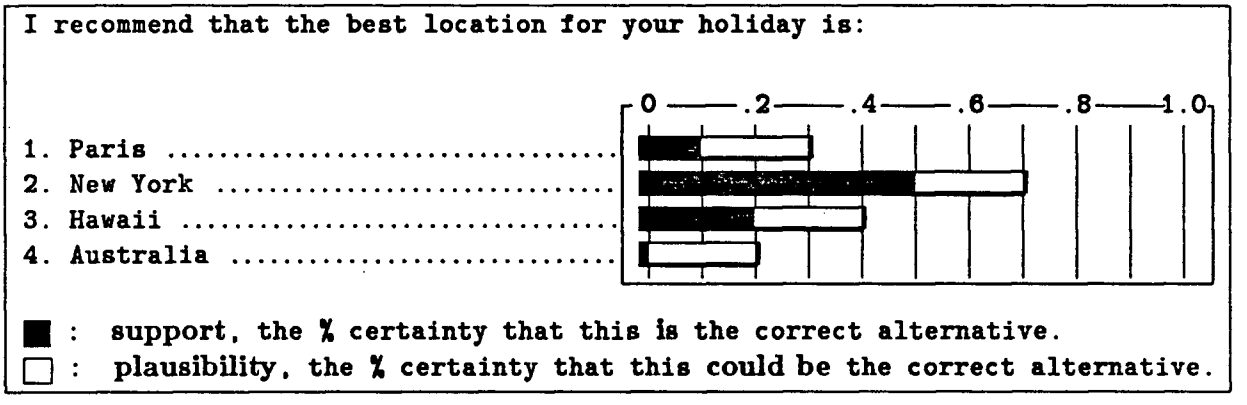


Figure 14.5: Belief output screen

Chapter 15: Using MKB

MKB (which stands for **M**ake **K**nowledge **B**ase) is an expert system written for the FRO shell which is designed to help create problem files for new FRO expert systems. Since the syntax of FRO design problem files is much more complex and awkward than FRO control knowledge bases, MKB is by far the easiest way of creating simple frame based systems. While MKB is not fully implemented yet, those features which are available are described below

In order to operate MKB, all FRO files as well as the following MKB files should be in the current DOS directory:

```
MKB
MKB-CREA
MKB-EDIT
MKB-LOAD
MKB-STOR
```

MKB is the main problem file and will load the other files as needed. To begin execution, simply follow normal FRO procedures and, at the DOS prompt, type:

```
FRO MKB <enter>
```

Note that a mouse **must** be used for rule-based FRO. After FRO and MKB have been loaded, the MKB main menu will appear. The system is menu driven and is fairly self-explanatory once the user is familiar with some D-S theory and with the uncertainty interface. The following is a listing of a sample session in which a system is being created to help select the best vacation destination. In this example, it is being assumed that the possible vacation destinations are Paris, New York, Hawaii, or Australia. It is also being assumed that the factors which determine the best alternative are the main attraction of the area (with the possible alternatives being either shopping and nightlife or sunshine and beaches) and cost (with the possible alternatives being cheap, medium, or expensive). These frames and alternatives are entered into the system first, followed by the uncertain relationships between them. Note that illustrations of the pop-up windows which are used in defining these relationships are given after

