The Design and Implementation of an Evidence Oracle for the Understanding of Arguments

Mark Anthony Young
Department of Computer Science

Research Report CS-87-33
June 1987

The Design and Implementation of an Evidence Oracle for the Understanding of Arguments

by

Mark Anthony Young

An essay
presented to the University of Waterloo
in fulfilment of the
essay requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, 1987

Abstract

When trying to understand the thrust of another person's argument, it is necessary to determine what his claim is, and what evidence he provides for it. It is necessary, therefore, to be able to recognise evidence relationships in terms of the speaker's beliefs. This essay concerns how one goes about building a system (an *oracle*) to do that, and provides a working version of a simple oracle.

Acknowledgements

I would like to thank my supervisor, Robin Cohen, for suggesting the topic of this essay and for helping me to see the work through to its end. The presentation is much clearer due to her helpful suggestions. I want to thank all my friends and family who supported and encouraged me both in this work and throughout my time here at UW.

Contents

Chapter 1

Introduction

The problem addressed in this essay is how best to construct an *Evidence Oracle* (*EO*)—a system which takes two propositions P and Q, and answers the question 'Is P meant to be evidence for Q?' The *EO* is intended to be used in a system for interpreting a class of natural language discourse referred to as arguments. The *EO* incrementally builds a representation of the speaker from the propositions it is given and the relations it finds between them.

This paper will describe the theory and practice of determining evidence relationships. The object is to implement a basic oracle that will make decisions based on a model of the world, including the speaker, built by itself. While the *EO* is intended to be used in a NL setting, we will simplify the situation under consideration by assuming that the input has already been converted into Prolog-like predicates representing their essential meaning. How this transformation is done is beyond the scope of this paper.

1.1 The Argument Understanding System

The *EO* described in this paper is based on the subsystem of the same name in Cohen's *Argument Understanding System* (*AUS*). The *AUS* (described in [Cohen83]) models a "patient listener", a person who hears a speaker out before presenting counter-arguments. It parses a NL argument (in the form of a monologue) into an 'argument tree', each node of which represents a statement of the argument. The root is the main claim of the speaker and the children of any node are those

statements given in evidence for the statement at that node. By identifying and using a limited number of coherent transmission strategies, Cohen was able to restrict the number of calls to the *EO* to be linear in the number of statements in the argument.

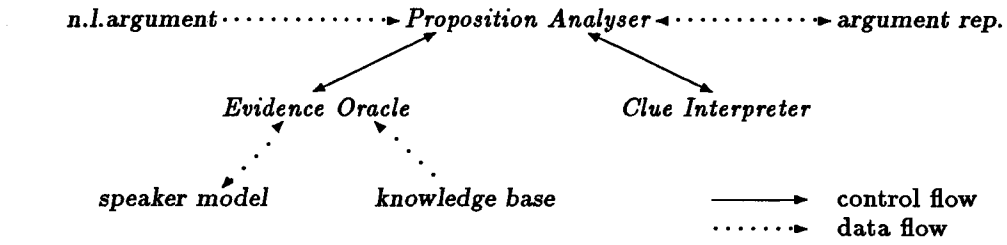

Figure 1.1: System Design

Figure 1.1 (adapted from [Cohen83]) shows the flow of control and data in the *AUS*. The propositions of the argument are passed to the *Proposition Analyser* (*PA*) which, based on the restrictions of the transmission strategy and taking into account any clues present (as determined by the *Clue Interpreter* (*CI*)), selects those nodes in the argument tree which may be related to the current proposition. For each of these nodes, proceeding toward the root, the question of evidence is asked of the *EO*. The search ends when an evidence relation is found. Sometimes the *PA* will pass clues to the *EO* to aid it in its analysis.

The *EO* has access to a general knowledge base and a model of the speaker. As the *EO* detects evidence relations between statements made by the speaker it can use the knowledge gained to modify the speaker model. In this way the oracle may start with little or no knowledge of the speaker and yet have a reasonably complete picture of his beliefs by the time the argument is fully parsed.

There are three Transmission Strategies the *AUS* uses. These are:

1. claim first—each claim is followed by its evidence.

2. claim last—each claim is preceded by its evidence.

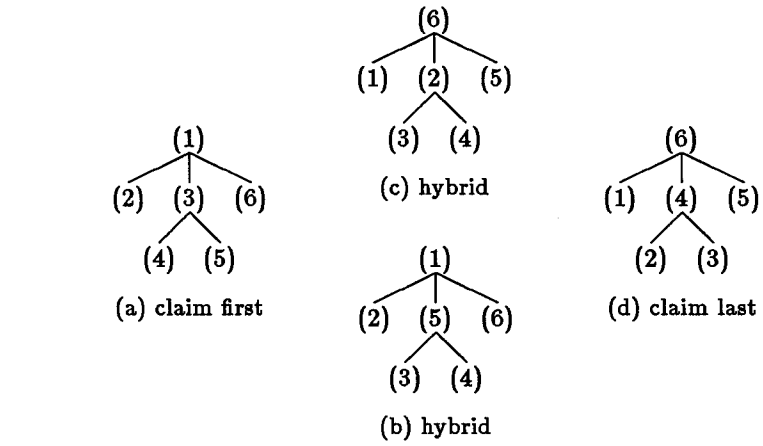

Figure 1.2: Transmission Strategies

3. hybrid—each claim is *either* preceded *or* followed by its evidence. That is, each sub-argument is either claim-first or claim-last.

Figure 1.2 shows examples of these transmission strategies, where the numbers on the nodes indicate the position of the corresponding statement in the argument.

1.2 The Response Unit

The *AUS* processes the NL argument into an argument tree. It is expected that this representation will be passed on to some sort of *Response Unit* (*RU*) which will look for weak spots in the argument and confront the speaker with them.

While the *RU* is not part of the *AUS* proper, its needs should be considered by the *EO* when it is doing its calculations. Some varieties of evidence will be inherently more believable than others. The *EO* should make notes of when weak or bad logic is used or when contradictions arise in the speaker's argument. This will save the *RU* the trouble of re-analysing the evidence relations appearing in the tree.

1.3 The Evidence Oracle

The *AUS* envisioned in [Cohen83] requires a module to make decisions regarding evidence relations. This Evidence Oracle is passed two propositions, P and Q, and would respond 'yes' or 'no' as to whether P is evidence for Q.

The following excerpts from [Cohen83, pp.8–9,p.68] give an initial definition of evidence.

> Basically, an evidence relation reflects that one proposition is being used to support another, through some logical chain of reasoning....
>
> The argument understanding system first of all considers a proposition to be the basic unit of analysis. Each proposition is then assigned an interpretation using the claim and evidence framework. An initial characterization of evidence is developed whereby: a proposition P is evidence for a proposition Q if they fit appropriate slots in one of the system frames representing various logical rules of inference, such that P is premise to Q's conclusion. The motivation is that there is some logical connection between P and Q. The use of frames is designed to encode various constraints on the relation between slot elements and to establish the frame inference rule by instantiating the slots according to their definitions.
>
> We now elaborate on our collection of rules of inference and the use of frames to establish evidence relations [see table 1.1]. This selection of frames is motivated by [Sadock77], which indicates those correct and incorrect logical rules that occur most often in conversation.

In our first version of the *EO* we will restrict our attention to *Modus Ponens* and *Modus Tollens*. Other frames of evidence will be discussed in chapter 2.

> Determining evidence relations becomes an interesting issue when one considers that, most often, not all slots of the applicable frame are spelled out. This phenomenon, called *Modus Brevis* in [Sadock77], includes such forms as: [see table 1.2]
>
> The first problem in establishing evidence is thus recognizing the rule of inference which serves to formulate the logical connection. We claim that the default frame and *modus brevis* version to test for is: Missing Major *Modus Ponens*. This is as well the

(correct)	MAJOR PREMISE	MINOR PREMISE	CONCLUSION
Modus Ponens	$A \rightarrow B$	A	B
Modus Tollens	$A \rightarrow B$	$\neg B$	$\neg A$
Modus Tollendo Ponens	$A \vee \neg B$	B	A
Modus Ponendo Tollens	$A \vee B$	B	$\neg A$
(incorrect)			
Asserting Consequent	$A \rightarrow B$	B	A
Denying Antecedent	$A \rightarrow B$	$\neg A$	$\neg B$

Table 1.1: Set of Frames:

	Given Premises	Conclusion
Normal	$P \rightarrow Q, P$	Q
Missing Minor	$P \rightarrow Q$	Q
Missing Major	P	Q
Only Major	$P \rightarrow Q$	(assume rest)
Only Minor	P	(assume rest)

Table 1.2: Classification of Frames (example: *Modus Ponens*)

relation exhibited most often in the stock of examples reviewed in the course of this research.

Again we restrict our attention for the purpose of the implementation, considering only the first three versions of *Modus Brevis*. The last two are hard to find because there is no direct link to any other utterance in the discourse. Because the oracle deals only with *pairs* of statements, these cases would have to be dealt with by the *PA*.

1.4 Simplifications Made

In order to simplify the construction of this oracle, a number of assumptions have been made and a few design decisions taken.

The first assumption made concerns the form of input the oracle receives. We do not want to deal with the problem of interpreting natural language itself, but only the relationship between pairs of utterances. Thus we assume that the input has been 'coded' into Prolog predicates. These predicates are assumed to represent the meaning of the statement without too much detail as to the form the statement was made in. Thus the translation of "Brian is an anglophone" could be "$anglophone(brian)$". The only requirement is consistency, that two statements that mean the same thing are translated to the same predicate.

To aid in our construction of the speaker model, we will assume that the speaker is reasonably competent at logic. We expect that if the speaker uses $E \rightarrow C$ as evidence for C, then he believes E to be true.

For the first version of the oracle we shall also assume that the speaker always uses logic correctly. That is, he never makes such common mistakes as asserting the consequent or denying the antecedent. In later versions this assumption should be relaxed to assuming competence unless given reason to doubt it.

For the first version of the oracle we will have the system stand on its own. The implementation is such that the user will query it on his own initiative and the model of the speaker will grow throughout the session. The user will be at the Prolog query level and enter the question of evidence (P for Q) as $?oracle(Q, P)$. Code for the *PA* has been written (see [Smedley86]) but the *EO* has not

been integrated with it. To show the system operating on an entire argument, a simple parser has been implemented. The parser makes successive calls to the oracle (based on claim first transmission) to determine the form of the entire argument.

The final simplification made is that clues will not be handled in the first version. This means that the oracle may return any relation it finds without reference to the clue words of the argument. Since clue words are often necessary to the understanding of arguments this restriction may mean that arguments must follow one of the two pure transmission strategies.

