PRS: A SYSTEM FOR PLAUSIBLE REASONING

BY

JAMES DONALD KELLY, JR.

B.S., Tufts University, 1982

89-16

08

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1989

Urbana, Illinois

BBN Systems and Technologies

ACKNOWLEDGEMENTS

I wish to thank my thesis advisor, Professor Ryszard Michalski, formerly of the Department of Computer Science at the University of Illinois at Urbana-Champaign, now at George Mason University in Fairfax. He encouraged me to undertake this project and supported it throughout. I would also like to Professor Thomas Fermanian of the Department of Horticulture at the University of Illinois. He spent a substantial amount of time working with me on the grass identification application in this work.

I would especially like to thank my family. Their constant support made this possible.

This thesis work was done at the Artificial Intelligence Laboratory at the University of Illinois and continued while the author was at Bolt, Beranek, and Newman, Inc. in Cambridge, MA.

This thesis was supported in part by the Office of Naval Research under grants N00014-82-K-0186, 14-87-K-0874, and N000-14-88-K-0226, in part by the National Science Foundation under grant NSF DCR 84-06801, and in part by grants administered by the Office of Naval Research, N00014-K-85-0878 and N000-14-88-K-0397. This work was also supported by the Illinois Turfgrass Foundation. The use of computer facilities at Bolt, Beranek, and Newman is gratefully acknowledged.

TABLE OF CONTENTS

1. INTRODUCTION

How people reason is one of the fundamental questions of human behavior. Just how is it that judgements are made, both the seemingly unconscious "everyday" decisions as well as the more complex "thought out" ones? Research in this area is necessarily widespread and diverse, as one must not only be concerned with particular inference processes but also other factors such as memory organization and learning. This thesis examines one theory about how human plausible reasoning occurs, that described by [Collins & Michalski 88].

Imagine you're in your car, following directions to a friend's house. The directions are somewhat sketchy but so far you think you're on the right track. The next instruction is to take a right onto High Street, right after passing the library. Well, although you didn't see a sign for High you pass a building that looks like it might be a library so you take a right. Since you didn't see the sign you have a slight feeling of uneasiness about whether you're still driving correctly, but are not too concerned. The next landmark is an Exxon station, at which point a left should be made onto Elm Street. From a distance you see a Sunoco station but cannot see any street sign. As you slowly approach you search for the street sign but still do not find it. The point arrives where a decision to continue straight or turn must be made - you decide to drive on. Going through the intersection you catch a glimpse of a street sign - it looks like "E" something but you just can't make it out fast enough. As you continue your uncertainty as to whether you're still on the right path increases. Was that Elm Street? Perhaps the directions had been meant to refer to the Sunoco station. On the other hand is this even the correct street in the first place? What made you think the building was a library, anyway? Should you turn around?

Continuing on, you see an Exxon sign and breath a sigh of relief. As you approach you signal for a left turn and look for the street sign. This time it's in plain sight but to your amazement it's not Elm Street but Oak. Now what? Did she just mix up Oak and Elm? Or perhaps Exxon and Sunoco. Again, what makes you think that the road you're on is even correct? What if you're not even close...

Besides being not that unfamiliar to most people this little scenario characterizes a number of issues which are addressed in this thesis. The use of multiple sources of evidence and inference methods, the expression of knowledge in uncertain terms, the variance of the certainty of a conclusion as evidence is gathered, the trade-off between positive and negative evidence, and the effect of time upon the reasoning process are the main issues explored within.

Many situations arise in which people reason from incomplete and ambiguous information. In addition, much of the reasoning process occurs at different levels of abstraction. In such situations the evidence for alternative conclusions is considered and one interpretation is selected as being the most probable. This in itself is a complex task but, in addition, many such reasoning situations are further constrained by the time available to consider evidence. This report describes PRS (Plausible Reasoning System) which is an implemented computer program that models the reasoning processes characterized above.

To be a little more specific the development of PRS was focused in several general areas: 1) Modeling a number of inference forms which previous researchers have found to be prevalent in human reasoning [Collins & Michalski 88]; 2) Examining the trade-offs in any reasoning process between certainty, cost, and specificity [Michalski & Winston 86]; and 3) Representing the uncertainty inherent in the real world.

Throughout this project several top-level goals directed the work: 1) To develop a working computer program which can be used to model, test, and refine theories; 2) To develop an implementation which is general and not tied to a specific domain; 3) To present a robust example which illustrates the utility of the program in a "real world" reasoning situation; and 4) to consider both formal theories and atheoretical yet intuitive methodologies.

1.1. Automated Reasoning

The term automated reasoning refers to the mechanization of reasoning tasks. To some readers automated reasoning may specifically connote a binary (true/false) reasoning process but here it is meant to encompass the broad range of ideas connected with automating reasoning processes. It is a complex area of study and accordingly has attracted a great deal of research. This research is important in three significant ways: One is in formulating and testing theories about how humans reason; two, for building useful systems which can perform human-like reasoning tasks; and three, for building systems which perform infallibly accurate reasoning. Following, some of these alternative ideas and theories will be briefly described.

A number of formalisms have been used in describing how reasoning operates. Many theories of automated reasoning are based upon the principles of classical logic. Classical logic has been shown to be useful in the performance of many tasks. One domain where it has seen some success is in having computers do theorem proving. In general, this works by giving the system a set of initial facts, represented in predicate calculus, and having the system perform deductive inference.

Researchers have found, however, that in many instances classical logic does not

The focus of this thesis is to examine a theory of human plausible reasoning, that of Collins and Michalski. The Collins-Michalski Theory compares and contrasts with the aforementioned ideas. It is a descriptive rather than prescriptive theory - it addresses how human reasoning works rather than how it might ideally work. The Theory is concerned with modeling human reasoning and generating conclusions corresponding to those a human would rather than following methodologies which might produce results considered to be more correct.

A major contribution of the Collins-Michalski Theory is the combination of numeric uncertainty information with semantic and structural information. The Theory identifies parameters which are used to quantitatively model human plausible reasoning.

Additionally the Theory formalizes a number of different types of inferences used by humans to gather evidence when reasoning. In any individual reasoning situation multiple inferences using some number of the identified types are performed which may result in multiple evidence being found for the possible conclusions. This evidence is combined, allowing a final conclusion to be reached with some calculated degree of belief.

In parallel with this thesis several other attempts have been made at implementing the Theory: [Baker, Burstein, & Collins 87] and [Dontas 88]. All of the implementations are similar in that they follow the reasoning style described by Collins and Michalski, implementing the inference types identified in the Theory. As would be expected differences are lower level implementation details, such as the mechanism for propagating uncertainty.

This thesis will explore the Collins-Michalski Theory in more detail, addressing issues raised in this introduction. As mentioned above, the focus of this report is on a computer implementation of the theory, PRS, which can be used to hypothesize and test various aspects of human plausible reasoning.

accurately model human reasoning. Thus other formalisms have been developed in the hope of more accurately mirroring humans.

One of the foundations of many reasoning systems is that facts remain constant over time - once a fact is known to be true it cannot become false. Non-monotonic logic, on the other hand, does not require that information remain in an unchanging state and be unequivocally known to be true or false prior to drawing conclusions. Conclusions can be drawn based upon default values or assumptions. Non-monotonic reasoning allows conclusions to be modified if additional information is obtained regarding the steps which support a conclusion. For example, when an assumption which was used in inferring some other fact turns out to be erroneous the fact must either be supportable by other means or labeled as false. [McDermott & Doyle 80] provides a good description of non-monotonic reasoning.

Other alternatives to classical logic which receive much attention are formalisms designed to address the notion that humans reason without knowing information as being either true or false but rather having some degree of belief attached to it. Uncertain reasoning is this idea that human reasoning is based upon degrees of belief rather than strictly true/false determinations. For example, instead of having to decide whether some answer is right or not some measure of the confidence in the answer being right is known. A great deal of work has gone into both the representational issues concerned with uncertain reasoning and the quantitative and qualitative aspects of it. Specific formalisms which have been developed include fuzzy logic [Zadeh 65], variable precision logic [Michalski & Winston 86], and Cohen's endorsement-based reasoning [Cohen 85].

A major issue in automated reasoning is in the control of the reasoning process. The two control schemes which have been the heart of many reasoning systems are forward and backward chaining. Forward chaining is the process of beginning with an initial knowledge base, reasoning with it, and formulating conclusions. Backward chaining begins with a fact, or conclusion, and uses the knowledge base in attempting to prove it. Systems have been developed which combine both forward and backward reasoning, which corresponds well with human reasoning.

In addition to the strictly theoretical underpinnings, automated reasoning has been at the forefront of more practical uses of artificial intelligence. Expert systems, the performing of human-like tasks by a computer, have been used for the testing of the theoretical issues as well as for tools in providing a useful service. Such systems are generally more concerned with accurate results rather than modeling human reasoning. Perhaps the best known system, XCON, or R1 as it was originally called, [McDermott 82], was developed by the Digital Equipment Corporation to configure VAX systems.

2. PLAUSIBLE REASONING

A good beginning point is to discuss what the term plausible reasoning actually means. The word plausible is used in a number of different contexts, however in general a user is attempting to evoke a feeling similar to the sense of the words believable, probable, reasonable, or presumable. It is used in situations where there is a general feeling of believability rather than uncompromising evidence. Reasoning can be thought of as some sort of thinking process which explains, justifies, or supports a conclusion. It does not necessarily imply any notion of accuracy or correctness. Plausible reasoning thus is the act of reaching conclusions which are supported by some evidence, although that evidence is not, in many cases, confirmatory. Plausible reasoning describes reasoning processes in which there is no clear cut, irrefutable evidence supporting a particular conclusion. It is perhaps best described as reasoning based upon supporting but not necessarily confirming evidence.

Plausible reasoning implies that there is some degree of uncertainty associated with a given conclusion and that there are alternative conclusions (although they may not be known). In this way it differs from deterministic reasoning procedures in which conclusions can be unequivocally reached given a set of initial facts. One problem with such processes is that rarely are all of the facts known about a situation enough to make an unequivocal determination. Think about some of the everyday judgements that you routinely make. Although in some cases you can be very certain of a conclusion there is always a chance that, perhaps due to some small quirk, the conclusion is not, in fact, valid.

Plausible reasoning is manifest throughout human behavior. People do it in both simple situations (e.g., "That must be John at the door since he said he would be over around now".) and more complex ones (e.g., "I bought stock in the Acme Company because their P/E ratio is good, the market for their products is strong,...".)

Much research has been done in the area of human plausible reasoning (e.g., [Collins 78], [Bobrow & Collins 75]). This thesis incorporates many ideas presented in previous work, primarily drawing from [Collins & Michalski 88]. The purpose of the following section is to provide the reader with a brief overview of the Collins-Michalski Theory of Plausible Reasoning - details will be presented in subsequent chapters.

2.1. Collins-Michalski Theory of Plausible Reasoning

Collins and Michalski define a formal theory describing how humans do plausible reasoning. Their Theory builds upon the idea that humans use many diverse methods when reasoning. Previous research has identified and examined a number of these methods, or inference procedures. For example, the human ability to make judgements based upon generalizing the current situation to the more abstract case is clear. Another type of inference which people use is a *lack of knowledge inference* [Gentner & Collins 81]. This refers to situations where unless something is explicitly known to be false it is assumed to be true (or vice-versa).

In addition to identifying certain inference patterns the theory proposes that human reasoning about a particular situation uses one or more of the identified inference types in arriving at a conclusion. A final judgement is made based upon the combination of the evidence obtained from using any inference.

Much of the theory evolved through the analysis of questioner-respondent interactions. The answers to a large number of questions were examined for patterns as to how humans reason. An example of one such dialogue is shown in figure 1 (taken directly from [Collins & Michalski 88]). This is an example of a *specialization transform*: the respondent knew that the Andes are in most South American countries (7 out of 9 of the Spanish speaking countries). Since Uruguay is a fairly typical South American country, he guesses that the Andes may be there too. The respondent is wrong, but the conclusion was quite plausible.

Q. Is Uruguay in the Andes Mountains?

R. I get mixed up on a lot of South American countries (pause). I'm not sure. I forget where Uruguay is in South America. It's a good guess to say that it's in the Andes Mountains because a lot of the countries are.

Figure 1: Plausible Reasoning Dialogue.

The Theory addresses both the abstract cognitive issues of reasoning as well as more implementation oriented ideas. It might be broken down into four interrelated areas: representation, inference types, inference control, and uncertainty.

2.1.1. Representation

Collins and Michalski propose that information is represented in *dynamic hierarchies* which are interconnected by *traces*. Essentially the theory describes a network of objects with relations between them indicated by links. The network changes to reflect new information. The change takes place in the links, new ones are created and the strength of existing ones changes. The idea of dynamic hierarchies captures a number of intuitions about how humans store and process information. It allows for objects to be related in any number of ways and for the information known about them to be constantly changing. It proposes that information is highly structured, related both hierarchically and laterally. Figure 2 illustrates the representation of a fact as a trace between two hierarchies.

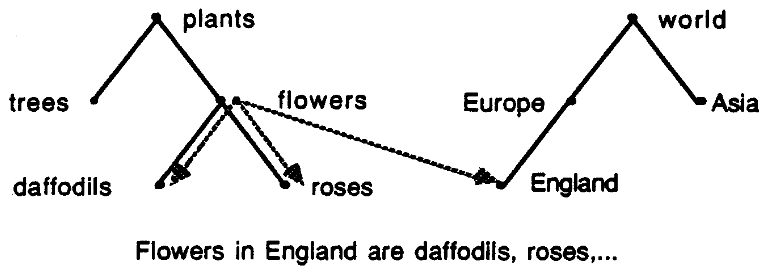


Figure 2: Trace.

2.1.2. Inference Types

A major contribution of this Theory and the work which it builds upon is the identification of the different inference types which people use. Inference types detailed in the Theory are:[1]

Mutual Implication: Implications, which are also referred to as rules in much of the literature, are statements meaning that some antecedent implies some consequent. Mutual implication is

[1]These are discussed in detail in Chapter 4.

an extension to this which additionally considers to what degree knowing just the consequent implies the antecedent. Thus in the Theory implications are bidirectional, with parameters used to measure both the forward and backward nature of the rule.

Mutual Dependency: Dependencies are relationships between attributes. They model a human's knowledge that certain attributes are related yet the exact relationship is unknown. For example, an implication might state that a place with a temperate climate will have roses while a similar dependency would state that the types of flowers in a place are related to the climate. Dependencies relate attributes while implications relate attribute-value pairs. Like mutual implications mutual dependencies are bidirectional.

Generalization: This is the inference process whereby knowing something about some object and also knowing that the object is member of a general class of things allows an inference to be made that the same thing holds for the class as a whole.

Specialization: Specialization is the inverse of generalization. It is the inference process whereby knowing something about some general class of things and also knowing that some object is a member of that class allows an inference to be made that the same thing holds for the specific object.

Similarity/Analogy: This describes the class of inferences whereby knowing that objects have a number of corresponding features other features which only one of them may be known to have can be inferred to also exist on the other objects.

Again, these inferences comprise the major types upon which the Theory is presented. However, it is not restricted to these. Recall from above one of the Theory's primary tenets is the idea that all sorts of diverse inferences types are combined when people reason.

2.1.3. Inference Control

Inference control issues are also addressed in the Theory. Control alludes to several hypotheses about the actual process of human reasoning. It is, of course, necessarily linked to the issues of representation, inference types (discussed above), and uncertainty (discussed below).

The main contentions here are that people often use multiple sources of evidence as well as multiple inference types in reasoning. They are more or less certain of things depending upon the evidence which they can bring to bear on the subject. A certain conclusion may initially be weakly believed but as the certainty of the evidence supporting it increases it becomes more accepted as valid. A final judgement is made considering all sources and strengths of evidence and combining them. Finally, the Theory contends that all evidence, both for and against a certain conclusion, is considered. This involves the identification of alternatives and weighing them against each other.

2.1.4. Uncertainty

The Collins and Michalski Theory goes beyond just the qualitative aspects of reasoning and proposes that numeric variables can be used to more fully model the actual human reasoning process. They identify nine parameters, or weights, which are used to more fully specify the information used in reasoning. The parameters are manipulated by the inference processes in making judgements. The nine parameters used in the theory are *forward and backward conditional likelihood, frequency, dominance, typicality, similarity, multiplicity,* both of the argument and referent, and *certainty,* or support. They will be discussed in detail in the following chapter.

The weights are used to characterize the degree of the relationship between concepts or objects. They are attached to data with the intent of modeling human uncertainty about their knowledge. As a quick example recall the questioner-respondent dialogue in figure 1. Two of the certainty parameters associated with specialization transforms are evident: *frequency* (he knows that the Andes are in *most* countries), and *typicality* (Uruguay is a *typical* South American country). As will be described in Chapter 3, support weights are associated with information as a higher level measure of the belief in both a fact and its' associated frequency, dominance, typicality, and/or likelihood weight.

With this introduction PRS can now be presented. The following chapters will build upon the presentation of the theory in this chapter with more detailed descriptions of the representation, inference types, control, and uncertainty as they relate to the implemented system.

3. REPRESENTATION

3.1. Descriptors, Arguments, Referents

All information in the system is represented as combinations of **descriptors, arguments,** and **referents.** Descriptors can be thought of as predicates or attributes, arguments as objects, and referents as attribute values. For example *color* can be used as a descriptor, *car* as an argument, and *blue* as a referent. Any object or concept in the universe can be any of the three, depending upon how it is used. For example, in different situations the word *car* could be used as a descriptor or an argument: "the car of John ...", "the color of the car ...". Built upon these primitives are three general categories of information: **facts, implications,** and **dependencies.**

3.1.1. Facts, Implications, Dependencies

Facts are represented in the form *(DES (ARG) = (REF)),* where *DES* is the descriptor (i.e. predicate or attribute), *ARG* is the argument (i.e. object), and *REF* is the referent (i.e. attribute value). An example of a fact is *(climate (England) = (temperate))*[2] which is interpreted as meaning that the climate in England is temperate. Any descriptor can have an inverse associated with it, which essentially just reverses the argument and referent. Referring to the example above, a descriptor *climate-value* could be defined to be the inverse of *climate.* Thus the fact *(climate-value (temperate) = (England))* is the inverse of *(climate (England) = (temperate)).* This is interpreted as meaning that a temperate climate exists in England. Both a fact and an inverse may convey essentially the same meaning. However, as will be described in following sections, the uncertainty weights attached to both a fact and its inverse, which are not identical, define the exact relationship between a descriptor, argument, and referent. To further familiarize the reader with how facts are interpreted figure 3 lists some facts and their corresponding meaning in English.[3]

Facts may also be written with a conjunction or disjunction of referents. In the above example it might be the case that the climate of England ranges from temperate to harsh. Hence

[2]Note: In order to make the connection between this thesis and previous work many of the examples used in this thesis are from [Collins 78] and [Collins & Michalski 88]

[3]As another example, referring back to chapter 2, the trace in figure 2 would be represented as (flowers (England) = (daffodils) (roses)).

(has-part (car) = (engine))	A car has the part engine.
(part-of (engine) = (car))	An engine is a part of a car.
(type-of (car) = (sedan))	The type of the car is a sedan.
(terrain (Illinois) = (flat))	The terrain of Illinois is flat.
(color (baseball) = (white))	The color of a baseball is white.

Figure 3: English interpretation of PRS Facts.

the above fact could be written *(climate (England) = (temperate) (harsh)).* Similarly since England is certainly not the only country with a temperate climate the inverse *(climate (England) = (temperate))* might be more appropriately written *(climate-value (temperate) = (England) (Holland) ...)).*

PRS uses four descriptors to represent hierarchical relationships. They are *type-of, has-type, part-of,* and *has-part.* Type-of and has-type are inverses of each other, as are part-of and has-part. As described above, an inverse means that the referent and argument are transposed when a fact is written with the inverse of the current descriptor (e.g., *(part-of (England) = (Europe))* and *(has-part (Europe) = (England))* are inverses of each other). Several examples of these relationships were shown in figure 3.

Implications are represented as *(ANT → CON)* where *ANT* is a list of antecedent facts and *CON* is a single consequent fact.[4] **Dependencies** relate two descriptors, indicating that there is some relationship between the two. Figure 4 lists some implications and dependencies and their interpretation. For example, the dependency *(climate ← → latitude)* indicates that there is some link between the climate of a place and its latitude.

[4]Note the infix notation of the implication. In the actual implementation and in some examples in this report prefix notation is used, for both implications and dependencies (e.g., *(→ ANT CON)*).

(→ ((#-of-wheels (?x) = (4)) (has-part (?x) = (engine))) (object-id (?x) = (car)))	If ?x has 4 wheels and has an engine then it is a car.
(← → climate latitude)	There is some relationship between the value of something's climate and the value of its latitude.

Figure 4: English interpretation of PRS Implications and Dependencies.

Figure 5 summarizes the elements of the representation system.[5]

3.2. Uncertainty

In order to model the uncertainty in the world weights, or certainty parameters, are attached to all information. In PRS all of these weights are in the range of 0 to 1. 0 indicates zero certainty and 1 indicates absolute certainty. In addition, each weight is represented as a range - i.e. with lower and upper bounds - rather than as a single number. This allows the precision of the weight to be made explicit: for example, 0.5 ± 0.2 is quite different from 0.5 ± 0.02. Thus 0.4 to 0.6 indicates that although the actual value is not known, however lies somewhere between 0.4 and 0.6. Two characteristics of any weight, or interval, are important: the magnitude - how high are the bounds, and the precision - the distance between the lower and upper bound. It should be noted that the addition of certainty ranges rather than single numbers is an extension of the Collins-Michalski Theory. This was done to more closely match some aspects of human knowledge that could not be done with a single value. As each weight can be represented as a single value by having equal upper and lower bounds (e.g., 0.5 to 0.5) this representation is a generalization of that described in [Collins-Michalski 88].

In general what the weights represent is the relationship between objects, or concepts, in the universe. More precisely each concept can be thought of as a finite set in the total universe.

[5]Following standard notation symbols or words followed by a "?" are variables.

NAME	FORM	EXAMPLE
DESCRIPTORS	DES_1, DES_2, DES_3	climate, language, latitude
ARGUMENTS	ARG_1, ARG_2	England, Europe
REFERENTS	REF_1, (REF_1, REF_2), {REF_3,...}	temperate, temperate and mild, tropical, ...
VARIABLES	BOOK?	a book
TERMS	$DES_1(ARG_1)$, $DES_2(ARG_2)$	climate(England), language(Europe)
STATEMENTS	$(DES_1(ARG_1) = (REF_1))$	The climate of England is temperate.
IMPLICATIONS	(BUY-LOC(BOOK?) = (England)) $\rightarrow$ (LANG(BOOK?) = (English))	If the book was purchased in England then it's written in English.
DEPENDENCIES	$DES_1(ARG_1) \leftarrow\rightarrow DES_3(ARG_1)$	The climate of England is related to the latitude of England.

Figure 5: Elements of PRS Expressions.

Sets may overlap indicating that they are related, which in PRS means in the context of a certain descriptor. Figure 6 illustrates this.

The weights are used to characterize the degree of intersection between concepts. Weights are represented by the following terminology: A weight is indicated by two numbers enclosed by square brackets, e.g., [0.7 0.9]. The general template is **[s p]** where **s** is the **support** and **p** is the **plausibility**.[6] By definition **s** and **p** are both equal to or greater than 0 and equal to or less than 1. Also **s** is less than or equal to **p**. When **s** = **p** it indicates that there is no imprecision about the weight.

A weight indicates belief in both a fact and its negation. The lower bound specifies both the lower bound for a fact and the upper bound for the negation of that fact. The upper bound specifies both the upper bound for a fact and the lower bound for its negation. For example, consider a weight [0.7 0.9] supporting a fact. This is equivalent to a weight of [0.1 0.3] supporting the negation of the fact.

[6]The terms support and plausibility are used to remain consistent with uncertainty literature (e.g., [Shafer 76]). The terms have no meaning other than lower and upper bounds.

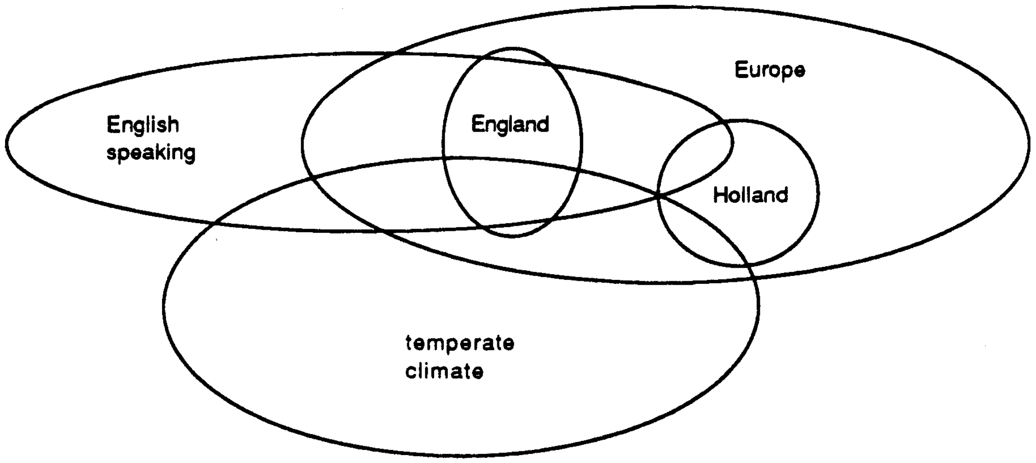


Figure 6: Concepts as Intersecting Sets.

Six different types of weights can be attached to datum: **frequency, dominance, typicality, likelihood, similarity,** and **support** (or certainty), each described in detail in the following sections.[7] (Note: As mentioned, in keeping with the notation in the uncertainty literature the term support is used to signify the lower bound in any uncertainty interval. It is also used to refer to the certainty parameter as identified in the Collins-Michalski Theory. The term support is used instead of the term certainty because it was felt to better convey the meaning of the weight. Do not confuse the two usages of the term support.)

Each bracketed weight is preceded by a letter which indicates what type of weight it is. F[0.7 0.9] is thus is a frequency weight. The other weights also use the first letter in their name as the identifier. Each of the bounds in a weight can also be represented by subscripting the

[7]The Theory identifies nine certainty parameters. This difference between nine and the six listed above is for two reasons: One, as will be described in section 3.2.5, the forward and backward likelihood weights are represented as the same weight on two implications, one for the forward nature of the implication and one for the backward nature. Two, the two multiplicity parameters were not part of the initial drafts of the Collins-Michalski Theory and subsequently this implementation does not include them. Section 3.2.6 describes the multiplicity parameter and assesses the effect of their absence.

bound with the weight type identifier. Thus F[s p] contains the bounds s_F and p_F. For example, in the dominance range D[0.3 0.6] $s_D = 0.3$ and $p_D = 0.6$. (Note: the terms **weight**, **range**, and **interval** will be used interchangeably to refer to the numeric values associated with a datum. **Belief** is used interchangeably with the term support).

It should be mentioned that in this thesis all the weights were set based upon the author's experience or intuitions with regard to any datum. In general any weights with wide ranges reflect a lack of knowledge about what they should actually be. As will be mentioned in the conclusion scientific methods for the setting of uncertainty values is one area for additional research.

3.2.1. Frequency

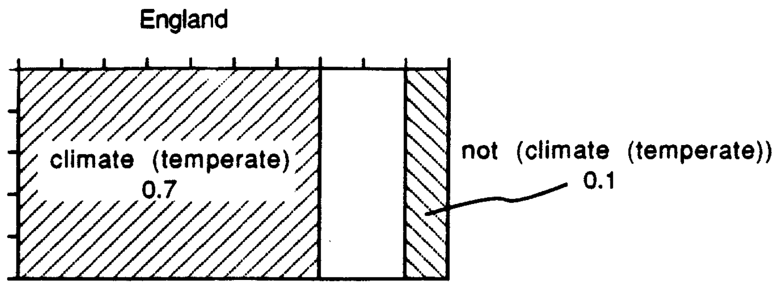
The frequency weight indicates the frequency of the referent occurring in the argument of the descriptor. For example, if the frequency weight F[0.7 0.9] was associated with the fact *(climate (England) = (temperate))* it would mean that the climate of 70 to 90 percent of England is temperate. (It might also be interpreted that the climate of England is temperate between 70 and 90 percent of the time. PRS currently interprets frequency weights using the "place" interpretation rather than the "time" interpretation). Frequency can thus be interpreted as the conditional probability of a referent given the argument: the probability that the climate of England is temperate is 0.7 to 0.9 (i.e., P(temperate | climate(England)) = 0.7 to 0.9).[8] Figure 7 gives some examples of frequency weights attached to facts. Figure 8 graphically illustrates the notion of frequency. The figure is interpreted as the outside rectangle being the climate of England which is partitioned based upon the different values which it can have. In this case we are only concerned with the value temperate.

(terrain (Illinois) = (flat F[0.8 0.9]))	The terrain of 80 to 90% of Illinois is flat.
(color (baseball) = (white F[0.97 0.99]))	The color of baseballs is 97 to 99% white.

Figure 7: Facts with associated Frequency.

To illustrate the difference in the weights attached to an inverse consider the previous

[8]Appendix A summarizes the representation and associated meaning for PRS facts. This may be a useful reference for the following chapters.



(climate (England) = (temperate F[0.7 0.9]))

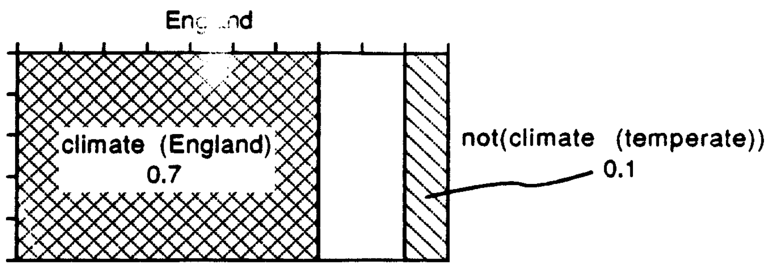
Figure 8: Graphical Representation of Frequency.

example, *(climate (England) = (temperate F[0.7 0.9]))*. The inverse fact is *(climate-value (temperate) = (England))*. Since England is but a small portion of all temperate areas the frequency attached to the inverse would be low. For the sake of this example let's say it's 0.1 to 0.2 (which is certainly somewhat inflated over the actual value). With this information the complete relationship between England and temperate in terms of the descriptor climate is specified. The top two diagrams in Figure 9 illustrate graphically these two facts. In the diagrams the area's marked with the crossed lines are in fact the same. This area is the intersection between England and a temperate climate. By normalizing the weights attached to the facts they can be represented in a single diagram. Thus 0.7 can be normalized to 0.1 (or vice versa), which, as will be described in later chapters, allows certain inferences to be made (e.g., if we know that London is a part of England what can be said about it having a temperate climate). The bottom diagram in figure 9 illustrates this normalization.

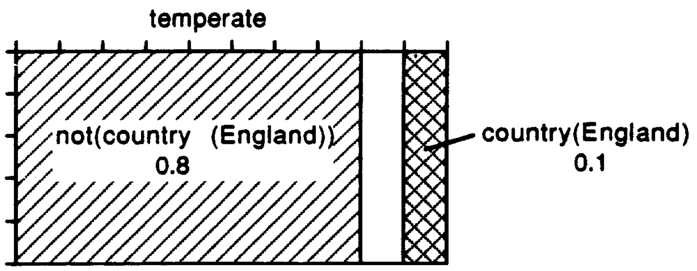
Frequency models the human ability to make judgements about the probability that a characteristic or attribute is present. As the examples in figure 7 indicate it allows the probability of an attribute having a specific value (or argument having a specific referent) to be expressed. It also allows for facts such as "a car is mostly red with some black" to be easily represented:

```
(color (car) =    (red F[0.4 0.8])
                  (black F[0.2 0.4]))
```

Intuitively the notion of frequency corresponds well with the way people think about much of the information they know. Although much of it may not be explicitly thought of in terms of weights and probabilities it is clear that people do reason with such information.



(climate (England) = (temperate F[0.7 0.9]))



(country (temperate) = (England F[0.1 0.2]))

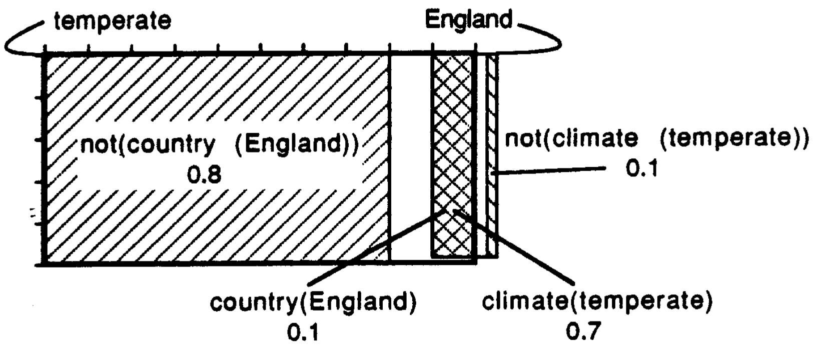


Figure 9: Graphical Representation of a Fact and its Inverse.

3.2.2. Dominance

The second type of weight is dominance. Dominance is identical to frequency except it refers to the relationship between sets and subsets and is used with any facts containing the hierarchical descriptors "type-of", "has-type", "part-of", or "has-part". It reflects the dominance of a set within another set. As with frequency, dominance can be interpreted as a conditional probability. Figure 10 gives some examples of dominance weights. Note the set/subset relationships between the arguments and referents in the statements. Figure 11 graphically represents dominance.

(has-part (car) = (engine D[1.0 1.0]))	100 % of cars have engines.
(part-of (engine) = (car D[0.4 0.6]))	40 to 60 % of engines are in cars.

Figure 10: Facts with associated Dominance.

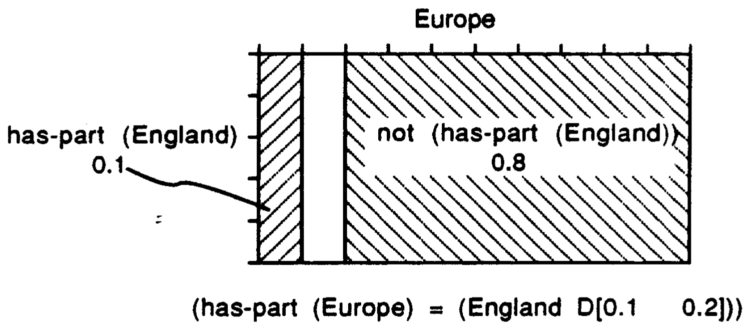


Figure 11: Graphical Representation of Dominance.

Like frequency dominance reflects people's knowledge about the relationship between sets, in this case hierarchical sets. In people much of this knowledge is used unconsciously yet it is a cornerstone of much reasoning.

3.2.3. Typicality

Typicality indicates how typical a subset is to a superset. It refers to the probability that characteristics of a superset also hold for a subset. Typicality is similar to dominance but it allows stronger inferences to be made in situations where a subset is only a very small portion of a superset yet most of the characteristics of the superset hold for the subset. As an example consider the sets of Golden Retrievers and dogs. Since there are many different kinds of dogs Golden Retrievers make up a small percentage of all dogs (low dominance). However they're fairly typical of the prototypical dog and therefore have a high typicality. As will be shown in Chapter 6 on uncertainty propagation the high typicality allows for stronger inferences than would otherwise be possible due to the low dominance.

Facts may have both dominance and typicality weights associated with them. Depending upon their values and the type of inference either of the weights or both may have a bearing on a particular reasoning process. (Again, this is discussed in Chapter 6).

Figure 11 lists some facts with typicality weights and their interpretation. (The graphic representation would be identical to that for dominance).

(has-type (car) = (sedan T[0.8 1.0]))	There is an 80 to 100% probability that facts associated with cars will also hold for sedans.
(has-part (England) = (Surrey T[0.3 0.8]))	There is a 30 to 80% probability that facts associated with England will also hold for Surrey.

Figure 12: Facts with associated Typicality.

3.2.4. Similarity

Similarity is analogous to typicality except it indicates a relationship between two non-hierarchically related sets. It refers to the probability that characteristics of one set hold for the other. It allows for inferences to be made about the presence of an attribute, or characteristic, in one set based upon a measure of its similarity to another set with that characteristic.

A number of issues make the concept of similarity extremely complex. One is that two objects are generally not just similar but similar in a certain context. For example, in a certain viewpoint cats are very similar to dogs: domesticated, relative sizes, number of legs, and so on. Yet obviously there are many ways in which they are dissimilar.