the entire session listing. The session continues with the definition of some initial level of belief stating belief that users will generally prefer cheaper vacations. Finally, all of the information entered into the system is written out to files, one corresponding to the design knowledge base and one corresponding to the control knowledge base.

```
************************************
*              MAKE-KB             *
************************************
* A system  to  help construct  the *
* knowledge base for expert systems *
*     which use D-S uncertainty     *
************************************
          version 1.00

**** MAIN SYSTEM MENU ****
What do you want to do?

  1.  Go through tutorial session
  2.  Create a new knowledge base
  3.  Load an existing knowledge base
  4.  View and/or Edit a knowledge base
  5.  Save a knowledge base
  6.  Finish
>>>> 2
       (create)


************************************************
SECTION NUMBER 1 : SET UP
************************************************

Enter a short description of what the expert system will be about.

>>>> Chosing a Vacation Destination

If desired, MAKE-KB can construct a simple "control knowledge base"
which can be used along with the "design (uncertainty) knowledge base"
to complete an expert system. Do you want a control knowledge base?

  1.  yes
  2.  no
>>>> 1
```

```
************************************************
SECTION NUMBER 2 : FRAME DEFINITION
************************************************

The first step is to define each "concept" and its possible values
which might be used in the system. This is done by defining "frames".
We will now begin entering the names and the possible values of
frames.


Enter a name for frame number 1
     (or just press "enter" if all frames have been given)

>>>> MAIN ATTRACTION


Enter alternative number 1 for frame "MAIN ATTRACTION"
     (or just press "enter" if all alternatives have been given)

>>>> shopping and nightlife


Enter alternative number 2 for frame "MAIN ATTRACTION"
     (or just press "enter" if all alternatives have been given)

>>>> sunshine and beaches


Enter alternative number 3 for frame "MAIN ATTRACTION"
     (or just press "enter" if all alternatives have been given)

>>>>
```

```
What will the frame "MAIN ATTRACTION" be used for?

  1.  Accepting information input from the user
  2.  Providing conclusion output to the user
  3.  Both input and output
  4.  Neither input nor output (e.g. intermediate conclusions)
  5.  Unknown
>>>> 1
        (input).

*** The frame "MAIN ATTRACTION" has been defined.



Enter a name for frame number 2
     (or just press "enter" if all frames have been given)

>>>> COSTS


Enter alternative number 1 for frame "COSTS"
     (or just press "enter" if all alternatives have been given)

>>>> cheap


Enter alternative number 2 for frame "COSTS"
     (or just press "enter" if all alternatives have been given)

>>>> medium


Enter alternative number 3 for frame "COSTS"
     (or just press "enter" if all alternatives have been given)

>>>> expensive
```

```
Enter alternative number 4 for frame "COSTS"
     (or just press "enter" if all alternatives have been given)

>>>>


What will the frame "COSTS" be used for?

  1.  Accepting information input from the user
  2.  Providing conclusion output to the user
  3.  Both input and output
  4.  Neither input nor output (e.g. intermediate conclusions)
  5.  Unknown
>>>> 1
        (input).

*** The frame "COSTS" has been defined.



Enter a name for frame number 3
     (or just press "enter" if all frames have been given)

>>>> LOCATION


Enter alternative number 1 for frame "LOCATION"
     (or just press "enter" if all alternatives have been given)

>>>> Paris


Enter alternative number 2 for frame "LOCATION"
     (or just press "enter" if all alternatives have been given)

>>>> New York
```

```
Enter alternative number 3 for frame "LOCATION"
     (or just press "enter" if all alternatives have been given)

>>>> Hawaii


Enter alternative number 4 for frame "LOCATION"
     (or just press "enter" if all alternatives have been given)

>>>> Australia


Enter alternative number 5 for frame "LOCATION"
     (or just press "enter" if all alternatives have been given)

>>>>


What will the frame "LOCATION" be used for?

  1.  Accepting information input from the user
  2.  Providing conclusion output to the user
  3.  Both input and output
  4.  Neither input nor output (e.g. intermediate conclusions)
  5.  Unknown
>>>> 2
        (output).

*** The frame "LOCATION" has been defined.



Enter a name for frame number 4
     (or just press "enter" if all frames have been given)

>>>>

*** All frames have been defined
```

```
**************************************************
SECTION NUMBER 3
**************************************************


The next step is to define the relationships between each of the
frames


Select a frame which should have a link to another frame,
     (or select alternative #1 if all links have been given)

  1.  *** Finished defining links ***
  2.  MAIN ATTRACTION
  3.  COSTS
  4.  LOCATION
>>>> 2
        (MAIN ATTRACTION).


Which frame should "MAIN ATTRACTION" be linked to?

  1.  COSTS
  2.  LOCATION
>>>> 2
        (LOCATION).


Which propositions from frame "LOCATION" are IMPLIED BY the
proposition "shopping and nightlife" from frame "MAIN ATTRACTION"?

  1.  Paris
  2.  New York
  3.  Hawaii
  4.  Australia
>>>> ( Uncertainty entered).                    (see figure 15.1)
```

```
Which propositions from frame "LOCATION" are IMPLIED BY the
proposition "sunshine and beaches" from frame "MAIN ATTRACTION"?

  1.  Paris
  2.  New York
  3.  Hawaii
  4.  Australia
>>>> ( Uncertainty entered).                          (see figure 15.2)


*** The link number 1 has been defined.



Select a frame which should have a link to another frame,
     (or select alternative #1 if all links have been given)

  1.  *** Finished defining links ***
  2.  MAIN ATTRACTION
  3.  COSTS
  4.  LOCATION
>>>> 3
        (COSTS).


Which frame should "COSTS" be linked to?

  1.  MAIN ATTRACTION
  2.  LOCATION
>>>> 2
        (LOCATION).
```

```
Which propositions from frame "LOCATION" are IMPLIED BY the
proposition "cheap" from frame "COSTS"?

  1. Paris
  2. New York
  3. Hawaii
  4. Australia
>>>> ( Uncertainty entered).                    (see figure 15.3)


Which propositions from frame "LOCATION" are IMPLIED BY the
proposition "medium" from frame "COSTS"?

  1. Paris
  2. New York
  3. Hawaii
  4. Australia
>>>> ( Uncertainty entered).                    (see figure 15.4)


Which propositions from frame "LOCATION" are IMPLIED BY the
proposition "expensive" from frame "COSTS"?

  1. Paris
  2. New York
  3. Hawaii
  4. Australia
>>>> ( Uncertainty entered).                    (see figure 15.5)


*** The link number 2 has been defined.
```

```
Select a frame which should have a link to another frame,
     (or select alternative #1 if all links have been given)

  1.  *** Finished defining links ***
  2.  MAIN ATTRACTION
  3.  COSTS
  4.  LOCATION
>>>> 1
        (*** Finished defining links ***).



**************************************************
SECTION NUMBER 4 : INITIAL BELIEF
**************************************************


If you have some belief about any of the frames, you can add that
belief into the system now, do you wish to do so?

  1.  yes
  2.  no
>>>> 1
        (desired).


Select a frame which should have a link to another frame,
     (or select alternative #1 if all links have been given)

  1.  *** Finished specifying initial belief ***
  2.  MAIN ATTRACTION
  3.  COSTS
  4.  LOCATION
>>>> 3
        (COSTS).
```

```
Which do believe to be the best alternative for frame "COSTS"?

  1.  cheap
  2.  medium
  3.  expensive
>>>> ( Uncertainty entered).                    (see figure 15.6)


*** Your belief about frame "COSTS" has been defined.

Select a frame which should have a link to another frame,
     (or select alternative #1 if all links have been given)

  1.  *** Finished specifying initial belief ***
  2.  MAIN ATTRACTION
  3.  COSTS
  4.  LOCATION
>>>> 1
        (*** Finished specifying initial belief ***).


*** All initial belief has been defined


**** MAIN SYSTEM MENU ****
What do you want to do?

  1.  Go through tutorial session
  2.  Create a new knowledge base
  3.  Load an existing knowledge base
  4.  View and/or Edit a knowledge base
  5.  Save a knowledge base
  6.  Finish
>>>> 5
```

```
************************************************
SECTION NUMBER 5 : WRITING INFORMATION INTO KB FILES
************************************************

Enter a file name to use for creating the design knowledge base.

>>>> SAMPL-11.DKB


Enter a file name to use for creating the control knowledge base.

>>>> SAMPL-11.CKB


Creating design knowledge base....
 adding frame:MAIN ATTRACTION
 adding frame:COSTS
 adding frame:LOCATION
 adding link:MAIN ATTRACTION to LOCATION
 adding link:COSTS to LOCATION

Creating control knowledge base....


**** MAIN SYSTEM MENU ****
What do you want to do?

  1.  Go through tutorial session
  2.  Create a new knowledge base
  3.  Load an existing knowledge base
  4.  View and/or Edit a knowledge base
  5.  Save a knowledge base
  6.  Finish
>>>> 6
        (finish).

*** Session Completed ***
```