1.5 Outline of Essay

Chapter 2 deals with the definition of evidence, the theory behind the speaker model. Chapter 3 deals with the implementation of the oracle, the form of the oracle's knowledge of the world and the way evidence relations are found. It then goes on to give some examples of the oracle at work.

Chapter 4 concerns the future of the oracle, what things it should do, but does not. Chapter 5 deals with details of the implementation that caused the most concern. The implications of the oracle for other fields is discussed in chapter 6.

Chapter 2

Theory

2.1 Evidence

In section 1.3 we give a brief description of evidence. In this section we wish to expand and clarify this definition, and introduce some strategies that might be used to discern evidence relations.

It is important to keep in mind that the evidence relations the listener finds should be those he thinks the speaker intends him to find. The speaker is expected to cooperate in this matter by presenting his argument in a coherent manner (*i.e.* one of the recognized transmission strategies), and with enough steps given for the listener to follow the logic.

We have given the preliminary definition of evidence that P is evidence for Q if there is some logical connection between the two, with P as premise and Q conclusion, where *Modus Ponens* and *Modus Tollens* are the logical connections recognized. These rules are expected to be used in the following manner: "A is true, and A implies B, therefore B is true." Here Q is replaced by B and P by A and $A \rightarrow B$ in turn. (If the argument were presented in the order given then it would satisfy the 'claim last' transmission strategy.)

We say that *MP* and *MT* are *frames of evidence.* Each frame has a slot for the conclusion and two or more for premises. For P to be evidence for Q there must be a frame of evidence with P in the premises and Q in the conclusion. Other frames will be described in later sections of this chapter.

Cohen provides four methods for determining whether the speaker intends an evidence relation between E (evidence) and C (claim). These are as follows:

Method 1 The hearer tries to verify that $E \rightarrow C$ within the set of axioms known as shared knowledge.

Method 2 If $E \rightarrow C$ cannot be derived, then it may be that a "relaxed" form of logic is being used. An example might be that the claim is only quantified "for-most" rather that "for-all".

Method 3 Stereotype the speaker and see if the proposed relation is likely for that stereotype.

Method 4 Create a model of a "hypothetical person" and judge the plausibility of the relation in that framework. This may be viewed as a "least detailed" speaker model.

We will modify these four methods somewhat in that "relaxed logic" will be made a special case of more rigorous logic and the "speaker model" will be more than just a stereotype of the speaker. These differences are explained in more detail in the following sections.

2.2 Generalizations

In casual conversation, the speaker often uses a looser version of logic. He will say 'always', when, in fact, he means 'almost always'. To deal with this usage we will use a variation of *Modus Ponens* which we will call a *generalization.* We represent the counterpart to $P \rightarrow Q$ by $P \rightarrow\!\!\triangleright Q$.

There are three differences between generalizations and the more precise rules of logic.

- Generalizations are not used in contrapositive form. Just because most anglophones are not Canadians does not mean that most Canadians are not anglophones
- Generalizations are not linked together. Most university students are adults, and most adults are employed, but most university students are not employed. If our speaker is reasonably logical he would not use such a linking. He may link two generalizations together, but only by providing enough steps for the oracle to find each one separately.
- The final difference occurs when the speaker is justifying a generalization that he has introduced (see section 2.5). In this case the speaker may grant counter-examples to the rule without

destroying its validity. The speaker may suggest that most Québecois are French while granting that Mulroney is a non-French native of that province. The oracle must recognize that the relation is only for *most*, not for all. Often the speaker will qualify a counter-example by giving the reason it is exceptional (a clever oracle could use this information to 'fix' the proposed generalization).

We skirt the issue of just what these 'for-most' relations really mean. They may represent defaults (as in [Reiter80] or [Poole&al86]) or relations which are merely "probable" (similar to Dempster/Shafer, described in [Barnett81]). They are used in the oracle merely as rules that may not apply to all the values of the variables under their scope.

2.3 The Problem of Missing Premises

The oracle is passed two arguments at a time. Yet most of the evidence relationships we have seen involve three parts. The oracle must fill in the missing part from its knowledge of the speaker. It is sometimes not difficult to fill in the missing piece of evidence. If we are given $E \wedge F \rightarrow C$ for C then we are missing E and F.

The problem arises when we are given E for C. We must fill in some rule with E in the antecedents and C in the consequents. The exact form of this rule is unknown. We do not even know whether such a rule exists. Where are we to find such a rule?

The first place to look is in the list of shared beliefs. This is the type of evidence that is most acceptable, that which is common between the speaker and the listener. In the *EO* these shared beliefs are held in a module called *facts*. This module is discussed in section 3.2.

If there is no appropriate shared belief, then we should look at what we know of the speaker to see if we can figure out what connection he sees. If the speaker has previously stated $E \rightarrow C$, it is entirely likely that E is meant to be evidence for C by application of that rule. Note that it is not necessary that the *listener* believe $E \rightarrow C$, only that he believe the speaker believes it. The knowledge of the speaker is in the form of a speaker model, discussed in section 3.1. The model consists of those things the speaker has said along with some conclusions we have drawn from what he has said.

If we do not know of any belief that the speaker holds that would be appropriate to the predicates then we should see if there is any plausible relation between them. The first step is to see if any other person holds such a belief. A stereotype of the speaker may be useful here (see section 4.2). If we know the speaker to be right-wing then we can expect him to have beliefs that are common among right-wing people. This stereotype model will help to reduce search time, but we cannot rely on it to provide all beliefs appropriate to the argument. Other stereotypes and hypothetical people must be brought in to the search for possible relations.

We should note here that the stereotype in the *EO* is not the same as that in [Cohen83]. In that paper the stereotype *was* the speaker model, in this it is an *addition* to the model. The beliefs indicated in the our stereotype may be viewed as beliefs we are prepared to accept the speaker having. Thus if the stereotype indicates a belief such as *pious(X)* $\rightarrow$ *good-person(X)* and the speaker says something like "John is a good man. He's pious", then we can fill in the connection by assuming the speaker has the belief indicated above. If the speaker did not mention piety anywhere in his discourse we would not make any assumptions about whether he thinks pious people to be good people.

If we still cannot find a relation between the two predicates then we might want to give up. But it is highly unlikely that we have an exhaustive list of beliefs. We should try to see if the indicated relationship is at all plausible. This will involve checking the relationship against the listener's model of the world. This is discussed in more detail in section 3.6.

2.4 Other Frames of Evidence

When we are modelling disputants in an argument the speaker may believe that the listener has an erroneous belief. One strategy the speaker may use to eliminate this erroneous belief is to overwhelm the listener, *i.e.* present such a case for his own beliefs that the listener feels compelled to reject his conflicting belief. Another is to undermine, to provide evidence for the negation of the "erroneous" belief. However, if the belief the speaker wishes to undermine is a rule of inference none of the frames we have presented as yet is appropriate to the situation.

We therefore present another frame, which we shall call *Refutation*, in which the speaker may present a counterexample to a rule in order to disprove it. Thus "Mulroney is a francophone" and

"Mulroney is not French" constitute evidence for "Not all francophones are French" by this frame of evidence.

When disproving a rule the speaker will often concede that there are some examples that can be presented to support it[1]. For example, in disproving the rule in the previous paragraph the speaker may grant that Trudeau and Chretien are both French francophones. We call this frame of evidence a *Concession*; it is a form of contrastive evidence (v. [Cohen83, pp.46–47]).

Table 2.1 shows the frames of evidence our oracle will accept.

	Minor Premise	Major Premise	Conclusion
Modus Ponens	A	$A \rightarrow B$	B
Modus Tollens	$\neg B$	$A \rightarrow B$	$\neg A$
Generalization	A	$A \mathrel{-\!\!\triangleright} B$	B
Refutation	A	$\neg B$	$\neg(A \rightarrow B)$
Concession	A	B	$\neg(A \rightarrow B)$

Table 2.1: Frames of Evidence

2.5 Partial Evidence

The speaker may, from time to time, use logical connections that the hearer does not believe. If the speaker is aware of this non- or disbelief he will probably try to justify its use. One way to do this is to provide examples to back the rule up. This "justification by example" we call *partial evidence*[2].

There are two forms of examples, *positive* and *contrapositive*. Positive examples have the form "A and B are evidence for $A \rightarrow B$", and contrapositive ones the form "$\neg B$ and $\neg A$ are evidence for $A \rightarrow B$". If A is a conjunction then the speaker may indicate that certain elements of the

[1]This is from politeness—a way of saying that the error was understandable in light of these examples.

[2]This form of evidence is discussed in more detail in [Cohen83, §4.8]. The justification given for considering examples as evidence is an implicit generalization, *sufficient_examples(R)* $\mathrel{-\!\!\triangleright} R$, where R is the rule being justified. The predicate *sufficient_examples(R)* is true if enough examples of R are available to the agent considering it for him to believe R. The examples the speaker gives are meant to help the listener to accept the predicate, and thereby accept the rule.

conjunction are true. Thus the oracle must be prepared to accept C as evidence for $C \wedge D \rightarrow E$ with *either* D and E *or* $\neg E$ and $\neg D$ missing. This problem of multiple evidence relations is discussed in section 3.3.

All of the above applies if the major premise under consideration is a generalization. But in this case the speaker may also grant *counterexamples* to the rule. Thus the speaker may say

> I will grant that Mulroney isn't French, but Trudeau and Chretien are, and so are most Quebeckers.

Here Trudeau and Chretien are examples, and Mulroney a counterexample, to the rule "Most Quebeckers are French". In the case of generalizations counterexamples do not refute, but merely serve to make a distinction with *Modus Ponens.*

The use of examples as a form of evidence creates a problem. Consider example 1.

Example 1

(1) Most adults own a car.

(2) Bob owns a car.

Is (1) the rule that allows us to reasonably assume (2)? Or is (2) an example given to support the introduction of (1)? Without context or other clues it is very hard to decide which is the preferred explanation. If (2) is followed by a list of other people who own cars, then we might like to say it is an example. Certainly if we know that the speaker believes that the system believes (2), then using (1) as evidence for (2) is superfluous. Yet if (2) is unknown to the system, then (1) would be acceptable evidence for it.

If (2) had been phrased "Bob, for example, owns a car," then the *Clue Analyser* should indicate a clue 'detail'[3] which would be passed on to the *EO.* If (2) had been phrased "So I conclude Bob has a car," then the clue would be 'inference'. The presence of a clue would strongly suggest one interpretation: 'detail' would support (2) for (1); 'inference' (1) for (2). The absence of clues leaves us wondering. Consideration of shared belief could be helpful, but may still not decide the issue.