Researchers have proposed a number of ideas of how human similarity mechanisms work and how they might be modeled. For example, one view is that similarity can be measured by looking at the corresponding attributes between two objects. Measuring similarity in this manner brings up a number of issues: salience or importance of attributes when making this match, underlying functional aspects of attributes, and consideration of the attributes which don't match. Because of the great complexity involved with correctly implementing a similarity mechanism it was not attempted in this thesis. Much research has been devoted specifically to the concept of similarity and its inclusion here would not do justice to all of this research in the context of the other foci of this thesis.

As a final note one point which will be made in the conclusion is that an implementation feature of PRS is that it is amenable to incremental additions. Although the development of mechanisms to handle similarity is not a trivial task it could be done on top of the current system and it would not entail radical system modifications.

3.2.5. Likelihood

The fifth type of weight is likelihood and is attached to implications and dependencies. For implications likelihood is a measure of the belief in the consequent given the antecedents. It is interpreted as the conditional probability of the consequent given the antecedents. For dependencies likelihood is interpreted as a measure of the correlation between the two descriptors. That is, a priori to knowing the values associated with attributes, or descriptors, it is known that they may be related. Figure 13 gives an example of each.

In the Theory likelihood is represented as two separate weights, one relating the probability of of the right-hand side of a mutual implication or dependency given the left-hand side and the other vice-versa. This requires two weights attached directly to each implication and

(→ ((#-of-wheels (?x) = (4)) (has-part (?x) = (engine))) (object-id (?x) = (car)) L[0.7 0.95])	If ?x has 4 wheels and has an engine then there is a 70 to 95% likelihood that it is a car.
(←→ climate latitude L[0.8 0.9])	For any argument there is an 80 to 90% likelihood that the value of climate is related to the value of latitude.

Figure 13: English Interpretation of PRS Implications and Dependencies.

dependency, for example A → B : L_1 L_2. The representation used in this thesis is to write each implication (and dependency) as two implications (or two dependencies) and attach a likelihood weight to each. Thus the aforementioned example would be represented as A → B : L_1 and B → A : L_2. This change is merely representational and has no effect on the results of reasoning.

The likelihood weight matches well with human knowledge for both implications and dependencies. Humans have knowledge about how likely a result is if certain conditions are true. Also, in the case of dependencies it is clear that humans use a notion of dependent variables in reasoning processes.

3.2.6. Multiplicity

The multiplicity parameters refer to either a referent or an argument. They express the cardinality of possible references or arguments. Consider the fact *(mineral (Venezuela) = (oil ...)).* A multiplicity weight reflecting the number of minerals Venezuela might have could be attached to the fact as well as one reflecting the number of countries which produce oil.

As mentioned at the beginning of this Chapter the multiplicity weight is not currently included in this implementation, as this was initially based on an earlier draft of the Theory which did not include multiplicity. Although the absence of this parameter does not prevent inferences from occurring multiplicity does factor into belief calculations and thus may affect the certainty of conclusions in some instances. One of the characteristics of the mechanisms for handling uncertainty, which will be described in Chapter 6, is that all calculations are done on a conservative basis, with no simplifying assumptions. Thus if multiplicity parameters were added to the system the conclusions reached during a reasoning task would become more precise (i.e., the uncertainty ranges would become smaller) - inconsistencies with current reasoning conclusions would not arise.

As noted in section 3.2.4 PRS is amenable to incremental additions and the addition of multiplicity parameters could be done on top of the existing system.

3.2.7. Support

The final type of weight is the *support* weight, referred to in the Theory as certainty. Up to this point the weights which have been discussed - frequency, dominance, typicality, similarity, likelihood, and multiplicity - have all been *first order weights.* What is meant by a first order weight is that it is directly associated with data (i.e., facts, implications, and dependencies). Consider the combination of datum and its first order weights as a pair. Support is associated with such pairs. It indicates the degree of belief in both the datum and the first order weights connected to that datum. Support is termed a *second order weight* in that it is measure of certainty about other weights. Figure 14 shows some of the examples from the previous sections, now with a support value attached.

(terrain (Illinois) = (flat F[0.8 0.9] S[0.95 1.0]))	There is a 95 to 100% probability that the terrain of 80 to 90 % of Illinois is flat.
(has-part (car) = (engine D[1.0 1.0] S[0.9 1.0]))	There is a 90 to 100% probability that 100% of cars have engines.
(→ ((#-of-wheels (?x) = (4)) (has-part (?x) = (engine))) (object-id (?x) = (car)) L[0.7 0.95] S[0.7 0.8])	There is a 70 to 80% probability that the rule if ?x has 4 wheels and has an engine then there is a 70 to 95% likelihood that it is a car, is true.

Figure 14: Examples of Support Weights.

Support corresponds with the notion of meta-knowledge: they not only have knowledge but they have knowledge about that knowledge [Bobrow & Collins 75].

3.3. Implementation

The purpose of this section is not to provide an in-depth description of the implemented knowledge representation, but rather to give a short overview of the general principles and features.

PRS was implemented using a number of ideas expressed through the years on human memory structure. Many of the thoughts were presented in [Quillian 68] and have come to be associated with the term *semantic networks*. In PRS every object (i.e., descriptor, argument, and referent) is represented as a class, or node. Links are made between classes indicating the relationships between them. Thus for representing facts the links connect a descriptor class and an argument class to a referent class. Each top level class is represented identically which allows any class to potentially be used as a descriptor, argument, or referent.

In the world any object or concept in the world may be related to any number of other objects in multiple ways. This is modeled in PRS by allowing multiple links from any class and attaching information to the link indicating the manner in which the classes are related. For example, if the class *bat* existed in a knowledge base links attached to it would connect it to other classes, indicating the the use of the class (i.e., as a descriptor, argument, or referent) in terms of each piece of information in which it is involved. This corresponds to way people know information related to the concept bat with regard to the various meanings it may have (e.g., the verb bat or the noun corresponding to a baseball bat) and the various ways in which each meaning is used.

Other information in the knowledge base, implications and dependencies, is represented in a similar manner. In the case of implications classes are linked together to make up individual antecedents and consequents and then these groups are linked together to form the implication. A dependency is represented as a direct link between two classes - the two classes being concepts which are used as descriptors.

In addition to strictly connecting classes, links hold information on the strength of the link. More specifically, this is the uncertainty information discussed in the previous section. Thus when a fact is retrieved it is not simply indicated that it exists but certain information about it is found.

In summary, there are number of important and useful characteristics of this implementation. The whole principle of having an entity for every object and concept and allowing them to be extensively interrelated corresponds to many long held views on human memory. The implementation allows all information relating to a given object to be easily found. In addition, the incremental expansion of a knowledge base is straightforward: new classes and links can be added. Likewise for the modification of a knowledge base: the information associated with links can be retrieved and revised, or perhaps just removed.

4. INFERENCE TYPES

Now that the representation of information in PRS has been discussed, the next issue is how it is used to make inferences. The basic idea behind the inference scheme is that if a fact is not explicitly known (i.e., present in the knowledge base) a number of transformations can take place which in effect creates a **path** from a known fact to the unknown **input query** (Terminology: a **query** is any fact which the system attempts to prove. It may be one of several types: An **input** or **initial query** is the query given to the system by the user to begin the inference process. An **unknown query** is any query which is not explicitly found in the knowledge base. A **variable query** is a query with a variable. It is generated by the system based upon one of the inference transformations in order to find evidence for an unknown query. A **secondary query** is a query which results from an unknown query and the match found by a variable query. A **solution path**, or **path**, is a string of queries and facts which connects an initial query to a known fact in the knowledge base. The protocols given throughout this chapter will make these distinctions clear). Once this path is discovered, the various weights associated with each datum along the path can be combined to arrive at a frequency and support range for the query. This propagation will be discussed in Chapter 6, after each inference type has been presented.

A simple example might be useful before discussing each inference type. Consider a system such that the facts *(part-of (Surrey) = (England))* and *(climate (England) = (temperate))* were included in knowledge base but *(climate (Surrey) = (temperate))* was not. If the system was given the initial query *(climate (Surrey) = (temperate))*? a line of reasoning could be made which would indicate that since Surrey is a part of England and the climate in England is temperate then there is some degree of evidence that the climate in Surrey is temperate (a generalization-based inference).[9] Based on the weights associated with the two facts along the path a frequency and support range is assigned to the query that the climate in Surrey is temperate. Using this inference method it is possible, and probable in many situations, that the use of multiple transformations on the input query will generate multiple solution paths. In the above example we might also know that a town in Surrey is Guildford and the climate there is temperate and thus a

[9]Note: This thesis uses the terms generalization and specialization based inference operations, or inference transformations, to refer to the mechanisms by which an unknown datum is generalized or specialized to something that is known. For example, if A is unknown it may be generalized to B which is known. This allows some degree of belief to be transferred from B to A. Thus since the inference operation was a generalization the evidence transfer operation was a specialization (from B to A). This thesis uses the operation names to refer to the inference operation necessary to move from unknown to known data not vice-versa. Keep this distinction clear.

second solution path could be created (using a specialization-based inference). In such situations the weights which each path brings to bear on the query are combined to produce final frequency and support ranges for the query. (Again, this propagation and combination will be discussed in Chapter 5).

Since many of inference types in PRS depend upon the hierarchical structure of information some further clarification of the hierarchical descriptors is in order. The four descriptors, *type-of, has-type, part-of,* and *has-part* (presented in Chapter 3), map out two types of hierarchies: type-of (or "isa") and part-of (or "part/whole"). There are a number of differences in the information represented in the two types and the inferences which humans make from the information varies accordingly. The inferences depend not only on the fact that information is ordered hierarchically but also on exactly what the hierarchical relationship is and what information is being transferred. As an example, even thought a spark plug is a part of an engine clearly the weight attribute of the engine cannot be inherited by the spark plug. Alternatively, London is a part of England and certain attributes, such as language, are easily transferred between the two sets. Type-of hierarchies are more amenable to inheritance between levels but they too do not always lead to correct inferences. To avoid the complexity involved in distinguishing between acceptable and unacceptable inferences all descriptors in PRS are treated equally and thus can just as easily be reasoned with in each kind of hierarchy. One area for future research is to expand the knowledge base to encompass more information on descriptors, for example indicating validity for hierarchically based inferences.

4.1. Argument-Based Generalization

The first type of inference transformation - an argument-based transformation - was illustrated in the climate example on the previous page. More specifically, this is an example of a argument-based generalization transform.[10] The strategy of this inference is to generalize the argument of an initial query, *(DES (ARG) = (REF))*. To generalize, the system looks for facts of the form (type-of (*ARG*) = (?x)) and (part-of (*ARG*) = (?x)). If any generalizations are found they are substituted for the argument in the unknown query and this generalized query then becomes the subject of the inference mechanism. Figure 15 shows the template of queries and facts which successful argument-based generalizations follow.

Figure 16 shows the example given at the outset of this chapter cast into this template.

[10]In the most recent version of the Theory, [Collins & Michalski 88], this is referred to as a generalization-based argument transform. The names of the other inferences to be described below have been similarly revised.

Input Query:	(DES (ARG) = (REF))
Inference Operation:	Argument-Based Generalization
Variable Query:	(part-of (ARG) = (?X))
Known:	(part-of (ARG) = (ARG2))
Secondary Query:	(DES (ARG2) = (REF))
Known:	(DES (ARG2) = (REF))
Inferred:	(DES (ARG) = (REF))

Figure 15: Template for Argument-Based Generalization Inference.

Input Query:	(climate (Surrey) = (temperate))
Inference Operation:	Argument-Based Generalization
Variable Query:	(part-of (Surrey) = (?X))
Known:	(part-of (Surrey) = (England))
Secondary Query:	(climate (England) = (temperate))
Known:	(climate (England) = (temperate))
Inferre	(climate (Surrey) = (temperate))

Figure 16: Example of an Argument-Based Generalization Inference.

4.2. Argument-Based Specialization

Specialization operations are similar to generalization operations except they are concerned with finding more specific objects and relating them to the initial object. In the case of argument-based specializations instead of looking for generalizations of the argument, specializations are found using the descriptors "has-part" and "has-type". Figure 17 shows the template for these inferences.

Input Query:	(DES (ARG) = (REF))

Inference Operation:	Argument-Based Specialization
Variable Query:	(has-part (ARG) = (?X))
Known:	(has-part (ARG) = (ARG2))
Secondary Query:	(DES (ARG2) = (REF))
Known:	(DES (ARG2) = (REF))

Inferred:	(DES (ARG) = (REF))

Figure 17: Template for Argument-Based Specialization Inference.

Figure 18 illustrates an example of this inference. The initial query is to find out whether the latitude of England is greater than 40 degrees. Although this fact is not in the knowledge base it is known that Surrey is a part of England and that the latitude of Surrey is indeed greater than 40 degrees. Thus, some degree of evidence exists that England's latitude is greater than 40 degrees.

4.3. Referent-Based Generalization

Referent-based transformations are the next type of inferences. These transformations look for generalizations and specializations of the referent in a query as opposed to the argument. When generalizing a query of the form *(DES (ARG) = (REF))* the secondary queries (type-of (*REF*) = (?x)) and (part-of (*REF*) = (?x)) are generated by the system. Figure 19 presents the template for referent-based generalizations.

Figure 20 illustrates a referent-based generalization inference. In the example it is unknown

Input Query:	(color (bicycle-X) = (maroon))

Inference Operation:	Referent-Based Generalization
Variable Query:	(type-of (maroon) = (?X))
Known:	(type-of (maroon) = (red))
Secondary Query:	(color (bicycle-X) = (red))
Known:	(color (bicycle-X) = (red))

Inferred:	(color (bicycle-X) = (maroon))

Figure 20: Example of Referent-Based Generalization Inference.

4.4. Referent-Based Specialization

As in argument-based specialization transformations, referent-based specialization uses the descriptors "has-type" and "has-part" to generate secondary queries only using the referent instead of the argument. Figure 21 shows the template for these inference patterns.

Input Query:	(DES (ARG) = (REF))

Inference Operation:	Referent-Based Specialization
Variable Query:	(has-part (REF) = (?X))
Known:	(has-part (REF) = (REF2))
Secondary Query:	(DES (ARG) = (REF2))
Known:	(DES (ARG) = (REF2))

Inferred:	(DES (ARG) = (REF))

Figure 21: Template for Referent-Based Specialization Inference.

Input Query:	(latitude (England) = (>40-degrees))
Inference Operation:	Argument-Based Specialization
Variable Query:	(has-part (England) = (?X))
Known:	(has-part (England) = (Surrey))
Secondary Query:	(latitude (Surrey) = (>40-degrees))
Known:	(latitude (Surrey) = (>40-degrees))
Inferred:	(latitude (England) = (>40-degrees))

Figure 18: Example of Argument-Based Specialization Inference.

Input Query:	(DES (ARG) = (REF))
Inference Operation:	Referent-Based Generalization
Variable Query:	(part-of (REF) = (?X))
Known:	(part-of (REF) = (REF2))
Secondary Query:	(DES (ARG) = (REF2))
Known:	(DES (ARG) = (REF2))
Inferred:	(DES (ARG) = (REF))

Figure 19: Template for Referent-Based Generalization Inference.

whether or not a bicycle is maroon however it is known that maroon is a type of red and that the bicycle is red. Thus some degree of evidence has been found for the bicycle being maroon. The assumption inherent in this reasoning should be apparent: that because something is red it is maroon. In actuality there are many shades of red which something that is red might be and the uncertainty mechanisms will constrain such assumptions.

Figure 22 illustrates one possible referent-based specialization inference. Although it is not known explicitly that in pine trees in general exist on Cape Cod, it is known that scotch pines, which are a type of pine, do. This allows the inference that pine trees are present on Cape Cod.

Input Query:	(tree-types (Cape-Cod) = (pine))

Inference Operation:	Referent-Based Specialization
Variable Query:	(has-type (pine) = (?X))
Known:	(has-type (pine) = (scotch-pine))
Secondary Query:	(tree-types (Cape-Cod) = (scotch-pine))
Known:	(tree-types (Cape-Cod) = (scotch-pine))

Inferred:	(tree-types (Cape-Cod) = (pine))

Figure 22: Example of Referent-Based Specialization Inference.

4.5. Implication

The fifth type of inference transformations in PRS involve implications. The presence of an implication which has an unknown query as its consequent invokes a backward chaining operation whereby secondary queries are generated from the antecedent facts in the rule. Figure 23 illustrates the general template which implication operations follow.

Figure 24 gives an example of an implication based inference. The goal of the query in the example is to determine if the computer is an Apple. Using the rule that Macintosh computers are made be Apple and the knowledge that the computer in question is a Macintosh the query is inferred.

Note that although in the example the implication has only a single antecedent is is possible to have multiple antecedents which all must be present in the knowledge base for a path to be generated.

Input Query:	(DES (ARG) = (REF))

Inference Operation:	Implication
Variable Query:	(→ ?X (DES (ARG) = (REF)))
Known:	(→ ((DES2 (ARG2) = (REF2))) (DES (ARG) = (REF)))
Secondary Query:	(DES2 (ARG2) = (REF2))
Known:	(DES2) (ARG2) = (REF2))

Inferred:	(DES (ARG) = (REF))

Figure 23: Template for Implication Inference.

Input Query:	(made-by (computer-A) = (Apple))

Inference Operation:	Implication
Variable Query:	(→ ?X (made-by (computer-A) = (Apple)))
Known:	(→ ((computer-type (?X) = (Macintosh))) (made-by (?X) = (Apple)))
Secondary Query:	(computer-type (computer-A) = (Macintosh))
Known:	(computer-type (computer-A) = (Macintosh))

Inferred:	(made-by (computer-A) = (Apple))

Figure 24: Example of Implication based Inference.

4.6. Dependency-Based Inferences

The next class of inferences, dependency-based inferences, are similar to implication-based inferences. The key difference is that dependencies are not as specific - they do not relate facts directly to each other as in as implication but instead relate just descriptors. Figure 25 illustrates the general template for dependency-based inferences.

Input Query:	(DES (ARG) = (REF))
Inference Operation:	Dependency
Variable Query:	(← → DES ?X)
Known:	(← → DES NEW-DES)
Variable Query:	(NEW-DES (ARG) = (?X))
Known:	(NEW-DES (ARG) = (NEW-REF))
Variable Query:	(NEW-DES (?X) = (NEW-REF))
Known:	(NEW-DES (NEW-ARG) = (NEW-REF))
Secondary Query:	(DES (NEW-ARG) = (REF))
Known:	(DES (NEW-ARG) = (REF))
Inferred:	(DES (ARG) = (REF))

Figure 25: Template for Dependency Inference.

In dependency inferences a general correlation between two descriptors is known as well as some facts involving them with several arguments. One way of thinking of these inferences is that by using the facts associated with the second argument an implication can be made with one fact implying the other. Using this derived implication with the fact associated with the first argument the initial query can be inferred. Obviously this is not as strong an inference as the case when an implication is already existing but in certain situations that degree of information is not available and dependency inferences can be useful.

Figure 26 gives an example of a dependency-based inference. This inference works by first finding a dependency relating the descriptor in the initial query to another descriptor. Latitude was found, which indicates some link between that and climate. The next step is finding if the argument in the query - Surrey - has a value, or referent, for latitude. Once that is found a search is made for other arguments with that same latitude and Holland is found. Holland's referent for the descriptor climate is checked and it is found to be temperate. Since it is known that the values of climate and latitude are related and Holland and Surrey have the same value for latitude it can be inferred that they have the same value for climate. Thus evidence has been found supporting the initial query that the climate of Surrey is temperate.

Input Query:	(climate (Surrey) = (temperate))

Inference Operation:	Dependency
Variable Query:	(←→ climate ?X)
Known:	(←→ climate latitude)
Variable Query:	(latitude (Surrey) = (?X))
Known:	(latitude (Surrey) = (>40-degrees))
Variable Query:	(latitude (?X) = (>40-degrees))
Known:	(latitude (Holland) = (>40-degrees))
Secondary Query:	(climate (Holland) = (temperate))
Known:	(climate (Holland) = (temperate))

Inferred:	(climate (Surrey) = (temperate))

Figure 26: Example of Dependency-Based Inference.

4.7. Combined Inferences

Combined inferences refer to two aspects of the system which extend its power. The first is the use of multiple inferences along one solution path. In the examples given previously the general format is to give the system a query and when it could not be found an inference operation transformed it to a secondary query. In the examples this secondary query was always found to be in the knowledge base and thus a solution path was generated. However it is not necessarily the case (or perhaps even likely) that the secondary query will be known. In that case another inference operation will be performed on the secondary query, generating another secondary query which the knowledge base will be checked for. If that query is present a path will have been found. If not the process can repeat itself again. Figure 27 gives an example of a two step argument-based generalization inference. This example is similar to that in figure 16 except that (climate (England) = (temperate)) is unknown and the argument is further generalized to Europe.

Input Query:	(climate (Surrey) = (temperate))

Inference Operation:	Argument-Based Generalization
Variable Query:	(part-of (Surrey) = (?X))
Known:	(part-of (Surrey) = (England))
Secondary Query:	(climate (England) = (temperate))
Inference Operation:	Argument-Based Generalization
Variable Query:	(part-of (England) = (?X))
Known:	(part-of (England) = (Europe))
Secondary Query:	(climate (Europe) = (temperate))
Known:	(climate (Europe) = (temperate))

Inferred:	(climate (Surrey) = (temperate))

Figure 27: Example of a two step Inference (Argument-Based Generalization).

Multiple step inferences are not confined to a single inference type. Thus if an implication was known implying that the climate of England is temperate knowing its antecedent would also create a solution path.

The second type of combined inference refers to the fact that at each point where a query is not found, not just one but all six of the possible inferences are tried. Thus, as has been mentioned previously, support for an initial query may come from multiple solution paths. As an example consider figures 16 and 26. In both examples the same initial query was given and using different inference methods two different solution paths were generated. As will be shown in Chapter 6 both paths bring some level of support for the initial query. Note that although six inference operations are attempted on each unknown query the number of paths leading from an unknown query is not a fixed number. One operation may not be applicable while another may produce several secondary queries. For example, consider the query *(tax-rate (Urbana) = (low))*. The knowledge base may have several generalizations of the argument *Urbana - town, Champaign County* - but no specializations.

Several points should be made about combined inferences. The first is that it is conceivable that the system will have an unknown query, perform an inference operation, generate another query, not find that query in the knowledge base, and repeat the process ad infinitum. Also, since each inference operation is attempted for each unknown query the number of paths spreading from one unknown query can grow very large. For both of these reasons a number of inference control mechanisms are used - they are described in the following chapter.

5. INFERENCE CONTROL

As discussed in Chapter 4 there are six inference operations which can produce secondary queries. Because each of the six are applied to any unknown query the number of diverging paths from an input query can grow quite large. To illustrate the combinatorics of the inference process consider if a query "A" is unknown. The six inference operations can be performed to find related information, resulting in a number of new queries.[11] For any of the new queries that are unknown the six inference operations can be applied to each and more queries are generated, and so on. Thus, continually applying each transformation to each unknown query quickly increases the number of queries which PRS must search for. Because of this increase some type of control is needed so that the system does not get bogged down in its search for data relevant to the initial query.

Prior to describing the specific control mechanisms an overview of the general control structure is necessary. The system uses two stacks for controlling the inference: an **active stack** and an **inactive stack**. Both are initially empty. Elements in both stacks are operations to be performed on a particular query. The active stack is processed as the system proceeds with its inference. The inactive stack holds elements which are no longer processed. Elements may be put in the inactive stack for several reasons which will be discussed below.

To initiate the reasoning process an initial query is entered. A check for the input query is added as the first element in the active stack. The stack is then processed by popping its top element and performing the associated operation. Thus if the input query is in the knowledge base the uncertainty values for it are returned. If it is unknown, however, inference operations on the query are added to the stack and the first element is again processed.

[11]Recall from the end of Chapter 4 that the number of secondary queries does not necessarily equal the number of inference operations. For example, several generalizations may exist for some argument or, conversely, no generalization may exist.

Using the inference operations described in the following chapter ten possible operations may be generated and become stack elements. All are operations to be performed on any input or unknown queries (shown in no particular order):[12]

1. (CHECK-QUERY query)

   Checks the knowledge base for the existence of the query. If the query is found the inference path from the query to the initial query is traversed, performing the uncertainty calculations. In the case of an initial query being directly found the exact uncertainty weights associated with it are returned. If it is not found the operations listed below are added to the stack.

2. (GET-ARGUMENT-GENERALIZATIONS query)

   Checks for facts of the form *(part-of (<argument>) = (?X))* or *(type-of (<argument>) = (?X))*. If found new queries of the form *(check-query <descriptor> (?X) = (<referent>)))* are added to the stack.

3. (GET-ARGUMENT-SPECIALIZATIONS query)

   Checks for facts of the form *(has-part (<argument>) = (?X))* or *(has-type (<argument>) = (?X))*. If found new queries of the form *(check-query <descriptor> (?X) = (<referent>)))* are added to the stack.

4. (GET-REFERENT-GENERALIZATIONS query)

   Checks for facts of the form *(part-of (<referent>) = (?X)) or (type-of (<referent>) = (?X))*. If found new queries of the form *(check-query <descriptor> (<argument>) = (?X)))* are added to the stack.

5. (GET-REFERENT-SPECIALIZATIONS query)

   Checks for facts of the form *(has-part (<referent>) = (?X))* or *(has-type (<referent>) = (?X))*. If found new queries of the form *(check-query <descriptor> (<argument>) = (?X)))* are added to the stack.

6. (BACKCHAIN query)

   Checks for implications which have query as their consequent. If any are found the antecedents are added to the stack with CHECK-QUERY operations.

7. (GET-DEPENDENCIES query)

   Checks for dependencies relating to the <descriptor> in query: *(<--> <descriptor> ?X)*. If found an operation (check-arg-with-new-des query ?X) is added to the stack, where ?X is the newly found dependent descriptor. (Note: Making inferences with dependencies is more complicated and requires a number of steps. Dependency-based inferences are implemented by this operation along with the following three. See the discussion of dependency-based inferences in section 4.6 for a more complete description.)

8. (CHECK-ARG-WITH-NEW-DES query NEW-DES)

   Checks for facts of the form *(NEW-DES (<argument>) = (?X))*. If found an operation *(get-matching-arg* query *NEW-DES ?X)* is added to the stack.

[12]Query, the argument in all of the operations, is of the form (<descriptor> (<argument>) = (<referent>)). Angle brackets surrounding a word, < and >, indicate a particular value, not a variable.

9. (GET-MATCHING-ARG **query** NEW-DES NEW-REF)

   Checks for arguments which have the referent NEW-REF for the descriptor NEW-DES (e.g., *(NEW-DES (?X) = (REF))*). If found an operation *(get-ref-with-init-des* **query** *?X)* is added to the stack.

10. (GET-REF-WITH-INIT-DES **query** NEW-ARG)

    Adds *(check-query (<descriptor> (NEW-ARG) = (<referent>))* to the stack.

In addition to dealing with the combinatoric problem due to the multiple possible operations, control of the stack is necessary in order to model human reasoning. Simply processing the stack in a standard first in first out manner does not model human reasoning. The mechanisms described in the following section correspond to a number of reasoning behaviors exhibited by people. One of the most obvious is the relationship between reasoning processes and time. People constantly reason in time constrained situations. They have the ability to reason quickly, although in some situations incorrectly, when time is not available to consider all evidence. They have the ability to decide that too much time has been spent on a particular solution method and something else should be tried. In addition, people shift their attention away from less promising avenues toward areas more likely to produce an answer. People's reasoning processes are directed, either consciously or subconsciously, towards using procedures which have better chances of success than others.

5.1. Control Mechanisms

A number of control mechanisms have been identified and integrated into the inference scheme to constrain and direct processing. The mechanisms involve four areas: time, solution path length, evidence limits, and inference type priorities. Following the description of all the mechanisms a table summarizes the control parameters, giving their default values. The default values, which are all adjustable, were developed based on a combination of experimentation with the uncertainty mechanisms (discussed in Chapter 6) and intuitions about human reasoning.

5.1.1. Time

The time spent on finding evidence for a query is the first means of inference control. Time limits are used in both an absolute and relative sense. The absolute time limit is the total allowable amount of time to be spent processing an input query. When this threshold is exceeded processing terminates and any evidence found so far is summarized. The relative time limit is the amount of time which has transpired since the operation which initiated the current one was popped off the stack. When this threshold is exceeded the element is not processed immediately but put at the bottom of the stack. When an element is put at the bottom of the stack the time at which it is subsequently processed becomes the time which descendent elements use for their relative time check. This mechanism allows for a combination of depth-first and breadth-first processing. Paths are processed in a depth-first manner until it is "decided" that perhaps too much time has been spent following this path with no results and another should be tried.

5.1.2. Solution Path Length

The next control mechanism is a constraint on the length of a solution path - it puts a maximum limit on the length of paths. Recall that a solution path is the sequence of queries, inferences, and known data which provides some degree of evidence for the initial query. In addition, a **current path** is a sequence of operations which have not yet reached a solution but are still being processed. The **path length** is the number of queries, inference operations, and data in any path. The path length is used to terminate search down a path after a globally defined length is exceeded. As the current path length increases the chances are that the amount of evidence which the particular path can possibly bring to bear on the input query will be minimal, due to the propagation of weights through multiple transformations. Intuitively this makes sense - the discussion of uncertainty in Chapter 6 will corroborate this. (Theoretically the degree of certainty does not necessarily decrease - it could stay constant as the path length increases. However, this would be a highly improbable situation as all the uncertainty weights along the path would have to be absolutely certain).

5.1.3. Evidence Limits

A limit is put on the number of separate pieces of evidence, or solution paths, to be found for any given input query. When it is reached the processing of the stack terminates and the evidence found is summarized. This is useful in constraining the search in robust knowledge bases where processing may proceed indefinitely, relating every fact in the knowledge base to every other.

5.1.4. Prioritizing Inference Types

Inference type priority is the most complex of the control mechanisms. It attempts to steer the processing in directions which have the greatest chance of finding useful evidence. Each stack element has an associated worth, on a scale of 100 (high) to 0, which is a measure of the element's likelihood of producing useful evidence. The initial query is given a worth of 100 and as new elements are made and added to the stack they are given a new, decremented worth value which is based upon the worth of the element from which they offshoot and a rating value for the operation associated with the new element. Each of the six inference operations has a rating value associated with it. These are used of rank the different inference operations and steer the system toward using operations which will more likely produce results. For example, if an implication is available relating to an unknown query it might be more beneficial to use it and see if any evidence can be obtained rather than performing a referent-based transformation. The rating value for each inference operation is set by using either subjective methods or empirical evidence as to what inferences tend to produce reliable evidence. As an example of how worth is decremented, consider an initial query "A" with a worth of 100. If an argument-based generalization operation, which has a rating value of 0.9 , is performed on it the worth of the resulting element is 90 (100 x 0.9). If another argument-based generalization is performed on that element the resulting worth would be 81 (90 x 0.9).

The worth associated with stack elements is used in several manners. It is used to sort the stack prior to popping an element from it. Elements with higher worth values are put at the top of the stack on each cycle. Thus the processing is also geared to somewhat of a best-first search - those operations which have been determined to more reliably generate evidence are tried before less reliable operations. There is also a notion of absolute and relative worth limits. The absolute worth limit is the threshold below which a stack element is put on the inactive stack instead of being processed, thereby terminating search down a particular path. The relative worth limit is the percentage of the worth of the current element to the worth of the parent element which was initially popped from the stack. If the current elements worth is below this percentage the element is not processed currently but put at the bottom of the stack. When an element is put at the bottom of the stack the relative worth control is reset: the relative worth check for elements which eventually descend from it is based upon its worth.

As an example of the relative worth mechanism consider a stack with the element "A" with a worth of 80 at the top. It is popped off the stack and processed. One of the operations performed on it has a rating value of 0.9 which generates a new element "B" with a worth of 72 (80 x 0.9). When "B" is popped off the stack and examined, its' worth, 72, is greater than the relative worth limit, set to its default of 0.7, times its parents worth, 80 (i.e., 72 > 80 x 0.7 (0.56)) so it is processed. An operation with a rating value of .75 is then applied which creates a new element "C" with a worth of 54 (72 x .75). When "C" is popped from the stack its worth is found to be less than 56 so it is put on the bottom of the stack. Putting an element on the bottom of the stack resets the relative worth limit check - when elements which are descendents of "C" are checked their worth will have to fall below 45 (56 x 0.7) for them to be put on the bottom of the stack.

Figures 28 and 29 summarize these parameters, giving their default values. Recall from the introduction to this section that the default values were set based on a combination of experimentation with the uncertainty mechanisms (discussed in Chapter 6) and intuitions about human reasoning.

VARIABLE	DESCRIPTION	PURPOSE
abs-time-limit	Maximum number of seconds to find evidence for an input query. Default value: 400.	Restricts inference from running indefinitely.
rel-time-limit	Limit on the length of time an element and its descendents are processed before putting them on the end of the stack. Default value: 10.	Allows one line of reasoning to be explored for some amount of time before shifting to another line.
path-limit	Limit on the length of paths of evidence. When this limit is exceeded the operation is put on the inactive stack instead of being processed. Default value: 10.	Prevents lengthy, probably insignificant lines of reasoning.
evid-limit	Limit on the number of separate pieces of evidence to find relating to an input query. Default value: 10.	Allows a decision to be made after a specified amount of evidence has been brought to bear on the query.

Figure 28: Inference Control Parameters (Figure 1 of 2).

5.2. PRS Control Algorithm

The control algorithm developed for PRS combines the mechanisms described in the previous section to optimize reasoning capabilities. As the specific applications and goals for PRS can be diverse optimal may take on a variety of meanings. Thus the algorithm is controlled by the aforementioned parameters (summarized in figures 28 and 29) which allow it to be adjusted and optimized to suit the needs at hand. The use of variables allows for generality, as well as the ability to tune the system to model or test specific inference theories. The algorithm is described below and also presented in figure 30.

The control algorithm works by first creating the two control stacks: the active stack and the inactive stack. (Recall from the outset of this chapter that the active stack holds operations still to be processed while the inactive stack holds operations which are no longer considered for processing.) An initial query of the form *(descriptor (argument) = (referent))* is input and given a worth value of 100. If this query is explicitly in the knowledge base its certainty values are retrieved and output. If the query is not known then operations performing the six possible inference types on the query are added to the active stack. Thus if an initial query given to the system is unknown six elements are added to the stack. Each operation, or element, is given a

VARIABLE	DESCRIPTION	PURPOSE
abs-worth-limit	The worth value below elements are taken from the active stack and put on the inactive stack. Default value: 50.	Prevents operations which are most likely insignificant from being performed.
rel-worth-limit	Limit on the percentage of the current worth of an operation to the initial worth of its parent before it is put on the end of the stack. Default value: 0.70.	Gives paths whose element's worths are stable a higher priority to check than those who are decreasing more steadily.
arg-gen-rating	Worth of query resulting from argument-based generalization inference is set to this percentage of the worth of the query it was performed on. Default value: 0.90.	Ranks the expected utility of the argument-based generalization inference as compared to the other 5.
arg-spec-rating	Worth of query resulting from argument-based specialization inference is set to this percentage of the worth of the query it was performed on. Default value: 0.8.	Ranks the expected utility of the argument-based specialization inference as compared to the other 5.
ref-gen-rating	Worth of query resulting from referent-based generalization inference is set to this percentage of the worth of the query it was performed on. Default value: 0.8.	Ranks the expected utility of the referent-based generalization inference as compared to the other 5.
ref-spec-rating	Worth of query resulting from referent-based specialization inference is set to this percentage of the worth of the query it was performed on. Default value: 0.9.	Ranks the expected utility of the referent-based specialization as compared to the other 5.
imply-rating	Worth of query resulting from an implication is set to this percentage of the worth of the query it was performed on. Default value: 0.95.	Ranks the expected utility of the implication inference as compared to the other 5.
depend-rating	Worth of query resulting from a dependency inference is set to this percentage of the worth of the query it was performed on. Default value: 0.75.	Ranks the expected utility of the dependency inference as compared to the other 5.

Figure 29: Inference Control Parameters (Figure 2 of 2).

worth value based upon the worth value of the element from which it is proceeding (in this case from the initial query which has a value of 100) and the worth rating associated with the type of inference operation (*arg-gen-rating*, *arg-spec-rating*, *ref-gen-rating*, *ref-spec-rating*, *imply-rating*, and *depend-rating*, as discussed in the previous section). Prior to adding the new elements to the stack they are sorted based upon their worth values. Thus new elements in the stack are ordered so that operations with higher worth values are performed first. (Note that the entire stack is not sorted, just the newly added elements.)

The system proceeds by popping the top operation off the stack, performing the operation, and adding new elements to perform additional inference when facts are not found in the knowledge base. If operations cannot be performed (e.g., there are no generalizations of the argument in the initial query) the element is put onto the inactive stack and the next element from the active stack is processed. When evidence is found for the input query - i.e., a path is found from the query to a fact in the knowledge base - the inferred uncertainty weights are calculated and output. (The uncertainty mechanism is discussed in the following chapter.)