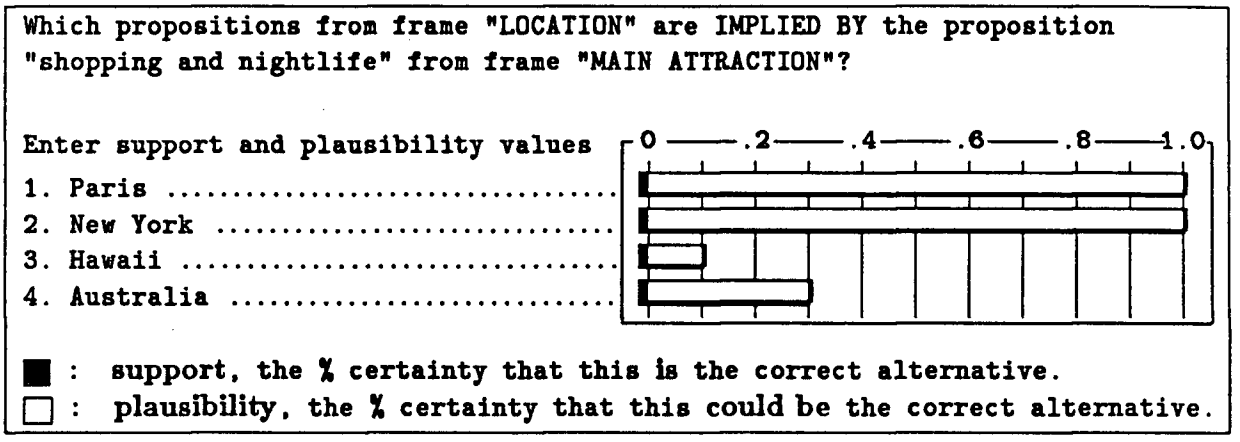


Figure 15.1: Entering the relationship between "shopping and nightlife" and the best vacation location

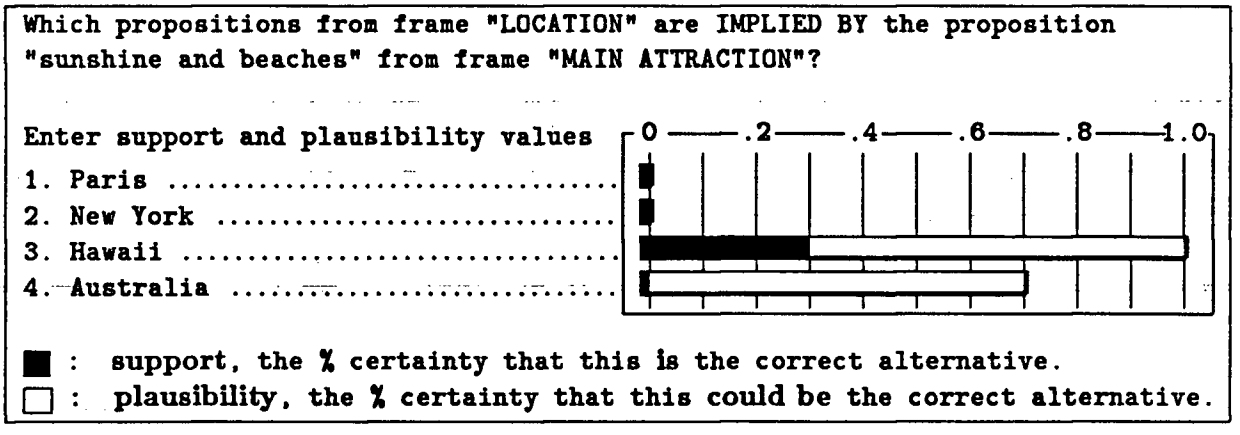


Figure 15.2: Entering the relationship between "sunshine and beaches" and the best vacation location