[3]Clue types taken from [Cohen83, Appendix II]

As has been noted, our oracle does not yet handle clues. Therefore we will take a different tack. Each time it is asked to decide evidence of a specific for a general (a ground instance for a rule or generalization) the oracle will check to see if some other predicate that the general is evidence to also takes the specific as evidence. If so the oracle will report no relation between the two. If not, it will proceed normally. Thus the oracle will find "and Fred is an adult" as evidence for "Fred owns a car" (minor premise of *Modus Ponens*) rather than for "Most adults own cars" (premise of Positive Example) in the following:

Example 2

(1) Fred owns a car.

(2) Most adults own cars—

(3) Bob, for example, owns one—

(4) and Fred is an adult.

Here (4) could be evidence for (2), Fred being an example of a car-owning adult. However, we find that (2) is evidence to (1) and that (4) would also be evidence for (1). Therefore we reject (4) as evidence for (2) and allow it to find its correct position as evidence for (1).

Table 2.2 shows the frames of partial evidence.

	Minor Premises		Conclusion		
Positive Example	A	B	$A \rightarrow B$	or	$A \mathrel{-\!\!\triangleright} B$
Contrapositive Example	$\neg B$	$\neg A$	$A \rightarrow B$	or	$A \mathrel{-\!\!\triangleright} B$
Counterexample	A	$\neg B$			$A \mathrel{-\!\!\triangleright} B$

Table 2.2: Frames of Partial Evidence

Chapter 3

Practice

3.1 The Speaker Model

The *EO* uses and maintains a model of the speaker to aid it in making decisions. The beliefs kept in this model are those that the *system* believes the speaker has. They will often be referred to as the speaker's beliefs, even though they are not properly his.

The model we keep of the speaker will be made up of our interpretations of the statements he has made plus whatever else we can glean from the form of his argument. We make the assumption that the speaker does not advocate beliefs that he does not hold[1].

If we are to reason in the speaker's belief space then we must have rules to tell us what we can infer from what he says. It would be nice if we could assume that the speaker is ideally rational. Then, anything that follows logically from what he says would be a belief that he holds. Unfortunately, it is not often we meet someone who knows (really knows) everything that follows from his beliefs. Usually the person is unsure of what follows from his beliefs. So the question is, what can we deduce about the speaker's beliefs based on what he says?

Levesque deals with this problem in [Levesque84]. He divides the speaker's beliefs into two parts, *explicit* and *implicit*. Explicit beliefs are those that the speaker is aware of, implicit those that follow

[1]There is no problem if this assumption is violated; the model will be of the persona the speaker has adopted for the argument, and the persona is defined by the beliefs he advocates. This is the form of debate.

from those he holds and the facts of the world he is in. Thus, if "*roman-orator(cicero)*" is an explicit belief, then "*roman-orator(tully)*" is implicit. The speaker may not be aware that Tully and Cicero are the same man, and may thus even deny that Tully is a roman orator (in this case the speaker's beliefs would be inconsistent, even though he is not aware of it).

Any belief that the speaker states must be an explicit one. It may be that it only occurred to him as he said it, but it is something that he knows he believes. We thus have a module, *explicit*, which keeps track of the statements made by the speaker in his argument. These beliefs are something we have concrete evidence for, namely, his statement of them.

How do we add to the speaker's beliefs? Suppose that, using the methods of section 2.3 (which are developed further in section 3.3), we decide that the speaker is offering us *eagle(sam)* as evidence for *flies(sam)*, and that the rule used to justify this evidence relation is $\forall X : eagle(X) \wedge adult(X) \rightarrow flies(X)$, then by our assumption of reasonable competence we can conclude that the speaker believes *adult(sam)*. We make this conclusion whether the speaker adds that Sam is an adult or not. Once we have determined all the beliefs that are implied in this manner, we collect them together and add them to another module, *missing*, which is also part of the speaker model.

We must be careful when adding beliefs in this manner. We do not want to introduce beliefs that are absurd in the context of the speaker's other beliefs. If we considered the speaker to be ideally rational then we could reject any set of beliefs that lead to an inconsistency in the model. Yet we have assumed the speaker to be only *reasonably* competent at logic. We will thus only reject a presumed belief F if we see that the speaker has a belief $\neg F$. A reasonable person would not believe both F and $\neg F$, so any evidence relation requiring that one of these be added to a model containing the other will be rejected as implausible[2].

In addition to the two modules given above, there will be another module which will serve as a stereotype for the speaker (see section 2.3). This module is not considered part of the speaker model because the speaker is not known to hold any of the beliefs in it. If, however, we find that a belief in the stereotype would indicate a relation between the predicates under consideration then it will be freely added to the speaker model (assuming it does not contradict another belief held by the speaker). In this first version of the oracle the system's beliefs will serve as the stereotype for

[2][Levesque84] allows for this sort of contradictory belief, but [Fagin&Halpern85] gives a modification which prevents this. Note that it is possible for both F and $\neg F$ to be logically implied by the speaker model; we only restrict the model by not allowing both to appear *explicitly*.

the speaker. This use is justified because, as listeners, we often assume that the speaker's beliefs are similar to our own until we are given reason to believe otherwise.

There are some beliefs that the speaker might have that might better be kept separate from *explicit* and *missing*. An example would be those beliefs the speaker believes to be common to himself and the listener. Since the speaker should not waste time by providing evidence to things that his listener already believes, then knowing that the speaker believes B to be a mutual belief will eliminate any need to test anything as evidence for B[3]. For the time being we will stick to the two modules given above.

3.2 The System's Beliefs

The *AUS* has a knowledge base that the *EO* consults when verifying evidence relations. This knowledge base may be looked at in one of two ways. One is that it is complete and correct. This would be the case if the oracle were working for a question answering system such as an expert advisor. The other is that it is a not-necessarily-correct picture of the world.

Which of the two versions represents the situation is of little importance to the oracle. In either case we are dealing with a fallible speaker, and there will therefore be disagreements between the speaker and the system. How the *RU* cares to deal with these inconsistencies is of no concern to the *EO*. The oracle takes the part of a listener in a discourse. It consults its own beliefs about the world in making its decisions on evidence. Whether these beliefs are objectively correct or not depends only on the purpose of the entire system (of which the *AUS* is only part). The design of the oracle allows for either interpretation.

There is one distinction that must be made in the system's beliefs. That is the distinction between definitional and assertional knowledge. The definitions of the predicates are important. It is hard to reason with any sort of consistency if the system's definition of a predicate does not match that of the speaker. Since we can't very well expect the speaker to define everything each time he uses the system we must rely on the coding of the argument to match the definitions used.

[3]There are exceptions to this rule; for example, if the speaker is a teacher and the listener a student the speaker may wish to reinforce the belief in B. This exception and others can be handled by appropriately excising beliefs from this module.

Since definitional knowledge must be common to the speaker and the listener, it is kept together with the "shared belief" axioms, in the module *facts*. Since the speaker and the listener are the only participants in this discourse definitional knowledge is indistinguishable from common knowledge[4].

Knowledge that the system has that is not known to be common with the speaker is kept in a module called *hearer*. These beliefs cannot be put in *facts* because the speaker may not believe them: one of the conversants may have an incorrect picture of the world. This may result in disagreements between them as to the validity of the argument. The system should, nevertheless, be able to understand the speaker's thinking.

3.3 Determining Evidence Relations

Now we turn to the actual workings of the oracle. Given statements P and Q, just what tests are carried out to find the answer to the question of evidence?

If there is an evidence relation from P to Q then P and Q should fit one of our frames of evidence (see tables 2.1 and 2.2) with P as premise and Q as conclusion. If none of these frames fit then no evidence relation is found and the oracle should return 'no'. On the other hand, P and Q may fit more than one of these frames, or may fit one frame in more than one way (for example, if there are rules $P \wedge R_1 \rightarrow Q$ and $P \wedge R_2 \rightarrow Q$). We collect these *candidate relations* into a list.

We now select one candidate for closer consideration. We want to determine whether it is plausible that this candidate is the one the speaker is using. The way we test this is to inspect the missing portions of the appropriate frame and test whether these are plausible beliefs of the speaker.

If the missing beliefs are from the set of shared beliefs then they are very plausible. The speaker will recognize the connection and will have confidence that the listener can also do so. If the beliefs appear in the speaker model then they are plausible to the speaker. We can reasonably assume that since we deduced their presence in the speaker's beliefs that the speaker believes we can recognize his use of them. (This is especially true if the missing beliefs are from those beliefs that he has stated).

If the beliefs are typical enough of people then we can reasonably assume that the speaker has

[4]If the discourse concerns the reasoning processes of other agents then it may become important to distinguish these two types of predicates within the *facts* module.

them and either assumes that we have them (and does not, therefore, state them), or will mention them later in his argument. If the beliefs are not typical, and yet are not denied by the speaker then they may still be recognizable. These beliefs may be used if we think that the two propositions are related, but we should probably look for more believable relations instead.

Now presume that we have decided that an evidence relation exists. Shall we just return a 'yes' and be done with it? As was stated before, these missing beliefs are required of the speaker, so we might as well make a note of them. The missing parts of the appropriate frame will be added to module *missing*.

We did not say earlier how we would select a relation from our list of candidates. It is possible that there will be more than one candidate, so we must have a way of choosing among them.

The easiest solution is to take them in any convenient order (for example, in the order they were generated). The first plausible relation chosen will be the one we use. If this choice leads to an inconsistency later on, then we will simply backtrack to this point and select the next valid alternative.

But if we're going to choose one we might as well try to choose the best one. In that way we minimize the danger of having to backtrack. We therefore introduce the notion of relative worth as a way of judging the "best" relation.

Table 3.1(a) shows the *belief level* that would be attributed to a predicate R, where R is one of the beliefs missing from the frame (lower numbers indicate more believable predicates). The numbers in table 3.1(a) are found by summing the weights in table 3.1(b). A predicate R is true in module M if $R \in M$, false if $\neg R \in M$ and unknown otherwise. If R is false in any of *facts, explicit* or *missing* then it is rejected as implausible.

Example 3 Suppose we have predicate A, which is true in *explicit* and unknown in *facts, missing* and *stereotype*, and predicate B, which is true in *missing* and unknown in the others. The weights that apply to A (from Table 3.1(b)) are 0, 0, 3 and 1, for a total 4 (*cf.* Table 3.1(a)). For B the weights are 0, 6, 0 and 1 for a total of 7. The predicate that was explicitly stated by the speaker is more believable than the one we deduced from his argument.