As discussed in the previous sections a number of controls are used to guide the system. A maximum time limit is checked each time an element is popped from the stack. If the allowable time for processing the input query (*abs-time-limit*) has been exceeded the remaining elements are transferred to the inactive stack and processing terminates. Likewise if the limit on the number of individual pieces of evidence to be found (*evid-limit*) has been reached processing terminates.

In addition to these two global controls four local controls are used in examining each element popped from the stack before its operation is performed. If the length of the path from the input query to the current element exceeds the set limit (*path-limit*) the element is put on the inactive stack and the next element from the active stack is processed. Likewise, if the worth value associated with the element has fallen below the set value (*abs-worth-limit*), due to a string of inference operations (which all cause the value to be decremented), the element is put on the inactive stack.

As discussed in earlier sections in this chapter two relative checks are done to compare the current element to the original stack element from which it descends. Initially the input query given the system is considered the original element. These checks are on time and worth - if the difference between either the time or worth of the original element and the current element exceeds the set values (*rel-time-limit*, *rel-worth-limit*) the element is put at the bottom of the stack and the next element is popped from the stack and processed. Recall from above that when new elements are added to the stack they are sorted among themselves but the whole stack is not sorted. This creates a depth first inference mechanism. These last two controls effectively cut off this depth first inference after a point, causing the system to consider other branches. When either of these relative limits is exceeded and an element is put at the bottom of the stack its relative time and worth values are reset. The relative time and worth of any elements which might subsequently proceed from them are compared to the reset values. The concept of original stack elements was introduced at the beginning of this paragraph. The initial query, as well as elements which are put at the bottom of the stack, due to one of these controls, are considered the original elements as far as any subsequent elements proceeding from them are concerned. Because their relative time and worth values are reset, operations can proceed from them in a depth first manner, in the same manner as with the initially input query, until these relative limits are again exceeded.

In figure 30 the items marked with asterisks are the variable parameters for adjusting the algorithm. Note that by setting certain parameters at extreme values they can be effectively removed from having an effect on the control. Thus, for example, if the path length threshold was set to a very high value that check would have no bearing on the algorithm. The parameters can also be adjusted in order to create standard search algorithms. For example, by setting a very low relative time limit breadth-first search would be done.

```
1. Create active stack, A-Stack, and inactive stack, I-Stack.
2. Get input query, In-Query, and put (CHECK-QUERY In-Query) as the first element
   in the A-Stack, with a worth value of 100.
      3. If A-Stack is empty go to 15 (terminate).
      4. If *abs-time-limit* for finding evidence for In-Query is exceeded then
         transfer all elements in A-Stack to I-Stack and go to 15.
      5. If the number of pieces of evidence (referred to as evidence nodes) already
         found which relate to In-Query equals *evid-limit* then move all elements in
         A-Stack to I-Stack and go to 15.
      6. Get next element, E, from A-Stack.
            7. Check the length of the path from In-Query to E - if greater than
               *path-limit* then put E on I-Stack and go to 3.
            8. Get worth of E - if below *abs-worth-limit* then put E on I-Stack
               and go to 3.
            9. Check time that stack element which lead to E was initially pulled off
               A-Stack - if it exceeds *rel-time-limit* then reset time on E, put E at
               bottom of A-Stack, and go to 3.
           10. Check difference between E's worth and worth of the initial stack
               element which led to E - if greater than *rel-worth-limit* then reset
               relative worth of E, put E at bottom of A-Stack, and go to 3.
           11. Perform operation associated with E. Possibilities are the operations
               associated with the six inference types.
                 12. If element operation produces new elements then set their
                     worth to a decremented value of E's worth (based on the
                     transformation type which produced them). If more than one
                     new element then sort them by worth, add them to top of
                     A-Stack, and go to 3.
                 13. If E is a (CHECK-QUERY fact) operation and the fact is
                     found in the knowledge base then create an evidence node
                     for the path from E to In-Query and go to 3.
                 14. If performing E generates no additional elements and is not a
                     CHECK-QUERY operation whose fact is found in the
                     knowledge base then put E in the I-Stack and go to 3.
15. Terminate: Summarize the evidence found for the In-Query.
```

Figure 30: PRS Inference Control Algorithm.

6. UNCERTAINTY

When an inference path is generated the weights associated with each datum in the path are combined to give both a frequency and support weight for the input query. To illustrate what PRS actually generates consider the following example inference, initially presented in figure 22 (Chapter 4), which is rewritten in figure 31 only now with uncertainty weights. As can be seen, not only does PRS find information related to an input query but also calculates uncertainty weights for that query.

Input Query:	(tree-types (Cape-Cod) = (pine))

Inference Operation:	Referent-Based Specialization
Variable Query:	(has-type (pine) = (?X))
Known:	(has-type (pine) = (scotch-pine D[0.1 0.2] S[1.0 1.0]))
Secondary Query:	(tree-types (Cape-Cod) = (scotch-pine))
Known:	(tree-types (Cape-Cod) = (scotch-pine F[0.8 0.9] S[0.9 1.0]))

Inferred:	(tree-types (Cape-Cod) = (pine F[0.8 1.0] S[0.9 1.0]))

Figure 31: Example of Inference path with weight calculations.

If multiple pieces of evidence are found the certainty which each brings to bear on the query is combined, producing a single set of frequency and support bounds for the query. The reasoning process is non-monotonic - as new evidence is found it may support the negation of the query as well as the query - and thus the certainty weights for a query may rise and fall as the reasoning process proceeds.

It was briefly mentioned in section 3.2.7 that frequency, dominance, typicality, and likelihood are first-order weights and support is a second-order weight. To further elaborate, the weights can be thought of as existing in two separate probability spaces, one relating to the first-order weights and the other relating to second order (support) weights. Thus the frequency and support values for the input query are calculated independently of each other. Prior to describing the uncertainty method implemented in PRS a brief introduction to the handling of uncertainty in other AI systems will be given. The general rules developed for PRS for combining uncertainty weights in both probability spaces will be then be detailed. Following this, an example of the propagation of uncertainty for each of the different inference types will be shown.

A number of uncertainty methods have used by AI researchers in their attempt to model the uncertainty in the world. The purpose here is not to review them in detail but to point out a few well-known deficiencies with the existing methods which precipitated the development of the mechanisms in PRS. For one reference see [Quinlan 83] or [Cohen 85] for a more detailed discussion of alternative uncertainty mechanisms.

Three means of handling uncertainty have received the greatest attention in AI systems of late: Bayesian Probability (for one reference see [Duda, Hart & Nilsson 76]), the Dempster-Shafer Theory of Evidence [Shafer 76], and fuzzy logic [Zadeh 65]. Each of these methods has its strengths and weaknesses. One of the concerns with any method is its tractability: is the information necessary to reach a conclusion available? For Bayesian probability schemes the answer is generally no. In order to reach conclusions the formal definition of Bayesian propagation requires that the conditional probabilities of every combination of datum in the knowledge base be known. Having this degree of information available in any real domain is unrealistic, making the Bayesian scheme less attractive.

One way in which this problem has been reduced is by making the assumption of conditional independence between data. The Dempster-Shafer Theory, a generalization of Bayesian probability, also makes this assumption. The problem is that it is an incorrect assumption. For example, if you had some rule which said that a person's height and weight implied what gender they are it is clearly erroneous to assume no relationship between height and weight.

Fuzzy logic is the idea that concepts do not have clear, defined definitions but rather have vague, context dependent boundaries. In some ways fuzzy logic is analogous to use of frequency and dominance weights in PRS to characterize the overlap between sets. Most fuzzy logic based systems are concerned with problems involving linguistic interpretation and as this was not the focus of PRS its mechanisms were not used.

The uncertainty method in PRS was developed with several underlying principles. The first is that a conservative approach was taken: in propagating weights no assumptions are made which could lead to an inflated lower bound or deflated upper bound. Also, it allows for varying degrees of known information: if information is available it is used, if not an uncertainty calculation can still be made, albeit with less precision.

6.1. Propagation Rules

The general rules for computing the negation, conjunction, and disjunction of uncertainty weights will be presented. In addition, the calculations necessary for propagating weights through implications, generalization and specialization based inferences, and dependencies will be detailed. For each operation the calculation for both first and second order weights will be shown. The equations for conjunction and disjunction are essentially the same as those given in [Quinlan 83].

6.1.1. Negation

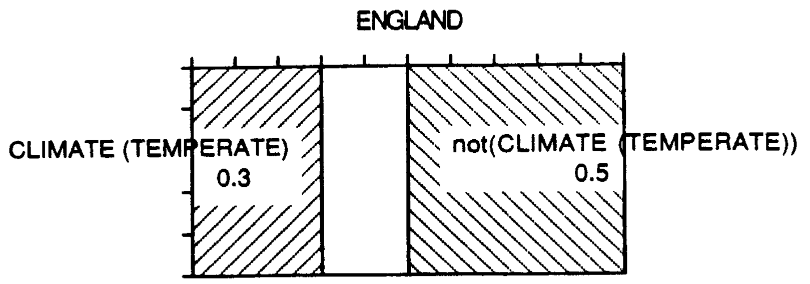
As described in Chapter 3 a frequency range is associated with each fact. The lower bound represents the minimum probability of the argument having the referent and the upper bound the maximum. Thus a fact such as (climate (England) = (temperate F[0.3 0.5])) indicates that there is between a 30 and 50% probability that England has a temperate climate and, conversely, a 50 to 70% probability that it does not. This is illustrated by figure 32. The following equations apply to first order frequency negation:[13]

$$s_x(\neg DATUM) = 1 - p_x(DATUM) \tag{1}$$

$$p_x(\neg DATUM) = 1 - s_x(DATUM) \tag{2}$$

Note the x subscript in equations 1 and 2, which indicates that they are applicable for negating all of the five types of weights, including second-order support weights. To illustrate the negation of support weights consider the support interval of S[0.6 0.8] attached to the fact given above, (climate (England) = (temperate F[0.3 0.5])). The support and its negation are shown graphically in figure 33.

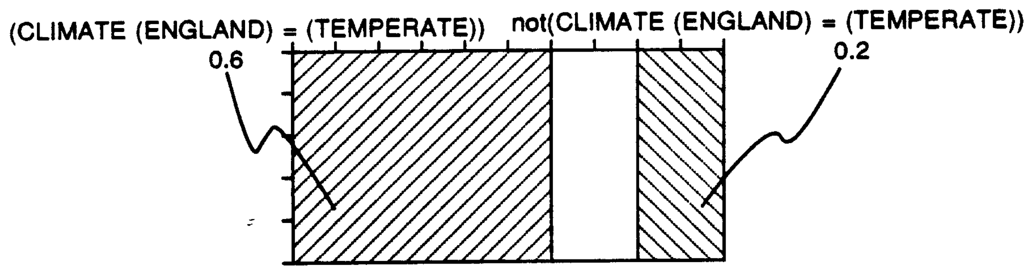
[13]Recall the notation from Chapter 3: [s p] defines a weight where s is the support, or lower bound, and p is the plausibility, or upper bound. A capital letter (F, D, T, L, or S) preceding an interval indicates the type of weight (e.g., F[s p]). A specific bound can be written as s or p with a subscripted letter signifying the type of weight (e.g., s_F). When x is used in an equation it indicates that it can be applied to any of the five types of weights. In the equations DATUM represents the piece of knowledge to which the weights relate.



(CLIMATE (ENGLAND) = (TEMPERATE F[0.3 0.5]))

(CLIMATE (ENGLAND) = (not (TEMPERATE) F[0.5 0.7]))

Figure 32: Graphic Representation of Frequency Weight and its Negation (1st Order).


(CLIMATE (ENGLAND) = (TEMPERATE))
0.6
not(CLIMATE (ENGLAND) = (TEMPERATE))
0.2

(CLIMATE (ENGLAND) = (TEMPERATE S[0.6 0.8]))

(not (CLIMATE (ENGLAND) = (TEMPERATE)) S[0.2 0.4])

Figure 33: Graphic Representation of Support Weight and its Negation (2nd Order).

6.1.2. Conjunction

Many situations require the conjoining of multiple facts and thus also require the conjoining of the associated weights. Conjunction is the same for both first and second order conjunction. The lower bound for a conjunction is determined by finding the minimal possible intersection between two facts, using each of their lower bounds. The upper bound is determined by finding the maximum possible intersection, using the upper bounds associated with each fact. The equations for this calculation are similar to those given in [Quinlan 83]:[14]

$$DATUM \equiv D_1 \wedge D_2 \wedge \cdots \wedge D_n \tag{3}$$

$$s_x(DATUM) \geq MAX(1 - \sum_{i=1}^{n}(1 - s_x(D_i)) \ , \ 0) \tag{4}$$

$$p_x(DATUM) \leq MIN(p_x(D_i)) \qquad i = 1 \ldots n \tag{5}$$

Figure 34 illustrates graphically the notions behind conjoining frequency weights for multiple facts. Figure 35 illustrates the same for support weights.[15]

[14]Using the described notation $s_x(D_i)$ indicates the support, or lower bound, for some datum D_i and $p_x(D_i)$ indicates the plausibility, or upper bound.

[15]In interpreting the graphical figures in this Chapter be aware of the direction of the diagonal filler lines. The directions remain consistent in all diagrams in a figure. Also the diagrams are ticked off in one-tenth increments on the top and one-fifth increments on the side so that the correspondence between uncertainty weights and area is clear. Consider figure 34 for example. In the top diagrams the lower frequency bound for each conjunct - climate(temperate) and altitude(<100-meters) - is indicated by lines slanting in opposite directions. These same directions are consistent in the bottom two figures, the crossed sections indicating overlapping areas. In the lower left figure the minimum possible overlap, or lower bound, is 0.1 and in the lower right, which uses the upper frequency bound for each conjunct, the maximum possible overlap, or upper bound, is 0.4.

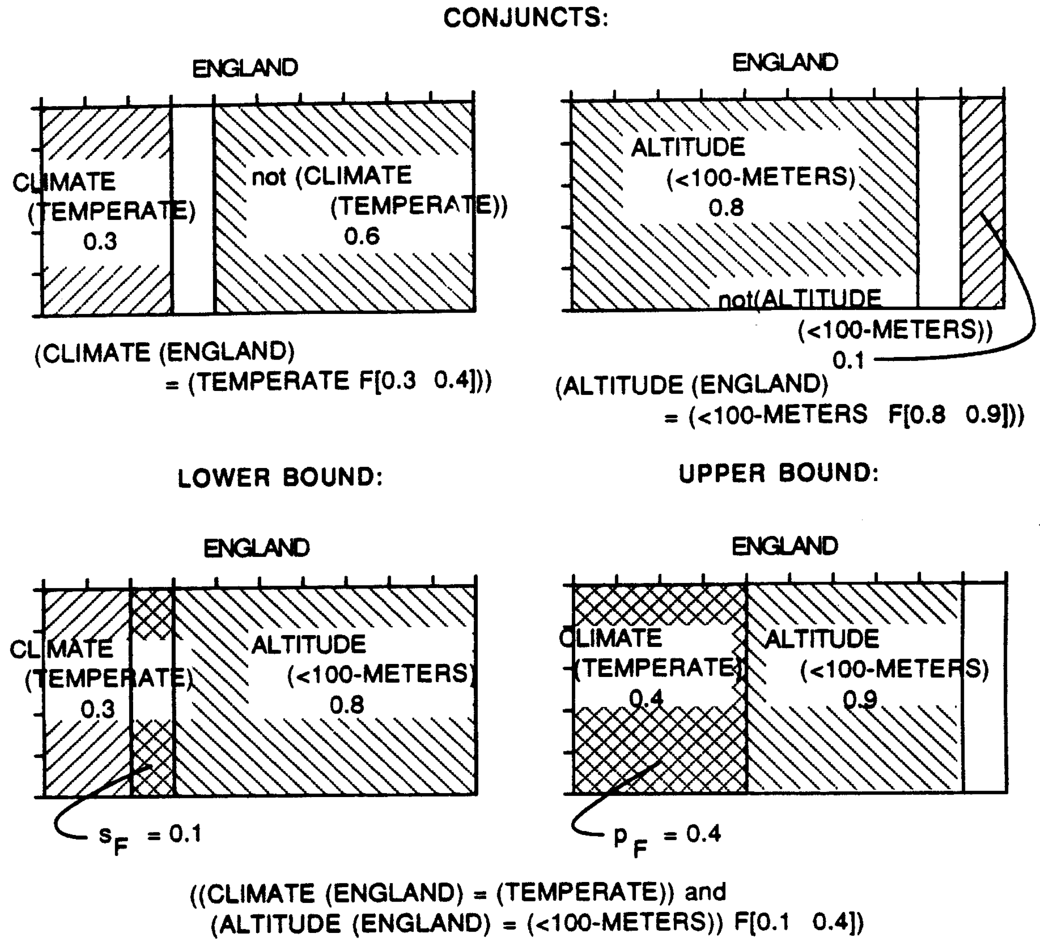

CONJUNCTS:
ENGLAND
CLIMATE (TEMPERATE) 0.3
not (CLIMATE (TEMPERATE)) 0.6
(CLIMATE (ENGLAND) = (TEMPERATE F[0.3 0.4]))
ENGLAND
ALTITUDE (<100-METERS) 0.8
not(ALTITUDE (<100-METERS)) 0.1
(ALTITUDE (ENGLAND) = (<100-METERS F[0.8 0.9]))
LOWER BOUND:
ENGLAND
CLIMATE (TEMPERATE) 0.3
ALTITUDE (<100-METERS) 0.8
s_F = 0.1
UPPER BOUND:
ENGLAND
CLIMATE (TEMPERATE) 0.4
ALTITUDE (<100-METERS) 0.9
p_F = 0.4
((CLIMATE (ENGLAND) = (TEMPERATE)) and
(ALTITUDE (ENGLAND) = (<100-METERS)) F[0.1 0.4])

Figure 34: Graphic Representation of Bounds on Conjunction of Frequency Weights (1st Order).

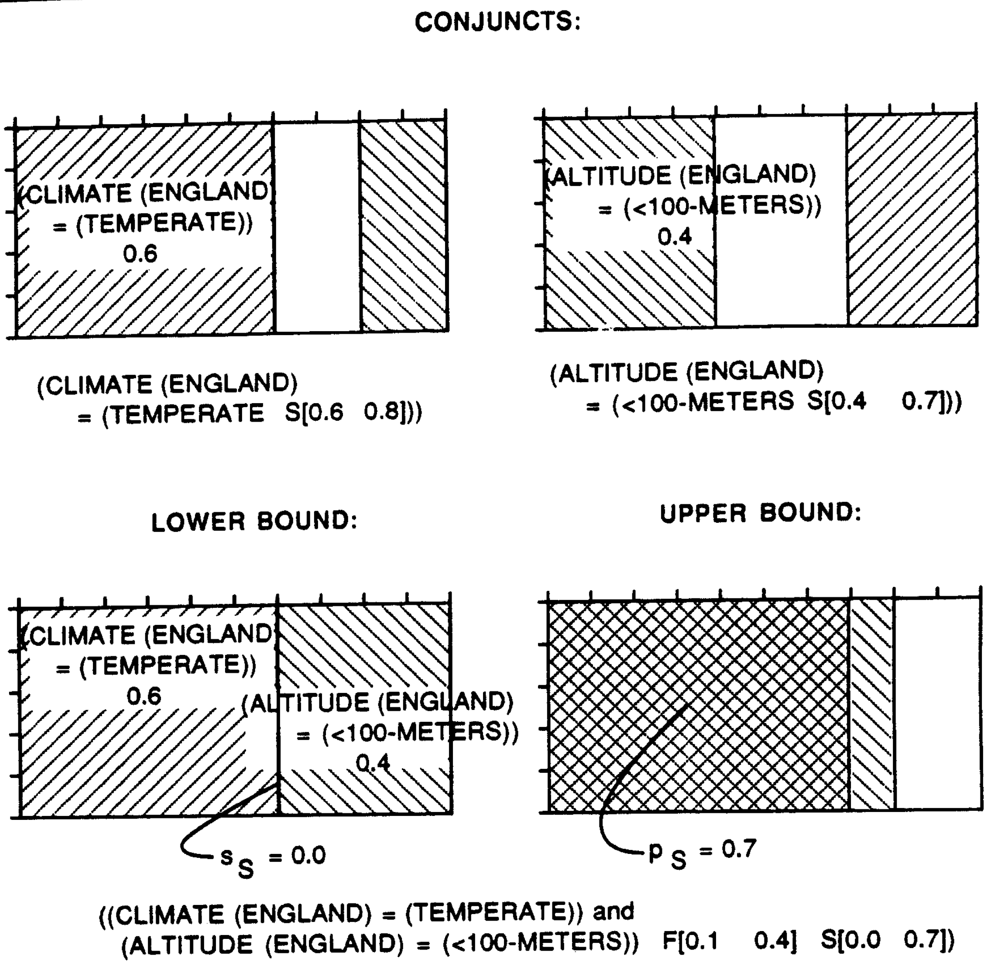

CONJUNCTS:
(CLIMATE (ENGLAND) = (TEMPERATE)) 0.6
(ALTITUDE (ENGLAND) = (<100-METERS)) 0.4
(CLIMATE (ENGLAND) = (TEMPERATE S[0.6 0.8]))
(ALTITUDE (ENGLAND) = (<100-METERS S[0.4 0.7]))
LOWER BOUND:
UPPER BOUND:
(CLIMATE (ENGLAND) = (TEMPERATE)) 0.6
(ALTITUDE (ENGLAND) = (<100-METERS)) 0.4
s S = 0.0
p S = 0.7
((CLIMATE (ENGLAND) = (TEMPERATE)) and (ALTITUDE (ENGLAND) = (<100-METERS)) F[0.1 0.4] S[0.0 0.7])

Figure 35: Graphic Representation of Bounds on Conjunction of Support Weights (2nd Order).

6.1.3. Disjunction

In addition to conjunction disjunctive operations are often necessary for grouping information and reasoning with it. As with both negation and conjunction, the first and second order disjunction are calculated identically. The lower bound is determined by intersecting the lower bounds of the disjuncts as much as possible. The upper bound is found by assuming the minimal possible intersection between the disjuncts, thus minimizing the uncovered area. The equations describing these calculations are:

$$DATUM \equiv D_1 \vee D_2 \vee \cdots \vee D_n \tag{6}$$

$$s_x(DATUM) \geq MAX(s_x(D_i)) \quad i = 1 \ldots n \tag{7}$$

$$p_x(DATUM) \leq MIN(\sum_{i=1}^{n} p_x(D_i) \ , \ 1) \tag{8}$$

Figure 36 illustrates the disjoining of the frequency intervals for two facts. Figure 37 illustrates the same for support values.

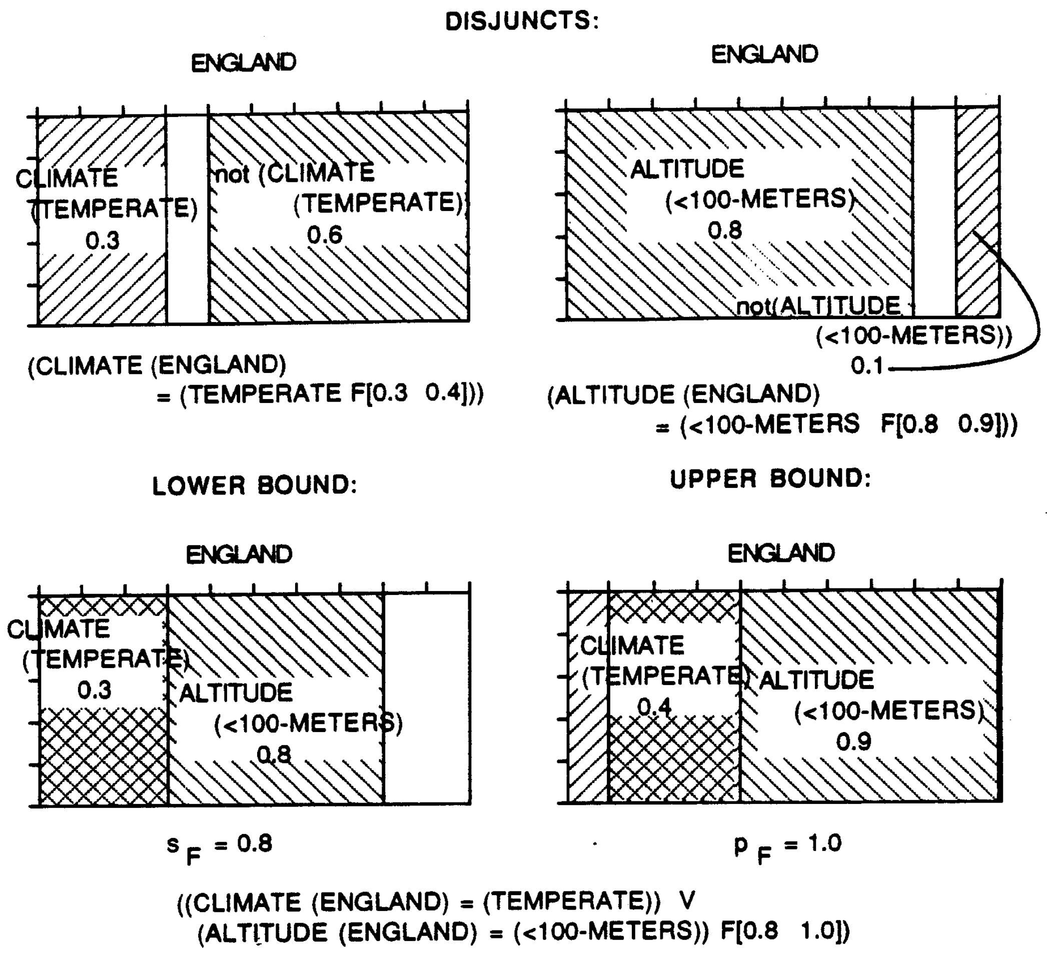
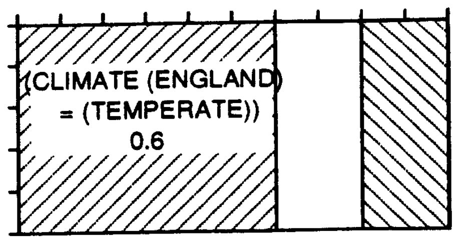
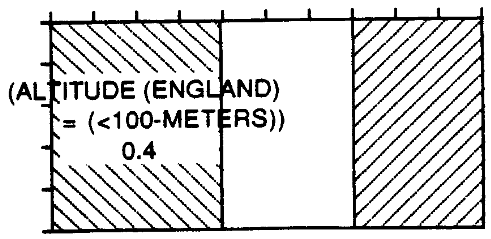

DISJUNCTS:
ENGLAND
CLIMATE (TEMPERATE) 0.3
not (CLIMATE (TEMPERATE)) 0.6
(CLIMATE (ENGLAND) = (TEMPERATE F[0.3 0.4]))
ENGLAND
ALTITUDE (<100-METERS) 0.8
not(ALTITUDE (<100-METERS)) 0.1
(ALTITUDE (ENGLAND) = (<100-METERS F[0.8 0.9]))
LOWER BOUND:
ENGLAND
CLIMATE (TEMPERATE) 0.3
ALTITUDE (<100-METERS) 0.8
s F = 0.8
UPPER BOUND:
ENGLAND
CLIMATE (TEMPERATE) 0.4
ALTITUDE (<100-METERS) 0.9
p F = 1.0
((CLIMATE (ENGLAND) = (TEMPERATE)) V
(ALTITUDE (ENGLAND) = (<100-METERS)) F[0.8 1.0])

Figure 36: Graphic Representation of Bounds on Disjunction of Frequency Weights (1st Order).

DISJUNCTS:



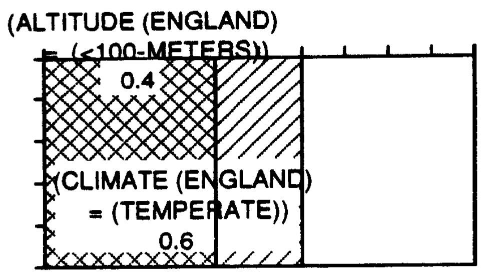


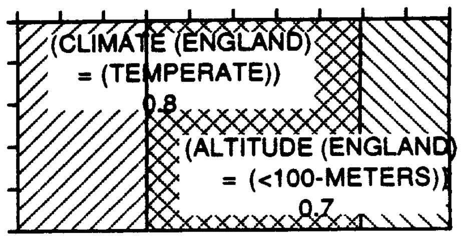
(CLIMATE (ENGLAND)
= (TEMPERATE S[0.6 0.8]))

(ALTITUDE (ENGLAND)
= (<100-METERS S[0.4 0.7]))

LOWER BOUND:

UPPER BOUND:





$s_S = 0.6$

$p_S = 1.0$

((CLIMATE (ENGLAND) = (TEMPERATE)) V
(ALTITUDE (ENGLAND) = (<100-METERS)) F[0.8 1.0] S[0.6 1.0])

Figure 37: Graphic Representation of Bounds on Disjunction of Support Weights (2nd Order).

6.1.4. Implication

Knowing the frequency weights associated with the antecedents in an implication and the likelihood weight for the implication itself allows a frequency interval for the consequent to be determined. The first step is to compute the frequency value for the antecedents as a whole, using the conjunction and disjunction equations as necessary. To use the likelihood weight associated with an implication recall that the likelihood intervals are lower and upper bounds on the conditional probability P(Consequent | Antecedents). The lower bound on the frequency of the consequent is determined by using the lower bound of both the frequency of the antecedent and likelihood (eq. 11). The upper bound is determined by first finding the lower bound on the antecedent implying the negation of the consequent (eq. 12). The inverse of this is the upper bound on the implication of the consequent. The equations are as follows:

$$(\rightarrow \mathit{ANTECEDENTS}\ \mathit{CONSEQUENT}\ L[s\ \ p]) \tag{9}$$

$$\mathit{ANTECEDENTS}\ F[s\ \ p] \tag{10}$$

$$s_f(\mathit{CONSEQUENT}) \geq s_f(\mathit{ANTECEDENTS}) \cdot s_l(\rightarrow \mathit{ANTECEDENTS}\ \mathit{CONSEQUENT}) \tag{11}$$

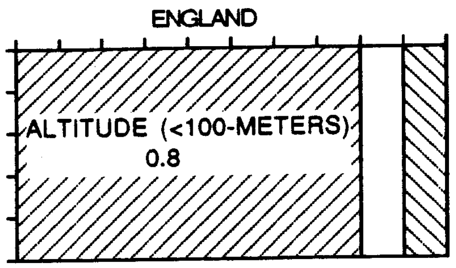
$$s_f(\neg \mathit{CONSEQUENT}) \geq s_f(\mathit{ANTECEDENTS}) \cdot (1 - p_l(\rightarrow \mathit{ANTECEDENTS}\ \mathit{CONSEQUENT})) \tag{12}$$

$$p_f(\mathit{CONSEQUENT}) \leq 1 - s_f(\neg \mathit{CONSEQUENT}) \tag{13}$$

Figure 38 illustrates graphically how the bounds are determined. In the figure the lower bound on the antecedent being true is 0.8. At least 0.8 (the lower bound on the likelihood of the implication holding) of that 0.8 the consequent is true thus the lower bound is 0.64 (0.8 x 0.8). To calculate the upper bound first the lower bound on the antecedent implying the negation of the consequent is determined: 1 - 0.9 = 0.1. The lower bound on the negation of the consequent holding is the product of this and the lower bound on the antecedent: 0.1 x 0.8 = 0.08. This is the inverse of the upper bound on the consequent - the upper bound is 1 - 0.08 or 0.92. To look at this upper bound calculation another way consider that the frequency of the antecedent is the lower bound 0.8. Using the upper bound on the implication - 0.9 - the frequency of the consequent is calculated to be 0.72. However, using 0.8 as the frequency of the antecedent means that 0.2 is the frequency of the negation of the antecedent being true. Since nothing is known which constrains the consequent from coexisting-existing with the negation of the antecedent assume that the consequent holds in that 0.2 and the upper bound is 0.72 + 0.2 = 0.92.

Appendix D illustrates the implication calculations for an example implication inference.

ANTECEDENT:

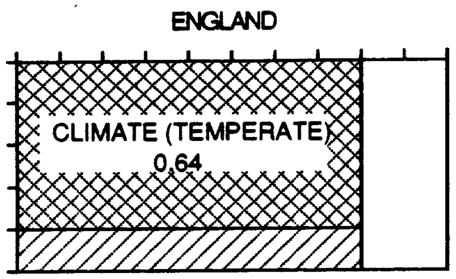


(ALTITUDE (ENGLAND) =
(<100-METERS F[0.8 0.9]))

IMPLICATION:

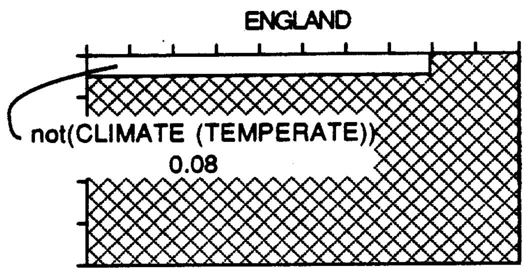
((ALTITUDE (?X) = (<100-METERS))
$\longrightarrow$ (CLIMATE (?X) = (TEMPERATE))
L[0.8 0.9])

LOWER BOUND:



$s_F = 0.64$

UPPER BOUND:



$p_F = 0.92$

(CLIMATE (ENGLAND) = (TEMPERATE F[0.64 0.92]))

Figure 38: Graphic Representation of Bounds on the Implication of Frequency Weights (1st Order).

The support interval for the consequent in an implication is determined by conjoining the support weights for the antecedents and implication using equations 4 and 5. This is clear when one considers that a consequent logically follows from knowing the antecedents and the implication (antecedents $\rightarrow$ consequents).

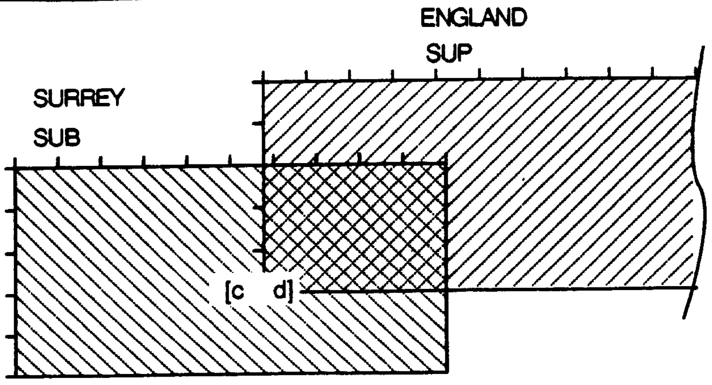
6.1.5. Dominance

Many of the inferences in PRS involve transferring a property which an object at one level of abstraction has to an object at a different level. The two objects are related through a generalization or specialization inference which has a dominance or typicality weight associated with it. The property, or referent, which one of the objects has and is to be transferred through the generalization or specialization inference to the second object has a frequency interval attached to it. The purpose of the dominance equations is to calculate a frequency relating the referent to the second object. To illustrate this propagation consider the inheritance of a referent associated with a superset by one of its subsets (an argument-based generalization inference operation).[16] What is being computed is the frequency of a fact (DES (SUB) = (REF)) (that is, the probability of DES(REF) given SUB or $P(DES(REF)|SUB)$) if it is known that (part-of (SUB) = (SUP)) and (DES (SUP) = (REF)). In order to present the equations in general terms the variables DES, INV-DES, ARG, REF, SUB, and SUP are used. They stand for DEScriptor, INVerse-DEScriptor, REFerent, ARGument, SUBset, and SUPerset. To map this to a concrete example DES = climate, INV-DES = country, ARG = SUB = Surrey, SUP = England, and REF = temperate. Thus what is being computed is the frequency of Surrey (ARG, SUB) having a temperate climate (DES(REF)) (that is, compute P(climate(temperate) | Surrey)) if it is known that Surrey is a part of England (SUP) and that the climate of England is temperate.

Figure 39 shows the information which the system knows prior to propagating the uncertainty. In the figure the lower case letters are variables for the lower and upper bounds on the weights. The location of the bracketed weights in the diagrams indicate what it actually refers to: In figure 39 the location of [c d] indicates that bounds on the frequency of the superset in the subset are known, not vice-versa (i.e., the bounds on Surrey being part of England not the bounds on England which is also Surrey).[17] Likewise the location of [e f] indicates the bounds on SUP having DES(REF) and not the bounds on things which are DES(REF) being SUP (i.e., the bounds on England having a temperate climate not the bounds on something which has a temperate climate being England).