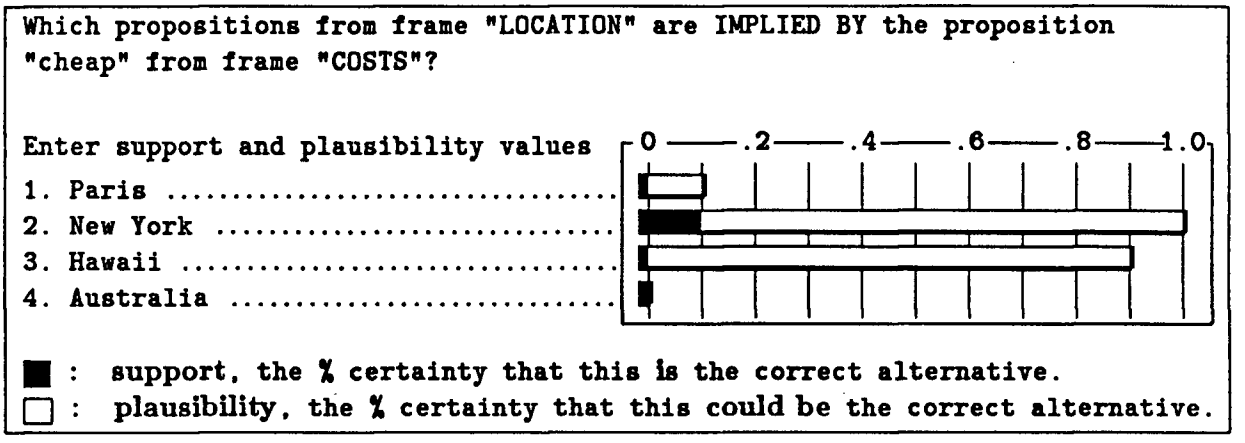


Figure 15.3: Entering the relationship between "cheap" cost and the best vacation location

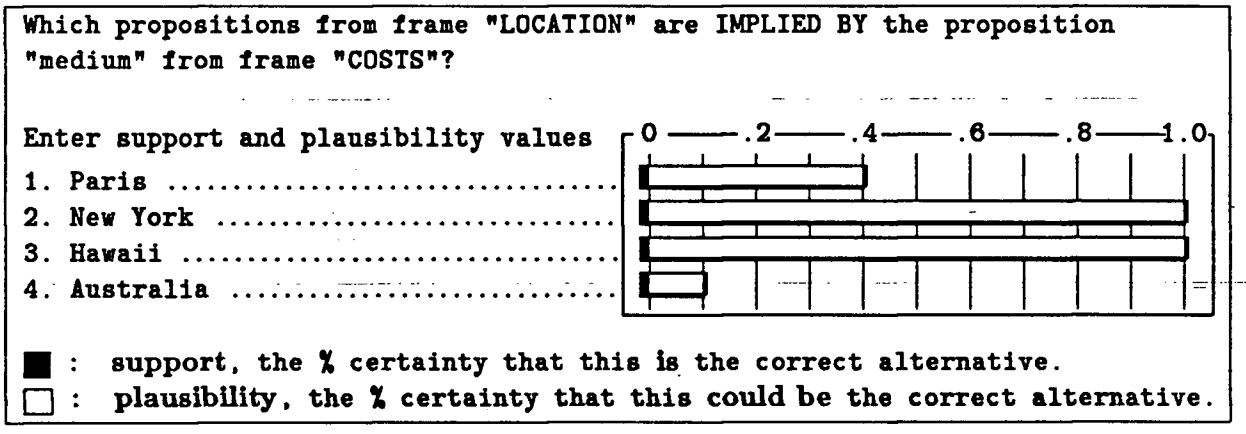


Figure 15.4: Entering the relationship between "medium" cost and the best vacation location

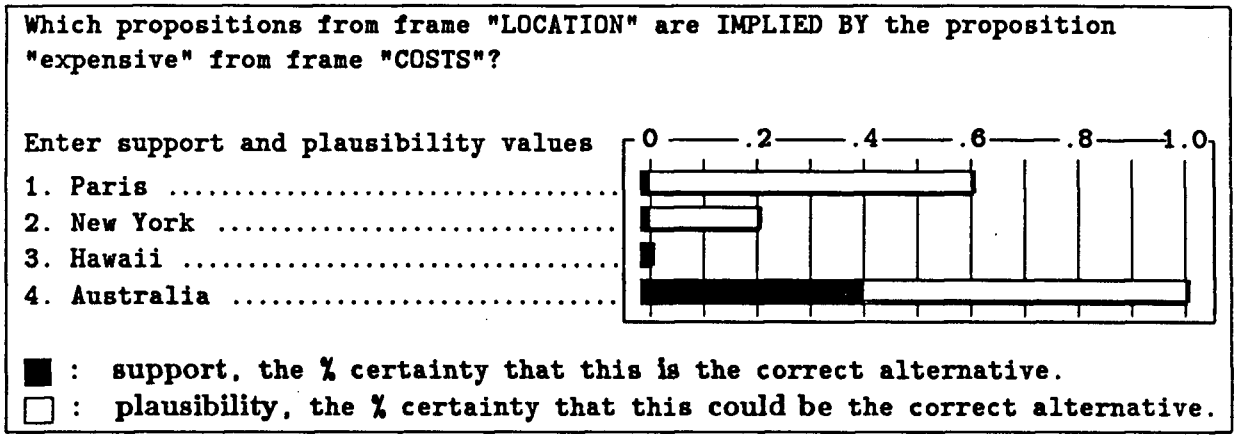


Figure 15.5: Entering the relationship between "expensive" cost and the best vacation location