Truth value in *explicit/missing*	Truth value in *stereotype*		
	True	Unknown	False
True/True	0	1	2
True/Unknown	3	4	5
Unknown/True	6	7	8
Unknown/Unknown	9	10	11

3.1(a) Belief Level

Module	Truth value of predicate		
	True	Unknown	False
facts	0	0	*reject*
explicit	0	6	*reject*
missing	0	3	*reject*
stereotype	0	1	2

3.1(b) Weights

Table 3.1: Relative Worth

Example 4 Suppose the modules are as follows:

$$\begin{aligned} facts = &\ \{\ \} \\ explicit = &\ \{\ greek(X) \wedge man(X) \rightarrow mortal(X) \\ &\quad\ greek(X) \wedge woman(X) \rightarrow mortal(X)\} \\ missing = &\ \{\ \} \\ stereotype = &\ \{\ man(socrates)\} \end{aligned}$$

We are asked to test *greek(socrates)* as evidence for *mortal(socrates)*. We find two instances of the *Modus Ponens* frame of evidence. The missing beliefs are, respectively,

$$M_1 = \{\ greek(socrates) \wedge man(socrates) \rightarrow mortal(socrates), man(socrates)\}$$

$$M_2 = \{\ greek(socrates) \wedge woman(socrates) \rightarrow mortal(socrates), woman(socrates)\}$$

The major premises are each true in explicit and unknown in other modules, so each has a belief level of $0+0+3+1=4$. *man(socrates)* is unknown in *explicit* and *missing*, and true in *stereotype*,

so has a belief level of 9. *woman(socrates)* is unknown in all modules, so has a belief level of 10. Comparing the belief levels of *man(socrates)* and *woman(socrates)* we decide that *man(socrates)* is more believable. A set is no more believable than its least believable element[5], so the belief levels of M_1 and M_2 are 9 and 10 respectively. On this basis we decide that the first frame given (the one with missing beliefs M_1) is the frame the speaker most likely has in mind. Therefore we add the elements of M_1 to missing and return 'yes'.

We did not discuss how we find the Major Premise when that slot of the frame is missing. The simplest rule, $P \rightarrow Q$, is not always the one the speaker intends. What we want is a rule that the speaker believes (not necessarily explicitly) which has P in the antecedents and Q as the consequent. Since we have a list of what the speaker believes a simple search might seem appropriate.

But sometimes the rule is not in the knowledge base in the form required. We must check the contrapositive forms of the rules as well as the positive form. The speaker may chain rules together, leaving out some steps so that the argument does not become too long. Thus we must look for rules made by combining other rules.

To prevent an explosion in rules, rules which appear in *facts* or *explicit* are not added to *missing*. Therefore, to simulate the believability levels given above the oracle searches these modules in the order indicated above (*i.e., facts, explicit, missing, stereotype*). When belief level is insufficient to separate a pair of evidence relations the order they were found in determines which will be selected.

3.4 The Simple Oracle

In this section we examine the earliest version of the oracle. This version implements only the 'for all' version of rules. The code for *evidence* is given in figure 3.1. Examples given have predicate form on the left and a natural language version on the right. After the interpretation of each example is a run of the oracle indicating how it handles the example. The oracle prints out a number of messages that are meant to help the reader follow the system's reasoning. These messages have been edited to make them easier to read and so that messages unimportant to the example at hand do not appear. Note that 'lnot(X)' is the logical not of X (as opposed to Wup's negation-as-failure) and 'for-all (P [Q R])' is the predicate form of $P \leftarrow Q \wedge R$.

[5] "A chain is no stronger than its weakest link."—proverb

```
%    Q <- P ^ M
% minor premise
evidence ( Q P M ) <-
        clause ( Q B )
        delete ( P B M );
% major premise
evidence ( Q for-all(Q M) M );

%    partial evidence
% consequent
evidence ( for-all(Q M) Q M );
% antecedent
evidence ( for-all(Q B) P [Q|M] ) <-
        delete ( P B M );
% contrapositive forms
evidence ( for-all(Q B) Not_Q [Not_P] ) <-
        lnot ( Q Not_Q )
        member ( P B )
        lnot ( P Not_P );

evidence ( for-all(Q B) Not_P [Not_Q] ) <-
        member ( P B )
        lnot ( P Not_P )
        lnot ( Q Not_Q );

evidence ( for-all(Q B) P [Not_Q Not_R] ) <-
        delete ( P B B' )
        member ( R B' )
        lnot ( R Not_R )
        lnot ( Q Not_Q );

%    denying a rule
% counterexamples to the rule
evidence ( lnot ( for-all(Q M) ) Not_Q M ) <-
        lnot ( Q Not_Q );

evidence ( lnot ( for-all(Q B) ) P [Not_Q|M] ) <-
        delete ( P B M )
        lnot ( Q Not_Q );
% examples of the rule
evidence ( lnot ( for-all(Q B) ) E M ) <-
        evidence ( for-all(Q B) E M );
```

Figure 3.1: The Evidence Relations

Example 5 Suppose that the hearer (our stereotype for the speaker) has the rule

$$francophone(X) \leftarrow french(X)$$

and the speaker's argument is

(1) francophone (trudeau) Trudeau is a francophone.

(2) french (trudeau) He is French

```
? oracle ( francophone(trudeau) french(trudeau) );

french(trudeau) is evidence for
  francophone(trudeau) with
    [for-all(francophone(trudeau) [french(trudeau)])] missing
  for-all(francophone(trudeau) [french(trudeau)]) has belief level 9
The conjunction has belief level 9

yes
? print-beliefs ( missing )

module missing:

        for-all(francophone(trudeau) [french(trudeau)])
yes
```

The orace is asked whether french(trudeau) is evidence for francophone(trudeau). The relation is found to be the missing major premise form of *Modus Ponens*. The appropriate rule does not appear in *facts* nor in the speaker model, but does appear in the stereotype; it has, thus, a belief level of 9.

Example 6 Compare the above to the following:

(1) francophone (trudeau) Trudeau is a francophone.

(2) francophone(X) ← All French people are.

french(X)

```
? oracle ( francophone(trudeau) for-all(francophone(X) [french(X)]) );
```

```
for-all(francophone(trudeau) [french(trudeau)]) is evidence for
  francophone(trudeau) with
    [french(trudeau)] missing
  french(trudeau) has belief level 9
The conjunction has belief level 9

yes
? print-beliefs ( missing );

module missing:

        french(trudeau)
yes
```

Here the minor premise is missing and the major given. We naturally fill in the missing belief that Trudeau is French.

3.5 Using Generalizations

When making an argument, especially in a relaxed setting, the speaker does not always use exact logic. He resorts to generalizations—relaxed logic. The speaker might say "Bilandic will win because he is the machine candidate," without believing that the machine candidate *always* wins. The belief that he usually wins is enough for normal conversation.

Example 7 Suppose the following generalizations appear in *missing*:

$$adult(X) \wedge american(X) \rightarrow\!\triangleright owns\text{-}car(X)$$

$$adult(X) \wedge canadian(X) \rightarrow\!\triangleright owns\text{-}car(X)$$

Suppose further that the hearer believes Greg to be a Canadian. We are given the following argument to parse:

(1) owns-car(greg)	Greg owns a car.
(2) adult(greg)	He is an adult.

```
? oracle ( owns-car(greg) adult(greg) );
```

```
adult(greg) is evidence for
  owns-car(greg) with
    [american(greg)] missing
  american(greg) has belief level 10
The conjunction has belief level 10

adult(greg) is evidence for
  owns-car(greg) with
    [canadian(greg)] missing
  canadian(greg) has belief level 9
The conjunction has belief level 9

yes
? print-beliefs ( missing );

module missing:

        canadian(greg)
yes
```

The oracle has a choice of generalizations to use. If the system has no knowledge of the nationality of Greg it will choose whichever of (B1) and (B2) was more recently asserted. In the case above, the hearer believes Greg to be Canadian, and is using his own beliefs as a stereotype for the speaker, therefore assumes that the speaker also believes Greg to be Canadian. If the *EO*'s choice is later contradicted (if the speaker later states that Greg is not Canadian) then the system would backtrack and select the next alternative[6]. For an example of bactracking see section 3.7.

When the speaker uses a generalization, a note is made of its use. This is because the generalization is not always true, and it may be possible to prove the generalization wrong in the specific case. This sort of information should be passed on to the *RU*.

3.6 Making Generalizations

Now suppose we have propositions P and Q and our procedures so far have failed to find an evidence relation between them. This does not mean that none exists—our model of the speaker is not

[6]See Section 4.4 for a description of how the system might work in the future.

complete and the intended relation may be one of the things missing. We should test P and Q to see if any relation between them is plausible.

We might want to see whether there is any relation from P to Q in our own beliefs (assuming they have not already been consulted as a stereotype for the speaker). We might also want to consult some sort of a 'bin' of relations that we have encountered in interactions with other speakers. If either of these methods yields a relation we can test it against the speaker's beliefs to see if it might apply.

But when all those routes have been exhausted we still have an avenue open to us. We can see whether it makes any sense at all for there to be a relation from P to Q in our world—whether a rational person *might* believe such a connection.

Example 8 Suppose we are testing P for Q where

$$P = \textit{shark(joey)}$$
$$Q = \textit{dangerous(joey)}$$

Noting that "Joey" is common to P and Q, we postulate the rule

$$\textit{shark(X)} \mathrel{-\!\!\triangleright} \textit{dangerous(X)}$$

We then look in our database to see whether this rule is reasonable. If we find multiple examples of dangerous sharks and few of non-dangerous ones we might decide that the rule is plausible.

Here we run into a problem: just how many examples and how few counterexamples do we need to find a rule "plausible"? If P and Q are instances of predicates that are almost always true then we will almost certainly have more examples than counterexamples. In a database dealing only with humans we might find numerous examples and no counterexamples to the rule $\neg\textit{lived-to-100(X)} \mathrel{-\!\!\triangleright} \textit{had-2-arms(X)}$, yet we would not likely want to find the rule valid.

We shall use the following two tests to measure the plausibility of $P \mathrel{-\!\!\triangleright} Q$:

(i) There are more instances of $P \wedge Q$ than of $P \wedge \neg Q$. That is, when P is true, Q is more likely to be true than false.

(ii) There are more instances of $\neg P \wedge \neg Q$ than of $P \wedge \neg Q$. That is, when Q is false, P is more likely to be false than true.[7]

Now in order for *¬lived-to-100(x)* to be evidence for *had-2-arms(x)* it must also be true that among the people who did not have two arms there must be more that lived to be one hundred than did not.