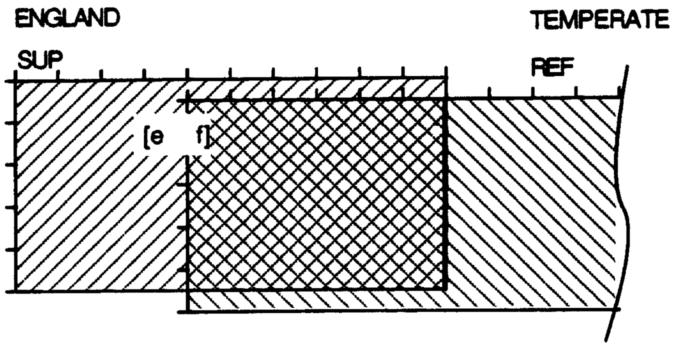
[16]Recall the discussion in the beginning of Chapter 4 regarding the naming of inference operations. As discussed there this thesis uses the inference names to refer to the inference operation necessary to move from unknown to known data not vice-versa.

[17]The diagram does not exactly fit the example since all of Surrey (SUB) is in reality within England (SUP), thus making [c d] = [1.0 1.0]. This total subsumption is not necessarily the case though, evident, for example, in the consideration of the sets such as Ford cars and sports cars.



(part-of (SURREY) = (ENGLAND D[c d]))

(part-of (SUB) = (SUP D[c d]))

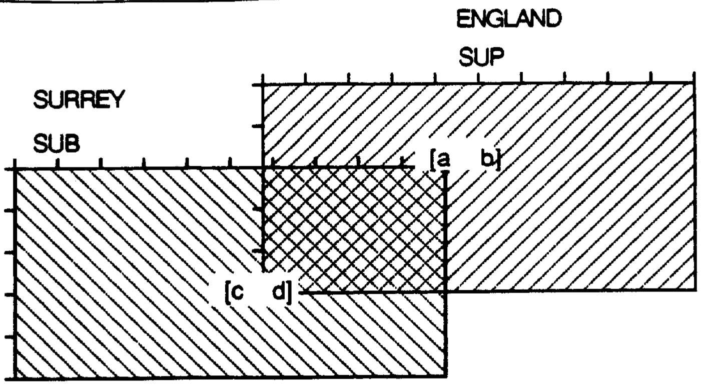


(CLIMATE (ENGLAND) = (TEMPERATE F[e f]))

(DES (SUP) = (REF F[e f]))

Figure 39: Graphic Representation of Minimum Initial Information for Calculation of Set/Subset Bounds (1st Order).

Note that in figure 39 the size of SUP and REF in relation to SUB and SUP respectively cannot be determined since the inverse probabilities are unknown. In order to fully specify the relationship two additional facts and their weights are needed. The fully constrained situation is shown in figure 40. Although more precise inferences can be made if all four facts are known, uncertainty intervals can be calculated with just the two original facts if the other two are unknown. In this case the default of [0.0 1.0] - total uncertainty- is used for the unknown weights.

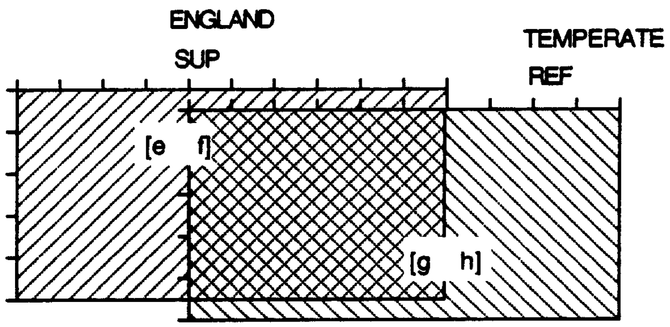


(part-of (SURREY) = (ENGLAND D[c d]))

(part-of (SUB) = (SUP D[c d]))

(has-part (ENGLAND) = (SURREY D[a b]))

(has-part (SUP) = (SUB D[a b]))



(CLIMATE (ENGLAND) = (TEMPERATE F[e f]))

(DES (SUP) = (REF F[e f]))

(COUNTRY (TEMPERATE) = (ENGLAND F[g h]))

(INV-DES (REF) = (SUP F[g h]))

Figure 40: Graphic Representation of Maximum Initial Information for Calculation of Set/Subset Bounds (1st Order).

Equations 14 through 17 list the information which is known from the first order weights. The information which may not be known is marked with an asterisk.[18]

$$(part\text{-}of\,(SUB) = (SUP \quad D[c \quad d])) \;;\; P(part\text{-}of(SUP)\,|\,SUB) \;=\; [a \quad b] \tag{14}$$

$$(has\text{-}part\,(SUP) = (SUB \quad D[a \quad b])) \;;\; P(has\text{-}part(SUB)\,|\,SUP) \;=\; [c \quad d]^{*} \tag{15}$$

$$(des\,(SUP) = (REF \quad F[e \quad f])) \;;\; P(des(REF)\,|\,SUP) \;=\; [e \quad f] \tag{16}$$

$$(inv\text{-}des\,(REF) = (SUP \quad F[g \quad h])) \;;\; P(inv\text{-}des(SUP)\,|\,REF) \;=\; [g \quad h]^{*} \tag{17}$$

In an argument-based generalization inference there is a property ((des(REF)) associated with a superset (SUP) which is inherited by a subset (SUB). A set of calculations can be done to determine both the lower and upper bounds on the frequency of des(REF) in SUB. The lower bound is determined by the following steps:[19] (Note: The left hand side of the equations refers to the size of the indicated sets. Also, two values separated by a vertical bar indicate that using either value might produce the lowest bound - however there is no way of determining which value to use without doing the actual calculations).

$$SUB \;=\; \frac{c|d}{a} \tag{18}$$

$$SUP\ not\ in\ SUB \;=\; 1 \;-\; c|d \tag{19}$$

$$SUP\ with\ REF \;=\; MAX(e \;-\; SUP\ not\ in\ SUB \;,\; 0) \tag{20}$$

$$LOWER\ \ BOUND\ \ P(REF\,|\,SUB) \;=\; \frac{SUB\ with\ REF}{SUB} \tag{21}$$

Equation 18 determines the size of SUB when the size of SUP is taken to be 1.0 (normalizes SUB). To get the lowest possible bound this should be as large as possible. Eq. 19 finds the amount of SUP which is not SUB - this should also be as large as possible since the objective is to find the lowest bound. Using the lower bound of REF in SUP eq. 20 determines how much REF in SUP, if any, necessarily overlaps into the intersection between SUB and SUP. If this value is less than zero the lower bound is 0.0. Eq. 21 renormalizes the amount of REF in SUB for

[18]The equations are presented in terms of an argument-based generalization inference. However, they are used for any of the four hierarchical inferences. The mapping between the inputs to the equations and each inference type is shown in figure 41, following the presentation of the equations.

[19]This could all be presented in a single equation but it is clearer if it is illustrated in steps

a SUB size of 1.0. In order to find the lowest possible bound the calculations need to be done twice, first using the c value and then the d value. This is because both eqs. 18 and 19 want to set their left hand sides as large as possible. For eq. 18 this would be by using the d value, and for 19 the c value. As the actual frequency which the interval [c d] constrains cannot be two values at once the calculations must be done twice to determine what the most conservative lower bound is.

The upper bound is calculated by the following steps:

$$SUB = \frac{c|d}{a|b} \qquad (22)$$

$$SUB\ not\ in\ SUP = SUB - (a|b \cdot SUB) \qquad (23)$$

$$REF = \frac{e|f}{g|h} \qquad (24)$$

$$REF\ not\ in\ SUP = REF - (g|h \cdot REF) \qquad (25)$$

$$SUB\ in\ SUP\ not\ with\ REF = \mathrm{MAX}(c|d - e|f\ ,\ 0) \qquad (26)$$

$$SUB\ not\ in\ SUP\ not\ with\ REF = \mathrm{MAX}(SUB\ not\ in\ SUP - REF\ not\ in\ SUP\ ,\ 0) \qquad (27)$$

$$SUB\ not\ REF = SUB\ in\ SUP\ not\ with\ REF + SUB\ not\ in\ SUP\ not\ with\ REF \qquad (28)$$

$$SUB\ with\ REF = SUB - SUB\ not\ REF \qquad (29)$$

$$UPPER\ \ BOUND\ \ \bar{P}(REF|SUB) = \frac{SUB\ with\ REF}{SUB} \qquad (30)$$

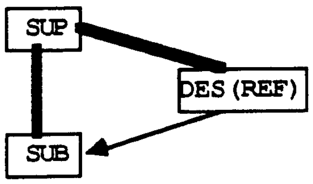
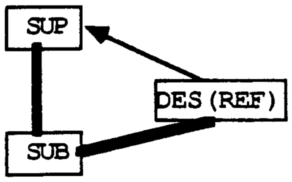
Clearly the upper bound calculation is more involved than that for the lower bound. This is because in addition to considering the portion within SUP that SUB and REF can overlap the possible intersection outside of SUP must be considered. What further complicates the procedure is that both SUB in SUP not with REF (eq. 26) and SUB not in SUP not with REF (eq. 27) should be minimized and it cannot be determined a priori which combination of the interval bounds will do this and thus produce the highest upper bound.

Although the calculation steps detailed above were presented in terms of an argument-based generalization inference they are applicable to all set/subset inferences. Since the equations themselves can be interpreted to be concerned only with set relationships and not with the fact that one set is a superset of the other (the subset/superset relations come from the semantics attached to each type of weight and the actual weights themselves) it is straightforward to map the equations to each inference type. Figure 41 correlates the inputs to the equations to the information for both argument and referent based inferences. Recalling the semantics discussed in Chapter 3 (and summarized in Appendix A), in the figure P(has-part(SUB)|SUP) corresponds to has-part or has-type relations and P(part-of(SUP)|SUB) to part-of or type-of relations. P(des(REF)|?X) maps to non-hierarchical facts, corresponding to the probability of some argument and descriptor having a particular referent, while P(?X|des(REF)) corresponds to the inverse.

In the diagrams in figure 41 the thick lines refer to information that is known and the thin lines with arrowheads refer to inferences being made. Thus the diagram in the upper left, for an argument-based generalization inference operation, DES(REF) is known for SUP and is being inferred for SUB. As an example, the climate of England (SUP) is known to be temperate (DES(REF)) and this is used to infer that the climate of Surrey (SUB) is likewise temperate. A similar example for an argument-based specialization (upper right diagram) would be that the climate of Surrey (again SUB) is known to be temperate (DES(REF)) and this is being used to infer that the climate of England (SUP) is temperate.

Similar examples can be given for the referent-based inferences. In the lower left diagram (referent-based generalization operation) the temperature of Antarctica (SET) might be known to be cold (DES(REF_{SUP})) and because cold is a generalization of frigid (DES(REF_{SUB})) an inference can be made that the temperature of Antarctica is frigid. Likewise for a referent-based specialization inference operation (lower right diagram) an example would be that since the temperature of Antarctica (SET) is frigid (DES(REF_{SUB})) it could be inferred that it is cold (DES(REF_{SUP})).

ARGUMENT-BASED INFERENCE OPERATIONS

	GENERALIZATION	SPECIALIZATION
		
Query:	$P(DES(REF) \mid SUB)$	$P(DES(REF) \mid SUP)$
[a b]	$P(SUP \mid SUB)$	$P(SUB \mid SUP)$
[c d]	$P(SUB \mid SUP)$	$P(SUP \mid SUB)$
[e f]	$P(DES(REF) \mid SUP)$	$P(DES(REF) \mid SUB)$
[g h]	$P(SUP \mid DES(REF))$	$P(SUB \mid DES(REF))$

REFERENT-BASED INFERENCE OPERATIONS

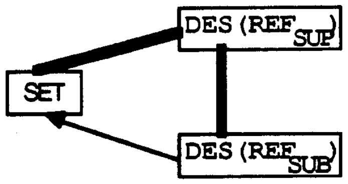
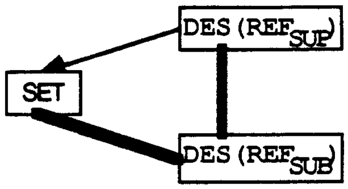
	GENERALIZATION	SPECIALIZATION
		
Query:	$P(DES(REF_{SUB}) \mid SET)$	$P(DES(REF_{SUP}) \mid SET)$
[a b]	$P(DES(REF_{SUP}) \mid SET)$	$P(DES(REF_{SUB}) \mid SET)$
[c d]	$P(SET \mid DES(REF_{SUP}))$	$P(SET \mid DES(REF_{SUB}))$
[e f]	$P(DES(REF_{SUB}) \mid DES(REF_{SUP}))$	$P(DES(REF_{SUP}) \mid DES(REF_{SUB}))$
[g h]	$P(DES(REF_{SUP}) \mid DES(REF_{SUB}))$	$P(DES(REF_{SUB}) \mid DES(REF_{SUP}))$

Figure 41: Correlation of known information to Set/Subset Propagation Equations.

Figure 42 lists some examples of bound calculations. This illustrates the relationship between the uncertainty weights for several combinations. The first four columns indicate various combinations of the known information and the last indicates the calculated frequency range of a subset having a referent value (argument-based generalization operations) or a superset having a referent value (argument-based specialization operations). (In order to fit the column headings Descriptor is abbreviated as just D instead of the usual DES.) Note the general increase in the magnitude of the resulting uncertainty weights as the magnitude of the contributing weights increases.

To give a better feel for how the calculations are done figures 43 and 44 graphically illustrate some examples of bound calculations for an argument-based generalization inference operation with varying degrees of known information. In these examples the facts (part-of (SUP) = (SUB)), (has-part (SUP) = (REF)), (DES (SUP) = (REF)), and (INV-DES (REF) = (SUP) are known with varying uncertainty weights (indicated by their degree of intersection). A bound on (DES (SUB) = (REF)) is being calculated.[20] Each row in the diagrams corresponds to various situations in which certain of the constraining weights are known or unknown. In the diagrams boxes with four exact sides indicate completely constrained situations while diagrams including boxes with continuation symbols are underconstrained. Under each diagram the bound is given, with many being either 0 or 1. "=X" indicates that a bound is calculable using equations 14 - 30 - it doesn't necessarily fall to either 0 or 1. The diagrams are meant to convey a general feel for how the sets relate and do not necessarily exactly depict what the bounds are in every possible instantiation of each situation.

In figure 43 A) the lower bound is 0 - it is possible for SUB to be contained in the portion of SUP which does not contain REF. The upper bound is determined by maximally "covering" the portion of SUB which is in SUP with REF. The exact bound is calculated depending on the percent of SUB which intersects with REF.

In B) the sizes of SUB and REF dictate that there must be some minimum intersection between SUB and REF inside of SUP and thus a lower bound can be calculated. The upper bound is determined similarly to in A), only additionally assuming that the portion SUB not in SUP is covered by REF.

C) illustrates a situation in which the relationship between SUP and REF is fully constrained but the relationship between SUP and SUB is not. To determine the smallest possible lower

[20]Again, this can be mapped to the same example used previously in this section: DES = climate, INV-DES = country, SUB = Surrey, SUP = England, and REF = temperate. What is being computed is the frequency of Surrey (SUB) having a temperate climate (DES(REF)) (that is, compute P (climate(temperate) | Surrey)) if it is known that Surrey is a part of England (SUP) and that the climate of England is temperate.

ARGUMENT-BASED GENERALIZATION OPERATIONS				
P(SUP|SUB)	P(SUB|SUP)	P(SUP|D(REF))	P(D(REF)|SUP)	P(D(REF)|SUB)
1.0 1.0	0.3 0.3	1.0 1.0	0.4 0.4	0.0 1.0
	0.5 0.5		0.4 0.4	0.0 0.8
	0.5 0.5		0.7 0.7	0.4 1.0
	0.7 0.7		0.7 0.7	0.58 1.0
0.5 0.5	0.3 0.3	1.0 1.0	0.4 0.4	0.0 0.5
	0.5 0.5		0.4 0.4	0.0 0.4
	0.5 0.5		0.7 0.7	0.2 0.5
	0.7 0.7		0.7 0.7	0.29 0.5
1.0 1.0	0.3 0.3	0.5 0.5	0.4 0.4	0.0 1.0
	0.5 0.5		0.4 0.4	0.0 0.8
	0.5 0.5		0.7 0.7	0.4 1.0
	0.7 0.7		0.7 0.7	0.58 1.0
ARGUMENT-BASED SPECIALIZATION OPERATIONS				
P(SUP|SUB)	P(SUB|SUP)	P(SUP|D(REF))	P(D(REF)|SUP)	P(D(REF)|SUP)
1.0 1.0	0.3 0.3	1.0 1.0	0.4 0.4	0.12 0.12
	0.5 0.5		0.4 0.4	0.2 0.2
	0.5 0.5		0.7 0.7	0.35 0.35
	0.7 0.7		0.7 0.7	0.49 0.49
0.5 0.5	0.3 0.3	1.0 1.0	0.4 0.4	0.0 0.24
	0.5 0.5		0.4 0.4	0.0 0.5
	0.5 0.5		0.7 0.7	0.2 0.5
	0.7 0.7		0.7 0.7	0.28 0.7
1.0 1.0	0.3 0.3	0.5 0.5	0.4 0.4	0.12 0.24
	0.5 0.5		0.4 0.4	0.2 0.4
	0.5 0.5		0.7 0.7	0.35 0.7
	0.7 0.7		0.7 0.7	0.49 0.79

Figure 42: Bound Calculation Examples.

bound consider the situation where SUB is very large (i.e., the unknown weight, P(SUP|SUB), is very small). As a result even though there is necessarily some overlap between SUB and REF inside of SUP it is possible that it (the overlap) is a trivially small portion of the whole of SUB and thus the lower bound is essentially 0. The upper bound is determined by assuming that the size of SUB outside of SUP equals the size of REF outside of SUP. The bound is calculated by assuming that those two areas intersect completely and that SUB and REF intersect as much as possible within SUP.

In figure 44 D) the size of SUP in SUB is known but not the inverse. From just this information the percentages of SUB which are inside and outside of SUP are known but not the exact sizes. (In the diagram this is illustrated by the doubly unconstrained SUB box. As the size expands or contracts in one direction it must do the same in the other, remaining in line with the percentage of SUP in SUB dictated by the known weight P(SUP|SUB)). To determine the smallest possible lower bound consider the situation where SUP is much larger than SUB (i.e., the unknown weight, P(SUB|SUP), is very small). In this case there will be some area of SUP where SUB can be present without REF and the lower bound is 0. (If REF had completely covered SUP this would be not be the case). The upper bound is 1 for a similar reason. Again, SUP may be much larger than SUB. Thus REF will also be larger than SUB and all of SUB, both inside and outside of SUP can be covered.

E) illustrates a situation similar to D). The size of SUP which is in REF is known but not the inverse. Thus the percentages of REF which are inside and outside of SUP are known but not the absolute size of REF in relation to SUB and SUP. It is possible that the amount of REF in SUP is trivially small and does not intersect at all with SUB, making the lower bound 0. The upper bound is determined by assuming that the unknown weight P(REF|SUB) is 1 and thus any SUB within SUP intersects with REF. With this assumption the size of REF in relation to SUB and SUP can be calculated and assuming maximal intersection between SUP and REF outside of SUP the upper bound can be calculated.

In F) since the size of both SUB and REF in SUP are known a lower bound can be calculated. As with the other examples, the lowest possible bound is when there is no intersection of SUB and REF outside of SUP. The upper bound can be determined by maximally covering SUB in SUP with REF and assuming that the size of REF allows any portion of SUB outside of SUP to also be covered.

The situations detailed illustrate situations in which none or one of the uncertainty weights is unknown. In cases where two or more are unknown the lower and upper bounds default to their poles, 0 and 1.

The support weight for set/subset relations is, like the support for implications, computed by conjoining the support values for each fact using equations 4 and 5. In this case though, a globally defined interval which indicates the support for the set to set transfer must also be

? = P(DES(REF)|SUB)	(e.g., (des (SUB) = (REF)))
[a b] = P(SUP|SUB)	(e.g., (part-of (SUB) = (SUP D[a b])))
[c d] = P(SUB|SUP)	(e.g., (has-part (SUP) = (SUB D[c d])))
[e f] = P(DES(REF)|SUP)	(e.g., (des (SUP) = (REF F[e f])))
[g h] = P(SUP|DES(REF))	(e.g., (inv-des (REF) = (SUP F[g h])))

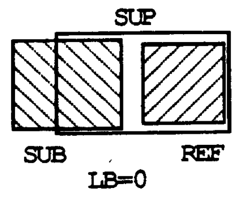
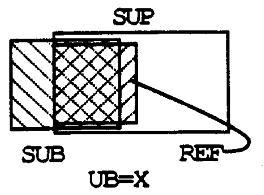
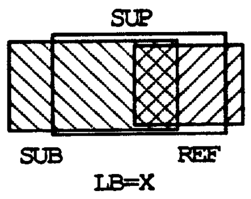
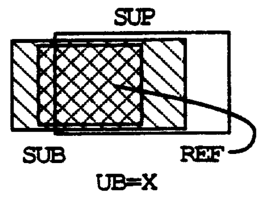
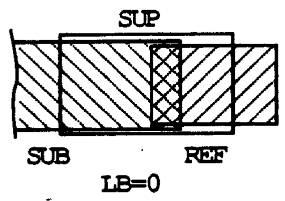
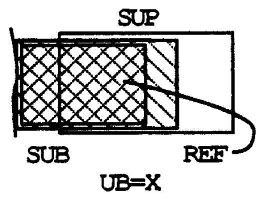
	LOWER BOUND	UPPER BOUND
A [a b]=x [c d]=x [e f]=x [g h]=x		
B [a b]=x [c d]=x [e f]=x [g h]=x		
C [a b]=? [c d]=x [e f]=x [g h]=x		

Figure 43: Examples of Bound Calculations.

conjoined. This is necessary since set/subset transfer, unlike implication, is not a principle of formal logic, and therefore is not necessarily true. Figure 45 lists these variables for for argument-based generalization inferences as well as each of the other types.

Appendix D illustrates the calculations for an example of an argument-based generalization inference.

? = P(DES(REF)|SUB)	(e.g., (des (SUB) = (REF)))
[a b] = P(SUP|SUB)	(e.g., (part-of (SUB) = (SUP D[a b])))
[c d] = P(SUB|SUP)	(e.g., (has-part (SUP) = (SUB D[c d])))
[e f] = P(DES(REF)|SUP)	(e.g., (des (SUP) = (REF F[e f])))
[g h] = P(SUP|DES(REF))	(e.g., (inv-des (REF) = (SUP F[g h])))

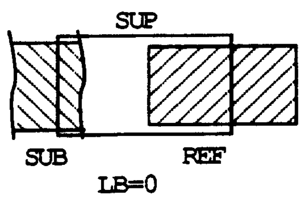
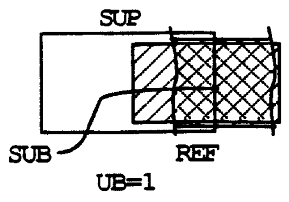
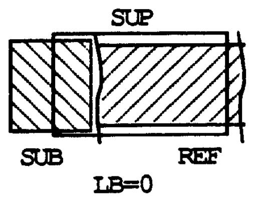
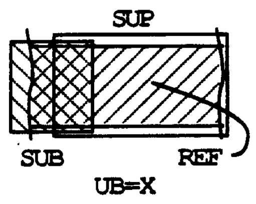
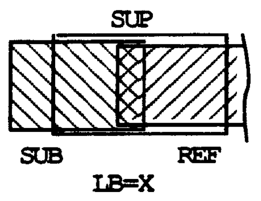
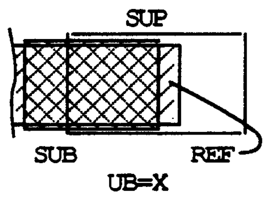
	LOWER BOUND	UPPER BOUND
D [a b]=x [c d]=? [e f]=x [g h]=x		
E [a b]=x [c d]=x [e f]=? [g h]=x		
F [a b]=x [c d]=x [e f]=x [g h]=?		

Figure 44: Examples of Bound Calculations (continued).

OPERATION	PARAMETER	DEFAULT VALUE
Argument-Based Generalization	*arg-gen-supp*	S[1.0 1.0]
Argument-Based Specialization	*arg-spec-supp*	S[1.0 1.0]
Referent-Based Generalization	*ref-gen-supp*	S[1.0 1.0]
Referent-Based Specialization	*ref-spec-supp*	S[1.0 1.0]
Dependency	*depend-supp*	S[1.0 1.0]

Figure 45: Support Parameters for Inference Operations.

6.1.6. Typicality

As described in section 3.2.3 typicality weights are associated with the relation between a set and its subsets. It is similar to dominance yet allows stronger inferences to made in situations where the relative sizes of two sets in a hierarchy are disparate yet they share many of the same attributes.

Propagation of frequency values through typicality relations works the same as with dominance relations, which were discussed in the preceding section. The same equations are used, except typicality weights are used instead of dominance weights for defining the relationship between two arguments.

The support for typicality relations is also calculated the same as with set/subset relations. Again, as with dominance, since typicality is not a principle of formal logic a variable is defined which is the global support for typicality relations. It is set to a default of S[1.0 1.0].

6.1.7. Dependencies

The propagation of uncertainty in dependency-based inferences works by building an implication based upon the dependency and known facts and using it to infer a frequency for the unknown query. Equations 31 through 34 below give the information gained through gathering the facts used in dependency inferences.[21] Equations 35 and 36 perform the uncertainty

[21]Recall the description of dependency inferences in section 4.6. The mapping between the example presented in figure 26 in that section and the equations presented here is DES1 = climate, DES2 = latitude, ARG1 = Surrey, ARG2 = Holland, REF1 = temperate, and REF2 = >40-degrees.

propagation. Equation 35 calculates the lower bound on the implication involving the two descriptors and the values associated with the second argument. The lower likelihood bound on the dependence between the two descriptors is used to lower this result below what would be obtained by just using the facts associated with the second argument. Equation 36 calculates the upper bound, using the upper bound on of the likelihood of the dependence as the correction factor. Once the likelihood associated with the generated implication are determined the implication equations (equations 9 - 13) are used to calculate the frequency involved with the initial query.

$$(\leftarrow\rightarrow \ DES1 \ \ DES2 \ \ L[s \ \ p]) \tag{31}$$

$$(DES2\,(ARG1) = (REF2 \ \ F[s \ \ p])) \ \ ; \ \ P(DES2(REF2)\,|\,ARG1) = [a \ \ b] \tag{32}$$

$$(DES2\,(ARG2) = (REF2 \ \ F[s \ \ p])) \ \ ; \ \ P(DES2(REF2)\,|\,ARG2) = [c \ \ d] \tag{33}$$

$$(DES1\,(ARG1) = (REF1 \ \ F[s \ \ p])) \ \ ; \ \ P(DES1(REF1)\,|\,ARG2) = [e \ \ f] \tag{34}$$

$$s_L(\,(\rightarrow((DES2\,(ARG2) = (REF2))\,) \ \ (DES1\,(ARG2) = (REF1)))\,) \ = \ \frac{e}{c}\cdot s_L \tag{35}$$

$$p_L(\,(\rightarrow(\,(DES2\,(ARG2) = (REF2))\,)\,(DES1\,(ARG2) = (REF1)))\,) \ = \ 1-(1-\frac{f-1+c}{c})\cdot(1-p_L) \tag{36}$$

6.2. Combined Inferences

At the end of Chapter 4 two types of combined inferences were discuss[illegible] combining multiple inferences in one solution path and combining the evidence obtained from multiple solution paths into some total measure of the belief in the initial query. The first of these is straightforward using the equations developed in this chapter. Beginning with the fact found at the end of a solution path the result of each uncertainty calculation along the path is used in the next calculation. For example, consider if a fact A is being queried for and the implication B $\rightarrow$ A is known. If B is known we're all set but perhaps it is unknown. It may be the case, however, that a generalization of B is known and thus it could be used to calculate a belief in B. This in turn can be used in the implication and a belief in A can be calculated. Obviously, the longer the solution paths are (and thus the greater the number of calculation steps) the lower the belief that a solution path will likely bring to bear on the query.

The second aspect of combined inferences is when multiple sources of evidence are found relating to a query. In these situations the evidence which each source brings to bear on the query is summarized to arrive at the final belief in the query. The method used it to first break each piece of evidence into positive and negative components. Thus if a particular piece of evidence generates a frequency of [a b] with a support of [c d] for a query A, A is true with the weights F[0 a] S[c d] and not(A) is true with the weights F[0 1-b] S[c d]. (Recall the semantics of uncertainty ranges, section 3.2.) The next step is to combine these weights by subdividing each frequency interval as necessary and adding the support weights associated with that interval. An example will make this clear - consider figure 46. Note the support values for the combined evidence. For the positive frequency range 0.0 to 0.2 this is the sum of the support for both evidence sources since they both cover this range. For the range 0.2 to 0.6 the support is just the support for the first evidence source since that is the only one which covers that range. Since both evidence sources cover the negative range 0.0 to 0.1 the combined support is the sum of the individual supports. Although the figure only shows the combination of two evidence sources any number of sources can be combined using the same strategy.

Evidence Source 1:	F[0.6 0.9]	S[0.3 0.7]
positive:	F[0.0 0.6]	S[0.3 0.7]
negative:	F[0.0 0.1]	S[0.3 0.7]
Evidence Source 2:	F[0.2 0.9]	S[0.2 0.5]
positive:	F[0.0 0.2]	S[0.2 0.5]
negative:	F[0.0 0.1]	S[0.2 0.5]
Combined Evidence:		
positive:	F[0.0 0.2]	S[0.5 1.0]
	F[0.2 0.6]	S[0.3 0.7]
negative:	F[0.0 0.1]	S[0.5 1.0]

Figure 46: Example of Summing Multiple Sources of Evidence.

6.3. Qualitative Uncertainty

Qualitative uncertainty refers to transforming quantitative measures of uncertainty to qualitative terms. Thus when evidence for a query is found the resulting frequency and support values can be output in English as well as numerically. Without any deep analysis a number of terms were selected and used to transform frequency and support values. Figure 47 shows the translations currently used in PRS for outputting results.

Weight	Translation for Frequency	Translation for Support
0.0	none	none
0.0 - 0.33	some	weak
0.33 - 0.67	half	medium
0.67 - 1.0	most	strong
1.0	all	strong

Figure 47: Translations between Quantitative and Qualitative Uncertainty Measures.

7. SYSTEM OPERATION

This chapter describes how PRS operates, from the perspective of a user interacting with it. The first part builds upon previous chapters, describing more specifics about how PRS runs, and includes an annotated listing of PRS running. This is followed by a description of the Symbolics interface.[22]

PRS was developed in common lisp and is composed of 13 source files and approximately 3700 lines of code, not including the underlying window interface code.

7.1. Creating a Knowledge Base

The first step in running PRS is setting up a knowledge base. The components of a knowledge base are facts, rules, and dependencies. These are entered using the lisp functions *ADD-FACTS, MAKERULE,* and *MAKEDEPENDENCY.* Appendix B lists and briefly describes these as well as the other functions which a user may invoke. An example of how these functions are used is shown in figure 48. A knowledge base can be developed incrementally and there is no required order in which information must be added. Appendix C lists the complete knowledge base used for the climate examples. Note that the size of a knowledge base is limited only by the memory in a Symbolics machine and thus for any practical task there is no constraint on the number of facts, rules, and dependencies.

7.2. Running PRS

Once knowledge is entered into the lisp environment PRS can be run. Evaluating the function *RUN-PRS* initiates the system and prompts for a query. Taking this query as the starting point PRS uses the inference operations (Chapter 4) to transform it and find related evidence in the knowledge base. As described in Chapter 5 this reasoning process is controlled by a number of adjustable parameters.

As an example, below is a trace of PRS running. In this example the system is trying to determine if the climate of Surrey is temperate. PRS finds four separate pieces of evidence

[22]Appendix B summarizes the lisp functions for operating PRS.

```
(ADD-FACTS
  '((part-of (England) = (Europe          D[ 0.1 0.2 ]   S[ 0.6 0.9 ]))

    (climate (England) = (temperate       F[ 0.8 0.9 ]   S[ 0.7 0.9 ]))
    ))

(MAKERULE '(--> ((latitude (?x) = (>40-degrees)))
                 (climate (?x) = (temperate))          L[ 0.8 0.9 ]
                                                       S[ 1.0 1.0 ]))

(MAKEDEPENDENCY '(<--> climate latitude  L[ 0.5 0.8 ]  S[ 0.6 0.9 ]))
```

Figure 48: Knowledge File.

regarding this query. In order to make this example more helpful comments have been added to the actual system output (in italics). User input is indicated with bold capitals. All else is tracing and result information which the system outputs as it runs.