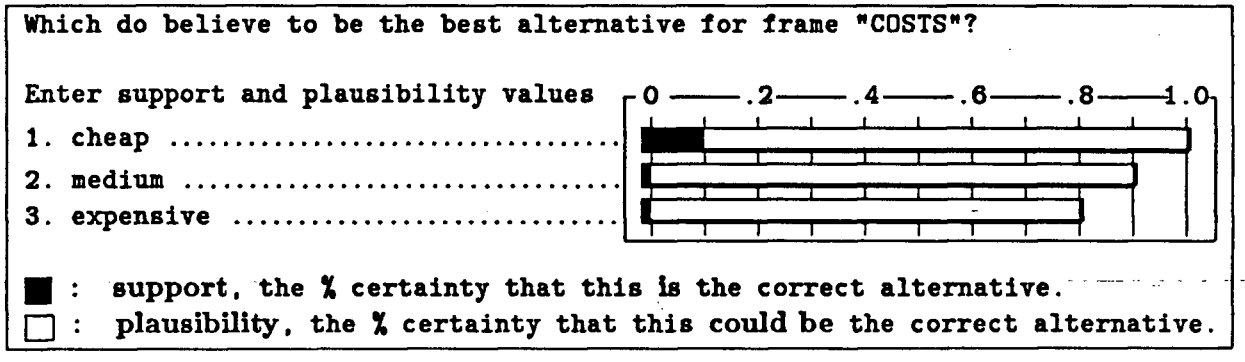


Figure 15.6: Entering the initial belief about desired vacation cost

MKB is designed to incorporate many more features than initial knowledge base creation, as can be seen from the options listed on the main menu. However several of these capabilities are not yet implemented in the system. While MKB will load existing knowledge bases and display their frames and links, it will not provide the tutorial session or edit existing terms. The user is free to explore the menu options of MKB as those options which are not yet implemented will display suitable messages.

Chapter 16: Design Problem Files

As stated in the previous section, the execution of the MKB program results in the creation of two problem files. Since these are standard ASCII files, they can also be created initially without the aid of MKB. Regardless of how they are first created, the problem files can be accessed and altered with any ASCII file editor. The control problem file is much like the problem files described in Part I of this manual and will be examined in subsequent chapters. This chapter deals with the design problem file which, while it too consists of a series of statements, is quite different in form from the control problem file.

The design problem file may contain comments using the same syntax as described for rule-based problem files in Part I. Apart from comments, the three types of statements which can be used are `add_frame`, `add_link`, and `add_bpa`. Each of these statement types is described below. Generally, the problem file should contain all of the `add_frame` statements first, followed by the `add_link` and then `add_bpa` statements. More precisely, an `add_link` or `add_bpa` statement cannot be given before any of the frames mentioned in the statement have been defined with an `add_frame` statement.

16.1 Adding Frames

The first type of statements to appear in a FRO design problem file are `add_frame` statements which define the frames which will be used in the system. These statements supply the name to be used and the names of each of the possible alternatives. The syntax for `add_frame` statements is as follows:

`add_frame(`*NAME*, [*ALT 1*, *ALT 2*, ...]).

Where *NAME* is the name of the frame and [*ALT 1*, *ALT 2*, ...] is the list of names for the possible alternatives. Both the frame name and the alternative names must be atoms or strings (both of which were defined in Part I). An example of an `add_frame` statement is as follows:

```
add_frame('LOCATION',
    ['Paris',
```

```
'New York',
'Hawaii',
'Australia']).
```

It can be seen that this is a less English-like syntax than those used in the rule-based problem files. Like rule-based statements, though, spaces and carriage returns are ignored by the system and the end of the statement is indicated by a period. The example above is an add_frame statement created by MKB and thus it shows MKB's style convention for adding spaces, carriage returns, etc. A similar style is recommended for legibility of the problem file though it is not required. For example, if the place names were not capitalized in the statement given above, then the following statement would produce equivalent results:

```
add_frame('LOCATION',[paris,'new york',hawaii,australia]).
```

16.2 Adding Links

The second type of statement in the design problem file is the add_link statement. These statements are used to define the uncertain inferential relationships between two frames. These are the most complex statements and are best created using the graphical interface of MKB, however they can also be created and manipulated directly. The format for an add_link statement is as follows:

add_link(*FRAME A, FRAME B,*
[(*ALT A1, BPA B1*), (*ALT A2, BPA B2*), ...]).

This statement shows how the alternatives of a first frame (*FRAME A*) imply belief about the alternatives of a second frame (*FRAME B*). *FRAME A* and *FRAME B*, then, are the names of these two frames. Next, for each of *FRAME A*'s alternatives, the statement gives the name of that alternative, *ALT A1*, and a corresponding basic probability assignment *BPA B1* stating what should be believed about the alternatives of *FRAME B* if it were known with certainty that alternative *ALT A1* where the correct answer for *FRAME A*. A basic probability assignment has the following syntax:

[([*ALT i1, ALT i2*, ...], *Mi*), ([*ALT j1, ALT j2*, ...], *Mj*), ...]