Example 9 Suppose the speaker makes the statements indicated in Example 8. The oracle deals with it as indicated above. The examples found by the oracle are printed out here for the benefit of the reader.

```
? print-beliefs(hearer);

 module: hearer

        shark(joey)
        dangerous(joey)
        shark(fred)
        dangerous(fred)
        whale(louis)
        lnot(dangerous(louis))
        whale(carl)
        dangerous(carl)
        for-all(lnot(shark(X)) [whale(X)])
yes

? oracle ( dangerous(joey) shark(joey) );

examples of (shark(X) ^ dangerous(X))
        example (shark(joey) ^ dangerous(joey))
        example (shark(fred) ^ dangerous(fred))
#(shark(X) ^ dangerous(X)) = 2

examples of (lnot(dangerous(X)) ^ lnot(shark(X)))
        example (lnot(dangerous(louis)) ^ lnot(shark(louis)))
#(lnot(dangerous(X)) ^ lnot(shark(X))) = 1

examples of (lnot(dangerous(X)) ^ shark(X))
#(lnot(dangerous(X)) ^ shark(X)) = 0
```

[7] One might want to count any instance P or $\neg Q$ for which the truth value of the corresponding Q or P is unknown as a counterexample. In our current implementation, however, these cases are merely ignored.

```
yes

? print-beliefs(missing);

 module: missing

        for-most([joey],dangerous(joey),[shark(joey)])
yes

? print-beliefs ( stereotype );

 module: stereotype

        for-most([X],dangerous(X),[shark(X)])
yes
```

In looking for examples and counterexamples the oracle makes deductions in its own knowledge base. It identifies Louis and Carl as non-sharks by the rule that whales are not sharks.

The indicated rule is found to be plausible. The non-unified version of the rule is added to module *stereotype*. This module has been empty to date, but now contains a belief that the system does not. The system will continue to consult *hearer* whenever a call to *stereotype* is made.

3.7 The Oracle So Far

The previous sections have shown how the oracle works. It attempts to match the given propositions to frames of evidence. It judges the plausibility of missing premises, rejecting relations that result in implausible beliefs. If more than one relation is valid, the oracle selects the one it judges most plausible. When all else fails, the *EO* will create a generalization and test how plausible that rule is.

In this section we show the current version of the oracle at work on various problems. Example 10 shows the Refutation and Concession frames in use, and considers the issue of choosing the best frame. Examples 11 and 12 show how partial evidence is handled by the oracle. A longer example using both *Modus Ponens* and Partial Evidence is example 13. Example 14 has the oracle fixing an incorrect choice by backtracking to that choice.

In these examples we use a simple parser. This parser assumes the speaker is using a claim first transmission strategy. It prints a message before and after each call to the oracle. Messages

have been edited to make them easier to read and to eliminate those that are not important to the example. After the parse is complete the parser prints a representation of the argument; the evidence to a claim appears below it and indented one level to the right[8].

Example 10

(1) lnot(genius(X) ← laughed-at(X))	Not everyone who is laughed at is a genius.
(2) laughed-at(columbus)	Sure, they laughed at Columbus;
(3) laughed-at(bozo-the-clown)	but they also laughed at Bozo the Clown.

```
? print-beliefs ( facts );

module: facts

        lnot(genius(bozo-the-clown))
yes

? print-beliefs ( hearer );

module: hearer

        genius(columbus)
yes

? parse ( [     lnot ( for-all ( genius(X) [laughed-at(X)] ) )
                laughed-at ( columbus )
                laughed-at ( bozo-the-clown )  ] );

considering laughed-at(columbus) for lnot(for-all(genius(X),[laughed-at(X)]))
  laughed-at(columbus) is evidence for
    lnot(for-all(genius(columbus),[laughed-at(columbus)])) with
      [lnot(genius(columbus))] missing.
    lnot(genius(columbus)) has belief level 11
  The conjunction has belief level 11

  laughed-at(columbus) is evidence for
    lnot(for-all(genius(columbus),[laughed-at(columbus)])) with
      [genius(columbus)] missing.
```

[8]The parser only makes the calls in the order required by claim first transmission, it does not implement the behaviour of the *AUS*. The oracle is still a stand-alone system. See section 4.4 for a discussion of how to modify the oracle to integrate with the *AUS*.

```
    genius(columbus) has belief level 9
  The conjunction has belief level 9

        success laughed-at(columbus)
          for lnot(for-all(genius(X),[laughed-at(X)]))

considering laughed-at(bozo-the-clown) for laughed-at(columbus)
        failure laughed-at(bozo-the-clown) for laughed-at(columbus)

considering laughed-at(bozo-the-clown)
    for lnot(for-all(genius(X),[laughed-at(X)]))
  laughed-at(bozo-the-clown) is evidence for
    lnot(for-all(genius(bozo-the-clown),[laughed-at(bozo-the-clown)])) with
      [lnot(genius(bozo-the-clown))] missing.
    lnot(genius(bozo-the-clown)) has belief level 0
  The conjunction has belief level 0

  laughed-at(bozo-the-clown) is evidence for
    lnot(for-all(genius(bozo-the-clown),[laughed-at(bozo-the-clown)])) with
      [genius(bozo-the-clown)] missing.
    genius(bozo-the-clown) is not believed by the model

        success laughed-at(bozo-the-clown)
          for lnot(for-all(genius(X),[laughed-at(X)]))

lnot(for-all(genius(X),[laughed-at(X)]))
        laughed-at(columbus)
        laughed-at(bozo-the-clown)
yes

? print-beliefs(missing);

 module: missing

        lnot(genius(bozo-the-clown))
        genius(columbus)
yes
```

When considering (2) as evidence for (1) there are two interpretations possible. One interpretation is that Columbus is given to prove that not all who are laughed at are geniuses; the other is that he is a concession—a genius who was laughed at. The presence of a clue word, "Sure", in the speaker's argument makes it plain to us that Columbus is a concession. The oracle, unfortunately, does not have access to that clue; it must decide which is meant by other means.

It turns out that the system believes Columbus to be a genius, so the latter interpretation is selected as the more likely. Had the system thought Columbus a fool (*i.e.*, not a genius), then the former interpretation would have been selected. Had the system no knowledge of Columbus, the

former would again have been selected, since it was the first found[9].

The same problem arises with Bozo the Clown, and again the speaker provides clues to the intended interpretation. In the case of Bozo, the latter interpretation is rejected as implausible, since it is common knowledge that Bozo the Clown is not a genius (though one might say he is a comic-genius).

Example 11 Here we see how partial evidence is handled by the oracle. The speaker seeks to justify the assertion that all sharks are dangerous.

(1) dangerous(X) ← shark(X)	All sharks are dangerous.
(2) dangerous(joey)	Joey is dangerous;
(3) dangerous(fred)	Fred is dangerous;
(4) lnot(shark(louis))	Louis, here, isn't really a shark.

```
? parse ( [
        for-all ( dangerous(X) [shark(X)] )
        dangerous(joey)
        dangerous(fred)
        lnot(shark(louis))
        ] );

for-all(dangerous(X) [shark(X)])
        dangerous(joey)
        dangerous(fred)
        lnot(shark(louis))
yes
? print-beliefs ( missing );

module missing:

        lnot(dangerous(louis))
        shark(fred)
        shark(joey)
yes
```

[9]The order of search was set up so that Refutations would be preferred to Concessions.

Here, (2), (3) and (4) are found to be (partial) evidence to (1). The oracle makes no judgement on how plausible the overall argument is, merely on how plausible the individual relations are.

Example 12 Examples may also be given in support of a generalization. A problem that must be considered is that the speaker may also grant counter-examples to the generalization he is trying to justify.

(1) dangerous(X) ⇐ shark(X)	Most sharks are dangerous.
(2) dangerous(doris)	Doris is dangerous.
(3) dangerous(martha)	Martha is dangerous.
(4) lnot(dangerous(jenny))	I will grant you, though, that Jenny isn't dangerous.

```
? print-beliefs ( hearer );

 module: hearer

        shark(jenny)
yes

? parse ( [     for-most ( [X] dangerous(X) [shark(X)] )
                dangerous ( doris )
                dangerous ( martha )
                lnot ( dangerous(jenny) ) ] );

considering dangerous(doris) for for-most([X] dangerous(X) [shark(X)])

  dangerous(doris) is evidence for
    for-most([doris] dangerous(doris) [shark(doris)]) with
      [shark(doris)] missing.
    shark(doris) has belief level 10
  The conjunction has belief level 10

        success dangerous(doris) for for-most([X] dangerous(X) [shark(X)])

considering dangerous(martha) for dangerous(doris)
        failure dangerous(martha) for dangerous(doris)

considering dangerous(martha) for for-most([X] dangerous(X) [shark(X)])
```

```
  dangerous(martha) is evidence for
    for-most([martha] dangerous(martha) [shark(martha)]) with
      [shark(martha)] missing.
    shark(martha) has belief level 10
  The conjunction has belief level 10

        success dangerous(martha) for for-most([X] dangerous(X) [shark(X)])

considering lnot(dangerous(jenny)) for dangerous(martha)
        failure lnot(dangerous(jenny)) for dangerous(martha)

considering lnot(dangerous(jenny)) for for-most([X] dangerous(X) [shark(X)])

  lnot(dangerous(jenny)) is evidence for
    for-most([jenny] dangerous(jenny) [shark(jenny)]) with
      [lnot(shark(jenny))] missing.
    lnot(shark(jenny)) has belief level 11
  The conjunction has belief level 11

  lnot(dangerous(jenny)) is evidence for
    for-most([jenny] dangerous(jenny) [shark(jenny)]) with
      [shark(jenny)] missing.
    shark(jenny) has belief level 9
  The conjunction has belief level 9

        success lnot(dangerous(jenny)) for for-most([X] dangerous(X) [shark(X)])

for-most([X] dangerous(X) [shark(X)])
        dangerous(doris)
        dangerous(martha)
        lnot(dangerous(jenny))
yes

? print-beliefs ( missing );

 module: missing

        shark(jenny)
        shark(martha)
        shark(doris)
yes
```

When the speaker concedes that Jenny is not dangerous the oracle decides that either Jenny is not a shark, or Jenny is an exception to the rule given. The frame that results in the speaker's beliefs matching the system's is the one chosen.

Example 13 is longer and uses both *Modus Ponens* and Partial Evidence frames of evidence. The main claim of the argument is that Chrétien will never lead the Liberals. He provides a rule to support his contention and some examples to back up his rule.