Load the climate Knowledge Base (Full listing in Appendix C). The climate KB contains facts, implications, and dependencies used to illustrate PRS working in a simple geographical domain.

```
Command: LOAD FILE CLIMATE

--------------------------Adding Facts-------------------------
```

Facts, implications, and dependencies are asserted into the environment:

```
 Asserting (PART-OF (ENGLAND) = (EUROPE))...
 Asserting (TYPE-OF (ENGLAND) = (COUNTRY))...
 Asserting (CLIMATE (ENGLAND) = (TEMPERATE))...
 .
 .
 .

 Making Rule (--> ((LATITUDE (?X) = (>-40-DEGREES)))
                  (CLIMATE (?X) = (TEMPERATE))) ...
 Asserting (PART-OF (ENGLAND) = (UNITED-KINGDOM))...
 Making Dependency (<--> CLIMATE LATITUDE) ...
 ... Done.
```

Initiate PRS session:

```
Command: (RUN-PRS)
```

```
Running PRS-QUERY system:
```

A user inputs an initial query to determine if the climate of Surrey is temperate:

```
Enter query: (climate (surrey) = (temperate))

Next Operation:   (1)
 (CHECK-QUERY '(CLIMATE (SURREY) = (TEMPERATE)))
    time = 0          path length = 0
    worth = 100       rel. time = 0
```

The fact (climate (Surrey) = (temperate)) is not in the KB so inference operations are added to the stack.

```
 Unknown...Adding stack elements:

 (GET-ARGUMENT-GENERALIZATIONS '(CLIMATE (SURREY) = (TEMPERATE)))
 (GET-ARGUMENT-SPECIALIZATIONS '(CLIMATE (SURREY) = (TEMPERATE)))
 (GET-REFERENT-GENERALIZATIONS '(CLIMATE (SURREY) = (TEMPERATE)))
 (GET-REFERENT-SPECIALIZATIONS '(CLIMATE (SURREY) = (TEMPERATE)))
 (BACKCHAIN '(CLIMATE (SURREY) = (TEMPERATE)))
 (GET-DEPENDENCIES '(CLIMATE (SURREY) = (TEMPERATE)))
```

The stack is sorted (see Chapter 5) and the next operation is popped off and executed. The implication operation has the highest worth (See Figure 29, Chapter 5).

```
Next Operation:   (2)
 (BACKCHAIN '(CLIMATE (SURREY) = (TEMPERATE)))
    time = 1          path length = 0
    worth = 95.0      rel. time = 1
```

Found an implication which implies that the climate of Surrey is temperate:

```
 Found (--> ((LATITUDE (?X) = (>40-DEGREES)))
            (CLIMATE (?X) = (TEMPERATE)))
 Adding to stack:
 (CHECK-QUERY '(LATITUDE (SURREY) = (>40-DEGREES)))
```

Now check to see if the antecedent of the implication is in known:

```
Next Operation:   (3)
 (CHECK-QUERY '(LATITUDE (SURREY) = (>40-DEGREES)))
    time = 2          path length = 2
    worth = 95.0      rel. time = 2
```

The antecedent is in the knowledge base:

```
 Found (LATITUDE (SURREY) = (>40-DEGREES))
```

Finding antecedent creates a solution path - one piece of evidence is found supporting the initial query. The system summarizes the inference path which led to this evidence and calculates the belief in the query based upon this evidence:

```
-------------------------- Evidence 1 ----------------------------
```

```
Found Evidence:
 It is known that
  (--> ((LATITUDE (?X) = (>40-DEGREES)))
       (CLIMATE (?X) = (TEMPERATE)))
       like : 0.80 .. 0.90                  supp: 1.00 .. 1.00
  (LATITUDE (SURREY) = (>40-DEGREES))
       freq : 0.80 .. 0.90                  supp: 0.80 .. 1.00
 On the basis of this evidence the input query is known with
     freq: 0.64 .. 0.92                     supp: 0.80 .. 1.00
```

Numbers are translated to qualitative terms (see Section 6.3):

```
 This translates to STRONG evidence that HALF to MOST of
  the CLIMATE of SURREY is TEMPERATE.

 Combining the 1 source of evidence the initial query,
  (CLIMATE (SURREY) = (TEMPERATE)), has the following belief:
Supporting Fact:
 freq: (0.00 0.64)  supp: (0.80 1.00)
Supporting Negation of Fact:
 freq: (0.00 0.08)  supp: (0.80 1.00)
```

```
------------------------------------------------------------------
```

Operation continues. Trying the next possible inference which was put on the stack above which is to find generalizations of the argument Surrey:

```
Next Operation:    (4)
 (GET-ARGUMENT-GENERALIZATIONS '(CLIMATE (SURREY) = (TEMPERATE)))
    time = 3         path length = 0
    worth = 90.0     rel. time = 3
```

Find that England is a generalization of Surrey:

```
Found (PART-OF (SURREY) = (ENGLAND))
Adding to stack:
(CHECK-QUERY '(CLIMATE (ENGLAND) = (TEMPERATE)))
```

Now check to see if England has a temperate climate:

```
Next Operation:    (5)
 (CHECK-QUERY '(CLIMATE (ENGLAND) = (TEMPERATE)))
    time = 4         path length = 2
    worth = 90.0     rel. time = 4
Found (CLIMATE (ENGLAND) = (TEMPERATE))
```

Since England does have a temperate climate an argument-based generalization inference has been completed. As with the first piece of evidence the path which produce the evidence is output and calculation of the degree of belief which this brings to bear on the initial query is calculated. Also the total belief in the initial query is calculated from the two pieces of evidence found so far:

```
-------------------------- Evidence 2 ----------------------------
```

```
Found Evidence:
 It is known that
  (PART-OF (SURREY) = (ENGLAND))
       dom: 1.00 .. 1.00  typ: 0.30 .. 0.80  supp: 1.00 .. 1.00
  (CLIMATE (ENGLAND) = (TEMPERATE))
       freq : 0.80 .. 0.90                   supp: 0.70 .. 0.90
 On the basis of this evidence the input query is known with
     freq: 0.33 .. 1.00                      supp: 0.70 .. 0.90
 This translates to STRONG evidence that SOME to ALL of
  the CLIMATE of SURREY is TEMPERATE.
```

Summarizes total evidence from both solution paths found so far...

```
 Combining the 2 sources of evidence the initial query,
  (CLIMATE (SURREY) = (TEMPERATE)), has the following belief:
Supporting Fact:
 freq: (0.00 0.33)  supp: (1.00 1.00)
 freq: (0.33 0.64)  supp: (0.80 1.00)
Supporting Negation of Fact:
 freq: (0.00 0.08)  supp: (0.80 1.00)
```

Operation continues. Now look for generalizations of the referent temperate:

```
Next Operation:   (6)
 (GET-REFERENT-GENERALIZATIONS '(CLIMATE (SURREY) = (TEMPERATE)))
   time = 7          path length = 0
   worth = 85.0      rel. time = 7
  None found.
```

No generalizations of temperate found. Now look for specializations of the argument Surrey:

```
Next Operation:   (7)
 (GET-ARGUMENT-SPECIALIZATIONS '(CLIMATE (SURREY) = (TEMPERATE)))
   time = 8          path length = 0
   worth = 85.0      rel. time = 8
  None found.
```

No specializations of Surrey found. Now look for specializations of the referent temperate:

```
Next Operation:   (8)
 (GET-REFERENT-SPECIALIZATIONS '(CLIMATE (SURREY) = (TEMPERATE)))
   time = 8          path length = 0
   worth = 80.0      rel. time = 8
 Found (HAS-PART (TEMPERATE) = (MODERATE-TEMPERATURE))
```

Found that moderate-temperature is a specialization of temperate. So now check if it is known that the climate of Surrey has a moderate temperature:

```
 Adding to stack:
 (CHECK-QUERY '(CLIMATE (SURREY) = (MODERATE-TEMPERATURE)))
```

```
Next Operation:    (9)
 (CHECK-QUERY '(CLIMATE (SURREY) = (MODERATE-TEMPERATURE)))
    time = 9          path length = 2
    worth = 80.0      rel. time = 9
 Found (CLIMATE (SURREY) = (MODERATE-TEMPERATURE))
```

Finding that Surrey does have a moderate temperature completes a reference-based specialization inference. Again the amount of evidence which this brings to bear on the climate of Surrey being temperate is calculated and the sum of the three pieces of evidence found so far is calculated:

```
-------------------------- Evidence 3 ----------------------------

 Found Evidence:
  It is known that
   (HAS-PART (TEMPERATE) = (MODERATE-TEMPERATURE))
        dom: 0.70 .. 0.80  typ: 0.00 .. 1.00  supp: 0.70 .. 0.90
   (CLIMATE (SURREY) = (MODERATE-TEMPERATURE))
        freq : 0.80 .. 0.90                   supp: 0.70 .. 0.80
  On the basis of this evidence the input query is known with
      freq: 0.80 .. 1.00                      supp: 0.40 .. 0.80
  This translates to MEDIUM to STRONG evidence that MOST to ALL of
   the CLIMATE of SURREY is TEMPERATE.

  Combining the 3 sources of evidence the initial query,
   (CLIMATE (SURREY) = (TEMPERATE)), has the following belief:
 Supporting Fact:
  freq: (0.00 0.64)  supp: (1.00 1.00)
  freq: (0.64 0.80)  supp: (0.40 0.80)
 Supporting Negation of Fact:
  freq: (0.00 0.08)  supp: (0.80 1.00)

------------------------------------------------------------------
```

Operation continues. Now looking for dependencies involving the descriptor climate:

```
Next Operation:    (10)
 (GET-DEPENDENCIES '(CLIMATE (SURREY) = (TEMPERATE)))
    time = 12         path length = 0
    worth = 75.0      rel. time = 12
```

Since the relative time is exceeded (i.e., the amount of time since the stack element which this descended from - (CHECK-QUERY '(CLIMATE (SURREY) = (TEMPERATE))) - is greater than the adjustable relative time limit parameter) the element is put on bottom of stack:

```
  Rel. time (12) exceeds limit (10) - Operation put at bottom of stack.
```

Since the element is the only element left in the stack it's processed immediately:

```
Next Operation:    (11)
 (GET-DEPENDENCIES '(CLIMATE (SURREY) = (TEMPERATE)))
    time = 12         path length = 0
    worth = 75.0      rel. time = 0
```

A dependency is found relating climate and latitude:

```
 Found (<--> CLIMATE LATITUDE)
```

Check if Surrey has a value (i.e., referent) for the descriptor latitude:

```
 Adding to stack:
 (CHECK-ARG-WITH-NEW-DES '(CLIMATE (SURREY) = (TEMPERATE)) <N-LATITUDE>)

Next Operation:   (12)
 (CHECK-ARG-WITH-NEW-DES '(CLIMATE (SURREY) = (TEMPERATE)) <N-LATITUDE>)
    time = 13        path length = 2
    worth = 67.5     rel. time = 1
```

Found that Surrey's latitude is greater than 40 degrees:

```
 Found (LATITUDE (SURREY) = (>40-DEGREES))


 Adding to stack:
 (GET-MATCHING-ARG '(CLIMATE (SURREY) = (TEMPERATE))
                          <N-LATITUDE> '>40-DEGREES)
```

Now check if there's another argument with the same referent as that found for Surrey (>40-degrees) for the descriptor latitude:

```
Next Operation:   (13)
 (GET-MATCHING-ARG '(CLIMATE (SURREY) = (TEMPERATE))
                          <N-LATITUDE> '>40-DEGREES)
    time = 14        path length = 4
    worth = 60.75    rel. time = 2
```

Found that Holland's latitude is also greater than 40 degrees:

```
 Found (LATITUDE (HOLLAND) = (>40-DEGREES))
```

Now check if new argument (Holland) has the same value (temperate) for the initial descriptor (climate) as Surrey:

```
 Adding to stack:
 (GET-REF-WITH-INIT-DES '(CLIMATE (SURREY) = (TEMPERATE)) 'HOLLAND)

Next Operation:   (14)
 (GET-REF-WITH-INIT-DES '(CLIMATE (SURREY) = (TEMPERATE)) 'HOLLAND)
    time = 15        path length = 6
    worth = 54.675   rel. time = 3
```

Found that Holland's climate is also temperate:

```
 Found (CLIMATE (HOLLAND) = (TEMPERATE))
```

Have now completed a dependency-based inference: since it is known that climate is related to latitude and that Surrey and Holland both have the same latitude an inference can be made that their climates are the same. Since Holland's climate is temperate this *is evidence that Surrey's is also. As with the previous three pieces of evidence a calculation is done to see the amount of belief which this piece of evidence brings to bear on the initial query and then the combined belief in Surrey having a temperate climate is calculated:*

```
-------------------------- Evidence 4 ----------------------------

 Found Evidence:
  It is known that
   (<--> CLIMATE LATITUDE)
        like : 0.50 .. 0.80                       supp: 0.60 .. 0.90
   (LATITUDE (SURREY) = (>40-DEGREES))
        freq : 0.80 .. 0.90                       supp: 0.80 .. 1.00
   (LATITUDE (HOLLAND) = (>40-DEGREES))
        freq : 0.90 .. 1.00                       supp: 0.90 .. 0.90
   (CLIMATE (HOLLAND) = (TEMPERATE))
        freq : 0.60 .. 0.80                       supp: 0.90 .. 0.90
  On the basis of this evidence the input query is known with
      freq: 0.60 .. 1.00                          supp: 0.20 .. 0.90
  This translates to WEAK to STRONG evidence that HALF to ALL of
   the CLIMATE of SURREY is TEMPERATE.

  Combining the 4 sources of evidence the initial query,
   (CLIMATE (SURREY) = (TEMPERATE)), has the following belief:
 Supporting Fact:
  freq: (0.00 0.64)  supp: (1.00 1.00)
  freq: (0.64 0.80)  supp: (0.40 0.80)
 Supporting Negation of Fact:
  freq: (0.00 0.08)  supp: (0.80 1.00)

------------------------------------------------------------------

PROCESSING TERMINATED - No more elements in stack.
```

Summarizes the evidence from each source:

```
SUMMARIZING EVIDENCE
  1) (CLIMATE (SURREY) = (TEMPERATE))
      freq: .64 .. .92  supp: .80 .. 1.00
  2) (CLIMATE (SURREY) = (TEMPERATE))
      freq: .33 .. 1.00  supp: .70 .. .90
  3) (CLIMATE (SURREY) = (TEMPERATE))
      freq: .80 .. 1.00  supp: .40 .. .80
  4) (CLIMATE (SURREY) = (TEMPERATE))
      freq: .60 .. 1.00  supp: .20 .. .90

 Type (see-evidence #) to see the inference path for a particular
   evidence number.

 Type (summarize-evidence-id) to see this summary.

(Total time: 20 seconds)

Command: (SEE-EVIDENCE 1)
```

Show stack elements and facts along the solution path for the first piece of evidence found:

```
(check-query '(climate (surrey) = (temperate)))
(backchain '(climate (surrey) = (temperate)))
  (--> ((LATITUDE (?X) = (>40-DEGREES)))
       (CLIMATE (?X) = (TEMPERATE)))
       like : 0.80 .. 0.90                   supp: 1.00 .. 1.00
(check-query '(latitude (surrey) = (>40-degrees)))
  (LATITUDE (SURREY) = (>40-DEGREES))
       freq : 0.80 .. 0.90                   supp: 0.80 .. 1.00

(CLIMATE (SURREY) = (TEMPERATE))
       freq : 0.64 .. 0.92                   supp: 0.80 .. 1.00
```

7.3. Symbolics Interface

Once loaded on a Symbolics 3600 series computer PRS is invoked with the *SELECT-SYMBOL-J* command. This builds the user interface screen, shown in Figure 49. This screen provides a simple interface to the lisp functions described above in order to easily operate PRS. As can be seen there are four basic panes in the window: *lisp interaction, command, debug,* and *summary*. The basic function of each will become clear in the following description of operating the system.[23]

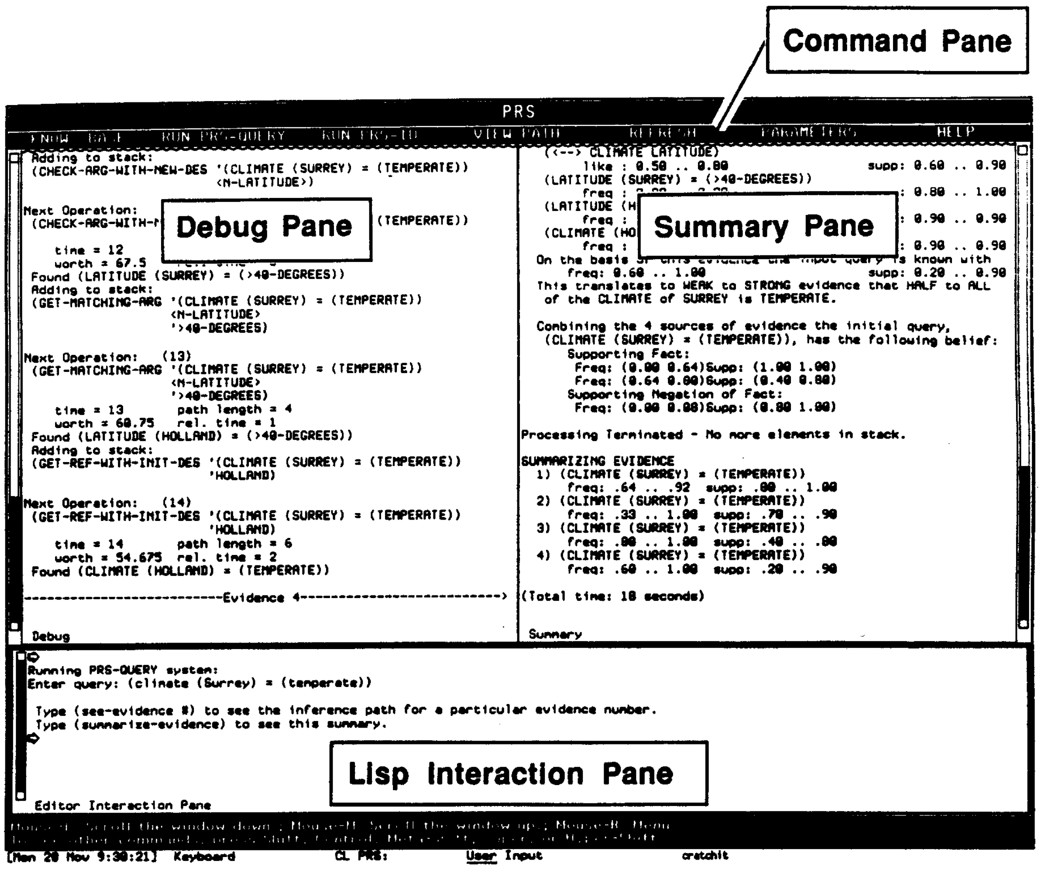
The *lisp interaction* pane at the bottom of the window is a lisp listener. It is the low level interface in which any lisp function can be called.

The *command pane* at the top of the window is the high level control interface. It allows the user to invoke a number of operations with the mouse. Each item in the command line, referred to below as buttons, is mouse sensitive. They may have multiple options depending upon which mouse button is clicked - the mouse documentation line at the bottom of the window indicates the possible operations.

The KNOW. BASE button allows a user to load in a knowledge file containing facts, rules, and dependencies or to reset the environment, removing all knowledge. Knowledge files are created by the user using an editor and contain function calls as shown previously in figure 48.

The RUN PRS-QUERY button initiates the running of the system. The user is first prompted

[23]The following description is not meant to describe the interface with great detail. With this summary the operation is very straightforward.



```
Adding to stack:
(CHECK-ARG-WITH-NEW-DES '(CLIMATE (SURREY) = (TEMPERATE))
                        <N-LATITUDE>)

Next Operation:
(CHECK-ARG-WITH-N                                  (TEMPERATE))

   time = 12
   worth = 67.5
Found (LATITUDE (SURREY) = (>40-DEGREES))
Adding to stack:
(GET-MATCHING-ARG '(CLIMATE (SURREY) = (TEMPERATE))
                  <N-LATITUDE>
                  '>40-DEGREES)

Next Operation:   (13)
(GET-MATCHING-ARG '(CLIMATE (SURREY) = (TEMPERATE))
                  <N-LATITUDE>
                  '>40-DEGREES)
   time = 13       path length = 4
   worth = 60.75   rel. time = 1
Found (LATITUDE (HOLLAND) = (>40-DEGREES))
Adding to stack:
(GET-REF-WITH-INIT-DES '(CLIMATE (SURREY) = (TEMPERATE))
                       'HOLLAND)

Next Operation:   (14)
(GET-REF-WITH-INIT-DES '(CLIMATE (SURREY) = (TEMPERATE))
                       'HOLLAND)
   time = 14       path length = 6
   worth = 54.675  rel. time = 2
Found (CLIMATE (HOLLAND) = (TEMPERATE))

--------------------------Evidence 4-------------------------->
```

```
  ((<--> CLIMATE LATITUDE)
       like : 0.50 .. 0.80                      supp: 0.60 .. 0.90
  (LATITUDE (SURREY) = (>40-DEGREES))
       freq :                                     : 0.80 .. 1.00
  (LATITUDE (H
       freq :                                     : 0.90 .. 0.90
  (CLIMATE (HO
       freq :                                     : 0.90 .. 0.90
 On the basis                                    s known with
     freq: 0.60 .. 1.00                         supp: 0.20 .. 0.90
 This translates to WEAK to STRONG evidence that HALF to ALL
  of the CLIMATE of SURREY is TEMPERATE.

 Combining the 4 sources of evidence the initial query,
  (CLIMATE (SURREY) = (TEMPERATE)), has the following belief:
     Supporting Fact:
      Freq: (0.00 0.64)Supp: (1.00 1.00)
      Freq: (0.64 0.80)Supp: (0.40 0.80)
     Supporting Negation of Fact:
      Freq: (0.00 0.08)Supp: (0.80 1.00)

Processing Terminated - No more elements in stack.

SUMMARIZING EVIDENCE
  1) (CLIMATE (SURREY) = (TEMPERATE))
      freq: .64 .. .92  supp: .80 .. 1.00
  2) (CLIMATE (SURREY) = (TEMPERATE))
      freq: .33 .. 1.00  supp: .70 .. .90
  3) (CLIMATE (SURREY) = (TEMPERATE))
      freq: .80 .. 1.00  supp: .40 .. .80
  4) (CLIMATE (SURREY) = (TEMPERATE))
      freq: .60 .. 1.00  supp: .20 .. .90

(Total time: 18 seconds)
```

```
Running PRS-QUERY system:
Enter query: (climate (Surrey) = (temperate))

 Type (see-evidence #) to see the inference path for a particular evidence number.
 Type (summarize-evidence) to see this summary.
```

```
[Mon 20 Nov 9:30:21]  Keyboard          CL PRS:          User Input          cratchit
```

Figure 49: Symbolics Interface.

for a fact to be reasoned about. As the system runs the *debug pane* lists each stack element being processed and the results of processing the element (unless debug messages are disabled via the parameters menu - described below). The *summary pane* lists each line of reasoning as it is inferred as well as a summary upon completion of processing.

The RUN PRS-ID button operates the application of PRS to the domain of turfgrass identification. This is described in detail in the following chapter.

The VIEW PATH button is used after the system has run. It allows the user to view the inference path for a particular piece of evidence which was found.

The REFRESH button allows a user to clear the output history in the *debug* or *summary* panes.

The PARAMETERS button pops up a menu which allows the user to view and modify inference control parameters as well as certain window operation parameters.

The HELP button provides a brief summary of PRS and its operation.

8. APPLICATION OF PRS

In addition to developing a model for human reasoning, one goal of this thesis was to show the utility of the Collins-Michalski Theory as a practical tool in a complex domain. An extension of PRS was developed - PRS-ID - which involves the task of making correct identifications of unknowns when given some initial information about the unknown. Specifically, the system is used in identifying turfgrass species given information regarding various characteristics.[24] Note that the development of a fully robust grass identification expert system was not the purpose of this work - it was done to demonstrate the applicability of the Theory in such a system.

8.1. Grass Type Identification

The motivation behind the development of a grass identification system is to build a tool which can be used by a turf manager to make positive identifications of unknown plants based on incomplete information. In addition, such a system could provide advice to persons not familiar with the grass identification domain. Correct identification is important as it is the first step in treating turf problems. Many factors make this task less than straightforward. For example, the unknown turf may be in such a condition that some important keys for identification cannot be determined. A common example of this is in attempting to identify grasses in mowed areas. Generally the floral part of the plants are cut off and thus cannot be used in the identification process, leaving only the vegetative features of the plants for identification. Also, the information upon which the identification is to be based may be from an unreliable source, perhaps from a person unfamiliar with the various species of turfgrasses and their characteristics.

[Morse 71] outlines five basic identification methods for determining unknown plant species:[25] i) Expert determination, which is generally regarded as the most reliable of all identification techniques. This method merely transfers the responsibility of identification to an appropriate expert. This service can be slow and costly, and is often limited by the availability of an expert. ii) Immediate recognition, which approaches expert determination and accuracy. This is the ability of an individual to recognize an unknown weed by past examples of identification.

[24]This work was done in collaboration with Professor Thomas W. Fermanian of the Department of Horticulture at the University of Illinois. Dr. Fermanian is actively involved in developing computer-based systems for agricultural tasks, especially in the area of turfgrass management.

[25]This and the following three paragraphs are included from [Fermanian & Michalski 88] with permission.

process itself matches with how people perform the identification task. Having a number of reasoning "tools" available to solve the task at hand is important as well as controlling the reasoning based on time and other constraints. The inference style of collecting evidence from multiple sources also fits in with the grass identification task. In [Fermanian & Michalski 88] it is indicated that untrained individuals could identify unknown grasses if multiple evidence was available.

8.2. Identification Mode of Operation

Although to this point only one mode of PRS operation has been discussed, as mentioned at the outset of thi^ chapter there are actually two, PRS and PRS-ID. PRS is the base system whose operation was described in the previous chapter: a user inputs an initial query and, using the previously input knowledge, inference is done to determine a measure of belief in that query.

PRS-ID is a specialization of PRS adapted to the task of turfgrass identification. Its general operation is identical to PRS. The goal behind it is to determine the identity of an unknown argument, given some initial information about the characteristics of the unknown. Based on the information in the knowledge base, a determination is made about which argument, from a set of given possibilities, has those initially input characteristics.

In order to make a useful identification tool several features were developed on top of the base PRS system. One is the ability to input information about a specific situation at hand in addition to having a knowledge base of domain information. This input information is referred to as **initial** or **id facts**. The ability to modify this initial information or add new information incrementally during system operation is also important. In keeping with the nature of PRS all input information is expressed using the same uncertainty weights as described previously.

The domain information which the system uses are facts regarding grass characteristics. This information can be thought of as two basic types, classificatory and characteristic. Classificatory involves the morphological structural information which defines the domain. Characteristic involves the various attributes and values associated with the classes. See [Morse 71] for more information on the grass domain. Figure 50 presents some examples of these facts. The system currently has 109 facts defined in its knowledge base. Appendix E lists the complete knowledge base.

Running PRS-ID involves first entering the characteristics of the unknown type of grass and the possible choices for the identification of the unknown. Next the inference process is started. The system operates by generating a query for the combination of each possible identification with each input characteristic (i.e., id fact). This is done by substituting the possibilities for the

For some taxonomic groups and immature plants, however, this method of identification is very difficult and in all cases requires extensive past experience. iii) Comparison of an unknown specimen with identified species or illustrations. It offers a rapid, simple diagnosis and is often useful for many commonly found weeds. iv) An identification key which is based on the development of appropriate descriptive phrases of morphological or biochemical character-variables (hereafter, referred to as variables). Identification keys generally take the form of groupings of similar morphological variables from which the user must select the variable which best matches that present on the unknown sample. The selection of this variable then leads to the next set of identifying characteristics . This process is followed until enough variables have been identified to suggest the identification of the specimen. v) The last identification technique is a diagnostic table or polyclave. Diagnostic tables are a matrix of rows of species and columns of identifying variables. Users of a diagnostic tables can identify the listed variables in any order they wish. Currently, the use of an identification key or expert determination are the most commonly used methods to identify grasses found in turf by inexperienced turf managers.

[Morse 71] lists two major faults of identification keys: i) They require a user to utilize certain variables whether or not they are convenient or can be identified. Because the keys are set up as decision trees where each branch requires a definite answer they are inflexible in not allowing for unknown information; ii) They implicitly rely on rigid descriptions of specimens. Occasional variation in a population can cause a gross misidentification.

In a list of relative merits of both identification keys and diagnostic tables [Payne & Preece 80], stated that the diagnostic table was superior to a key because it offered a choice in the order of characters to be used for identification. They also suggested, however, that keys are more convenient to use.

Computer systems might offer the benefits of both identification keys and diagnostic tables in one tool. The use of expert systems techniques offers a new, unique method for assisting with species identification [Atkinson & Gammerman 87]; [Fermanian, Michalski, Katz & Kelly 88]. The relative merits of an expert system is its ability to allow the user to select variables that are available on the unknown specimen. They can operate on various levels of uncertainty providing a more efficient mechanism for identification. They can be easily modified to reflect local variation in the values of variables for included species.

PRS is well suited to both the specific grass identification task and identification tasks in general. First, the representation allows for the hierarchical nature of grass classification to be made explicit. Characteristics of grasses can be entered at various levels of abstraction. The use of various certainty weights corresponds well with how characteristics are associated with plants - as with most "real world" domains little information is black and white. The inference

```
(has-type (grass) = (poa-genus  D[0.2  0.25]  T[0.85  0.95]  S[0.65  1.0] ))

(vein-type (grass) = (parallel  F[1.0  1.0]  S[0.95  1.0] ))

(has-type (poa-genus) = (kentucky-bluegrass  D[0.4  0.7]  T[0.85  0.95]  S[0.85  1.0] ))

(leaf-type (poa-genus) = (keel-shape  F[0.9  0.95]  S[0.9  0.95] ))

(collar-type (kentucky-bluegrass) = (narrow  F[0.45  0.65]  S[0.85  0.95] ))

(sheath-type (kentucky-bluegrass) = (compressed  F[0.8  0.9]  S[0.9  1.0] ))
```

Figure 50: Examples from Grass Identification Knowledge Base.

unknown as the argument in each id fact and adding a **CHECK-QUERY** operation to the stack for each of these generated queries. As an example, if there are two possible identifications for the unknown and three known features about the situation six queries would be generated. Once these operations are added to the stack inference proceeds as described in Chapter 5 on inference control. In processing these operations PRS-ID collects evidence that each of the possible identifications possesses the initially entered facts, and thus is the correct identification for the unknown. When processing terminates the evidence gathered supporting each possibility is summarized and a judgement can be made as to which possibility most closely matches with the initial features describing the unknown. At this point several options exist. One is to examine the inference path which led to each conclusion. Another is to analyze the evidence and try to draw a conclusion as to the correct identification. The system has a facility for matching the inferred evidence to the id facts and selecting the most probable identification. The user can also enter new facts or revise the old ones and see how the evidence is affected.

Figure 51 gives an example of the functions used to set up and run PRS-ID (A full listing is in Appendix B). **RESET-PRS-ID** clears any previous id facts and possibilities from the environment. **ADD-ID-FACTS** and **SET-POSSIBILITIES** set the initial facts and identification possibilities. The function **RUN-PRS-ID** initiates the processing of the given id facts and possibilities. **RUN-PRS-ID-INCREMENTALLY** allows a user to build upon previous results rather than starting from scratch. The idea is to allow a user to run PRS-ID and get some results. Then additional facts can be given or the initial ones can be revised and the system can be run again with the previously inferred facts as the starting point.

```
(RESET-PRS-ID)

(ADD-ID-FACTS
  '((collar-type (?x) = (narrow F[ 0.3 0.6 ]    S[ 0.9 1.0 ]))
    (ligule-shape (?x) = (truncate F[ 0.8 0.9 ] S[ 0.9 1.0 ]))))

(SET-POSSIBILITIES 'poa-genus 'agrostis-genus)

(RUN-PRS-ID)

(RUN-PRS-ID-INCREMENTALLY)

(MATCH-EVIDENCE-TO-FACTS)
```

Figure 51: PRS-ID Functions.

Using the function **MATCH-EVIDENCE-TO-FACTS** the evidence gathered for each possibility is matched to the given facts. This is done by considering the intersection between the frequencies of corresponding given and inferred facts as positive evidence and the non-intersection as negative evidence. The generated match ratings can be used to conclude that the unknown is one of the given possibilities, or perhaps to indicate that enough evidence has not yet been found supporting a particular conclusion. Figure 52 illustrates the match rating principle. The result generated by equation 39 is a range with both values between -1 and 1. These values are then normalized into the range of 0 to 1. Note that the ratings are only used for comparison purposes and have no higher meaning. The equations for the calculation are as follows:

$$\textit{Positive Evidence} = \textit{\% of inferred freq which intersects with given freq} \tag{37}$$

$$\begin{aligned}\textit{Negative Evidence} = \textit{MAX}(-1\ ,\ & \textit{\% of given freq which doesn't intersect with inferred freq} \\ & + (\textit{\% of inferred freq which doesn't intersect with given freq} \\ & \quad \cdot \textit{avg. distance from closest given bound}))\end{aligned} \tag{38}$$

$$\textit{Match Rating} = (\textit{Positive Evidence} - \textit{Negative Evidence}) \cdot \textit{Support Bounds} \tag{39}$$

MATCH-EVIDENCE-TO-FACTS lists the possibilities in decreasing order of likelihood as the correct identification. This order is determined by considering first the number of pieces of evidence found for each possible identification and then the average match rating of all the evidence found for that identification. See the end of the output trace in Appendix F for an example of the output of this function.

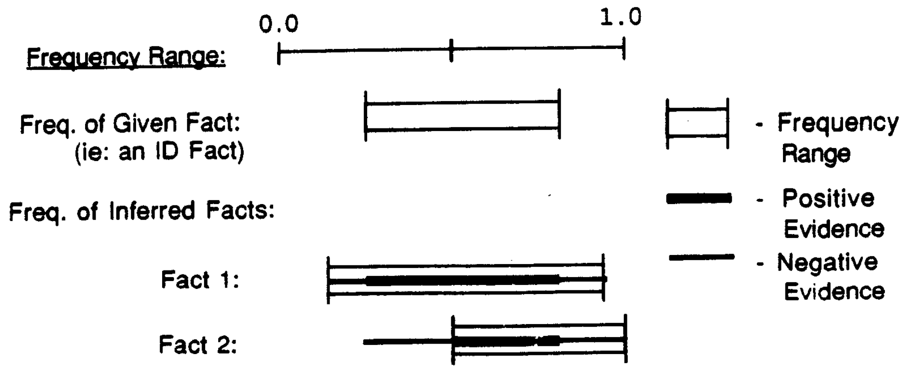


Figure 52: PRS-ID Match Rating.

In the Symbolics interface window, described in section 7.3, the RUN PRS-ID button operates the grass identification system. The user first sets the initial facts (id facts) and the possibilities for the unknown (recall figure 51). This can be done in the *lisp interaction pane*. Alternatively these functions could be put in a file and entered via the KNOW. BASE button. PRS-ID can run either from an initial "clean" state (left mouse click) or incrementally (middle click). It is run incrementally by adding new information about the unknown (with the function *ADD-ID-FACTS*) without resetting the environment. The currently known facts about the unknown as well as the identification possibilities can also be listed (right click).

Appendix F is an annotated trace of the system running in PRS-ID mode.

8.3. Analysis of Grass Identification Application

PRS-ID was analyzed by comparing system output to the conclusions of humans given the same initial information.[26] This testing involved setting up experiments, or *runs*, with various combinations of both initial facts and identification possibilities.[27] The initial information in the experiments ranged from two initial facts and two identification possibilities to six facts and seven

[26]The human analysis of each situation was done by the domain expert, Dr. Fermanian.

[27]The id facts and possibilities comprising each experiment are listed in Appendix G.

possibilities. A set of twenty runs comprised the test set. Each experiment was run by the system and the output recorded. Parallel to the system examination of each of the experiments the domain expert analyzed the twenty situations and ranked the identification possibilities in terms of their correspondence with the input facts. The conclusions of both the system and expert were then compared and contrasted.[28]

In the analysis of the results, an experiments might be viewed as being of one of three general types: 1) When there are no, or minimal, conflicting initial facts and one identification possibility is correct (In any particular experiment a *conflict* is when one id fact is evidence for one identification and another id fact is evidence for a different identification); 2) When there are conflicting facts but one possibility is still clearly correct; and 3) When there are many conflicting facts and no choice can be determined to be the correct identification.

In experiments of the first type the system output always matched the results of human reasoning given the same initial information: the same identification possibility was selected as being the most probable by both the human and the system. In experiments of the second type, where one choice was correct even though some of the initial facts conflicted, the system results again corresponded with human reasoning. However one deficiency of the system was evident in reviewing the experiments of this type. This shortcoming is in the match rating mechanism which matches the inferred evidence to the initial facts. Although the mechanism correctly ordered the correct identification choices as being more likely than the other possibilities there is not an accurate feel for what this rating actually means. It is unclear as to within what range two choices should be judged as having essentially the same degree of belief. Likewise, it should be made clear when one choice is above some threshold and should be deemed the correct choice. Recall from the previous section that the match rating was only developed to compare evidence. A facility for interpreting the meaning of rating values, allowing direct comparison with human generated values, would be useful.

In addition, some means of weighing the importance of attributes would allow the matching facility to more closely correspond with human reasoning. When presented with multiple conflicting characteristics humans have the ability to rank them as being more or less important relative to the task at hand. This importance factor plays a role in the analysis of conflicting evidence.

The need for additional analysis of the correspondence between initial and inferred facts is also evident in the review of experiments of the third type (when there are conflicts and no one choice is correct). For all identification possibilities the match ratings generated by the system

[28]A summary of the results of both the system and expert analysis of each experiment is given in Appendix H.

were closely clustered however there is no automatic facility for determining that within a certain range match rating values are essentially the same. Thus in situations when there are similar amounts of evidence for multiple identifications - when no one identification is decidedly correct - it is important to more formally analyze how the evidence for each possibility compares to the known facts so that the possibilities can be ordered correctly.

Another area of the system which the experiments illustrated as requiring additional refinement is the overall method by which the id possibilities are ranked. Recall from the description of the function MATCH-EVIDENCE-TO-FACTS in the previous section that this is done by considering the number of pieces of evidence found for each possibility and then the average match rating associated with the possibilities. However, in the words of the expert: "... several poor matches might not be justifiably better than one close match." Some method for considering both the amount of evidence and rating values in parallel rather than allowing one to have precedence over the other when ranking choices should be examined.

Another dimension of analysis of PRS-ID examined the effect of the varying number of initial facts and and identification possibilities. The inference control mechanisms described in Chapter 5 manage both small and large amounts of initial information so system can handle both situations. When the amount of initial information is at either the small or large extreme, though, the possibility of unclear results increases (too little information may be ambiguous, too much may lead to be conflicts), giving rise to the problems mentioned in the previous paragraphs.

Beyond the examination of specific experiments the analysis of the system considered a number of additional issues, including the correspondence of the Collins-Michalski Theory to human reasoning in this domain, and the utility of PRS-ID as an agricultural tool, and the utility of PRS-ID as a general purpose reasoning tool.

First, the organization of grass information into hierarchical knowledge structures and the attachment of attributes at any level in the hierarchies corresponds with how humans view the the domain. The uncertainty parameters which the Theory identifies allows the information which humans bring to bear on this task to be easily represented. The inference types used in PRS clearly match with the existing methods for turfgrass identification. For example, identification keys move users between general and specific features, corresponding to the generalization and specialization based inferences.

The inference style also corresponds with certain aspects of human reasoning. This is evident considering both the gathering of multiple evidence as well as the alternating focus on each identification possibility (described in Chapter 5 on inference control). Further, the aforementioned ability to incrementally add information models how a human reasons in most situations.

One area for some further attention is to examine more closely the manner in which queries are initially generated and subsequently processed. Recall this is done by combining each of the identification possibilities with each of the known characteristics. This method does fulfill the goal of getting evidence about each of the possibilities actually being the correct identification yet in situations where there are large numbers of possibilities and initial facts the number of queries which the system must find evidence for is large. Ideally as a reasoning process proceeds the control mechanism would focus attention on those hypotheses which have already gathered some level of evidence, not bothering to process those queries associated with possibilities that seem highly unlikely to be the correct identification.

Several comments can be made in terms of the utility of PRS-ID as an agricultural tool. Because it allows varying degrees of known facts about a situation PRS-ID is useful in the many different circumstances which a turfgrass manager might encounter. In situations where much information is known about the unknown obviously a more accurate conclusion can be reached but the system can also make judgements in situations where there are less constraints, albeit with less certainty. PRS-ID was developed to be a useful tool for both domain experts as well as people relatively unfamiliar with turfgrasses. The ability to input the initially known facts about a situation with uncertainty weights was developed for this reason. This feature is important in situations where a non-expert is examining the plant and the accuracy of their description of the features is not without doubt. Another related feature is the explanation facility. After the system finishes its processing of queries a user can look at the inferences and facts involved in each piece of evidence.