Where [*ALT i1, ALT i2*, ...] is a list of alternatives which make up a D-S proposition (i.e. the correct alternative is believed to be in this list) and *Mi* is a real number representing the basic probability number for that proposition. The sum of all basic probability numbers for the entire basic probability assignment must equal 1.0. An example of an add_link statement is as follows:

```
add_link('MAIN ATTRACTION', 'LOCATION',
 [('shopping and nightlife',
     [(['New York', 'Paris'], 0.7),
      (['Australia', 'New York', 'Paris'], 0.2),
      (['Australia', 'Hawaii', 'New York', 'Paris'], 0.1)]),
  ('sunshine and beaches',
     [(['Hawaii'], 0.3),
      (['Australia', 'Hawaii'], 0.7)])
 ]).
```

This statement is created by MKB and is, in fact, the statement generated by figures 15.1 and 15.2. Notice again that while spaces and carriage returns are ignored by the system, they are very important for making the statement legible. It should be noted that while `add_link` statements can be created with very little understanding of D-S theory using MKB, this cannot be said when the statements are accessed directly as shown here.

16.3 Adding Initial Belief

The final statement type is the add_bpa statement. These statements can be used for adding uncertain belief about the best alternative for some frame, should such belief be known at the time of the problem file development. The format for add_bpa statements is as follows:

add_bpa(*FRAME, BPA*).

Where *FRAME* is the name of the frame to which the belief is being added and *BPA* is a basic probability assignment on that frame defined using the same syntax as shown above. An example of an add_bpa statement which represents the uncertain belief displayed in figure 15.6 is as follows:

```
add_bpa('COSTS',
     [(['cheap'], 0.1),
      (['cheap', 'medium'], 0.1),
      (['cheap', 'expensive', 'medium'], 0.8)]).
```

Chapter 17: Control Problem Files

The previous chapter described the design problem file which defines all of the major uncertain variables in the system and the relationships between them. It has been stated earlier that when the user enters belief for some frame, the system automatically propagates the belief to every related frame. However the design knowledge base contains no information about which questions to ask, when to ask questions, and how to output or interpret the results. These functions are performed by the control problem file.

The control problem file for problems which contain uncertainty is largely the same as the rule-based problem files described in Part I of this manual. Control problem files contain goal statements, fact statements, rule statements, and questions statements in exactly the same format as outlined in Part I. However certain additional features may be used. First, an alternative form of question definition statement may be used for enabling the system to use IBIS screens to ask questions with uncertainty. Second, an additional series of command proposition terms may be used for adding uncertain belief to the frame network and for obtaining resultant belief back from it. These features are described in the following sections.

17.1 Question Statements for Uncertainty

In order for the FRO system to ask questions involving uncertainty, it must know the exact wording of the question to ask, the possible alternatives, and so on. This information is, in many ways, similar to the information required for asking categorical questions as defined in Part I. As such, an uncertain question definition statement must be defined which is similar to the categorical question definition statement defined in Part I. The syntax for an uncertain question definition statement is as follows:

```
question_un # ::
  FRAME
      ask_un QUESTION
      alt [ALT 1, ALT 2, ...]
```

ralt [*RALT 1, RALT 2, ...*]
expl *EXPLANATION.*

or, alternatively:

FRAME
ask_un *QUESTION*
alt [*ALT 1, ALT 2, ...*]
ralt [*RALT 1, RALT 2, ...*]
expl *EXPLANATION.*

In these statements, "*#*" is the statement number assigned to the statement by the user, "*FRAME*" is the name of the frame which is being asked about, "*QUESTION*" is the actual question which should be asked of the user, [*ALT 1, ALT 2, ...*] is the list of alternatives which the system offers the user as the possible answers to the question, [*RALT 1, RALT 2, ...*] is the list of return alternatives the system uses internally to represent the possible alternatives, and *EXPLANATION* is an explanation of the question which the user can access upon request. Both the list of return alternatives and the explanation are optional, but if they are used they must appear in the order shown here. If the list of return alternatives is used, then it must correspond exactly to the list of alternatives used to define the frame and the list of alternatives may contain any character strings. If the list of return alternatives is not used, then the list of alternatives must correspond exactly to the list used to define the frame. Note that both of these lists must be enclosed in square brackets, not parentheses.

Consider, for example, a question statement that is to be used to inquire about some frame `'LOCATION'` which offered the following list of alternatives: `[paris, 'new york', hawaii, australia]`. Either of the following statements, then, would adequately define an appropriate question:

```
'LOCATION' ask_un $Where would you prefer to go?$
     alt [paris, 'new york', hawaii, australia].

question_un 42 ::
  'LOCATION'
  ask_un $Where would you prefer to go?
     alt [$Paris, France$,
          $New York, USA$,
          $Hawaii, USA$,
          $Australia, (the grand Australian tour)$ ]
    ralt [paris, 'new york', hawaii, australia ]
    expl $The system would like to know where your preferred
          vacation destination is.  Your options are limited
```

```
to one and only one of the locations listed here.$.
```

17.2 Additional Command Propositions

17.2.1 Loading the Design Problem File

syntax: `load_un` *FILE_NAME*

A FRO-based system which includes uncertainty is executed by loading the rule-based control problem file in the same manner as described in Part I. One of the first tasks of the control problem file is to load the design problem file. For example, if the design problem file is called `SAMPLE.DKB`, the first few lines of the control problem file may be the following:

```
goal 1 :: session is completed.

rule 1 ::
  session is completed
     if load_un $SAMPLE.DKB$
    and .....
```

etc ...