Example 13

(1) lnot(leader(chretien grits T0))	Chrétien will never lead the grits.
(2) lnot(leader(X P T1)) ←	No one can lead any party
lnot(english(X))	unless he is English
lnot(charismatic(X))	or charismatic.
(3) lnot(english(wagner))	Wagner, for example, was not English,
(4) lnot(leader(wagner tories T2))	and so did not get to lead the Conservatives.
(5) lnot(charismatic(clark))	Clark wasn't charismatic,
(6) english(clark)	but he was English.
(7) charismatic(trudeau)	Trudeau was charismatic.

```
? print-beliefs ( hearer );

 module: hearer

        leader(mulroney,tories,t0)
        leader(turner,grits,t0)
        leader(trudeau,grits,t1)
        leader(mulroney,tories,t1)
        leader(clark,tories,t2)
        leader(trudeau,grits,t2)
        english(mulroney)
        english(turner)
        french(trudeau)
        english(clark)
yes

?parse ( [ lnot ( leader ( chretien grits T0 ) )
        for-all ( lnot(leader(P Q R)) [lnot(english(P)) lnot(charismatic(P))] )
        lnot ( english(wagner) )
        lnot ( leader(wagner tories T1) )
```

```
        lnot ( charismatic(clark) )
        english(clark)
        charismatic(trudeau) ] );

considering for-all(lnot(leader(P,Q,R)),[lnot(english(P)),lnot(charismatic(P))])
  for lnot(leader(chretien,grits,T0))
        success for-all(lnot(leader(P,Q,R)),[lnot(english(P)),lnot(charismatic(P))])
          for lnot(leader(chretien,grits,T0))
        [lnot(english(chretien)),lnot(charismatic(chretien))] missing.

considering lnot(english(wagner))
  for for-all(lnot(leader(P,Q,R)),[lnot(english(P)),lnot(charismatic(P))])
        success lnot(english(wagner))
          for for-all(lnot(leader(P,Q,R)),[lnot(english(P)),lnot(charismatic(P))])
        [lnot(leader(wagner,Q,R)),lnot(charismatic(wagner))] missing.

considering lnot(leader(wagner,tories,T1)) for lnot(english(wagner))
        failure lnot(leader(wagner,tories,T1)) for lnot(english(wagner))

considering lnot(leader(wagner,tories,T1))
  for for-all(lnot(leader(P,Q,R)),[lnot(english(P)),lnot(charismatic(P))])
        success lnot(leader(wagner,tories,T1))
          for for-all(lnot(leader(P,Q,R)),[lnot(english(P)),lnot(charismatic(P))])
        [lnot(english(wagner)),lnot(charismatic(wagner))] missing.

considering lnot(charismatic(clark)) for lnot(leader(wagner,tories,T1))
        failure lnot(charismatic(clark)) for lnot(leader(wagner,tories,T1))

considering lnot(charismatic(clark))
  for for-all(lnot(leader(P,Q,R)),[lnot(english(P)),lnot(charismatic(P))])
        success lnot(charismatic(clark))
          for for-all(lnot(leader(P,Q,R)),[lnot(english(P)),lnot(charismatic(P))])
    [leader(clark,Q,R),english(clark)] missing.

considering english(clark) for lnot(charismatic(clark))
        failure english(clark) for lnot(charismatic(clark))

considering english(clark)
  for for-all(lnot(leader(P,Q,R)),[lnot(english(P)),lnot(charismatic(P))])
        success english(clark)
          for for-all(lnot(leader(P,Q,R)),[lnot(english(P)),lnot(charismatic(P))])
        [leader(clark,Q,R)] missing.

considering charismatic(trudeau) for english(clark)
        failure charismatic(trudeau) for english(clark)

considering charismatic(trudeau)
  for for-all(lnot(leader(P,Q,R)),[lnot(english(P)),lnot(charismatic(P))])
        success charismatic(trudeau)
          for for-all(lnot(leader(P,Q,R)),[lnot(english(P)),lnot(charismatic(P))])
        [leader(trudeau,Q,R)] missing.
```

```
lnot(leader(chretien,grits,T0))
        for-all(lnot(leader(P,Q,R)),[lnot(english(P)),lnot(charismatic(P))])
                lnot(english(wagner))
                lnot(leader(wagner,tories,T1))
                lnot(charismatic(clark))
                english(clark)
                charismatic(trudeau)
yes

? print-beliefs ( explicit );

 module: explicit

        charismatic(trudeau)
        english(clark)
        lnot(charismatic(clark))
        lnot(leader(wagner,tories,T1))
        lnot(english(wagner))
        lnot(leader(chretien,grits,T0))
        for-all(lnot(leader(P,Q,R)),[lnot(english(P)),lnot(charismatic(P))])
yes
? print-beliefs ( missing );

 module: missing

        leader(trudeau,grits,t1)
        leader(clark,tories,t2)
        english(clark)
        lnot(english(wagner))
        lnot(leader(wagner,Q,R))
        lnot(charismatic(wagner))
        lnot(english(chretien))
        lnot(charismatic(chretien))
yes
```

Statement (3) has two interpretations, as does statement (5). In both cases the indicated man could either have been a leader (Contrapositive Example) or not been a leader (Positive Example). In the case of Clark the system knows that he was a leader of the Conservative party, so it selects the former frame. The system has no knowledge of Wagner, so selects Positive Example by default (the search has been ordered so that this is so).

The statement "Trudeau was charismatic" is interpreted as giving Trudeau as a Contrapositive Example for the rule. The system makes no assumption about whether the speaker believes Trudeau to be English or not, as this distinction is not important to the frame.

The use of (2) as evidence for (1) forces the system to conclude that the speaker believes Chretien

to be neither English nor charismatic. Again, the example shows that this conclusion is drawn by the oracle.

This example also indicates the importance of clues to the parsing of an argument. If the parsing method being used were hybrid, then, after finding (4) as evidence for (2), (3) would be tested as evidence for (4). The test would succeed, and no beliefs would have to be changed, but (4) would be closed off from further testing by the *PA*. If the speaker *had* provided evidence for (4), then that evidence would not have found its correct place in the representation. The problem arises because the oracle gets only two propositions at a time; the system does not realize that (3) and (4) are two parts of the same frame and so treats them independently. Future versions of the *PA* should perhaps take note of the frames being used and try to fit propositions into the current frame. Propositions which fit different premise positions of the current frame need not be tested as evidence for each other.

Example 14 The next example shows how backtracking handles incorrect assumptions made by the oracle. Shared knowledge indicates three ways that a party, P, can become the government of the day. These are

- P has a majority in Parliament.
- No party has a majority, but P is the largest party and no coalition forms against them.
- No party has a majority, P is not the largest party, but leads a coalition that has a majority in parliament.

The speaker's argument proceeds as follows:

(1) govt(grits 26)	The Liberals formed the government in 1926.
(2) nosuch([Q] majority(Q 26))	No party had the majority in Parliament,
(3) lnot(plurality(grits 26))	and the Liberals weren't even the largest party,
(4) support(grits [progressives] 26)	but they had the support of the Progressives.

```
? add-beliefs ( facts
[
        for-all ( govt(P T)
        [       majority(P T) ] )
        for-all ( govt(P T)
        [       nosuch([Q] majority(Q T))
                plurality(P T)
                lnot(coalition-majority([R|S] T)) ] )
        for-all ( govt(P T)
        [       nosuch([Q] majority(Q T))
                lnot(plurality(P T))
                supports(P S T)
                coalition-majority([P|S] T) ] )
] );
yes
? parse (
        [ govt(grits 26)
          nosuch([Q] majority(Q 26))
          lnot(plurality(grits 26))
          supports(grits [progressives] 26)
        ]);
considering nosuch([Q] majority(Q 26)) for govt(grits 26)
        success nosuch([Q] majority(Q 26)) for govt(grits 26)
        missing [plurality(grits 26) lnot(coalition-majority([R|S] 26))]
considering lnot(plurality(grits 26)) for nosuch([Q] majority(Q 26))
        failure lnot(plurality(grits 26)) for nosuch([Q] majority(Q 26))
considering lnot(plurality(grits 26)) for govt(grits 26)
        failure lnot(plurality(grits 26)) for govt(grits 26)
%
%         The first call to the oracle resulted in the oracle deciding
%         that the speaker believes the Liberals to have been the largest
%         party in parliament in 1926. Statement (3) directly contradicts
%         this assumption. The two calls to the oracle involving (3) are
%         failed without checking for frames of evidence.
%         Since claim first transmission requires that every utterance
%         fit into the argument at once, the parser is required to back-
%         track to the first call and try again.
        success nosuch([Q] majority(Q 26)) for govt(grits 26)
        missing [lnot(plurality(grits 26)) supports(grits S 26)
                 coalition-majority([grits|S] 26)]
considering lnot(plurality(grits 26)) for nosuch([Q] majority(Q 26))
        failure lnot(plurality(grits 26)) for nosuch([Q] majority(Q 26))
considering lnot(plurality(grits 26)) for govt(grits 26)
        success lnot(plurality(grits 26)) for govt(grits 26)
        missing [nosuch([Q] majority(Q 26)) supports(grits S 26)
                 coalition-majority([grits|S] 26)]
considering supports(grits [progressives] 26) for lnot(plurality(grits 26))
        failure supports(grits [progressives] 26) for lnot(plurality(grits 26))
considering supports(grits [progressives] 26) for govt(grits 26)
        success supports(grits [progressives] 26) for govt(grits 26)
        missing [nosuch([Q] majority(Q 26)) lnot(plurality(grits 26))
```

```
                coalition-majority([grits progressives] 26)]

govt(grits 26)
        nosuch([Q] majority(Q 26))
        lnot(plurality(grits 26))
        supports(grits [progressives] 26)
yes
? print-beliefs(explicit);

 module: explicit

        supports(grits [progressives] 26)
        lnot(plurality(grits 26))
        govt(grits 26)
        nosuch([Q] majority(Q 26))
yes
? print-beliefs(missing);

 module: missing

        nosuch([Q] majority(Q 26))
        lnot(plurality(grits 26))
        supports(grits [progressives] 26)
        coalition-majority([grits progressives] 26)
yes
```

After changing the selection made at the first call, the rest of the argument falls into place. Module *missing* indicates that the speaker believes that the Progressives and Liberals together had enough seats to form the government.

Chapter 4

Future Work

4.1 Clues

An important aspect of natural language arguments that the *EO* does not deal with yet is the presence of clues—special words and phrases that indicate the structure of the argument. For the most part these clues will affect only the *PA*, allowing it to reduce the number of pairs of propositions that it sends to the oracle. Other clues will, however, impact on the operation of the oracle.

Cohen gives ([Cohen83, p.44]) six types of *connective* clues. Evidence relations involving a proposition that is marked by one of these clues must conform to the special semantics of the connective. The clue will be passed on to the oracle so that inappropriate relations may be rejected. The classes are shown in Table 4.1. In the table, S refers to the phrase with the connective clue and P to the prior proposition the clue relates to.

Some of the frames of evidence we have discussed previously fit these categories. Examples (positive and contrapositive) are examples of 'detail'. Concession and counterexamples are types of 'contrast'.

Addition of clues gives a new verification technique to the *EO*(see page 9):

Method 5 If a connective clue is given then verify the semantic relation indicated.