One area of focus for additional work is the general expansion of the knowledge base. The system currently operates on two general classes of grass (genera) and seven specific grass species divided between the two classes. The number of features used to describe these general classes and individual grasses is nine. A more fully developed system would include approximately 15 genera, 40 individual grass species, and 20 identifying features.

Using this example of turfgrass identification as a basis several points can be made about the applicability of PRS-ID in other domains. The inference style of moving between levels of abstraction by generalization and specialization is obviously dependent upon information being related hierarchically. Also, as the general inference mechanism works in a backward chaining manner - given a query evidence is found supporting its belief - the unknown possibilities must be made explicit at the outset. Certain aspects of reasoning not integrated into the Theory, such as a notion of sequences of events, are likewise not handled in PRS-ID and thus it is clearly not applicable to tasks which involve such reasoning. As has been a recurring theme in this thesis as well as in the Theory the inference methods employed by humans are used across diverse domains. The PRS-ID system might be thought of as essentially a pre and post processor to the inference methods detailed by Collins and Michalski and thus is applicable to identification tasks whose representation and inference style corresponds to that detailed in the Theory.

From this application several important conclusions can be made about both PRS-ID and PRS in general. In analyzing the results of PRS-ID it is clear that the inference style used produces plausible results, closely corresponding with human reasoning. Although the purpose of the grass identification system was not to develop a fully robust expert system clearly solid headway has been made toward that end. In terms of expert systems technology a number of features were illustrated. One is the ability to revise input and use the system in an interactive and incremental fashion. Secondly, by storing the traces of inference paths, PRS can explain its conclusions, always an important aspect for any expert system. A unique feature of PRS-ID is the explicit uncertainty allowable in the input. The focus of many expert systems has been on how uncertainty can be expressed in a knowledge base but allowing the actual interactions with a user to be expressed with uncertainty is also important. Although no additional domains were examined in detail there is no reason why the techniques used in PRS-ID and results obtained in the application of the system to the task of grass identification could not extended to other domains. As illustrated with this application the inference processes described in the Collins-Michalski Theory correspond with many important aspects of human reasoning.

9. CONCLUSION

The purpose of this thesis was to implement a working computer system which can serve for hypothesizing and testing various aspects of human plausible reasoning. In this research a number of important issues have been explored. In working on these problems some clear accomplishments have been made and certain directions for future research have become apparent.

First and foremost a running computer system modeling many characteristics of human plausible reasoning was developed. The focus, of course, was to look at one particular theory of plausible reasoning, that of Collins and Michalski. The implemented system, PRS, corresponds with the ideas presented in the Theory: Multiple inference methods are used to obtain evidence. Generalization and specialization based inferences as well as implications and dependencies are used to find information related to an unknown query. Conclusions are reached by combining and comparing the evidence obtained from multiple and perhaps disparate sources of evidence.

The Collins-Michalski Theory presents a viewpoint that human reasoning can be effectively modeled in a quantitative fashion. A major focus of this thesis is on the issue of uncertain reasoning and a method is presented which details how uncertainty can be represented and handled in a computer system. Although final determinations can be made qualitatively the evidence behind those decisions is examined and combined in a quantitative manner. An important extension to the Theory developed within is the idea that these numeric parameters are not single numbers but rather a range. This allows information to be represented capturing both the degree, or magnitude, of belief in a fact and also its precision. This is useful in more accurately modeling certain human knowledge - humans have the ability to express knowledge in terms of a range of values rather than just a single value when necessary. Allowing for a range of values does allow for single values - having single values is just a specialization where the lower and upper bounds are equal.

A large portion of the work in this thesis was spent developing the uncertainty propagation equations. A conservative approach was taken in the uncertainty mechanism. In taking a conservative viewpoint in the equations no assumptions were made which might bias the results, such as conditional independence. It is important as a starting point in any uncertainty mechanism to generate answers with no bias. If the initial results are not specific enough then it might be appropriate to make some assumptions which might generate more certain conclusions.

Another area receiving attention in this work was control of inference processes. In general reasoning can be viewed as involving trade-offs between cost, or time, certainty, and generality. As variance is made along one of these dimensions the other two are affected. For example, as more time is spent gathering evidence about a situation the certainty of the conclusion should increase. The control mechanisms developed within attempted to model these characteristics of human reasoning, based upon general intuitions. Clearly the individual mechanisms used, such as weighing certain inference types as more likely to generate evidence than others, have a basis in human plausible reasoning. An area for additional work is a more formal development and analysis of these inference control methods.

An important result of this thesis involved the analysis of the system output. Both in the simple geographic examples and in the more complex grass domain PRS does generate plausible answers. Analysis of the evidence found and conclusions reached by PRS showed it to correspond to general intuitions of human performance given the same information. One area for additional work is in a more formal analysis of the results, perhaps involving controlled experiments comparing system results to humans.

PRS can be used as a tool to test psychological theories of plausible reasoning. The parameterization of system variables allow it to be tuned to focus on specific aspects of reasoning for running controlled experiments. For example, by adjusting the search to follow a depth first algorithm the issue of getting sidetracked on seemingly promising yet eventually useless paths can be explored.

As a test domain for the implemented system the problem of making identifications with limited initial information was explored. Specifically the identification of grass types given less than complete identifying characteristics was examined. For this application a mode of operation was developed where a list of possible identifications for the unknown grass and a set of initial facts describing features of the unknown are input. From this information a decision can be reached on the most plausible identification. The analysis of the output of this application showed that the reasoning style as well as the conclusions reached closely matched those of humans.

Several features of the grass identification system are noteworthy. The allowing of uncertainty both in the input facts of the system as well as the knowledge base corresponds with the information humans have at their disposal in this and similar reasoning tasks. Also the ability to incrementally input facts and continue running from that point makes the system much more efficient, as well as, again, closely modeling this ability in humans. Although, as mentioned in the description of the implementation in Chapter 8, the number of facts must be expanded in order to have a fully robust system, it is clear that with that additional knowledge a useful tool would exist.

One of the basic tenets of the Collins-Michalski Theory is the general applicability of the described reasoning methodology in modeling humans. This is corroborated by the results of both the grass identification problem as well as the other small examples used in this report and during system development. The use of multiple inference methods and multiple evidence sources when making decisions is a major tool of human reasoning in any domain. The uncertainty parameters identified by the Theory provided a useful and accurate syntax for representing human knowledge. As PRS uses these general principles at its core in principle any domain can be implemented by generating a knowledge base and, if necessary, providing some basic input/output facilities.

The breadth of issues surrounding the implementation of a system for plausible reasoning necessitated focusing on certain areas and not on others. Several aspects of the Collins-Michalski Theory were not examined in this thesis. Specifically similarity based inferences which are based on lateral relationships between objects rather than hierarchical relationships were not implemented. The use of similarity and analogy in reasoning processes is currently an area of much research and the breadth of ideas and theories justifies more examination than was possible in the framework of this thesis. In addition, as discussed in Chapter 3 on knowledge representation, the multiplicity parameter was not implemented.

In the discussion of these issues it was stated that PRS is amenable to certain extensions, such as the addition of inference methods and uncertainty parameters. Several factors allow this statement to be made. First and foremost is the use of a stack-like mechanism to hold inference operations. As additional inference types are hypothesized, formalized, and implemented they are initiated by adding an element to the stack which, when processed, calls the newly defined function on an unknown query to perform the inference. In addition, because the control parameters are readily adjustable rather than hardcoded any tuning of the system to handle new operations is straightforward. Uncertainty calculations are performed after an inference is complete, after all information about a particular path has been found. This allows the current uncertainty mechanisms to be extended to handle additional uncertainty parameters and to be substituted for the current methods where appropriate. Finally, as described in Chapter 3, the knowledge representation in PRS is such that any concept may be used as any element in a datum - its use depends on the links between them. Because links can be added incrementally and there is no limit to the number of links to or from any concept extending a knowledge base is not complicated.

Another area which was not explored in detail is how formal methods could be used in setting the numeric uncertainty weights. As mentioned, all the weights on information used in this thesis were set subjectively, based on experience and intuition with regard to a particular datum. The setting of weights is nonetheless a very important issue and it would be a worthwhile avenue for additional work. Such work must first decide whether the focus should be to model the world as it is, for example by collecting data and computing correct conditional probabilities between objects and attributes, or to more closely model the information which humans use, by somehow eliciting such information directly from people.

Plausible reasoning is inseparably related to other aspects of human thinking and behavior. Human knowledge - how it is organized, how it changes - is obviously an important issue in reasoning. In this thesis a highly connective knowledge base was developed as the model for human memory. This agrees with many features of human memory yet more examination of this area is in order to more fully correspond with the human model.

One research area which in the past few years has become the focus of much attention is parallel processing. In many situations human reasoning relies on information being gathered with the various senses concurrently. PRS was implemented with the thought of the parallel processing of inference paths as an area for future work and such an extension would not require major changes to the structure of the system.

In summary, this thesis illustrates the ideas of the Collins-Michalski Theory of plausible reasoning in a working computer system. This work shows that not only are the ideas in the Theory valid in terms of modeling human plausible reasoning but also that a computer system can be built which generates results corresponding to human reasoning.

REFERENCES

[Atkinson & Gammerman 87]
Atkinson, W.D. & Gammerman, A. An Application of Expert System Technology to Biological Identification. *Taxon* 36(4):705-714, 1987.

[Baker, Burstein, & Collins 87]
Baker, M., Burstein, M., & Collins, A. Implementing a Model of Human Plausible Reasoning. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence.* August, 1987.

[Bobrow & Collins 75]
Bobrow, D.G. & Collins, A. (editors). *Representation & Understanding: Studies in Cognitive Science.* Academic Press, New York, 1975.

[Cohen 85] Cohen, P.R. *Heuristic Reasoning about Uncertainty: An Artificial Intelligence Approach.* Pitman, Boston, 1985.

[Collins 78] Collins, A. *Human Plausible Reasoning.* Technical Report 3810, Bolt, Beranek, and Newman, Inc., Cambridge, MA, February, 1978.

[Collins & Loftus 75]
Collins, A. & Loftus, E. A Spreading Activation Theory of Semantic Processing. *Psychological Review* 82(6):407-428, 1975.

[Collins & Michalski 88]
Collins, A. & Michalski, R.S. The Logic of Plausible Reasoning: A Core Theory. 1988.Accepted for Publication.

[Dontas 88] Dontas, K. An Implementation of the Collins-Michalski Core Theory of Plausible Reasoning. Master's thesis, University of Tennessee, Knoxville, August, 1988.

[Duda, Hart, & Nilsson 76]
Duda, R.O., Hart, P.E., & Nilsson, N.J. *Subjective Bayesian Methods for Rule-Based Inference Systems.* Technical Note 124, Stanford Research Institute, Artificial Intelligence Center, Menlo Park, CA, January, 1976.

[Duda, Hart, Konolige, & Reboh 79]
Duda, R.O., Hart, P.E., Konolige, K., & Reboh, R. *A Computer-Based Consultant for Mineral Exploration.* Final Report 6415, Stanford Research Institute, Artificial Intelligence Center, Menlo Park, CA, September, 1979.

[Fermanian & Michalski 88]
Fermanian, T.W. & Michalski, R.S. WEEDER: An Advisory System for the Identification of Grasses in Turf. *Agronomy Journal*, 1988. Accepted for Publication.

[Fermanian, Michalski, Katz, & Kelly 88]
Fermanian, T.W., Michalski, R.S., Katz, B. & Kelly, J.D. AGASSISTANT: An Artificial Intelligence System for Discovering Patterns in Agricultural Knowledge and Creating Diagnostic Advisory Systems. *Agronomy Journal*, 1988. Accepted for Publication.

[Fikes, Hart, & Nilsson 72]
Fikes, R.E., Hart, P.E., & Nilsson, N.J. Learning and Executing Generalized Robot Plans. *Artificial Intelligence* (3), 1972.

[Gentner & Collins 81]
Gentner, D. & Collins, A. Studies of inference from lack of knowledge. *Memory & Cognition* 99(4):434-443, 1981.

[Haddawy 86] Haddawy, P.F. A Variable Precision Logic Inference System Employing the Dempster-Shafer Uncertainty Calculus. Master's thesis, University of Illinois, December, 1986.

[Haddawy & Kelly 87]
Haddawy, P.F. & Kelly, J.D. ESTIMATOR: An Application of Variable Precision Logic to Construction Cost Estimation. In *Proceedings of the Second International Conference on Applications of Artificial Intelligence in Engineering Problems*. Boston, August, 1987.

[McDermott 82] McDermott, J. R1: A Rule-Based Configurer of Computer Systems. *Artificial Intelligence* 19:39-88, 1982.

[McDermott & Doyle 80]
McDermott, D.V. & Doyle, J. Non-Monotonic Logic I. *Artificial Intelligence* 13:41-72, 1980.

[Michalski & Winston 86]
Michalski, R.S. & Winston, P.H. Variable Precision Logic. *Artificial Intelligence* 29(2):99, 1986.

[Morse 71] Morse, L.E. Specimen Identification and Key Construction with Time-Sharing Computers. *Taxon* 20(1):269-282, 1971.

[Payne & Preece 80]
Payne, R.W. & Preece, D.A. Identification Keys and Diagnostic Tables: A Review. *Journal of the Royal Statistical Society* 143(3):253-292, 1980.

[Quillian 68] Quillian, M.R. Semantic Memory. *Semantic Information Processing.* M.I.T. Press, Cambridge, 1968.

[Quinlan 83] Quinlan, J. R. Inferno: A Cautious Approach to Uncertain Inference. *The Computer Journal* 26(3):255-269, 1983.

[Shafer 76] Shafer, G. *A Mathematical Theory of Evidence.* Princeton University Press, Princeton, N.J., 1976.

[Shortliffe 76] Shortliffe, E.H. *Computer-Based Medical Consultations: MYCIN.* American Elsevier, New York, 1976.

[Tversky & Kahneman 73]
Tversky, A. & Kahneman, D. Availability: A Heuristic for Judging Frequency and Probability. *Cognitive Psychology* 5(99):207-232, 1973.

[Wise & Henrion 84]
Wise, B. P. & Henrion, B. A Framework for Comparing Uncertain Inference Systems to Probability. In *Proceedings of the AAAI Workshop on Uncertainty and Probability in Artificial Intelligence*. American Association for Artificial Intelligence, Menlo Park, CA, August, 1984.

[Zadeh 65] Zadeh, L.A. Fuzzy Sets. *Information and Control* 8:338-353, 1965.

APPENDIX A. REPRESENTATION SUMMARY

The following is a quick reference for mapping between the representation and associated meaning for PRS facts. For each category the syntax and semantics are given, first for unweighted and then weighted facts. Also, for each category one example is given for a fully weighted fact.

1. **DESCRIPTORS:**

   a. (DES (ARG) = (REF))

      "The DES of ARG is REF."

   b. (DES (ARG) = (REF F[A B]))

      "A to B% of the DES of ARG is REF."

      P(DES(REF) | ARG) = A .. B

   c. (DES (ARG) = (REF F[A B] S[C D]))

      "There is a C to D% probability that A to B% of the DES of ARG is REF".

      P(P(DES(REF) | ARG)) = C .. D

   d. (climate (England) = (temperate F[0.6 1.0] S[0.8 1.0])) : "There is an 80 to 100% probability that 60 to 100% of the climate of England is temperate."

2. **INVERSE DESCRIPTORS:**

   a. (inv-DES (REF) = (ARG))

      "DES that is REF is in ARG"

   b. (inv-DES (REF) = (ARG F[A B]))

"A to B of DES that is REF is in ARG."

P(ARG | DES(REF)) = A .. B

c. (inv-DES (REF) = (ARG F[A B] S[C D]))

"There is a C to D% probability that A to B of DES that is REF is in ARG."

P(P(ARG | DES(REF))) = C .. D

d. (country (temperate) = (England F[0.0 0.2] S[0.7 0.9])) : "There is a 70 to 90% probability that 0 to 20 of climate that is temperate is in England."

3. **SPECIALIZATIONS:**

a. (has-part (ARG) = (REF))

"ARG has part REF."

b. (has-part (ARG) = (REF D[A B]))

"A to B% of ARG is REF."

P(has-part(REF) | ARG) = A .. B

c. (has-part (ARG) = (REF T[E F]))

"E to F% of ARG's features hold for REF.

P(has-part(REF) | ARG) = E .. F

APPENDIX B. SYSTEM LISP FUNCTIONS

This appendix lists and describes the functions used in running PRS and PRS-ID. Examples of some of the functions from the climate and grass identification domains are in Chapters 7 and 8 as well as Appendices C, E, F, and G. See Chapters 7 and 8 for details on running the system in either mode.

B.1. General Functions

DEBUG *status*	Sets debugging flags to argument.
ASSERT! *descriptor argument referent weights*	Enters given fact into environment. If the fact already exists the weights are set to the new values. An example of ASSERT! being called is in Appedix C.
ADD-FACTS *&rest facts*	Calls ASSERT! to add given facts into environment. *Facts* are any number of facts with their associated weights. See Chapters 7 and 8 as well as Appendices C and E for examples.
MAKERULE *rule*	Enters given implication (*rule*) into the environment. The syntax for *rule* is an implication in prefix form and its associated weights. See Chapters 7 and 8 as well as Appendices C and E for examples.
MAKEDEPENDENCY *dependency*	Enters given dependency (i{dependency}) into the environment. The syntax for *rule* is an implication in prefix form and its associated weights. See Chapter 7 or Appendix C for an example.
RESET-DATA *()*	Clears knowledge from environment.
LOAD-PRS-DATA *()*	Prompts for and then loads a knowledge-base file.

SET-INVERSE *descriptor1 descriptor2*	Sets descriptors as inverses of each other. (Arguments are symbols, not structures).
SET-DEFAULT-INVERSES *()*	Sets inverses of hierarchical descriptors (type-of, has-type, part-of, and has-part).
SET-INVERSES *()*	Prompts user for setting descriptors as inverses of each other.
CHECK-DATA *descriptor argument referent*	Checks knowledge-base for given fact. (Arguments are symbols).

B.2. PRS

RUN-PRS *&optional (outflag nil)*	Runs prs-query system. Outflag is t when running in PRS Interface window.
SEE-EVIDENCE *num &optional (all-elements nil)*	Lists the inference operations and facts found which provided evidence for an input query (prs-query mode) or for a particular uknown (prs-id mode). If *all-elements* is t listing includes queries for inverse facts (inverse fact weights are necessary for uncertainty calculations).
SUMMARIZE-EVIDENCE *()*	Lists each piece of evidence found relating to an input query.
SUM-EVIDENCE *()*	Combines all of the evidence found for an input query and summarizes it in terms of support for the query and for the negation of the query.

B.3. PRS-ID

SET-POSSIBILITIES *possibilities*	Defines the possibilities for the identification of the unknown.
ADD-ID-FACTS *forms*	Adds facts about unknown. If facts are already entered as an id fact this just modifies its weights. See Chapter 8 as well as Appendix E for examples.
DEL-ID-FACTS *forms*	Deletes facts about unknown and any evidence which relates to the deleted facts.
LIST-ID-FACTS-&-CHOICES *()*	Lists the possibilities for the unknown and related facts.
RESET-PRS-ID *()*	Re-initializes prs-id system: removes facts about unknown, clears evidence.
RUN-PRS-ID *()*	Begins running PRS-ID system.
RUN-PRS-ID-INCREMENTALLY *()*	Runs PRS-ID with previous results already known.
SEE-EVIDENCE *num &optional (all-elements nil)*	Same as in PRS-QUERY mode.
SUMMARIZE-EVIDENCE-ID *()*	Summarizes the evidence found for each possible choice of the unknown.
MATCH-EVIDENCE *()*	Computes a rating for each inferred fact to the given fact to which it corresponds.

B.4. Symbolics

SELECT-SYMBOL-j	Selects PRS Interface window.

APPENDIX C. CLIMATE KNOWLEDGE BASE

Below are the facts, rules, and dependencies used in the climate examples used throughout this thesis. (For a detailed example see Chapter 7). The type of weight associated with datum is indicated by the capital letter preceding a bracketed range: F - frequency, D - dominance, T - typicality, L - likelihood, S - support (described in Chapter 3).

The information is some very basic geographic and climatic information. The weights were set by some cursory research combined with the author's intuitions. The purpose of this knowledge base is to demonstrate PRS and certain aspects of the Collins-Michalski Theory and not to generate infallible lines of reasoning. See Chapter 7 for an annotated listing of system output involving this domain.

The ADD-FACTS command asserts facts into the environment:

```
(ADD-FACTS
 '((part-of (England) = (Europe        D[ 0.1 0.2 ]  S[ 0.6 0.9 ]))

   (climate (England) = (temperate     F[ 0.8 0.9 ]  S[ 0.7 0.9 ]))

   (has-part (England) = (Surrey       D[ 0.1 0.2 ]  T[ 0.3 0.8 ]
                                       S[ 1.0 1.0 ]))

   (part-of (Surrey) = (England        D[ 1.0 1.0 ]  S[ 1.0 1.0 ]))

   (latitude (Surrey) = (>40-degrees   F[ 0.8 0.9 ]  S[ 0.8 1.0 ]))

   (climate (Surrey) = (moderate-temperature         F[ 0.8 0.9 ]
                                                     S[ 0.7 0.8 ]))

   (forest-type (Surrey) = (deciduous  F[ 0.6 0.9 ]  S[ 1.0 1.0 ]))

   (has-type (fir-type) = (pine        D[ 0.5 0.5 ]  S[ 0.9 1.0 ]))

   (climate (Holland) = (temperate     F[ 0.6 0.8 ]  S[ 0.9 0.9 ]))

   (latitude (Holland) = (>40-degrees  F[ 0.9 1.0 ]  S[ 0.9 0.9 ]))

   (part-of (East-Surrey) = (Surrey    D[ 0.8 0.9 ]  S[ 0.9 0.9 ]))

   (has-part (temperate) = (moderate-temperature     D[ 0.7 0.8 ]
                                                     S[ 0.7 0.9 ]))
   (type-of (moderate-temperature) = (temperate        D[ 1.0 1.0 ]
                                                       S[ 1.0 1.0 ]))
   ))
```

ASSERT! can be used directly to get the same result as if using ADD-FACTS with a single fact:

```
(ASSERT! 'part-of 'England 'United-Kingdom
        '( D[ 0.9 0.9 ] S[ 0.95 1.0 ]))


(MAKERULE '(--> ((latitude (?x) = (>40-degrees)))
                (climate (?x) = (temperate))           L[ 0.8 0.9 ]
                                                       S[ 1.0 1.0 ]))


(MAKEDEPENDENCY '(<--> climate latitude  L[ 0.5 0.8 ]  S[ 0.6 0.9 ]))
```

APPENDIX D. EXAMPLES OF INFERENCES

This appendix presents the uncertainty calculations for two examples, a argument-based generalization inference and an implication. The inferences are based upon examples presented in Chapter 4 and the output trace in Chapter 7. The equation numbers correspond with those presented in Chapter 6 on uncertainty.

D.1. Argument-Based Generalization

```
Input Query:           (climate (Surrey) = (temperate))
------------------
Inference Operation:   Argument-Based Generalization

Variable Query:        (part-of (Surrey) = (?X))

Known:                 (part-of (Surrey) = (England      D[1.0 1.0]                S[1.0 1.0]))
          (has-part (England) = (Surrey                   D[0.1 0.2]   T[0.3 0.8]   S[1.0 1.0]

Secondary Query:       (climate (England) = (temperate))

Known:                 (climate (England) = (temperate   F[0.8 0.9]                S[0.7 0.9]))
          (inv-climate (temperate) = (England              F[0.0 1.0]                S[1.0 1.0]))
------------------
Inferred:              (climate (Surrey) = (temperate    F[0.33 1.0]               S[0.7 0.9]))
```

Frequency calculation:

In Equations 18 - 30, Chapter 6:
[a b] = [1.0 1.0] (From (part-of (Surrey) = (England)) above).
[c d] = [0.1 0.2] or [0.3 0.8] (From (has-part (England)=(Surrey)) above).
[e f] = [0.8 0.9] (From (climate (England) = (temperate)) above).
[g h] = [0.0 1.0] (From (inv-climate (temperate) = (England)) above).

$$SUB = \frac{c|d}{a} \tag{18}$$

$$= \frac{0.3}{1} = 0.3$$

$$SUP\ not\ in\ SUB = 1 - c|d \tag{19}$$

$$= 1 - 0.3 = 0.7$$

$$SUP\ with\ REF = MAX(e - SUP\ not\ in\ SUB\ , 0) = MAX(0.8 - 0.7\ , 0) = 0.1 \tag{20}$$

$$LOWER\ BOUND\ P(REF|SUB) = \frac{SUB\ with\ REF}{SUB} = \frac{0.1}{0.3} = 0.33 \tag{21}$$

$$SUB = \frac{c|d}{a|b} = \frac{0.3}{1.0} = 0.3 \tag{22}$$

$$SUB\ not\ in\ SUP = SUB - (a|b \cdot SUB) = 0.3 - (1.0 \cdot 0.3) = 0 \tag{23}$$

$$REF = \frac{e|f}{g|h} = \frac{0.8}{1.0} \tag{24}$$

$$REF\ not\ in\ SUP = REF - (g|h \cdot REF) = 0.8 - (1.0 \cdot 0.8) = 0 \tag{25}$$

$$SUB\ in\ SUP\ not\ with\ REF = MAX(c|d - e|f\ , 0) = MAX(0.3 - 0.8\ , 0) = 0 \tag{26}$$

$$SUB\ not\ in\ SUP\ not\ with\ REF = MAX(SUB\ not\ in\ SUP - REF\ not\ in\ SUP\ , 0) = MAX(0 - 0\ , 0) = 0 \tag{27}$$

$$SUB\ not\ REF = SUB\ in\ SUP\ not\ with\ REF + SUB\ not\ in\ SUP\ not\ with\ REF = 0 + 0 = 0 \tag{28}$$

$$SUB\ with\ REF = SUB - SUB\ not\ REF = 0.3 - 0 = 0.3 \tag{29}$$

$$UPPER\ BOUND\ P(REF|SUB) = \frac{SUB\ with\ REF}{SUB} = \frac{0.3}{0.3} = 1.0 \tag{30}$$

Support Calculation:

In Equations 4 and 5, Chapter 6:

S_1 = [1.0 1.0]
S_2 = [1.0 1.0]
S_3 = [0.7 0.9]
S_4 = [1.0 1.0]
S_5 = *arg-gen-supp* = [1.0 1.0]

$$s_x(DATUM) \geq MAX(1 - \textstyle\sum_{i=1}^{n}(1 - s_x(D_i)) , 0) \tag{4}$$

$$\geq MAX(1 - [(1-1)+(1-1)+(1-0.7)+(1-1)+(1-1)] , 0) = 0.7$$

$$p_x(DATUM) \leq MIN(p_x(D_i)) \quad i = 1 \ldots n \tag{5}$$
$$\leq MIN(1.0, 1.0, 0.9, 1.0, 1.0) \leq 0.9$$

D.2. Implication

Input Query:	(climate (Surrey) = (temperate))		

Inference Operation:	Implication		
Variable Query:	(→ ?X (climate (Surrey) = (temperate)))		
Known:	(→ ((latitude (?X) = (>40-degrees))) (climate (?X) = (temperate))	L[0.8 0.9]	S[1.0 1.0]
Secondary Query:	(latitude (Surrey) = (>40-degrees))		
Known:	(latitude (Surrey) = (>40-degrees	F[0.8 0.9]	S[0.8 1.0]))
	(inv-latitude (>40-degrees) = (Surrey	F[0.0 1.0]	S[1.0 1.0]))

Inferred:	(climate (Surrey) = (temperate	**F[0.64 0.92]**	**S[0.8 1.0]))**

Frequency calculation:

In Equations 11 and 13, Chapter 6:
L[t f] = [0.8 0.9] (From the known implication above).
F[t f] = [0.8 0.9] (From (latitude (Surrey) = (>40-degrees)) above).

$$s_f(CONSEQUENT) \geq s_f(ANTECEDENTS) \cdot s_f(\rightarrow ANTECEDENTS\ CONSEQUENT) \qquad (11)$$

$$\geq 0.8 \cdot 0.8 = 0.64$$

$$s_f(\neg CONSEQUENT) \geq s_f(ANTECEDENTS) \cdot (1 - p_f(\rightarrow ANTECEDENTS\ CONSEQUENT)) \qquad (12)$$

$$\geq 0.8 \cdot (1 - 0.9) \geq 0.08$$

$$p_f(CONSEQUENT) \leq 1 - s_f(\neg CONSEQUENT) \qquad (13)$$

$$\leq 1 - 0.08 \leq 0.92$$

Support calculation:

In Equations 4 and 5, Chapter 6:
S_1 = [1.0 1.0]
S_2 = [0.8 1.0]
S_3 = [1.0 1.0]

$$s_x(DATUM) \geq MAX(1 - \textstyle\sum_{i=1}^{n}(1 - s_x(D_i)) , 0) \qquad (4)$$

