Determining intended evidence relations in natural language arguments

MARK A. YOUNG[1]

Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109, U.S.A.

AND

ROBIN COHEN

Department of Computer Science, University of Waterloo, Waterloo, Ont., Canada N2L 3G1

Received August 15, 1989

Revision accepted March 5, 1991

When trying to understand a speaker's argument, it is necessary to determine what her claim is and what evidence she provides for it. It is necessary, therefore, to be able to recognize evidence relations in terms of the speaker's beliefs. This paper describes an implementation of an *evidence oracle*, which tests for evidence between statements and builds a model of the speaker based on the evidence relations found. This implementation is intended to be an advance in the development of practical discourse analysis systems, proposing a basis for verifying certain relationships between utterances. Another contribution of the work is a stratified speaker model which allows for varying levels of acceptance of beliefs attributed to the speaker. Integration of the implemented evidence oracle into a full discourse analyser is presented, together with output illustrating the analysis for several sample arguments. Some extensions of this approach for plan inference are also discussed.

Key words: evidence, user modeling, beliefs, natural language, inference.

Lorsque l'on essaie de comprendre l'argument d'un locuteur, il importe de déterminer la nature de sa prétention et le type d'évidence qui l'accompagne. Par conséquent, il est nécessaire de pouvoir distinguer des relations d'évidence les croyances du locuteur. Cet article décrit la mise en oeuvre d'un oracle qui recherche l'évidence entre des énoncés et construit un modèle du locuteur en fonction des relations d'évidence constatées. Cette mise en oeuvre propose une base pour vérifier certaines relations entre des énoncés; elle se veut une contribution au développement d'un système pratique d'analyse du discours. Une autre contribution de cette recherche est l'élaboration d'un modèle de locuteur stratifié qui tient compte de niveaux variables d'acceptation des croyances attribuées au locuteur. L'intégration de l'oracle d'évidence sous forme d'analyseur de discours est présentée, ainsi que des illustrations de l'analyse de plusieurs arguments types. Une extension de cette approche à l'inférence de plans est également discutée.

Mots clés : évidence, modèle de l'utilisateur, croyances, langage naturel, inférence.

[Traduit par la rédaction]

Comput. Intell. 7, 110–118 (1991)

1. Introduction

This paper addresses the problem of determining relations between utterances in a discourse. The type of discourse studied is the monologue argument, where the speaker has as goal to convince the listener of some point of view. The relations between utterances which must be determined in analysis are therefore intended evidence relations.

For the analysis, it is assumed that the argument will be input as a sequence of statements or propositions. Each statement made plays the role of evidence or claim (possibly both). Before judging the validity of the argument, one must determine its structure — which statements are evidence for which claims. Since large parts of arguments often go unstated, a knowledge of the speaker's beliefs is essential.

This paper describes one component of an argument understanding system. The evidence oracle (EO) takes two propositions, Q and P, from the speaker's argument, and answers the question "Does the speaker intend P to be evidence for Q?" In coming to its answer the oracle considers its own model of the world and a model of the speaker. Based on the answer it derives, it may build on the speaker model. The evidence oracle is more fully described in Young (1987).

[1]This paper describes work completed while the author was at the University of Waterloo.

Printed in Canada / Imprimé au Canada

1.1. The argument understanding system

The EO is based on the subsystem of the same name in Cohen's *argument understanding system* (AUS). The AUS (described in Cohen (1983, 1987*a*) and partially implemented in Smedley (1986, 1987)) parses a natural language argument (monologue) into a tree representation. The root of the tree is the main claim of the speaker, and the children of each node are the statements made in evidence for 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. Each transmission strategy represents a way of ordering the statements in the argument. The transmission strategies that the AUS accepts are (*i*) claim first — each claim followed by the evidence for it; (*ii*) claim last — each claim preceded by the evidence for it; and (*iii*) hybrid — each sub-argument is either claim first or claim last. (For more details on why these orderings constitute coherent transmissions, see Cohen (1983, 1984*a*).)