17.2.2 Asking Questions

syntax: `query_un` *FRAME*

As shown above, question definition statements are included in the control problem file for uncertainty questions in a similar manner to that of categorical questions. However the circumstances under which these questions are asked is quite different in the two systems. While the system automatically asks categorical questions whenever they concern the current goal, the system must be instructed exactly when to ask uncertainty questions. This is done through the use of the `query_un` command. When the system encounters such a command in the premise of some rule, it will locate the uncertainty question corresponding to the *FRAME* name supplied in the `query_un` command and proceed to ask the question. The user's response is then automatically added to the frame and the belief is propagated around the frame network.

17.2.3 Printing Uncertain Belief

syntax: `print_un(`*MESSAGE, FRAME, PROP_LIST*`)`

The `print_un` command can be used to instruct FRO to report the uncertain belief for any frame. The belief is displayed in a pop-up IBIS window as described in chapter 14. *MESSAGE* is a character string message which is printed across the top of the pop-up window, *FRAME* is the name of the frame which is to be reported on, and *PROP_LIST* is a list of the propositions for which the belief is to be given. The syntax for the *PROP_LIST* is as follows:

[*ALT 1* , [*ALT 2*], ...]

where "[[*ALT 1*], [*ALT 2*], ...]" is a list of the alternatives which make up some proposition. For example, the `print_un` statement which was generated by the MKB session shown in chapter 15 is as follows:

```
and print_un ($The best alternative for "LOCATION" is:$,
              'LOCATION',
  [ ['Paris'],
    ['New York'],
    ['Hawaii'],
    ['Australia'] ])
```

While this is the most common usage of the `print_un` command, some variations also exist, as follows:

syntax: `print_un(`*MESSAGE, BELIEF_FN*`)`

This form of `print_un` can be used to display a belief function directly, rather than referring to a frame from which to retrieve a belief function for display. In this form, *MESSAGE* is a message to appear at the top of the pop-up window and *BELIEF_FN* is a belief function which has the following syntax:

([*ALT 1* , *S1, PL1*), ([*ALT 2*], *S2, PL2*), ...]

Where "[*ALT 1*]" is an alternative, *S1* is a real number representing the support value for that alternative, and *PL1* is a real number representing the plausibility value. Note that this is not the form of a true D-S belief function, but rather it is more akin to a "singleton support interval function". It should also be noted that this form of `print_un` would probably be used very rarely.

syntax: `print_un(`*MESSAGE, BPA, PROP_LIST*`)`

This form of `print_un` is similar to the previous form but it can be used when the information to be displayed is in the form of a basic probability assignment rather than a belief function. The syntax for both *BPA*'s and *PROP_LIST*'s has been given previously.

17.2.4 Retrieving Uncertain Belief

syntax: `get_bpa_un(`*QUESTION, PROP_LIST*`) is` *BPA*

This command allows the system to obtain uncertain input from the user without automatically adding that belief to a frame and causing it to be propagated around the frame network. The *QUESTION* statement provides the text of the question to be asked of the user and the *PROP_LIST* (the syntax of which is given above) provides the alternatives to offer. The information entered by the user is returned in the value of the proposition is the form of a basic probability assignment, *BPA* (note that this is the only command proposition which has a value other than an assumed "`is true`").

17.2.5 Storing Uncertain Belief

syntax: `store_un(`*FRAME, BPA*`)`

This command causes a basic probability assignment to be added to a frame and propagated around the frame network. *FRAME* is the name of the frame to which the belief is to be added and *BPA* is the basic probability assignment being added (the syntax for basic probability assignments is given above).

17.2.6 Retrieving the Best Alternative

syntax: `best_un(`*FRAME, PROP_LIST, BEST_PROP, QUALITY*`)`

This command accesses the frame network to obtain the alternative which has the most belief for being the correct answer, as well as a measure of how much confidence can be placed on the assessment. The *FRAME* is the name of the frame for which the best alternative is desired and the *PROP_LIST* is the list of alternatives for that frame (syntax as above). The *BEST_PROP* and the *QUALITY* are supplied as variables in the control problem file and when the term is solved as the current goal they will have the correct values assigned to them (*BEST_PROP* is returned as the name of an alternative enclosed in square brackets and *QUALITY* is returned as a real number between 0 and 1). The rule in following example obtains the best value for some frame and then prints that information on the screen:

```
rule 42 ::
  the_best_alternative is found
     if best_un('LOCATION',
                [ ['Paris'],
```

```
                ['New York'],
                ['Hawaii'],
                ['Australia'] ], [BEST], QUALITY)
    and print [$The best location for your vacation would be $,
               BEST, $ and my degree of confidence in this
               decision is $, QUALITY].
```

Chapter 18: Running FRO

A FRO program which includes frame-based inexact reasoning is operated in an identical manner to rule-only FRO programs (described in Part I, chapter 11 of this documentation). The only difference is the use of IBIS input screens, which has been described in chapter 14. Note that because of these screens, a mouse is required for operating frame-based FRO whereas it is an option with rule-based FRO. Note also that although the control rules used in a frame-based FRO application can be traced, "why" queried, and so on, there is no real tracing of the D-S belief propagation at FRO's present stage of development.