$$\geq MAX(1 - [(1-1)+(1-0.8)+(1-1)].\ 0) = 0.8$$

$$p_x(DATUM) \leq MIN(p_x(D_i)) \qquad i = 1 \ldots n \tag{5}$$

$$\leq MIN(1, 1, 1) = 1.0$$

APPENDIX E. GRASS IDENTIFICATION: KNOWLEDGE BASE

This is the knowledge base generated for the grass identification system. It was developed by Professor Thomas W. Fermanian of the University of Illinois. He is involved on a daily basis with the identification of tufgrasses as well as the computerized automation of this and other agricultural tasks. It is by no means an exhaustive listing of all the facts associated with the turfgrass domain but rather a reasonable portion which allowed the testing of the system. The weights were set by Dr. Fermanian based upon his intuitions.

```
(ADD-FACTS
  '(

;; GRASS

    (has-type (grass) = (poa-genus
              D[ 0.2 0.25 ]    T[ 0.85 0.95 ]   S[ 0.65 1.0 ]))
    (has-type (grass) = (agrostis-genus
              D[ 0.05 0.15 ]   T[ 0.85 0.95 ]   S[ 0.85 1.0 ]))
    (num-leaf-groups (grass) = (two
              F[ 1.0 1.0 ]                      S[ 1.0 1.0 ]))
    (vein-type (grass) = (parallel
              F[ 1.0 1.0 ]                      S[ 0.95 1.0 ]))

;; POA-GENUS

    (type-of (poa-genus) = (grass
              D[ 1.0 1.0 ]                      S[ 1.0 1.0 ]))
    (has-type (poa-genus) = (annual-bluegrass
              D[ 0.2 0.35 ]    T[ 0.65 0.85 ]   S[ 0.85 0.95 ]))
    (has-type (poa-genus) = (kentucky-bluegrass
              D[ 0.4 0.7 ]     T[ 0.85 0.95 ]   S[ 0.85 1.0 ]))
    (has-type (poa-genus) = (canada-bluegrass
              D[ 0.1 0.15 ] T[ 0.85 0.95 ]      S[ 0.65 1.0 ]))
    (has-type (poa-genus) = (roughstalk-bluegrass
              D[ 0.2 0.25 ]    T[ 0.85 0.95 ]   S[ 0.65 1.0 ]))
    (leaf-type (poa-genus) = (keel-shape
              F[ 0.90 0.95 ]                    S[ 0.9 0.95 ]))
    (inv-leaf-type (keel-shape) = (poa-genus
              F[ 0.8 0.85 ]                     S[ 0.9 0.95 ]))
    (ligule-type (poa-genus) = (membranous
              F[ 0.95 1.0 ]                     S[ 0.95 1.0 ]))
    (inv-ligule-type (membraneous) = (poa-genus
              F[ 0.1 0.15 ]                     S[ 0.65 0.75 ]))
    (vernation-type (poa-genus) = (folded
              F[ 0.95 1.0 ]                     S[ 0.85 0.95 ]))
    (inv-vernation-type (folded) = (poa-genus
               F[ 0.2 0.25 ]                      S[ 0.7 1.0 ]))
    (auricle-type (poa-genus) = (absent
               F[ 0.95 1.0 ]                      S[ 0.95 1.0 ]))
    (inv-auricle-type (absent) = (poa-genus
               F[ 0.05 0.08 ]                     S[ 0.7 1.0 ]))

;; ANNUAL-BLUEGRASS

    (type-of (annual-bluegrass) = (poa-genus
               D[ 1.0 1.0 ]                       S[ 1.0 1.0 ]))
    (ligule-shape (annual-bluegrass) = (acute
               F[ 0.65 0.8 ]                      S[ 0.8 1.0 ]))
    (inv-ligule-shape (acute) = (annual-blugrass
               F[ 0.25 0.35 ]                     S[ 0.75 1.0 ]))
    (collar-type (annual-bluegrass) = (narrow
               F[ 0.8 0.9 ]                       S[ 0.65 1.0 ]))
    (inv-collar-type (narrow) = (annual-bluegrass
               F[ 0.15 0.25 ]                     S[ 0.65 1.0 ]))
    (sheath-type (annual-bluegrass) = (compressed
               F[ 0.9 1.0 ]                       S[ 0.8 0.95 ]))
    (inv-sheath-type (compressed) = (annual-bluegrass
               F[ 0.05 0.1 ]                      S[ 0.85 0.95 ]))
    (growth-habit (annual-bluegrass) = (bunch
               F[ 0.5 0.6 ]                       S[ 0.75 1.0 ]))
    (inv-growth-habit (bunch) = (annual-bluegrass
               F[ 0.01 0.05 ]                     S[ 0.65 1.0 ]))

;; KENTUCKY-BLUEGRASS

    (type-of (kentucky-bluegrass) = (poa-genus
               D[ 1.0 1.0 ]                       S[ 1.0 1.0 ]))
    (ligule-shape (kentucky-bluegrass) = (truncate
               F[ 0.89 0.95 ]                     S[ 0.85 0.95 ]))
    (inv-ligule-shape (truncate) = (kentucky-bluegrass
               F[ 0.25 0.3 ]                      S[ 0.75 1.0 ]))
    (collar-type (kentucky-bluegrass) = (narrow
               F[ 0.45 0.65 ]                     S[ 0.85 0.95 ]))
    (inv-collar-type (narrow) = (kentucky-bluegrass
               F[ 0.05 0.15 ]                     S[ 0.75 1.0 ]))
    (collar-type (kentucky-bluegrass) = (divided
               F[ 0.45 0.8 ]                      S[ 0.8 1.0 ]))
    (inv-collar-type (divided) = (kentucky-bluegrass
               F[ 0.05 0.15 ]                     S[ 0.7 1.0 ]))
    (sheath-type (kentucky-bluegrass) = (compressed
               F[ 0.8 0.9 ]                       S[ 0.9 1.0 ]))
    (inv-sheath-type (compressed) = (kentucky-bluegrass
               F[ 0.15 0.2 ]                      S[ 0.6 1.0 ]))
    (growth-habit (kentucky-bluegrass) = (rhizome
               F[ 0.65 0.85 ]                     S[ 0.9 0.95 ]))
    (inv-growth-habit (rhizome) = (kentucky-bluegrass
               F[ 0.2 0.3 ]                       S[ 0.6 1.0 ]))

;; CANADA-BLUEGRASS
```

```
    (type-of (canada-bluegrass) = (poa-genus
                D[ 1.0 1.0 ]                S[ 1.0 1.0 ]))
    (ligule-shape (canada-bluegrass) = (acute
                F[ 0.75 0.95 ]              S[ 0.6 0.75 ]))
    (inv-ligule-shape (acute) = (canada-bluegrass
                F[ 0.1 0.15 ]               S[ 0.5 0.65 ]))
    (collar-type (canada-bluegrass) = (divided
                F[ 0.65 0.85 ]              S[ 0.5 0.65 ]))
    (inv-collar-type (divided) = (canada-bluegrass
                F[ 0.05 0.1 ]               S[ 0.5 0.65 ]))
    (sheath-type (canada-bluegrass) = (compressed
                F[ 0.6 0.8 ]                S[ 0.7 0.8 ]))
    (inv-sheath-type (compressed) = (canada-bluegrass
                F[ 0.05 0.1 ]               S[ 0.7 0.8 ]))
    (growth-habit (canada-bluegrass) = (rhizome
                F[ 0.8 0.9 ]                S[ 0.9 0.95 ]))
    (inv-growth-habit (rhizome) = (canada-bluegrass
                F[ 0.15 0.2 ]               S[ 0.8 0.9 ]))

;; ROUGHSTALK-BLUEGRASS

    (type-of (roughstalk-bluegrass) = (poa-genus
                D[ 1.0 1.0 ]                S[ 0.95 1.0 ]))
    (ligule-shape (roughstalk-bluegrass) = (toothed
                F[ 0.7 0.95 ]               S[ 0.8 0.95 ]))
    (inv-ligule-shape (toothed) = (roughstalk-bluegrass
                F[ 0.2 0.25 ]               S[ 0.65 0.8 ]))
    (collar-type (roughstalk-bluegrass) = (broad
                F[ 0.8 0.95 ]               S[ 0.7 0.9 ]))
    (inv-collar-type (broad) = (roughstalk-bluegrass
                F[ 0.05 0.1 ]               S[ 0.5 0.65 ]))
    (sheath-type (roughstalk-bluegrass) = (compressed-rough
                F[ 0.8 0.95 ]               S[ 0.9 0.95 ]))
    (inv-sheath-type (compressed-rough) = (roughstalk-bluegrass
                F[ 0.05 0.1 ]       S[ 0.7 0.75 ]))
    (growth-habit (roughstalk-bluegrass) = (stolons
                F[ 0.95 1.0 ]               S[ 0.65 0.75 ]))
    (inv-growth-habit (stolons) = (roughstalk-bluegrass
                F[ 0.1 0.15 ]               S[ 0.55 0.65 ]))

;; AGROSTIS-GENUS

    (type-of (agrostis-genus) = (grass
                D[ 1.0 1.0 ]                S[ 1.0 1.0 ]))
    (has-type (agrostis-genus) = (creeping-bentgrass
                D[ 0.4 0.45 ]  T[ 0.85 0.95 ]  S[ 0.85 0.95 ]))
    (has-type (agrostis-genus) = (colonial-bentgrass
                D[ 0.1 0.2 ]   T[ 0.6 0.7 ]    S[ 0.75 1.0 ]))
    (has-type (agrostis-genus) = (redtop
                D[ 0.3 0.35 ]  T[ 0.75 0.8 ]   S[ 0.75 1.0 ]))
    (leaf-type (agrostis-genus) = (pointed
                F[ 0.9 0.95 ]               S[ 0.75 1.0 ]))
    (inv-leaf-type (pointed) = (agrostis
                F[ 0.05 0.08 ]              S[ 0.5 0.55 ]))
    (ligule-type (agrostis-genus) = (membranous
              F[ 0.95 1.0 ]                     S[ 0.9 0.95 ]))
    (inv-ligule-type (membranous) = (agrostis-genus
              F[ 0.05 0.08 ]                    S[ 0.85 0.9 ]))
    (vernation-type (agrostis-genus) = (rolled
              F[ 0.95 1.0 ]                     S[ 0.95 1.0 ]))
    (inv-vernation-type (rolled) = (agrostis-genus
              F[ 0.1 0.15 ]                     S[ 0.65 1.0 ]))
    (auricle-type (agrostis-genus) = (absent
              F[ 0.95 1.0 ]                     S[ 0.95 1.0 ]))
    (inv-auricle-type (absent) = (agrostis
              F[ 0.01 0.05 ]                    S[ 0.95 1.0 ]))

;; CREEPING-BENTGRASS

    (type-of (creeping-bentgrass) = (agrostis-genus
              D[ 1.0 1.0 ]                      S[ 0.85 1.0 ]))
    (ligule-shape (creeping-bentgrass) = (round
              F[ 0.65 0.8 ]                     S[ 0.65 1.0 ]))
    (inv-ligule-shape (round) = (creeping-bentgrass
              F[ 0.05 0.1 ]                     S[ 0.65 1.0 ]))
    (ligule-size (creeping-bentgrass) = (tall
              F[ 0.8 0.9 ]                      S[ 0.85 0.95 ]))
    (inv-ligule-size (tall) = (creeping-bentgrass
              F[ 0.25 0.35 ]                    S[ 0.9 0.95 ]))
    (collar-type (creeping-bentgrass) = (narrow
              F[ 0.8 0.9 ]                      S[ 0.75 1.0 ]))
    (inv-collar-type (narrow) = (creeping-bentgrass
              F[ 0.1 0.15 ]                     S[ 0.5 1.0 ]))
    (sheath-type (creeping-bentgrass) = (round
              F[ 0.75 0.85 ]                    S[ 0.65 1.0 ]))
    (inv-sheath-type (round) = (creeping-bentgrass
              F[ 0.05 0.1 ]                     S[ 0.65 1.0 ]))
    (growth-habit (creeping-bentgrass) = (stolen
              F[ 0.75 0.8 ]                     S[ 0.85 1.0 ]))
    (inv-growth-habit (stolen) = (creeping-bentgrass
              F[ 0.2 0.25 ]                     S[ 0.75 1.0 ]))

;; COLONIAL-BENTGRASS

    (type-of (colonial-bentgrass) = (agrostis-genus
              D[ 1.0 1.0 ]                      S[ 0.65 1.0 ]))
    (ligule-shape (colonial-bentgrass) = (round
              F[ 0.75 0.85 ]                    S[ 0.75 1.0 ]))
    (inv-ligule-shape (round) = (colonial-bentgrass
              F[ 0.1 0.15 ]                     S[ 0.6 1.0 ]))
    (ligule-size (colonial-bentgrass) = (short
              F[ 0.75 0.85 ]                    S[ 0.65 1.0 ]))
    (inv-ligule-size (short) = (colonial-bentgrass
              F[ 0.05 0.1 ]                     S[ 0.6 1.0 ]))
    (collar-type (colonial-bentgrass) = (narrow
              F[ 0.85 0.95 ]                    S[ 0.65 1.0 ]))
    (inv-collar-type (narrow) = (colonial-bentgrass
              F[ 0.1 0.15 ]                     S[ 0.45 1.0 ]))
    (sheath-type (colonial-bentgrass) = (round
                F[ 0.85 0.95 ]                  S[ 0.6 1.0 ]))
    (inv-sheath-type (round) = (colonial-bentgrass
                F[ 0.05 0.1 ]                   S[ 0.5 1.0 ]))
    (growth-habit (colonial-bentgrass) = (stolen
                F[ 0.65 0.8 ]                   S[ 0.75 1.0 ]))
    (inv-growth-habit (stolen) = (colonial-bentgrass
                F[ 0.1 0.15 ]                   S[ 0.7 1.0 ]))

;; REDTOP

    (type-of (redtop) = (agrostis-genus
                D[ 1.0 1.0 ]                    S[ 0.65 1.0 ]))
    (ligule-shape (redtop) = (round
                F[ 0.85 0.95 ]                  S[ 0.8 1.0 ]))
    (inv-ligule-shape (round) = (redtop
                F[ 0.15 0.2 ]                   S[ 0.7 1.0 ]))
    (ligule-size (redtop) = (tall
                F[ 0.85 0.9 ]                   S[ 0.9 0.95 ]))
    (inv-ligule-size (tall) = (redtop
                F[ 0.2 0.25 ]                   S[ 0.85 0.95 ]))
    (collar-type (redtop) = (divided
                F[ 0.75 0.85 ]                  S[ 0.65 1.0 ]))
    (inv-collar-type (divided) = (redtop
                F[ 0.1 0.15 ]                   S[ 0.6 1.0 ]))
    (sheath-type (redtop) = (round
                F[ 0.8 0.95 ]                   S[ 0.75 1.0 ]))
    (inv-sheath-type (round) = (redtop
                F[ 0.05 0.1 ]                   S[ 0.65 1.0 ]))
    (growth-habit (redtop) = (rhizome
                F[ 0.8 0.95 ]                   S[ 0.9 0.95 ]))
    (inv-growth-habit (rhizome) = (redtop
                F[ 0.1 0.2 ]                    S[ 0.75 1.0 ]))))

(SET-INVERSE 'leaf-type 'inv-leaf-type)
(SET-INVERSE 'ligule-shape 'inv-ligule-shape)
(SET-INVERSE 'ligule-size 'inv-ligule-size)
(SET-INVERSE 'ligule-type 'inv-ligule-type)
(SET-INVERSE 'collar-type 'inv-collar-type)
(SET-INVERSE 'sheath-type 'inv-sheath-type)
(SET-INVERSE 'growth-habit 'inv-growth-habit)
(SET-INVERSE 'auricle-type 'inv-auricle-type)
(SET-INVERSE 'vernation-type 'inv-vernation-type)
```

APPENDIX F. GRASS IDENTIFICATION: EXAMPLE OUTPUT

This appendix lists an annotated listing of system output when operating in identification mode (described in Chapter 7) in the grass domain (Chapter 8). The example used was selected because it is representative of system operation in this mode and domain.

```
Command: (RESET-PRS-ID)

Command: (ADD-ID-FACTS
          '((collar-type (?x) = (narrow F[ 0.3 0.6 ]    S[ 0.9 1.0 ]))
            (ligule-shape (?x) = (truncate F[ 0.8 0.9 ] S[ 0.9 1.0 ]))))

Command: (SET-POSSIBILITIES 'poa-genus 'agrostis-genus)

Command: (RUN-PRS-ID)

The possibilities for the identification of the unknown are:
   POA-GENUS
   AGROSTIS-GENUS

The known facts are:
   1. (COLLAR-TYPE (?X) = (NARROW))
        freq: 0.30 .. 0.60   supp: 0.90 .. 1.00
   2. (LIGULE-SHAPE (?X) = (TRUNCATE))
        freq: 0.80 .. 0.90   supp: 0.90 .. 1.00
```

Cross each possibility with each given fact and attempt to infer the result.
In this example evidence will be found for four queries. First query:

```
Next Operation:    (1)
 (CHECK-QUERY '(COLLAR-TYPE (AGROSTIS-GENUS) = (NARROW)))
    time = 0          path length = 0
    worth = 100       rel. time = 0
 Unknown...Adding stack elements:
 (GET-ARGUMENT-GENERALIZATIONS '(COLLAR-TYPE (AGROSTIS-GENUS) =
                                                        (NARROW)))
 (GET-ARGUMENT-SPECIALIZATIONS '(COLLAR-TYPE (AGROSTIS-GENUS) =
                                                        (NARROW)))
 (GET-REFERENT-GENERALIZATIONS '(COLLAR-TYPE (AGROSTIS-GENUS) =
                                                        (NARROW)))
 (GET-REFERENT-SPECIALIZATIONS '(COLLAR-TYPE (AGROSTIS-GENUS) =
                                                        (NARROW)))
 (BACKCHAIN '(COLLAR-TYPE (AGROSTIS-GENUS) = (NARROW)))
 (GET-DEPENDENCIES '(COLLAR-TYPE (AGROSTIS-GENUS) = (NARROW)))

Next Operation:    (2)
 (BACKCHAIN '(COLLAR-TYPE (AGROSTIS-GENUS) = (NARROW)))
```

```
    time = 1          path length = 0
    worth = 90.0      rel. time = 1
  None found.

Next Operation:   (3)
 (GET-ARGUMENT-GENERALIZATIONS '(COLLAR-TYPE (AGROSTIS-GENUS) =
                                                          (NARROW)))
    time = 1          path length = 0
    worth = 90.0      rel. time = 1
 Found (TYPE-OF (AGROSTIS-GENUS) = (GRASS))
 Adding to stack:
 (CHECK-QUERY '(COLLAR-TYPE (GRASS) = (NARROW)))

Next Operation:   (4)
 (CHECK-QUERY '(COLLAR-TYPE (GRASS) = (NARROW)))
    time = 1          path length = 2
    worth = 90.0      rel. time = 1
 Unknown...Adding stack elements:
 (GET-ARGUMENT-GENERALIZATIONS '(COLLAR-TYPE (GRASS) = (NARROW)))
 (GET-ARGUMENT-SPECIALIZATIONS '(COLLAR-TYPE (GRASS) = (NARROW)))
 (GET-REFERENT-GENERALIZATIONS '(COLLAR-TYPE (GRASS) = (NARROW)))
 (GET-REFERENT-SPECIALIZATIONS '(COLLAR-TYPE (GRASS) = (NARROW)))
 (BACKCHAIN '(COLLAR-TYPE (GRASS) = (NARROW)))
 (GET-DEPENDENCIES '(COLLAR-TYPE (GRASS) = (NARROW)))

Next Operation:   (5)
 (BACKCHAIN '(COLLAR-TYPE (GRASS) = (NARROW)))
    time = 2          path length = 2
    worth = 81.0      rel. time = 2
  None found.

  .
  .
  .

Next Operation:   (12)
 (GET-ARGUMENT-SPECIALIZATIONS '(COLLAR-TYPE (AGROSTIS-GENUS) =
                                                          (NARROW)))
    time = 6          path length = 0
    worth = 85.0      rel. time = 6
```

Found three specializations of Agrostis-Genus:

```
 Found (HAS-TYPE (AGROSTIS-GENUS) = (CREEPING-BENTGRASS))
 Adding to stack:
 (CHECK-QUERY '(COLLAR-TYPE (CREEPING-BENTGRASS) = (NARROW)))
 Found (HAS-TYPE (AGROSTIS-GENUS) = (COLONIAL-BENTGRASS))
 Adding to stack:
 (CHECK-QUERY '(COLLAR-TYPE (COLONIAL-BENTGRASS) = (NARROW)))
 Found (HAS-TYPE (AGROSTIS-GENUS) = (REDTOP))
 Adding to stack:
 (CHECK-QUERY '(COLLAR-TYPE (REDTOP) = (NARROW)))
```

```
Next Operation:   (13)
 (CHECK-QUERY '(COLLAR-TYPE (REDTOP) = (NARROW)))
    time = 7         path length = 2
    worth = 85.0     rel. time = 7
 Unknown...Adding stack elements:
 (GET-ARGUMENT-GENERALIZATIONS '(COLLAR-TYPE (REDTOP) = (NARROW)))
 (GET-ARGUMENT-SPECIALIZATIONS '(COLLAR-TYPE (REDTOP) = (NARROW)))
 (GET-REFERENT-GENERALIZATIONS '(COLLAR-TYPE (REDTOP) = (NARROW)))
 (GET-REFERENT-SPECIALIZATIONS '(COLLAR-TYPE (REDTOP) = (NARROW)))
 (BACKCHAIN '(COLLAR-TYPE (REDTOP) = (NARROW)))
 (GET-DEPENDENCIES '(COLLAR-TYPE (REDTOP) = (NARROW)))

Next Operation:   (14)
 (BACKCHAIN '(COLLAR-TYPE (REDTOP) = (NARROW)))
    time = 9         path length = 2
    worth = 76.5     rel. time = 9
  None found.

 .
 .
 .

Next Operation:   (17)
 (GET-ARGUMENT-SPECIALIZATIONS '(COLLAR-TYPE (REDTOP) = (NARROW)))
    time = 11        path length = 2
    worth = 72.25    rel. time = 11
```

Relative time exceeded (Rel. time is the time when initial operation - in this case (check-query '(collar-type (agrostis-genus) = (narrow))) - was popped off stack. Rel. time is reset when element is put at bottom of stack.

```
  Rel. time (11) exceeds limit (10) - Operation put at bottom of stack.

 .
 .
 .
```

Second query, out of the four to be processed:

```
Next Operation:   (24)
 (CHECK-QUERY '(LIGULE-SHAPE (AGROSTIS-GENUS) = (TRUNCATE)))
    time = 14        path length = 0
    worth = 100      rel. time = 0
 Unknown...Adding stack elements:
 (GET-ARGUMENT-GENERALIZATIONS '(LIGULE-SHAPE (AGROSTIS-GENUS) =
                                                      (TRUNCATE)))
 (GET-ARGUMENT-SPECIALIZATIONS '(LIGULE-SHAPE (AGROSTIS-GENUS) =
                                                      (TRUNCATE)))
 (GET-REFERENT-GENERALIZATIONS '(LIGULE-SHAPE (AGROSTIS-GENUS) =
                                                      (TRUNCATE)))
 (GET-REFERENT-SPECIALIZATIONS '(LIGULE-SHAPE (AGROSTIS-GENUS) =
                                                      (TRUNCATE)))
 (BACKCHAIN '(LIGULE-SHAPE (AGROSTIS-GENUS) = (TRUNCATE)))
 (GET-DEPENDENCIES '(LIGULE-SHAPE (AGROSTIS-GENUS) = (TRUNCATE)))

Next Operation:   (25)
 (BACKCHAIN '(LIGULE-SHAPE (AGROSTIS-GENUS) = (TRUNCATE)))
    time = 15       path length = 0
    worth = 90.0    rel. time = 1
  None found.

Next Operation:   (26)
 (GET-ARGUMENT-GENERALIZATIONS '(LIGULE-SHAPE (AGROSTIS-GENUS) =
                                                      (TRUNCATE)))
    time = 16       path length = 0
    worth = 90.0    rel. time = 2
 Found (TYPE-OF (AGROSTIS-GENUS) = (GRASS))
 Adding to stack:
 (CHECK-QUERY '(LIGULE-SHAPE (GRASS) = (TRUNCATE)))

Next Operation:   (27)
 (CHECK-QUERY '(LIGULE-SHAPE (GRASS) = (TRUNCATE)))
    time = 17       path length = 2
    worth = 90.0    rel. time = 3
 Unknown...Adding stack elements:
 (GET-ARGUMENT-GENERALIZATIONS '(LIGULE-SHAPE (GRASS) = (TRUNCATE)))
 (GET-ARGUMENT-SPECIALIZATIONS '(LIGULE-SHAPE (GRASS) = (TRUNCATE)))
 (GET-REFERENT-GENERALIZATIONS '(LIGULE-SHAPE (GRASS) = (TRUNCATE)))
 (GET-REFERENT-SPECIALIZATIONS '(LIGULE-SHAPE (GRASS) = (TRUNCATE)))
 (BACKCHAIN '(LIGULE-SHAPE (GRASS) = (TRUNCATE)))
 (GET-DEPENDENCIES '(LIGULE-SHAPE (GRASS) = (TRUNCATE)))

Next Operation:   (28)
 (BACKCHAIN '(LIGULE-SHAPE (GRASS) = (TRUNCATE)))
    time = 18       path length = 2
    worth = 81.0    rel. time = 4
  None found.

  .
  .
  .

Next Operation:   (33)
 (GET-DEPENDENCIES '(LIGULE-SHAPE (GRASS) = (TRUNCATE)))
    time = 21       path length = 2
    worth = 67.5    rel. time = 7
```

Relative worth limit is 70% of worth of element initially popped from stack - in this case 100 from (check-query '(ligule-shape (agrostis-genus) = (truncate)))

```
  Worth (67.5) below relative limit (70.0) - Operation put at
                                             bottom of stack.

Next Operation:   (34)
 (GET-REFERENT-GENERALIZATIONS '(LIGULE-SHAPE (AGROSTIS-GENUS) =
                                                         (TRUNCATE)))
   time = 21         path length = 0
   worth = 85.0      rel. time = 7
 None found.

Next Operation:   (35)
 (GET-ARGUMENT-SPECIALIZATIONS '(LIGULE-SHAPE (AGROSTIS-GENUS) =
                                                         (TRUNCATE)))
   time = 22         path length = 0
   worth = 85.0      rel. time = 8
```

Again, three specializations of Agrostis-Genus:

```
 Found (HAS-TYPE (AGROSTIS-GENUS) = (CREEPING-BENTGRASS))
 Adding to stack:
 (CHECK-QUERY '(LIGULE-SHAPE (CREEPING-BENTGRASS) = (TRUNCATE)))
 Found (HAS-TYPE (AGROSTIS-GENUS) = (COLONIAL-BENTGRASS))
 Adding to stack:
 (CHECK-QUERY '(LIGULE-SHAPE (COLONIAL-BENTGRASS) = (TRUNCATE)))
 Found (HAS-TYPE (AGROSTIS-GENUS) = (REDTOP))
 Adding to stack:
 (CHECK-QUERY '(LIGULE-SHAPE (REDTOP) = (TRUNCATE)))

Next Operation:   (36)
 (CHECK-QUERY '(LIGULE-SHAPE (REDTOP) = (TRUNCATE)))
   time = 23         path length = 2
   worth = 85.0      rel. time = 9
 Unknown...Adding stack elements:
 (GET-ARGUMENT-GENERALIZATIONS '(LIGULE-SHAPE (REDTOP) = (TRUNCATE)))
 (GET-ARGUMENT-SPECIALIZATIONS '(LIGULE-SHAPE (REDTOP) = (TRUNCATE)))
 (GET-REFERENT-GENERALIZATIONS '(LIGULE-SHAPE (REDTOP) = (TRUNCATE)))
 (GET-REFERENT-SPECIALIZATIONS '(LIGULE-SHAPE (REDTOP) = (TRUNCATE)))
 (BACKCHAIN '(LIGULE-SHAPE (REDTOP) = (TRUNCATE)))
 (GET-DEPENDENCIES '(LIGULE-SHAPE (REDTOP) = (TRUNCATE)))

Next Operation:   (37)
 (BACKCHAIN '(LIGULE-SHAPE (REDTOP) = (TRUNCATE)))
   time = 25         path length = 2
   worth = 76.5      rel. time = 11
  Rel. time (11) exceeds limit (10) - Operation put at bottom of stack.

Next Operation:   (38)
 (GET-ARGUMENT-GENERALIZATIONS '(LIGULE-SHAPE (REDTOP) = (TRUNCATE)))
   time = 25         path length = 2
   worth = 76.5      rel. time = 11
```

Time when (check-query '(ligule-shape (agrostis-genus) = (truncate))) came off stack was 14. Current time minus that is over threshold:

```
  Rel. time (11) exceeds limit (10) - Operation put at bottom of stack.
```

.
.
.

Third query:

```
Next Operation:   (47)
 (CHECK-QUERY '(COLLAR-TYPE (POA-GENUS) = (NARROW)))
    time = 30        path length = 0
    worth = 100      rel. time = 0
 Unknown...Adding stack elements:
 (GET-ARGUMENT-GENERALIZATIONS '(COLLAR-TYPE (POA-GENUS) = (NARROW)))
 (GET-ARGUMENT-SPECIALIZATIONS '(COLLAR-TYPE (POA-GENUS) = (NARROW)))
 (GET-REFERENT-GENERALIZATIONS '(COLLAR-TYPE (POA-GENUS) = (NARROW)))
 (GET-REFERENT-SPECIALIZATIONS '(COLLAR-TYPE (POA-GENUS) = (NARROW)))
 (BACKCHAIN '(COLLAR-TYPE (POA-GENUS) = (NARROW)))
 (GET-DEPENDENCIES '(COLLAR-TYPE (POA-GENUS) = (NARROW)))

Next Operation:   (48)
 (BACKCHAIN '(COLLAR-TYPE (POA-GENUS) = (NARROW)))
    time = 31        path length = 0
    worth = 90.0     rel. time = 1
  None found.

Next Operation:   (49)
 (GET-ARGUMENT-GENERALIZATIONS '(COLLAR-TYPE (POA-GENUS) = (NARROW)))
    time = 32        path length = 0
    worth = 90.0     rel. time = 2
```

Found one generalization of Poa-Genus:

```
 Found (TYPE-OF (POA-GENUS) = (GRASS))
 Adding to stack:
 (CHECK-QUERY '(COLLAR-TYPE (GRASS) = (NARROW)))

  .
  .
  .

Next Operation:   (52)
 (GET-ARGUMENT-SPECIALIZATIONS '(COLLAR-TYPE (POA-GENUS) = (NARROW)))
    time = 33        path length = 0
    worth = 85.0     rel. time = 3
```

Four specializations of Poa-Genus:

```
 Found (HAS-TYPE (POA-GENUS) = (ANNUAL-BLUEGRASS))
 Adding to stack:
 (CHECK-QUERY '(COLLAR-TYPE (ANNUAL-BLUEGRASS) = (NARROW)))
 Found (HAS-TYPE (POA-GENUS) = (KENTUCKY-BLUEGRASS))
 Adding to stack:
 (CHECK-QUERY '(COLLAR-TYPE (KENTUCKY-BLUEGRASS) = (NARROW)))
 Found (HAS-TYPE (POA-GENUS) = (CANADA-BLUEGRASS))
 Adding to stack:
 (CHECK-QUERY '(COLLAR-TYPE (CANADA-BLUEGRASS) = (NARROW)))
 Found (HAS-TYPE (POA-GENUS) = (ROUGHSTALK))
 Adding to stack:
  (CHECK-QUERY '(COLLAR-TYPE (ROUGHSTALK) = (NARROW)))

  .
  .
  .

Next Operation:   (60)
  (CHECK-QUERY '(COLLAR-TYPE (CANADA-BLUEGRASS) = (NARROW)))
     time = 40         path length = 2
     worth = 85.0      rel. time = 10
 Unknown...Adding stack elements:
  (GET-ARGUMENT-GENERALIZATIONS '(COLLAR-TYPE (CANADA-BLUEGRASS) =
                                                          (NARROW)))
  (GET-ARGUMENT-SPECIALIZATIONS '(COLLAR-TYPE (CANADA-BLUEGRASS) =
                                                          (NARROW)))
  (GET-REFERENT-GENERALIZATIONS '(COLLAR-TYPE (CANADA-BLUEGRASS) =
                                                          (NARROW)))
  (GET-REFERENT-SPECIALIZATIONS '(COLLAR-TYPE (CANADA-BLUEGRASS) =
                                                          (NARROW)))
  (BACKCHAIN '(COLLAR-TYPE (CANADA-BLUEGRASS) = (NARROW)))
  (GET-DEPENDENCIES '(COLLAR-TYPE (CANADA-BLUEGRASS) = (NARROW)))

  .
  .
  .
```

Fourth query:

```
Next Operation:   (71)
  (CHECK-QUERY '(LIGULE-SHAPE (POA-GENUS) = (TRUNCATE)))
     time = 46         path length = 0
     worth = 100       rel. time = 0
 Unknown...Adding stack elements:
  (GET-ARGUMENT-GENERALIZATIONS '(LIGULE-SHAPE (POA-GENUS) = (TRUNCATE)))
  (GET-ARGUMENT-SPECIALIZATIONS '(LIGULE-SHAPE (POA-GENUS) = (TRUNCATE)))
  (GET-REFERENT-GENERALIZATIONS '(LIGULE-SHAPE (POA-GENUS) = (TRUNCATE)))
  (GET-REFERENT-SPECIALIZATIONS '(LIGULE-SHAPE (POA-GENUS) = (TRUNCATE)))
  (BACKCHAIN '(LIGULE-SHAPE (POA-GENUS) = (TRUNCATE)))
  (GET-DEPENDENCIES '(LIGULE-SHAPE (POA-GENUS) = (TRUNCATE)))

Next Operation:   (72)
  (BACKCHAIN '(LIGULE-SHAPE (POA-GENUS) = (TRUNCATE)))
     time = 48         path length = 0
     worth = 90.0      rel. time = 2
  None found.

Next Operation:   (73)
  (GET-ARGUMENT-GENERALIZATIONS '(LIGULE-SHAPE (POA-GENUS) = (TRUNCATE)))
     time = 48         path length = 0
     worth = 90.0      rel. time = 2
 Found (TYPE-OF (POA-GENUS) = (GRASS))
 Adding to stack:
```

```
 (CHECK-QUERY '(LIGULE-SHAPE (GRASS) = (TRUNCATE)))

Next Operation:   (74)
 (CHECK-QUERY '(LIGULE-SHAPE (GRASS) = (TRUNCATE)))
    time = 49        path length = 2
    worth = 90.0     rel. time = 3
```

This operation matches an operation previously tried - do not repeat (this path will use any results found from corresponding operation).

```
  Operation matches previous operation 27 .

Next Operation:   (75)
 (GET-REFERENT-GENERALIZATIONS '(LIGULE-SHAPE (POA-GENUS) = (TRUNCATE)))
    time = 50        path length = 0
    worth = 85.0     rel. time = 4
  None found.

Next Operation:   (76)
 (GET-ARGUMENT-SPECIALIZATIONS '(LIGULE-SHAPE (POA-GENUS) = (TRUNCATE)))
    time = 50        path length = 0
    worth = 85.0     rel. time = 4
 Found (HAS-TYPE (POA-GENUS) = (ANNUAL-BLUEGRASS))
 Adding to stack:
 (CHECK-QUERY '(LIGULE-SHAPE (ANNUAL-BLUEGRASS) = (TRUNCATE)))
 Found (HAS-TYPE (POA-GENUS) = (KENTUCKY-BLUEGRASS))
 Adding to stack:
 (CHECK-QUERY '(LIGULE-SHAPE (KENTUCKY-BLUEGRASS) = (TRUNCATE)))
 Found (HAS-TYPE (POA-GENUS) = (CANADA-BLUEGRASS))
 Adding to stack:
 (CHECK-QUERY '(LIGULE-SHAPE (CANADA-BLUEGRASS) = (TRUNCATE)))
 Found (HAS-TYPE (POA-GENUS) = (ROUGHSTALK))
 Adding to stack:
 (CHECK-QUERY '(LIGULE-SHAPE (ROUGHSTALK) = (TRUNCATE)))

Next Operation:   (77)
 (CHECK-QUERY '(LIGULE-SHAPE (ROUGHSTALK) = (TRUNCATE)))
    time = 52        path length = 2
    worth = 85.0     rel. time = 6
 Unknown...Adding stack elements:
 (GET-ARGUMENT-GENERALIZATIONS '(LIGULE-SHAPE (ROUGHSTALK) =
                                                         (TRUNCATE)))
 (GET-ARGUMENT-SPECIALIZATIONS '(LIGULE-SHAPE (ROUGHSTALK) =
                                                         (TRUNCATE)))
 (GET-REFERENT-GENERALIZATIONS '(LIGULE-SHAPE (ROUGHSTALK) =
                                                         (TRUNCATE)))
 (GET-REFERENT-SPECIALIZATIONS '(LIGULE-SHAPE (ROUGHSTALK) =
                                                         (TRUNCATE)))
 (BACKCHAIN '(LIGULE-SHAPE (ROUGHSTALK) = (TRUNCATE)))
 (GET-DEPENDENCIES '(LIGULE-SHAPE (ROUGHSTALK) = (TRUNCATE)))

Next Operation:   (78)
 (BACKCHAIN '(LIGULE-SHAPE (ROUGHSTALK) = (TRUNCATE)))
    time = 53        path length = 2
    worth = 76.5    rel. time = 7
  None found.

   .
   .
   .

Next Operation:   (84)
 (CHECK-QUERY '(LIGULE-SHAPE (CANADA-BLUEGRASS) = (TRUNCATE)))
    time = 56        path length = 2
    worth = 85.0     rel. time = 10
 Unknown...Adding stack elements:
 (GET-ARGUMENT-GENERALIZATIONS '(LIGULE-SHAPE (CANADA-BLUEGRASS) =
                                                        (TRUNCATE)))
 (GET-ARGUMENT-SPECIALIZATIONS '(LIGULE-SHAPE (CANADA-BLUEGRASS) =
                                                        (TRUNCATE)))
 (GET-REFERENT-GENERALIZATIONS '(LIGULE-SHAPE (CANADA-BLUEGRASS) =
                                                        (TRUNCATE)))
 (GET-REFERENT-SPECIALIZATIONS '(LIGULE-SHAPE (CANADA-BLUEGRASS) =
                                                        (TRUNCATE)))
 (BACKCHAIN '(LIGULE-SHAPE (CANADA-BLUEGRASS) = (TRUNCATE)))
 (GET-DEPENDENCIES '(LIGULE-SHAPE (CANADA-BLUEGRASS) = (TRUNCATE)))

   .
   .
   .

Next Operation:   (99)
 (CHECK-QUERY '(COLLAR-TYPE (COLONIAL-BENTGRASS) = (NARROW)))
    time = 65        path length = 2
    worth = 85.0     rel. time = 0
 Found (COLLAR-TYPE (COLONIAL-BENTGRASS) = (NARROW))


--------------------------Evidence 1--------------------------

 Found Evidence:
  It is known that
   (HAS-TYPE (AGROSTIS-GENUS) = (COLONIAL-BENTGRASS))
        dom: 0.10 .. 0.20  typ: 0.60 .. 0.70 supp: 0.75 .. 1.00
   (COLLAR-TYPE (COLONIAL-BENTGRASS) = (NARROW))
        freq : 0.85 .. 0.95                  supp: 0.65 .. 1.00

  Thus (COLLAR-TYPE (AGROSTIS-GENUS) = (NARROW))
      freq: 0.51 .. 0.97                     supp: 0.00 .. 1.00
  This is evidence that the identification of the unknown is
    AGROSTIS-GENUS.


Next Operation:   (100)
 (CHECK-QUERY '(COLLAR-TYPE (CREEPING-BENTGRASS) = (NARROW)))
    time = 67        path length = 2
    worth = 85.0     rel. time = 0
 Found (COLLAR-TYPE (CREEPING-BENTGRASS) = (NARROW))


--------------------------Evidence 2--------------------------

 Found Evidence:
  It is known that
   (HAS-TYPE (AGROSTIS-GENUS) = (CREEPING-BENTGRASS))
        dom: 0.40 .. 0.45  typ: 0.85 .. 0.95 supp: 0.85 .. 0.95
   (COLLAR-TYPE (CREEPING-BENTGRASS) = (NARROW))
        freq : 0.80 .. 0.90                 supp: 0.75 .. 1.00

  Thus (COLLAR-TYPE (AGROSTIS-GENUS) = (NARROW))
      freq: 0.68 .. 0.91                    supp: 0.00 .. 0.95
  This is evidence that the identification of the unknown is
    AGROSTIS-GENUS.


Next Operation:   (101)
 (GET-REFERENT-SPECIALIZATIONS '(COLLAR-TYPE (AGROSTIS-GENUS) =
                                                            (NARROW)))
    time = 69        path length = 0
    worth = 80.0     rel. time = 0
  None found.

  .
  .
  .


Next Operation:   (109)
 (GET-DEPENDENCIES '(LIGULE-SHAPE (REDTOP) = (TRUNCATE)))
    time = 73        path length = 2
    worth = 63.75    rel. time = 0
  None found.

Next Operation:   (110)
 (CHECK-QUERY '(LIGULE-SHAPE (COLONIAL-BENTGRASS) = (TRUNCATE)))
    time = 74        path length = 2
    worth = 85.0     rel. time = 0
 Unknown...Adding stack elements:
 (GET-ARGUMENT-GENERALIZATIONS '(LIGULE-SHAPE (COLONIAL-BENTGRASS)
                                                  = (TRUNCATE)))
 (GET-ARGUMENT-SPECIALIZATIONS '(LIGULE-SHAPE (COLONIAL-BENTGRASS)
                                                  = (TRUNCATE)))
 (GET-REFERENT-GENERALIZATIONS '(LIGULE-SHAPE (COLONIAL-BENTGRASS)
                                                  = (TRUNCATE)))
 (GET-REFERENT-SPECIALIZATIONS '(LIGULE-SHAPE (COLONIAL-BENTGRASS)
                                                  = (TRUNCATE)))
 (BACKCHAIN '(LIGULE-SHAPE (COLONIAL-BEN[illegible]SS) = (TRUNCATE)))
 (GET-DEPENDENCIES '(LIGULE-SHAPE (COLONI[illegible]-BENTGRASS) = (TRUNCATE)))

Next Operation:   (111)
```

```
 (BACKCHAIN '(LIGULE-SHAPE (COLONIAL-BENTGRASS) = (TRUNCATE)))
   time = 75        path length = 2
   worth = 76.5     rel. time = 1
  None found.

 .
 .
 .

Next Operation:   (117)
 (CHECK-QUERY '(LIGULE-SHAPE (CREEPING-BENTGRASS) = (TRUNCATE)))
   time = 78        path length = 2
   worth = 85.0     rel. time = 0
 Unknown...Adding stack elements:
 (GET-ARGUMENT-GENERALIZATIONS '(LIGULE-SHAPE (CREEPING-BENTGRASS)
                                                    = (TRUNCATE)))
 (GET-ARGUMENT-SPECIALIZATIONS '(LIGULE-SHAPE (CREEPING-BENTGRASS)
                                                    = (TRUNCATE)))
 (GET-REFERENT-GENERALIZATIONS '(LIGULE-SHAPE (CREEPING-BENTGRASS)
                                                    = (TRUNCATE)))
 (GET-REFERENT-SPECIALIZATIONS '(LIGULE-SHAPE (CREEPING-BENTGRASS)
                                                    = (TRUNCATE)))
 (BACKCHAIN '(LIGULE-SHAPE (CREEPING-BENTGRASS) = (TRUNCATE)))
 (GET-DEPENDENCIES '(LIGULE-SHAPE (CREEPING-BENTGRASS) = (TRUNCATE)))

 .
 .
 .

Next Operation:   (134)
 (CHECK-QUERY '(COLLAR-TYPE (KENTUCKY-BLUEGRASS) = (NARROW)))
   time = 88        path length = 2
   worth = 85.0     rel. time = 0
 Found (COLLAR-TYPE (KENTUCKY-BLUEGRASS) = (NARROW))

--------------------------Evidence 3--------------------------

 Found Evidence:
  It is known that
   (HAS-TYPE (POA-GENUS) = (KENTUCKY-BLUEGRASS))
        dom: 0.40 .. 0.70  typ: 0.85 .. 0.95 supp: 0.85 .. 1.00
   (COLLAR-TYPE (KENTUCKY-BLUEGRASS) = (NARROW))
        freq : 0.45 .. 0.65                  supp: 0.85 .. 0.95

  Thus (COLLAR-TYPE (POA-GENUS) = (NARROW))
      freq: 0.38 .. 0.70                     supp: 0.45 .. 0.95
  This is evidence that the identification of the unknown is
    POA-GENUS.

Next Operation:   (135)
```

```
 (CHECK-QUERY '(COLLAR-TYPE (ANNUAL-BLUEGRASS) = (NARROW)))
    time = 90        path length = 2
    worth = 85.0     rel. time = 0
 Found (COLLAR-TYPE (ANNUAL-BLUEGRASS) = (NARROW))


-------------------------Evidence 4-------------------------

 Found Evidence:
  It is known that
   (HAS-TYPE (POA-GENUS) = (ANNUAL-BLUEGRASS))
        dom: 0.20 .. 0.35  typ: 0.65 .. 0.85 supp: 0.85 .. 0.95
   (COLLAR-TYPE (ANNUAL-BLUEGRASS) = (NARROW))
        freq : 0.80 .. 0.90                  supp: 0.65 .. 1.00

  Thus (COLLAR-TYPE (POA-GENUS) = (NARROW))
      freq: 0.52 .. 0.94                     supp: 0.15 .. 0.95
  This is evidence that the identification of the unknown is
    POA-GENUS.


Next Operation:   (136)
 (GET-REFERENT-SPECIALIZATIONS '(COLLAR-TYPE (POA-GENUS) = (NARROW)))
    time = 92        path length = 0
    worth = 80.0     rel. time = 0
  None found.

Next Operation:   (137)
 (GET-DEPENDENCIES '(COLLAR-TYPE (POA-GENUS) = (NARROW)))
    time = 92        path length = 0
    worth = 75.0     rel. time = 0
  None found.

 .
 .
 .


Next Operation:   (146)
 (CHECK-QUERY '(LIGULE-SHAPE (KENTUCKY-BLUEGRASS) = (TRUNCATE)))
    time = 97        path length = 2
    worth = 85.0     rel. time = 0
 Found (LIGULE-SHAPE (KENTUCKY-BLUEGRASS) = (TRUNCATE))


-------------------------Evidence 5-------------------------

 Found Evidence:
  It is known that
   (HAS-TYPE (POA-GENUS) = (KENTUCKY-BLUEGRASS))
        dom: 0.40 .. 0.70  typ: 0.85 .. 0.95 supp: 0.85 .. 1.00
   (LIGULE-SHAPE (KENTUCKY-BLUEGRASS) = (TRUNCATE))
        freq : 0.89 .. 0.95                  supp: 0.85 .. 0.95
  Thus (LIGULE-SHAPE (POA-GENUS) = (TRUNCATE))
      freq: 0.76 .. 0.96                      supp: 0.45 .. 0.95
  This is evidence that the identification of the unknown is
    POA-GENUS.


Next Operation:   (147)
 (CHECK-QUERY '(LIGULE-SHAPE (ANNUAL-BLUEGRASS) = (TRUNCATE)))
    time = 99        path length = 2
    worth = 85.0     rel. time = 0
 Unknown...Adding stack elements:
 (GET-ARGUMENT-GENERALIZATIONS '(LIGULE-SHAPE (ANNUAL-BLUEGRASS) =
                                                        (TRUNCATE)))
 (GET-ARGUMENT-SPECIALIZATIONS '(LIGULE-SHAPE (ANNUAL-BLUEGRASS) =
                                                        (TRUNCATE)))
 (GET-REFERENT-GENERALIZATIONS '(LIGULE-SHAPE (ANNUAL-BLUEGRASS) =
                                                        (TRUNCATE)))
 (GET-REFERENT-SPECIALIZATIONS '(LIGULE-SHAPE (ANNUAL-BLUEGRASS) =
                                                        (TRUNCATE)))
 (BACKCHAIN '(LIGULE-SHAPE (ANNUAL-BLUEGRASS) = (TRUNCATE)))
 (GET-DEPENDENCIES '(LIGULE-SHAPE (ANNUAL-BLUEGRASS) = (TRUNCATE)))

Next Operation:   (148)
 (BACKCHAIN '(LIGULE-SHAPE (ANNUAL-BLUEGRASS) = (TRUNCATE)))
    time = 100       path length = 2
    worth = 76.5     rel. time = 1
  None found.

Next Operation:   (149)
 (GET-ARGUMENT-GENERALIZATIONS '(LIGULE-SHAPE (ANNUAL-BLUEGRASS) =
                                                        (TRUNCATE)))
    time = 101       path length = 2
    worth = 76.5     rel. time = 2
  None found.

   .
   .
   .


Next Operation:   (155)
 (GET-DEPENDENCIES '(LIGULE-SHAPE (POA-GENUS) = (TRUNCATE)))
    time = 104       path length = 0
    worth = 75.0     rel. time = 0
  None found.

PROCESSING TERMINATED - No more elements in stack.

SUMMARIZING EVIDENCE:

The possibilities for the identification of the unknown are:
   POA-GENUS
   AGROSTIS-GENUS
```

```
The known facts are:
   1. (COLLAR-TYPE (?X) = (NARROW))
       freq: 0.30 .. 0.60   supp: 0.90 .. 1.00
   2. (LIGULE-SHAPE (?X) = (TRUNCATE))
       freq: 0.80 .. 0.90   supp: 0.90 .. 1.00

  Inferred evidence for each choice is as follows:
    AGROSTIS-GENUS
     1) (COLLAR-TYPE (AGROSTIS-GENUS) = (NARROW))
         freq: 0.51 .. 0.97   supp: 0.00 .. 1.00
     2) (COLLAR-TYPE (AGROSTIS-GENUS) = (NARROW))
         freq: 0.68 .. 0.91   supp: 0.00 .. 0.95
    POA-GENUS
     3) (COLLAR-TYPE (POA-GENUS) = (NARROW))
         freq: 0.38 .. 0.70   supp: 0.45 .. 0.95
     4) (COLLAR-TYPE (POA-GENUS) = (NARROW))
         freq: 0.52 .. 0.94   supp: 0.15 .. 0.95
     5) (LIGULE-SHAPE (POA-GENUS) = (TRUNCATE))
         freq: 0.76 .. 0.96   supp: 0.45 .. 0.95

 Type (see-evidence #) to see the inference path for a particular
    evidence number.
 Type (match-evidence-to-facts) to match inferred facts to given
    facts.
 Use  (add-id-facts '((...) to add new initial facts or revise
    existing facts.
 Type (run-prs-id t) to run PRS incrementally, after adding or
    modifying facts.
 Type (summarize-evidence-id) to see this summary.

(Total time: 107 seconds)
```