Category	Relation	Example phrase
parallel	S brother to P	"Secondly"
inference	S son to P	"As a result"
detail	S father to P	"In particular"
summary	S father to $P_1, P_2, \ldots$	"In conclusion"
reformulation	S son and father to P	"In other words"
contrast	S brother or son to P	"On the other hand"

Table 4.1: Connective Clues

This could be carried out after all possible evidence relations have been found, but would better be used to restrict the search for relations (see section 5.2). Even when the clues are not sufficient to reject some frames out of hand, they might be useful in selecting between competing frames of evidence.

4.2 Stereotypes

When looking for evidence relations in a speaker's argument one must often reason beyond one's own beliefs. This is true because the speaker is usually trying to convince the listener of something the listener does not already believe. Moreover, the speaker will often leave his reasoning unstated, assuming the listener will be able to reconstruct it. Sometimes this will involve filling in beliefs that the listener does not hold.

One way of filling in beliefs is to stereotype the speaker. In the oracle as implemented to date the system's beliefs serve as a stereotype for the speaker. For a question answering system or a discussion between people with similar beliefs this is an acceptable stereotype. But if the speaker and listener have radically different beliefs then this will be an insufficient model. If we have some further information on the speaker's ideology we may be able to fill in some evidence relations from a "standard" stereotype for that ideology.

For example, consider the following pair of statements.

Example 15

(1) This results in greater revenue for businesses.

(2) Unemployment will be reduced.

If the speaker has been stereotyped as a conservative on economics then we should be able to trace a link from increased revenues to more hiring to reduced unemployment, even if our own beliefs would indicate that either link is highly unlikely.

If the system keeps a number of stereotypes around then it may even be possible for the system to select one on the basis of the propositions presented by the speaker. Even if the speaker in example 15 had not previously been stereotyped as a conservative we might feel that it would be appropriate to do so *because* of the juxtaposition of (1) and (2). If the system is unable to find any evidence relation outside of the stereotypes nor in any of the other stereotypes then it would be a reasonable assumption that the speaker conforms to (at least part of) that stereotype.

If the oracle has access to a model of the speaker that was compiled at a different time (either in another session or in a previous argument in the current session) then this will serve as an excellent stereotype. The model should not be incorporated directly into the model since beliefs *do* change over time[1]—the oracle should be able to accept changing beliefs on the part of the speaker[2]. This ability to keep track of beliefs that we "think" the speaker has along with those we "know" he has was the main motivation for having a separate stereotype module.

4.3 Extended Testing

When the speaker is presenting his argument to the listener he has a plan. The plan is to convince the listener of the beliefs advocated. To accomplish this plan he should start from shared beliefs to build his argument, and he should relate his utterances to the beliefs of the listener.

These two strategies put some restrictions on what the speaker can give as claim and evidence. First, he should never give evidence for any statement that he thinks the listener already believes.

[1]Beliefs might also change when the focus of the conversation changes. [Fagin&Halpern??] provides for "local reasoning", where the user is viewed as a "society of minds", each member of that society having its own beliefs, possibly contradicting other members.

[2]This would seem to be especially true of question answering systems. If the system perceives an incorrect belief in the user then it should try to correct that misconception. See section 6.2 for further discussion.

```
evidence_oracle ( E A _ ) <-
        asserted_evidence ( E A );
evidence_oracle ( E A _ ) <-
        not ( asserted_evidence ( E A ) )
        not ( unasserted_evidence ( E A ) )
        call_oracle ( E A Result )
        assert_evidence ( Result E A )
        !
        eq ( Result 'y' );

call_oracle ( E A 'y' ) <-
        oracle ( A E );
call_oracle ( E A 'n' );
```

Figure 4.1: Code to integrate *EO* and *PA*

In particular, this rules out giving evidence for anything in module *facts*. Second, he should never advocate any belief that he knows the listener cannot accept without becoming inconsistent. These two restrictions indicate that the listener's beliefs affect the structure of the speaker's argument[3]. So far our implementation has used the hearer's beliefs as a stereotype for the speaker's. When more appropriate stereotypes are allowed the speaker's beliefs will still be consulted, but with a different purpose in mind.

4.4 Integration with the AUS

The *EO* as it is written could be integrated with the *PA* of [Smedley86]. Smedley's version of the *PA* asks the user to provide the answer to the question of evidence. This answer is saved and consulted at various times in the parse. Figure 4.1 shows the code required in the *PA* to replace Smedley's *evidence_oracle* predicate.

Note that the method does not allow backtracking to the oracle call. Our implementation of the *EO* was built on the premise that backtracking would allow us to correct misconceptions about missing beliefs (see example 14, section 3.7). Integration with the *PA* will destroy this property.

[3]Actually it is what the speaker believes the listener to believe that is important. If the distinction is made the system should still use its own beliefs as a stereotype of what the speaker believes about it.

Furthermore, it is important to the complexity arguments of [Cohen83] that backtracking not occur. If the selection of an evidence relation on the first call to the oracle results in an inability to find a believable relation on the last (i.e, the last one that should succeed), then the system will have to backtrack all the way to the first call to undo its effects. This change in beliefs at the first call will affect all future calls, so they will all have to be redone. It becomes impossible to keep the number of calls down to linear in the number of predicates because the first decision could affect the last for each relation found in the first.

To ensure that our oracle does not cause a valid argument to be rejected as incoherent, we must somehow modify the way the oracle adds beliefs to *missing*. There are three ways I see of doing this:

- We could not add anything to *missing*, being thereby sure that we have not ascribed any wrong beliefs to him. This solution will result in a less complete speaker model, and may result in acceptance of some relations that a more discerning listener would reject.

- We could keep all relations found and eliminate those that are contradicted later in the argument. This will require keeping disjunctive knowledge in *missing*, and result in more complex insertion algorithms.

- We could postpone handling the problem. We could have the *RU* try to "fix" the representation after the parse has resulted in an incoherent representation[4]. By redoing the failed evidence tests and allowing for contradictions within *missing* it may be possible to isolate the error that caused the problem, which may then be fixed. If the error rate is low then it is likely that this "fixing up" will not have to be done very often and therefore the method will be inexpensive on average. Of course, errors may "cascade", the fix of one resulting in an error in another. The frequency of errors and the danger of cascading must be measured before it can be determined whether this method or the previous one is the better.

The *AUS* is implemented in three pieces. To date we have a *PA* and an *EO*, and work is proceeding on the *Clue Interpreter* ([Smedley87]). The *CI* is not necessary for the operation of the *AUS*, but could result in substantial improvements in its operation.

[4]A correctly parsed coherent argument is represented by an ordered tree, an incoherent by an ordered forest. If a coherent argument is parsed incorrectly it may also yield a forest.

Chapter 5

Implementation Details

5.1 Storing Modules

The *EO* uses and maintains a speaker model to aid it in its deliberations. In this section we discuss how this model (and the rest of the model of the world) is stored.

The normal way in Prolog to build a representation of some item is to represent the current and final models of that item as arguments of the predicate that builds the model. Our model would be a structure of some sort that incorporated lists of beliefs within it. The problem with this representation is Prolog's binding of variables might make it impossible to use a rule more than once. If the rule $p(X) \rightarrow q(X)$ is used to show that $p(a)$ is evidence for $q(a)$ then it (the rule) will be replaced by $p(a) \rightarrow q(a)$. Special processing would be required to maintain the rules in a non-bound state.

One way around this is to keep the beliefs as Prolog assertions. In this way only the local copies of the rules are modified by variable bindings. This was the way the earliest version of the oracle was implemented and the terminology of "belief modules" is still used. The oracle maintained modules called *facts, speaker* and *hearer* wherein were kept the appropriate beliefs. Module *speaker* was subdivided into *explicit* and *missing.*

The original oracle had a problem when it came to modifying beliefs. Because of the way Wup handles asserting and retracting we could never be sure that the correct beliefs were being removed

when backtracking occurred. Consider the following (contrived) example.

Example 16

(1) older(sam X) ← welder(X) Sam is older than any welder.

(2) older(Y fred) ← welder(Y) All welders are older than Fred.

Suppose (1) had been added to *missing* and the later addition of (2) caused a contradiction. We backtrack and try to remove (2). Wup's assertion predicate has placed (2) *after* (1) in module *missing*, but the retraction predicate selects the *first* match. Thus (1) will be removed and (2) will remain. This is a serious error in processing and is not easily fixed.

The current implementation solves the above problem by combining the list and assertion methods mentioned. Beliefs are kept in lists which are asserted to a module called *beliefs*. Each list is identified as to which belief module it represents. Variable bindings take place only in the local copy of the belief list, so rules may be re-used. Furthermore, new beliefs are added at the beginning of the list and so may be readily identified on backtracking.

5.2 Depth of Search

When checking the speaker for consistency in his views, the oracle does only a surface search, *i.e.*, it does not check the implications of the beliefs for consistency. The only time that implications are considered is when we are searching for possible evidence relations. The theoretical justification is that the speaker is not *logically omniscient*, so is not necessarily aware of the implications of what he believes. When he gives evidence to a claim, however, he must be following some line of inference, so the oracle tries to recreate it.

For simplicity, *facts* is handled the same way as the rest of the speaker's beliefs. We might want to handle these differently, though. Since these facts are common knowledge, their implications should be fairly well known as well. It will also allow for the definition of such common knowledge as the membership function, which our system cannot at this time deal with. Allowing this sort of reasoning makes testing for belief equivalent to a full first order theorem-prover, and thus only semi-decidable.

When testing the system's beliefs we do test the implications. A naive cycle check is implemented to reduce the chance of looping behaviour. This means that some implications of the hearer's beliefs may not be found. The incompleteness of the system's reasoning is not a great problem, though a similar fault in the *RU* might be so.

We actually cheat a bit in the speaker's model. We allow for a predicate *lexc(L)*, where L is a list of predicates which are logically exclusive. If R and S bind with two elements of L, then the oracle will reject a system of beliefs that includes both R and S as inconsistent. If the speaker says that no object is both red and blue, this is represented by *lexc([red(X) blue(X)])* and belief by the speaker of *blue(pyramid(a))* will force *red(pyramid(a))* to be false.

The performance of the oracle would benefit from operating in a restricted domain. Testing for a missing Major Premise involves a sort of 'proof'. The way beliefs are stored results in a retrieval time that is linear in the number of beliefs held. Having unrelated beliefs laying around will thus degrade the oracle's performance.

Chapter 6

Implications

6.1 Argument Understanding

The *EO* is the second of the modules of the *AUS* to be implemented. The implementation of the *PA* in [Smedley86] required the user to answer the question of evidence. With the *EO*, this burden is moved onto the machine, and the system begins to work on its own. The remaining module is the *Clue Interpreter*, on which work is progressing ([Smedley87]).