Chapter 19: Sample Files

19.1 Sample file 1: Design Knowledge Base

The following is a simple design knowledge base created by the MKB session listed in chapter 15. The domain is selecting an appropriate vacation destination.

File: "SAMPL-11.DKB"

```
%%%%%%%%%%%% SAMPL-11.DKB %%%%%%%%%%%%
%%% Chosing a vacation destination

%%%%%%%%%%%% frames %%%%%%%%%%%%
add_frame('MAIN ATTRACTION',
    ['shopping and nightlife',
     'sunshine and beaches']).
add_frame('COSTS',
    ['cheap',
     'medium',
     'expensive']).
add_frame('LOCATION',
    ['Paris',
     'New York',
     'Hawaii',
     'Australia']).


%%%%%%%%%%%% links %%%%%%%%%%%%
add_link('MAIN ATTRACTION', 'LOCATION',
 [('shopping and nightlife',
     [(['New York', 'Paris'], 0.7),
      (['Australia', 'New York', 'Paris'], 0.2),
      (['Australia', 'Hawaii', 'New York', 'Paris'], 0.1)]),
```

```
  ('sunshine and beaches',
     [(['Hawaii'], 0.3),
      (['Australia', 'Hawaii'], 0.7)])
 ]).

add_link('COSTS', 'LOCATION',
 [('cheap',
     [(['New York'], 0.1),
      (['Hawaii', 'New York'], 0.8),
      (['Hawaii', 'New York', 'Paris'], 0.1)]),
  ('medium',
     [(['Hawaii', 'New York'], 0.6),
      (['Hawaii', 'New York', 'Paris'], 0.3),
      (['Australia', 'Hawaii', 'New York', 'Paris'], 0.1)]),
  ('expensive',
     [(['Australia'], 0.4),
      (['Australia', 'Paris'], 0.4),
      (['Australia', 'New York', 'Paris'], 0.2)])
 ]).


%%%%%%%%%%%% initial BPA's %%%%%%%%%%%%
add_bpa('COSTS',
     [(['cheap'], 0.1),
      (['cheap', 'medium'], 0.1),
      (['cheap', 'expensive', 'medium'], 0.8)]).
```

19.2 Sample file 2: Control Knowledge Base

The following is the control knowledge base created by the MKB session listed in chapter 15 which accompanies the design knowledge base listed in the previous section.

File: "SAMPL-11.CKB"

```
%%%%%%%%%%%% SAMPLE-11.CKB %%%%%%%%%%%%
%%% Chosing a vacation destination

%%%%%%%%%%%% goal %%%%%%%%%%%%
goal  session is complete.

%%%%%%%%%%%% rules %%%%%%%%%%%%
rule  session is complete
     if load_un $SAMPL-11.DKB$
    and query_un 'MAIN ATTRACTION'
    and query_un 'COSTS'
    and print_un ($The best alternative for "LOCATION" is:$,
                  'LOCATION', [
       ['Paris'],
       ['New York'],
       ['Hawaii'],
       ['Australia'] ]).

%%%%%%%%%%%% questions %%%%%%%%%%%%
'MAIN ATTRACTION' ask_un $What is the best alternative for "MAIN
ATTRACTION"?$
    alt  [
         'shopping and nightlife',
         'sunshine and beaches'].

'COSTS' ask_un $What is the best alternative for "COSTS"?$
    alt  [
         'cheap',
         'medium',
         'expensive'].
```

19.3 Sample file 3: Revised Control Knowledge Base

The following is a revision of the previously listed control knowledge base which is customize for the particular application. The wording of the questions has been changed and the uncertain D-S results are interpreted by the program.

File: "SAMPL-12.CKB"

```
%%%%%%%%%%%% SAMPLE-11.CKB %%%%%%%%%%%%
%%% Chosing a vacation destination

%%%%%%%%%%%% goal %%%%%%%%%%%%
goal 1 :: location is suggested.

%%%%%%%%%%%% rules %%%%%%%%%%%%
rule 1 ::
   location is suggested
     if load_un $SAMPL-11.DKB$
    and query_un 'MAIN ATTRACTION'
    and query_un 'COSTS'
    and best_un ('LOCATION',
                 [ ['Paris'],
                   ['New York'],
                   ['Hawaii'],
                   ['Australia'] ], [BEST],QUALITY )
    and result(BEST,QUALITY) is analyzed.

rule 2 ::
 result(LOCATION,QUALITY) is analyzed
     if QUALITY > 0.7
    and print [$I can strongly recommend that you go to $,LOCATION].

rule 3 ::
  result(LOCATION,QUALITY) is analyzed
     if QUALITY > 0.4
    and print [$My  best guess is that you try $,LOCATION].

rule 4 ::
  result(LOCATION,_) is analyzed
     if print [$It seems as though $,LOCATION,
               $ would suit your needs, but I suggest that
```

```
                you try to get some more information first
                because I'm not very sure$].

%%%%%%%%%%%% questions %%%%%%%%%%%%
'MAIN ATTRACTION' ask_un $What is the main attraction that you
                          are interested in?$
    alt  [ 'shopping and nightlife',
           'sunshine and beaches'].

'COSTS' ask_un $What price range are we talking about?$
    alt  [ 'cheap',
           'medium',
           'expensive'].
```