Calling SEE-EVIDENCE with a 2nd argument of t shows "hidden" queries also - "hidden" queries are the internal queries used to find inverses.

```
Command: (SEE-EVIDENCE 1 t)

---------------------------Evidence #1---------------------------

 (check-query '(collar-type (agrostis-genus) = (narrow)))
 (get-argument-specializations '(collar-type (agrostis-genus) =
                                                          (narrow)))
   (HAS-TYPE (AGROSTIS-GENUS) = (COLONIAL-BENTGRASS))
        dom: 0.10 .. 0.20  typ: 0.60 .. 0.70 supp: 0.75 .. 1.00
 (get-inverse '(has-type (agrostis-genus) = (colonial-bentgrass)))
   (TYPE-OF (COLONIAL-BENTGRASS) = (AGROSTIS-GENUS))
        dom: 1.00 .. 1.00                    supp: 0.65 .. 1.00
 (check-query '(collar-type (colonial-bentgrass) = (narrow)))
   (COLLAR-TYPE (COLONIAL-BENTGRASS) = (NARROW))
        freq : 0.85 .. 0.95                  supp: 0.65 .. 1.00
 (get-inverse '(collar-type (colonial-bentgrass) = (narrow)))
   (INV-COLLAR-TYPE (NARROW) = (COLONIAL-BENTGRASS))
        freq : 0.10 .. 0.15                  supp: 0.45 .. 1.00
```

```
(COLLAR-TYPE (AGROSTIS-GENUS) = (NARROW))
      freq : 0.51 .. 0.97                    supp: 0.00 .. 1.00
```

Command: (MATCH-EVIDENCE-TO-FACTS)

A review of the evidence ratings indicates that the unknown is most likely Poa-Genus:

```
ID possibilities for run 1 listed in decreasing likelihood:

?X = POA-GENUS  (3 piece(s) of evidence found)
    Match rating:  avg: (0.47  0.63) max: (0.61  0.74)
      Evidence 3 matches with id fact 1: (0.59 0.69)
      Evidence 4 matches with id fa : 1: (0.22 0.46)
      Evidence 5 matches with id fact 2: (0.61 0.74)

?X = AGROSTIS-GENUS  (2 piece(s) of evidence found)
    Match rating:  avg: (0.11  0.50) max: (0.00  0.50)
      Evidence 1 matches with id fact 1: (0.21 0.50)
      Evidence 2 matches with id fact 1: (0.00 0.50)
```

APPENDIX G. GRASS IDENTIFICATION: EXPERIMENT LISTING

The following functions define the twenty examples which were used to compare system output and expert analysis. These examples were developed both by having a particular identification possibility in mind when composing the groups of id facts as well as by randomly selecting facts to use together. See also Chapter 8, which discusses the grass identification system, and Appendix H, which presents a summary of the results of both the system and expert analysis for these experiments.

```
;; RUN-1 output is Appendix F.

(defun RUN-1 ()
  (reset-prs-id)
  (add-id-facts
        '((collar-type (?x) = (narrow F{ 0.3 0.6 } S{ 0.9 1.0 }))
          (ligule-shape (?x) = (truncate F{ 0.8 0.9 } S{ 0.9 1.0 }))))
  (set-possibilities 'poa-genus 'agrostis-genus)
  (setq *run-id* 1))



(defun RUN-2 ()
  (reset-prs-id)
  (add-id-facts
        '((vernation-type (?x) = (folded F{ 0.9 1.0 } S{ 0.7 1.0 })
                                 (rolled F{ 0.9 1.0 } S{ 0.1 0.2 }))
          (sheath-type (?x) = (compressed F{ 0.8 0.9 } S{ 0.8 1.0 })
                              (round F{ 0.8 0.9 } S{ 0.1 0.2 }))))
  (set-possibilities 'poa-genus 'agrostis-genus)
  (setq *run-id* 2))



(defun RUN-3 ()
  (reset-prs-id)
  (add-id-facts
        '((collar-type (?x) = (narrow F{ 0.3 0.6 } S{ 0.9 1.0 }))
          (ligule-shape (?x) = (truncate F{ 0.8 0.9 } S{ 0.9 1.0 }))
          (leaf-type (?x) = (keel-shape F{ 0.6 0.7 } S{ 0.9 1.0 }))
          (vernation-type (?x) = (folded F{ 0.75 0.9 } S{ 0.9 1.0 }))))
  (set-possibilities 'creeping-bentgrass 'canada-bluegrass
                     'annual-bluegrass 'kentucky-bluegrass
                     'colonial-bentgrass)
  (setq *run-id* 3))



(defun RUN-4 ()
  (reset-prs-id)
  (add-id-facts
        '((collar-type (?x) = (narrow F{ 0.3 0.6 } S{ 0.9 1.0 }))
          (ligule-shape (?x) = (truncate F{ 0.8 0.9 } S{ 0.9 1.0 }))
          (leaf-type (?x) = (pointed F{ 0.6 0.7 } S{ 0.9 1.0 }))
          (vernation-type (?x) = (rolled F{ 0.75 0.9 } S{ 0.9 1.0 }))))
  (set-possibilities 'creeping-bentgrass 'canada-bluegrass
                     'annual-bluegrass 'kentucky-bluegrass
                     'colonial-bentgrass)
  (setq *run-id* 4))



(defun RUN-5 ()
  (reset-prs-id)
  (add-id-facts
        '((ligule-shape (?X) = (toothed F{ 0.5 0.9 } S{ 0.6 0.9 }))
          (collar-type (?X) = (narrow F{ 0.4 0.7 } S{ 0.2 0.7 }))
          (sheath-type (?X) =
                  (compressed-rough F{ 0.7 1.0 } S{ 0.8 1.0 }))
          (growth-habit (?X) = (bunch F{ 0.6 0.8 } S{ 0.5 1.0 }))))
  (set-possibilities 'Annual-Bluegrass 'Kentucky-Bluegrass
                     'Canada-Bluegrass 'Roughstalk-Bluegrass
                     'Creeping-Bentgrass 'Colonial-Bentgrass 'Redtop)
  (setq *run-id* 5))



(defun RUN-6 ()
  (reset-prs-id)
  (add-id-facts
        '((ligule-shape (?X) = (toothed F{ 0.7 1.0 } S{ 0.8 1.0 }))
          (collar-type (?X) = (broad F{ 0.5 0.9 } S{ 0.6 0.9 }))
          (sheath-type (?X) = (compressed F{ 0.5 0.9 } S{ 0.5 1.0 }))
          (growth-habit (?X) = (stolons F{ 0.7 1.0 } S{ 0.2 0.7 }))))
  (set-possibilities 'Annual-Bluegrass 'Kentucky-Bluegrass
                     'Canada-Bluegrass 'Roughstalk-Bluegrass
                     'Creeping-Bentgrass 'Colonial-Bentgrass 'Redtop)
  (setq *run-id* 6))



(defun RUN-7 ()
  (reset-prs-id)
  (add-id-facts
        '((ligule-shape (?X) = (round F{ 0.7 1.0 } S{ 0.5 1.0 }))
          (collar-type (?X) = (divided F{ 0.4 0.7 } S{ 0.6 0.9 }))
          (sheath-type (?X) =
                         (compressed-rough F{ 0.6 0.8 } S{ 0.6 0.9 }))
          (growth-habit (?X) = (rhizome F{ 0.5 0.9 } S{ 0.8 1.0 }))))
  (set-possibilities 'Annual-Bluegrass 'Kentucky-Bluegrass
                     'Roughstalk-Bluegrass 'Creeping-Bentgrass
                     'Colonial-Bentgrass 'Redtop 'Canada-Bluegrass)
  (setq *run-id* 7))


(defun RUN-8 ()
  (reset-prs-id)
  (add-id-facts
        '((leaf-type (?X) = (pointed F{ 0.5 0.9 } S{ 0.5 1.0 }))
          (vernation-type (?X) = (rolled F{ 0.7 1.0 } S{ 0.2 0.7 }))
          (ligule-shape (?X) = (acute F{ 0.4 0.7 } S{ 0.6 0.9 }))
          (collar-type (?X) = (narrow F{ 0.4 0.7 } S{ 0.8 1.0 }))
          (sheath-type (?X) = (round F{ 0.5 0.9 } S{ 0.2 0.7 }))
          (growth-habit (?X) = (stolons F{ 0.7 1.0 } S{ 0.6 0.9 }))))
  (set-possibilities 'Annual-Bluegrass 'Kentucky-Bluegrass
                     'Roughstalk-Bluegrass 'Creeping-Bentgrass
                     'Colonial-Bentgrass 'Redtop 'Canada-Bluegrass)
  (setq *run-id* 8))


(defun RUN-9 ()
  (reset-prs-id)
  (add-id-facts
        '((leaf-type (?X) = (keel-shape F{ 0.4 0.7 } S{ 0.6 0.9 }))
          (vernation-type (?X) = (folded F{ 0.6 0.8 } S{ 0.8 1.0 }))
          (ligule-shape (?X) = (truncate F{ 0.7 1.0 } S{ 0.6 0.9 }))
          (collar-type (?X) = (broad F{ 0.7 1.0 } S{ 0.2 0.7 }))
          (sheath-type (?X) = (round F{ 0.5 0.9 } S{ 0.5 1.0 }))
          (growth-habit (?X) = (bunch F{ 0.5 0.9 } S{ 0.2 0.7 }))))
  (set-possibilities 'Annual-Bluegrass 'Kentucky-Bluegrass
                     'Roughstalk-Bluegrass 'Creeping-Bentgrass
                     'Colonial-Bentgrass 'Redtop 'Canada-Bluegrass)
  (setq *run-id* 9))


(defun RUN-10 ()
  (reset-prs-id)
  (add-id-facts
        '((leaf-type (?X) = (pointed F{ 0.6 0.8 } S{ 0.6 0.9 }))
          (vernation-type (?X) = (folded F{ 0.7 1.0 } S{ 0.6 0.9 }))
          (ligule-shape (?X) = (round F{ 0.4 0.7 } S{ 0.6 0.9 }))
          (collar-type (?X) = (broad F{ 0.6 0.8 } S{ 0.5 1.0 }))
          (sheath-type (?X) = (compressed F{ 0.5 0.9 } S{ 0.2 0.7 }))
          (growth-habit (?X) = (bunch F{ 0.4 0.7 } S{ 0.8 1.0 }))))
  (set-possibilities 'Annual-Bluegrass 'Kentucky-Bluegrass
                     'Roughstalk-Bluegrass 'Creeping-Bentgrass
                     'Colonial-Bentgrass 'Redtop 'Canada-Bluegrass)
  (setq *run-id* 10 ))


(defun RUN-11 ()
  (reset-prs-id)
  (add-id-facts
        '((leaf-type (?X) = (pointed F{ 0.4 0.7 } S{ 0.3 0.7 }))
          (vernation-type (?X) = (rolled F{ 0.5 0.9 } S{ 0.2 0.7 }))
          (ligule-shape (?X) = (toothed F{ 0.5 0.9 } S{ 0.5 1.0 }))
          (collar-type (?X) = (divided F{ 0.7 1.0 } S{ 0.2 0.7 }))
          (sheath-type (?X) =
                          (compressed-rough F{ 0.7 1.0 } S{ 0.6 0.9 }))
          (growth-habit (?X) = (rhizome F{ 0.6 0.8 } S{ 0.6 0.9 }))))
  (set-possibilities 'Annual-Bluegrass 'Kentucky-Bluegrass
                     'Roughstalk-Bluegrass 'Creeping-Bentgrass
                     'Colonial-Bentgrass 'Redtop 'Canada-Bluegrass)
  (setq *run-id* 11))


(defun RUN-12 ()
  (reset-prs-id)
  (add-id-facts
        '((leaf-type (?X) = (keel-shape F{ 0.7 1.0 } S{ 0.6 0.9 }))
          (vernation-type (?X) = (rolled F{ 0.7 1.0 } S{ 0.2 0.7 }))
          (ligule-shape (?X) = (acute F{ 0.4 0.7 } S{ 0.8 1.0 }))
          (collar-type (?X) = (divided F{ 0.6 0.8 } S{ 0.8 1.0 }))
          (sheath-type (?X) = (round F{ 0.5 0.9 } S{ 0.6 0.9 }))
          (growth-habit (?X) = (stolons F{ 0.5 0.9 } S{ 0.5 1.0 }))))
  (set-possibilities 'Annual-Bluegrass 'Kentucky-Bluegrass
                     'Roughstalk-Bluegrass 'Creeping-Bentgrass
                     'Colonial-Bentgrass 'Redtop 'Canada-Bluegrass)
  (setq *run-id* 12))


(defun RUN-13 ()
  (reset-prs-id)
  (add-id-facts
        '((leaf-type (?X) = (keel-shape F{ 0.5 0.9 } S{ 0.6 0.9 }))
          (vernation-type (?X) = (folded F{ 0.4 0.7 } S{ 0.5 1.0 }))
          (ligule-shape (?X) = (acute F{ 0.4 0.7 } S{ 0.5 1.0 }))
          (collar-type (?X) = (narrow F{ 0.7 1.0 } S{ 0.6 0.9 }))
          (sheath-type (?X) = (compressed F{ 0.6 0.8 } S{ 0.8 1.0 }))
          (growth-habit (?X) = (bunch F{ 0.6 0.8 } S{ 0.2 0.7 }))))
  (set-possibilities 'Annual-Bluegrass 'Kentucky-Bluegrass
                     'Roughstalk-Bluegrass 'Creeping-Bentgrass
                     'Colonial-Bentgrass 'Redtop 'Canada-Bluegrass)
  (setq *run-id* 13))



(defun RUN-14 ()
  (reset-prs-id)
  (add-id-facts
        '((ligule-shape (?x) = (toothed F{ 0.3 0.6 } S{ 0.5 1.0 }))
          (collar-type (?x) = (divided F{ 0.6 0.8 } S{ 0.9 1.0 }))
          (sheath-type (?x) =
                       (compressed-rough F{ 0.8 0.9 } S{ 0.9 1.0 }))))
  (set-possibilities 'poa-genus 'agrostis-genus)
  (setq *run-id* 14))



(defun RUN-15 ()
  (reset-prs-id)
  (add-id-facts
        '((vernation-type (?x) = (rolled F{ 0.3 0.6 } S{ 0.9 1.0 }))
          (growth-habit (?x) = (bunch F{ 0.8 0.9 } S{ 0.9 1.0 }))
          (sheath-type (?x) = (round F{ 0.6 0.8 } S{ 0.9 1.0 }))
          (collar-type (?x) = (broad F{ 0.6 0.8 } S{ 0.9 1.0 }))))
  (set-possibilities 'poa-genus 'agrostis-genus)
  (setq *run-id* 15))



(defun RUN-16 ()
  (reset-prs-id)
  (add-id-facts
        '((growth-habit (?x) = (bunch F{ 0.3 0.6 } S{ 0.9 1.0 }))
          (sheath-type (?x) = (compressed F{ 0.8 0.9 } S{ 0.9 1.0 }))
          (collar-type (?x) = (narrow F{ 0.6 0.8 } S{ 0.9 1.0 }))
          (ligule-size (?x) = (short F{ 0.8 0.9 } S{ 0.9 1.0 }))
          (ligule-shape (?x) = (truncate F{ 0.8 0.9 } S{ 0.9 1.0 }))))
  (set-possibilities 'poa-genus 'agrostis-genus)
  (setq *run-id* 16))



(defun RUN-17 ()
  (reset-prs-id)
  (add-id-facts
        '((sheath-type (?x) =
                       (compressed-rough F{ 0.6 0.8 } S{ 0.9 1.0 }))
          (collar-type (?x) = (broad F{ 0.8 0.9 } S{ 0.9 1.0 }))
          (ligule-size (?x) = (tall F{ 0.8 0.9 } S{ 0.9 1.0 }))
          (ligule-shape (?x) = (round F{ 0.6 0.8 } S{ 0.9 1.0 }))))
  (set-possibilities 'poa-genus 'agrostis-genus)
  (setq *run-id* 17))



(defun RUN-18 ()
  (reset-prs-id)
  (add-id-facts
        '((growth-habit (?x) = (stolons F{ 0.3 0.6 } S{ 0.9 1.0 }))
          (sheath-type (?x) = (compressed F{ 0.8 0.9 } S{ 0.9 1.0 }))
          (collar-type (?x) = (narrow F{ 0.6 0.8 } S{ 0.5 1.0 }))
          (collar-type (?x) = (divided F{ 0.6 0.8 } S{ 0.5 1.0 }))))
  (set-possibilities 'poa-genus 'agrostis-genus)
  (setq *run-id* 18))



(defun RUN-19 ()
  (reset-prs-id)
  (add-id-facts
        '((growth-habit (?x) = (rhizome F{ 0.3 0.6 } S{ 0.9 1.0 }))
          (sheath-type (?x) =
                          (compressed-rough F{ 0.6 0.8 } S{ 0.5 1.0 }))
          (sheath-type (?x) = (compressed F{ 0.6 0.8 } S{ 0.5 1.0 }))
          (collar-type (?x) = (broad F{ 0.8 0.9 } S{ 0.9 1.0 }))
          (ligule-size (?x) = (short F{ 0.6 0.8 } S{ 0.9 1.0 }))
          (ligule-shape (?x) = (acute F{ 0.8 0.9 } S{ 0.9 1.0 }))))
  (set-possibilities 'poa-genus 'agrostis-genus)
  (setq *run-id* 19))



(defun RUN-20 ()
  (reset-prs-id)
  (add-id-facts
        '((growth-habit (?x) = (bunch F{ 0.3 0.6 } S{ 0.9 1.0 }))
          (sheath-type (?x) = (round F{ 0.8 0.9 } S{ 0.9 1.0 }))
          (collar-type (?x) = (divided F{ 0.8 0.9 } S{ 0.9 1.0 }))
          (ligule-size (?x) = (short F{ 0.6 0.8 } S{ 0.9 1.0 }))
          (ligule-shape (?x) = (toothed F{ 0.6 0.8 } S{ 0.9 1.0 }))))
  (set-possibilities 'poa-genus 'agrostis-genus)
  (setq *run-id* 20))
```

APPENDIX H. GRASS IDENTIFICATION: RESULTS SUMMARY

This appendix summarizes the system and expert conclusions for the twenty experiments used to test the grass identification system. (The information used to define each experiment is listed in Appendix G). A brief description of the information listed for each run follows, which refers to terms and principles described in Chapter 8. For a more complete discussion of the grass identification system and the results see Chapter 8.

The information given below is separated into the individual experiments. For each experiment, or run, the number of id facts and identification possibilities are given. The identification possibilities are then listed in the order in which they were ranked by the system, using the abbreviations given below. Following each possibility is the number of pieces of evidence found and the average and maximum match ratings of the inferred facts. The last columns show the ranking given by the domain expert to the possibilities as well as a measure of his belief in his selection. Following each run are comments.

For readability several of the issues discussed in Chapter 8 will be briefly mentioned here. One is that match ratings are only for comparison purposes between other match ratings. Likewise the belief measures given by the expert are comparable only among themselves. Nonetheless, they can be used to give an indication of the confidence in an assigned rank in relation to the values given for the other rankings. As discussed in Chapter 8 an area for future work is the further examination of the relationship between system match ratings and human belief values. Another point is that the system currently ranks choices based first upon amount of evidence found and then considering the average match rating. Again, as discussed in Chapter 8, this method could use refinement in some cases.

In a number of the experiments the expert did not rank all of the choices. This can be taken to indicate that the evidence for the unranked choices does not approach that of the ranked choices or that the given facts have a high level of conflict which makes determinations difficult. (Recall from Chapter 8 that in any particular experiment a conflict is when one id fact is evidence for one identification and another id fact is evidence for a different identification.)

ABBREVIATIONS:

AB -	Annual Bluegras	KB -	Kentucky Bluegrass
AG -	Agrostis Genus	PG -	Poa Genus
CaB -	Canadian Bluegrass	RT -	Redtop
CoB -	Colonial Bentgrass	RB -	Roughstalk Bluegrass
CrB -	Creeping Bentgrass		

RUN #	# of ID FACTS	# of POSSI-BILITIES	SYSTEM RANKING	# of PIECES of EVIDENCE	MATCH RATING: Avg	MATCH RATING: Max	EXPERT RANKING	
Run 1	2	2	PG	3	(0.47 0.63)	(0.61 0.74)	1	0.60
			AG	2	(0.11 0.50)	(0.22 0.50)	2	0.60
	System and expert rankings match exactly.							
Run 2	4	2	PG	4	(0.52 0.63)	(0.72 0.74)	1	0.75
			AG	4	(0.56 0.62)	(0.74 0.75)	2	0.50
	Equal amount of evidence for either choice, however the support values for the facts related to PG are higher than those related to AG and thus the system ranks PG higher.							
Run 3	4	5	KB	4	(0.20 0.32)	(0.61 0.62)	1	0.80
			AB	3	(0.01 0.21)	(0.02 0.30)	3	0.60
			CaB	2	(0.01 0.34)	(0.02 0.40)	2	0.60
			CoB	1	(0.00 0.17)	(0.00 0.17)		
			CrB	1	(0.00 0.12)	(0.00 0.12)		
	First choice matches. Expert ranked CaB above AB because research has shown that the values for one of the variables (ligule shape) is easily confused and thus CaB might actually fit the "real" situation better than AB.							

RUN #	# of ID FACTS	# of POSSI-BILITIES	SYSTEM RANKING	# of PIECES of EVIDENCE	MATCH RATING: Avg	Max	EXPERT RANKING	
Run 4	4	5	CoB	3	(0.07 0.39)	(0.22 0.50)	1	0.80
			CrB	3	(0.08 0.32)	(0.22 0.50)	1	0.80
			KB	2	(0.38 0.41)	(0.61 0.62)	2	0.70
			AB	1	(0.00 0.17)	(0.00 0.17)		
			CaB	0				

System and expert rankings match exactly for first 3 possibilities.

RUN #	# of ID FACTS	# of POSSI-BILITIES	SYSTEM RANKING	# of PIECES of EVIDENCE	MATCH RATING: Avg	Max	EXPERT RANKING	
Run 5	4	7	RB	2	(0.67 0.69)	(0.73 0.74)	1	0.60
			AB	2	(0.00 0.15)	(0.00 0.17)	2	0.40
			KB	1	(0.78 0.81)	(0.78 0.81)		
			CoB	1	(0.00 0.17)	(0.00 0.17)		
			CrB	1	(0.00 0.12)	(0.00 0.12)		
			CaB	0				
	RT	0						

System and expert rankings match for first 2 possibilities. (Expert didn't rank remaining choices: "their order is ambiguous".)

RUN #	# of ID FACTS	# of POSSI-BILITIES	SYSTEM RANKING	# of PIECES of EVIDENCE	MATCH RATING: Avg	Max	EXPERT RANKING	
Run 6	4	7	RB	3	(0.62 0.64)	(0.84 0.89)	1	0.90
			CaB	1	(0.68 0.70)	(0.68 0.70)		
			KB	1	(0.61 0.63)	(0.61 0.63)		
			CrB	1	(0.57 0.58)	(0.57 0.58)		
			CoB	1	(0.50 0.50)	(0.50 0.50)		
			AB	1	(0.03 0.10)	(0.03 0.10)		
			RT	0				

System and expert ranking matches for first choice. Expert: "RB is a strong match to the facts". (Other possibilities weren't ranked by expert).

RUN #	# of ID FACTS	# of POSSI-BILITIES	SYSTEM RANKING	# of PIECES of EVIDENCE	MATCH RATING: Avg	Max	EXPERT RANKING	
Run 7	4	7	RT	3	(0.37 0.43)	(0.63 0.67)	3	0.40
			KB	2	(0.72 0.75)	(0.72 0.77)	1	0.70
			CaB	2	(0.46 0.49)	(0.61 0.62)	2	0.55
			CoB	1	(0.63 0.67)	(0.63 0.67)		
			CrB	1	(0.50 0.50)	(0.50 0.50)		
			RB	1	(0.02 0.04)	(0.02 0.04)		
			AB	0				

First 3 selections are all close. KB was picked first by expert based upon typicality of KB as a possible id. Also, although KB has less evidence found than RT it has higher match ratings.

Run 8	6	7	CrB	5	(0.39 0.48)	(0.58 0.63)	1	0.90
			CoB	5	(0.34 0.44)	(0.50 0.64)		
			RT	3	(0.42 0.53)	(0.50 0.61)		
			AB	2	(0.13 0.23)	(0.25 0.30)		
			KB	1	(0.78 0.81)	(0.78 0.81)		
			RB	1	(0.56 0.56)	(0.56 0.56)		
			CaB	1	(0.12 0.19)	(0.12 0.19)		

System and expert agree on CrB. However it is not clear why the expert did not also select CoB as it matches on the same facts as CrB.

Run 9	6	7	RB	3	(0.24 0.48)	(0.68 0.73)		
			AB	3	(0.21 0.36)	(0.60 0.63)		
			KB	3	(0.20 0.35)	(0.58 0.60)	1	0.45
			CaB	2	(0.02 0.33)	(0.02 0.40)		
			CrB	1	(0.58 0.63)	(0.58 0.63)		
			RT	1	(0.46 0.47)	(0.46 0.47)		
			CoB	1	(0.31 0.38)	(0.31 0.38)		

Expert: "This example contains conflicting initial facts and ranking possibilities is difficult. If forced to make choices, though, RB is also a reasonable selection - missed it on first pass".

RUN #	# of ID FACTS	# of POSSIBILITIES	SYSTEM RANKING	# of PIECES of EVIDENCE	MATCH RATING: Avg	Max	EXPERT RANKING
Run 10	6	7	AB	3	(0.40 0.46)	(0.63 0.67)	
			CaB	2	(0.60 0.65)	(0.68 0.70)	
			KB	2	(0.58 0.61)	(0.61 0.63)	
			CrB	2	(0.23 0.42)	(0.22 0.50)	
			RB	2	(0.28 0.37)	(0.51 0.60)	
			CoB	2	(0.11 0.31)	(0.22 0.50)	1 0.60
			RT	2	(0.11 0.30)	(0.22 0.50)	

Expert: "Knowledge base is missing a fact attached to CoB which would have allowed an additional piece of evidence to be found - this would have caused CoB to be ranked highest by the system". There is a lot of conflicting evidence in this example. With one exception each possibility had an equal amount of evidence.

RUN #	# of ID FACTS	# of POSSIBILITIES	SYSTEM RANKING	# of PIECES of EVIDENCE	MATCH RATING: Avg	Max	EXPERT RANKING
Run 11	6	7	RT	4	(0.21 0.43)	(0.61 0.67)	
			RB	2	(0.67 0.69)	(0.73 0.74)	
			KB	2	(0.51 0.53)	(0.73 0.74)	
			CoB	2	(0.11 0.50)	(0.22 0.50)	
			CrB	2	(0.12 0.43)	(0.22 0.50)	
			CaB	2	(0.29 0.31)	(0.56 0.58)	
			AB	0			

Expert: "The given facts made the identification of any species very difficult. There were just too many discrepancies in the values selected for the facts".

RUN #	# of ID FACTS	# of POSSIBILITIES	SYSTEM RANKING	# of PIECES of EVIDENCE	MATCH RATING: Avg	Max	EXPERT RANKING
Run 12	6	7	CrB	3	(0.56 0.60)	(0.58 0.63)	1 0.50
			CoB	3	(0.48 0.57)	(0.64 0.69)	
			CaB	3	(0.44 0.51)	(0.59 0.69)	
			RT	3	(0.45 0.50)	(0.50 0.61)	
			KB	2	(0.68 0.73)	(0.73 0.78)	
			AB	2	(0.46 0.52)	(0.67 0.75)	
			RB	2	(0.35 0.43)	(0.58 0.69)	

Expert: "Evidence is very conflicting - no choice clearly outweighs others". If pressed though, expert would choose CrB which matches with the system.

RUN #	# of ID FACTS	# of POSSI-BILITIES	SYSTEM RANKING	# of PIECES of EVIDENCE	MATCH RATING: Avg	Max	EXPERT RANKING	
Run 13	6	7	AB	6	(0.19 0.30)	(0.61 0.67)	1	0.85
			CaB	4	(0.27 0.45)	(0.85 0.90)		
			KB	4	(0.03 0.16)	(0.01 0.30)		
			RB	2	(0.06 0.38)	(0.01 0.43)		
			CrB	1	(0.63 0.67)	(0.63 0.67)		
			CoB	1	(0.61 0.67)	(0.61 0.67)		
			RT	0				
	System and expert agree on AB being a strong match.							
Run 14	3	2	PG	4	(0.42 0.63)	(0.57 0.71)	1	0.60
			AG	1	(0.50 0.81)	(0.50 0.81)	2	
	System matches expert exactly.							
Run 15	4	2	AG	4	(0.38 0.56)	(0.50 0.77)	1	0.55
			PG	2	(0.25 0.43)	(0.50 0.50)	2	
	System matches expert exactly.							
Run 16	5	2	PG	7	(0.50 0.64)	(0.57 0.77)	1	0.65
			AG	3	(0.50 0.61)	(0.50 0.71)	2	
	System matches expert exactly.							
Run 17	4	2	AG	5	(0.52 0.71)	(0.50 0.84)	1	0.60
			PG	2	(0.50 0.56)	(0.50 0.62)	2	
	System matches expert exactly.							
Run 18	4	2	AG	5	(0.40 0.59)	(0.50 0.81)	1	0.45
			PG	5	(0.39 0.55)	(0.57 0.71)	2	
	System matches expert exactly. (Note: evidence found is equal for either possibility and conclusion is essentially a toss-up.)							

RUN #	# of ID FACTS	# of POSSI-BILITIES	SYSTEM RANKING	# of PIECES of EVIDENCE	MATCH RATING: Avg	Max	EXPERT RANKING	
Run 19	6	2	PG	8	(0.42 0.55)	(0.51 0.75)	1	0.70
			AG	2	(0.25 0.59)	(0.50 0.71)	2	
	System matches expert exactly.							
Run 20	5	2	AG	5	(0.50 0.60)	(0.50 0.71)	2	
			PG	4	(0.42 0.59)	(0.57 0.77)	1	0.35
	Although system and expert disagree on what possibility should be ranked higher the evidence is almost equal for either choice.							