Figure 1 shows the flow of control and data in the AUS. The propositions of the argument are passed one at a time to the *proposition analyser* (PA), which, based on the restrictions of the transmission strategy, selects those nodes in the argument representation that may be related to the current proposition. For each of the selected nodes, the question of evidence is asked of the EO. When an evidence relation

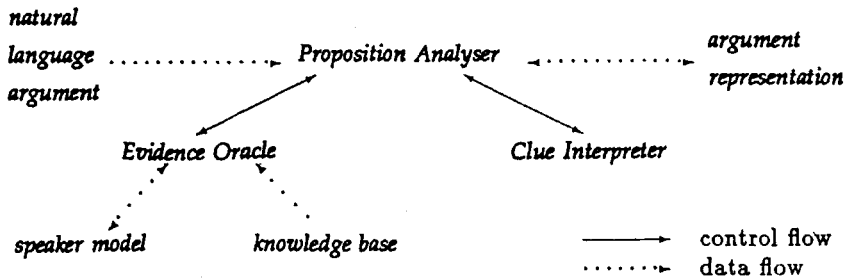


FIG. 1. System design.

is found, the current node is inserted at the appropriate position and the AUS goes on to the next proposition.

The *clue interpreter* finds and analyses clue words in the argument. These are special words and phrases used by the speaker to indicate the structure of the argument (for example, connectives). These clues may restrict the nodes considered in the search of the proposition analyser, or the types of evidence relations considered for each node pair (passing the clue information to the evidence oracle through the PA). (For more discussion on the role of clues in arguments, see Cohen (1984*b*).)

If the argument is coherent and the analysis is correct, then the representation returned will be a tree. It is expected that this tree will be passed on to some sort of *response unit* (RU), which will generate a reply to the argument. It is the RU that will deal with the believability of the overall argument; the EO, in contrast, is concerned with the believability of the individual evidence relations in the argument — the basis for deciding the yes/no answer for the PA.

2. The evidence oracle

2.1. Frames of evidence

To recognize evidence relations the EO uses frames with slots for a conclusion and one or more premises. We say that E is *evidence* to C (*claim*) if E and C fit some frame of evidence with E as a premise and C as the conclusion. The frames used appear in Tables 1 and 2. *Modus ponens* and *modus tollens* represent the rules of logic. The *generalization* is a variation on modus ponens in which the correspondence is not perfect; that is, it need not be true for all values of its free variables. It may be viewed either as a default rule (similar to Reiter (1980) or Poole *et al.* (1987)) or as a relation which is merely probable. The generalized rule corresponding to $A \rightarrow B$ is represented by $A \rightarrow\!\!\!\!\!\rightarrow B$.

Refutation and concession appear in counterarguments and serve to deny a rule proposed by another speaker. The "rule" is contradicted by giving a counterexample — the refutation. Before showing the counterexample, though, it is usual to lead the way with concessions — examples of the "rule" at work. These concessions are a form of contrastive evidence, as discussed in Cohen (1983).

Table 2 shows frames of *partial* evidence. These forms of evidence allow for "giving examples." There is an implied rule at work here:

> If the examples for a generalization outweigh the examples against, then that generalization is (probably) true.

The above rule may not be particularly convincing (to say nothing of the generalization it is being used to support), but the usage is common.

TABLE 1. Frames of evidence

	Minor premise	Major premise	Conclusion
Modus ponens	A	$A \rightarrow B$	B
Modus tollens	$\neg B$	$A \rightarrow B$	$\neg A$
Generalization	A	$A \rightarrow\!\!\!\!\!\rightarrow B$	B
Refutation	A	$\neg B$	$\neg(A \rightarrow B)$
Concession	A	B	$\neg(A \rightarrow B)$

The oracle should also be prepared to recognize incorrect rules of logic, such as "asserting the consequence," if there is evidence that the speaker is using them. For now we will restrict our attention to the above frames.

2.2. Missing premises

The speaker often does not fill in all slots of the applicable frame. This phenomenon is called *modus brevis* in Sadock (1977). Thus, while the speaker could say

> Socrates is a man. All men are mortal. Therefore, Socrates is mortal.

she is more likely so say either

> All men are mortal. Therefore, Socrates is mortal.

or (even worse)

> Socrates is a man, and therefore mortal.

The listener is expected to fill in the empty slots from his own knowledge.