When the *EO* and the *PA* are integrated testing may begin on the *AUS*. By selecting a limited domain and seeing how useful the system is in understanding arguments in that domain some idea can be gained of how useful the system will be in general. The effect of design decisions made in implementing the oracle can also be studied.

There are still problems to be overcome with the *AUS* in general and the *EO* in particular. The *AUS* has no way of eliminating possible evidence pairs based on redirection clues, but must stay with the strict definitions of the transmission strategies. This may result in some pairs being tested which should not be, and thus some relations being found that were not intended.

The *EO* considers many frames of evidence, some of which could be eliminated by consideration of connective words and phrases. The cost of the oracle's computations will be reduced by the addition of clues. Clues may also help the oracle choose between valid frames of reference, reducing the chance of errors.

6.2 Plan Recognition

Pollack suggests in [Pollack86] that it would be useful to have a model of Plan Inference (*PI*) that distinguishes the beliefs of the actor (the agent with the plan) and the observer (the agent trying to infer what the plan is). The reason for making this distinction is to allow the system to recognise incorrect plans and to generate appropriate responses. The *EO* makes such a distinction in the context where the actor's plan is to convince the observer that a specified claim is true. It seems that this system could be modified to deal with *PI* in more general situations.

The *EO* could be used to recognise plans and isolate beliefs that could lead to incorrect plan formation. The main goal of the plan would correspond to the main claim of the argument. Subgoals of and conditions on a goal would correspond to evidence for that goal.

Example 17

(1) prevent(mmfile read tom)	I want to prevent Tom from reading my mail file.
(2) set-permissions(mmfile read faculty)	How can I set the permissions on it to faculty read only.

Suppose the system has the following rules:

prevent(F P U) ←

set-permissions(F P G)
lnot(member(U G))
lnot(system-mgr(U))

set-permissions(F P G) ←

valid-permission(P G)
type("SET PROTECTION (G : P) F")

Using its own beliefs as a basis for the stereotype of the actor the system infers that she intends to do (1) by doing (2). Assuming that the major premise is the one intended, the missing beliefs are that Tom not be a member of the faculty, and that he not be the system manager. If one of these beliefs is wrong (in the system's view) then the plan is, in Pollack's terminology, *ill-formed.* The *RU* should generate a reply such as

> The command is "SET PROTECTION (faculty : read) mmfile", but that won't keep Tom out; he's the system manager, so file permissions don't apply to him.

The division of beliefs into *explicit* and *missing* may provide more information as to the source of the error. If the rule itself is in *missing*, then the source of the confusion may be an incorrect understanding of the conditions on the plan.

If all missing beliefs are believed true by the system the plan may still be incorrect. It may be the case that one step is *unexecutable*. In the example above it could be that *valid-permission(faculty read)* is false. In this case the *RU* should respond that the plan as stated cannot be carried out, and explain why.

If there are no rules to link the actor's utterances together then the plan is *incoherent*. If the speaker says, for example,

I need to speak to Cathy. How do I stand on my head?

then it is unlikely that the system will see any relation between the two utterances. The appropriate response in this situation is to indicate confusion.

The way the oracle deals with utterances allows the user to enter questions in a wide variety of ways with the same basic rules being used. For example, the user above might have asked "Is Tom a faculty member?" From this query and the same rules given above the system could deduce that the user intends to set permissions to faculty-read only[1]. This oblique reference to a plan may not be recognised by Pollack's system because it does not indicate an action to be performed, but rather a condition that must hold before the implied action will work.

The *PI* system would likely have a stereotype of a naive user, one which would represent a number of common errors. The system might record, for example, a version of the "prevent" plan given above in which the user does not realize that the system manager is exempt from file permissions.

Carberry has identified ([Carberry87]) four assumptions that Plan Recognition Systems make to avoid errors in modelling the user. One of these is the assumption that the user does not have incorrect beliefs about the domain. This assumption is made neither by Pollack's system above nor the *EO*. Another assumption is that the user's queries always address aspects of the task within the system's knowledge of the domain. The *EO* allows for some addition of relations in argument understanding. How useful this primitive ability will be to plan recognition remains to be seen.

[1]Since an "Is X true?" question can be answered either "yes" or "no", it should be tested for evidence both as X and $\neg X$. The user may have a plan for either answer, or for both.

The third assumption Carberry cites is that the user's statements are correct and not misleading. The *EO* does not deal with the whether what the user says is correct, only with how the statements interrelate. If the speaker makes a misleading statement then the oracle will likely be misled. It tries at all times, though, to indicate where contradictions arise so that a *RU* can isolate problems with the speaker's logic.

The fourth assumption is that the inference mechanisms used by the system do not introduce errors into the user model. Again, we can never be sure that no errors have been introduced, but the oracle will do its best to ensure that the inferences it makes are consistent, and it leaves data around to ease the problem of repairing the representation.

The oracle makes the distinction between *explicit* and *missing*, corresponding to the difference between *intended* and *keyhole* recognition. The oracle gives more credence to those beliefs it believes the speaker *intends* it to recognise than to those it merely deduced from the form of the argument (seen through the keyhole). If disparity occurs between the speaker's plan and the plan the system attributes to the speaker then beliefs in *missing* are more likely to be doubted than those in *explicit*.

The oracle has many of the characteristics Pollack and Carberry think should be present in a plan recognition system. While the oracle has not been applied to this problem it would seem to be an appropriate task to set it to.

6.3 Evidence

There are many differing definitions of evidence. A large part of these are probabilistic in nature. These definitions are useful when one is given the evidence and must decide what conclusion is best drawn from it. In our system the conclusion has already been drawn—by the speaker.

Since the conclusion of the argument has been decided for us, and we are only concerned with finding out what evidence has been mustered for that conclusion, we do not need to consider how good the evidence is. Consideration of how believable the argument is is beyond our purpose. We require only that the speaker have some logic to his argument—that the points brought up are pertinent and are presented in a coherent manner.

By setting out frames of evidence, the oracle also allows for recognition of faulty evidence. [Cohen83] provides incorrect as well as correct frames of evidence (see table 1.1). There is no valid

chain of logic from premise to conclusion in these frames: merely the mistaken belief on the part of the speaker that the conclusion is justified. This is a different sort of evidence than that which numerical systems study.

When trying to judge the plausibility of a generated rule (see section 3.6), we do use a sort of probabilistic evidence. The oracle simply counts examples and counterexamples to a proposed rule, and puts the numbers together to decide whether the rule is reasonable. The original version of the oracle used tests based on the definition of evidence given by Nathan in [Nathan80], but these tests led to results that I found counter-intuitive. Therefore I developed tests that fit my intuition better. The tests carried out are rudimentary and this is the part of the oracle that could use the most work.

Chapter 7

Summary

The purpose of this essay was to describe the implemention of a module for deciding a question of evidence in the context of an argument understanding system. In particular, the module was to answer the question "Does the speaker intend P to be evidence for Q in this argument?" The module was to have access to a knowledge base and a speaker model to aid it in its computations.

To decide the question of evidence the oracle uses frames of evidence, prototypes for evidence relations. Each evidence relation intended by the speaker should fit one of these frames. If more than one possible frame is found then considerations of speaker and listener beliefs are used to decide which one is intended.

The frame that the oracle decides on will have unfilled slots which will be filled in and noted by the oracle in an attempt to better understand the speaker. This "keyhole" recognition seems to be a method used by humans to expand their knowledge of the speaker ([Carberry87]). The beliefs added in this manner are slightly suspect, however, and are the first to be questioned when it becomes obvious that some mistake has been made in attributing beliefs to the speaker. The system implemented keeps explicitly stated beliefs separate from those it deduces "through the keyhole".

When making keyhole observations the oracle judges the plausibility of the observations made. Observations that it feeels are implausible are rejected. If more than one interpretation is possible the oracle compares the options and selects the one that it judges to be most plausible. The beliefs that it considers plausible are (in order of decreasing plausibility):

1. Beliefs the speaker has attributed to himself;
2. Beliefs the system has attributed to the speaker;
3. Beliefs the system expects the speaker to have;
4. Beliefs the system has[1];
5. Beliefs that do not explicitly contradict those the system believes the speaker to hold.

The argument understanding problem is an important subproblem of natural language understanding. The insights gained by modelling this problem will aid in the solution to the greater problem. It will help determine how people go about understanding discourse, and how computers can best do so.

The oracle might also be useful in recognising plans (section 6.2), particularly those that are only hinted at by the planner. The problem is similar in that some coherent structure applies and not all components are explicitly mentioned.

[1]This item is identical to the previous in the current implementation since we expect the speaker's beliefs to be the same as the system's.

Bibliography

[Barnett81] J Barnett, "Computational Methods for a Mathematical Theory of Evidence," *Proceedings of IJCAI-81,* 1981.

[Carberry87] Sandra Carberry, "Plan Recognition in User Modeling," to appear in *Computational Linguistics,* 1987.

[Cohen83] Robin Cohen, "A Computational Model for the Analysis of Arguments," University of Toronto Technical Report CSRG-151, October, 1983.

[Fagin&Halpern85] Ronald Fagin and Joseph Y Halpern, "Belief, Awareness, and Limited Reasoning: Preliminary Report," *Proceedings of IJCAI-85,* 1985, pp 491–501.

[Levesque84] Hector J Levesque, "A Logic of Implicit and Explicit Belief," Fairchild Technical Report No. 653, and FLAIR Technical Report No. 32, August, 1984.

[Nathan80] N M L Nathan, *Evidence and Assurance,* Cambridge University Press, 1980.

[Pollack86] Martha Pollack, "A Model of Plan Inference that Distinguishes between the Beliefs of Actors and Observers," *Proceedings of ACL-86,* New York, NY., 1986.

[Poole&al86] David Poole, Randy Goebel and Romas Aleliunas, "Theorist: A logical reasoning system for defaults and diagnosis," University of Waterloo Research Report CS-86-06, February, 1986.

[Reiter80] R Reiter, "A logic for default reasoning," *Artificial Intelligence 13 (1&2),* 1980, pp. 81–132.

[Sadock77] J Sadock, "Modus Brevis: The Truncated Argument", in *Papers from the 13th Regional Meeting, Chicago Linguistic Society,* 1977.

[Smedley86] Trevor J Smedley, "An Implementation of a Computational Model for the Analysis of Arguments," University of Waterloo Research Report CS-86-26, July, 1986.

[Smedley87] Trevor J Smedley, "Integrating Connective Clue Processing into the Argument Analysis Algorithm Implementation," University of Waterloo Research Report CS-87-34, 1987.