The sort of information that is usually not conveyed is that which is common knowledge: if everyone knows that Socrates is a man, then there is no point in telling anyone so. Sometimes something the speaker considers "obvious" will not be considered so by the listener. The oracle must be prepared to deal with beliefs used by the speaker that it does not share. A further complication is that the oracle deals with predicates by twos. Thus it is possible (indeed, quite likely) that the missing premise will be stated later in the argument.

It is important that the missing premises could plausibly be held by the speaker. That the missing premises contradict shared beliefs or earlier statements of the speaker could indicate that the relation found is not the one intended.

Cohen (1983) suggests that we determine whether a belief is plausible by consulting the following sets of knowledge: (*i*) shared beliefs; (*ii*) the hearer's beliefs; (*iii*) a stereotype of the speaker; and (*iv*) a model of a hypothetical person — a "least detailed" speaker model. To facilitate the implementation, we shall take a slightly different approach. The

TABLE 2. Frames of partial evidence

	Minor premises		Conclusion
Positive example	A	B	$A - B$ or $A \rightarrow B$
Contrapositive example	$\neg B$	$\neg A$	$A - B$ or $A \rightarrow B$
Counterexample	A	$\neg B$	$A \rightarrow B$

TABLE 3. An instantiation of the modus ponens frame

(1-1) *All men are mortal.*	Major premise	*mortal*(X) – *man*(X)
(1-2) *Socrates is a man.*	Minor premise	*man*(*socrates*)
(1-3) *Therefore, Socrates is mortal.*	Conclusion	*mortal*(*socrates*)

oracle will consult: (*i*) shared beliefs; (*ii*) a model of the speaker, including a stereotype; and (*iii*) the system's beliefs/knowledge.

We assume the speaker to be a fairly competent reasoner. Therefore, if she says that P is evidence for Q by virtue of the relation $P \wedge R - Q$, then she believes not only that P, Q, and $P \wedge R - Q$ are true, but also that R is true. On this basis, any premises missing from an accepted evidence relation may be added to the speaker model.

2.3. Model of the world

The oracle maintains a model of the world to help it judge the plausibility of missing premises. Part of that world model is a model of the speaker. The world model is broken into several modules, depending on who holds the beliefs represented:

facts the beliefs common to the conversants[2].
speaker a stratified model of the speaker's beliefs, broken down into:
- **explicit** those statements made by the speaker, and thus attributed to herself.
- **missing** beliefs we have attributed to the speaker on the basis of evidence relations determined earlier in the argument.
- **stereotype** default beliefs for the speaker. The system's beliefs will initially serve as a basis for the speaker stereotype.

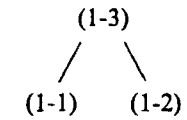
hearer the private beliefs (or knowledge) of the system.

The ordering of modules given above reflects the order of search for missing beliefs. Roughly speaking, the further down the list one must search for a missing belief, the less plausible it is that the speaker has that belief. Predicates that do not appear in the list, however, may also be judged plausible. In particular, if a predicate is not explicitly contradicted by one of *facts*, *explicit*, or *missing*, then it can be considered plausible.

2.4. Determining evidence relations

To show how frames are used to determine evidence relations, consider the instantiation of the *modus ponens* frame given in Table 3. The speaker gives the speech on the left (the statements are numbered for later reference), which is translated into the predicates on the right. (Our implementation of the evidence oracle, in Waterloo Unix Prolog (WUP), assumes that the input has already been preprocessed into the predicate notation[3].) The oracle can then fit the predicates into the slots for modus ponens and recognize the relation intended.

Assuming a claim-last transmission, and that the statements are made in the order given above, the PA first asks the EO whether (1-1) is evidence for (1-2). The oracle answers no, since there is no frame of evidence appropriate for the proposed relation. The PA defers handling (1-1) for the present. The next question asked is whether (1-2) is evidence for (1-3). The oracle discovers that (1-2) and (1-3) fit the modus ponens frame with (1-1) as the missing premise. Since (1-1) appears in *explicit* (it was put there on the first call to the oracle), it is deemed a plausible belief and the relation is accepted. The PA then completes processing the argument by asking whether (1-1) is evidence for (1-3). This succeeds in the same manner as (1-2) for (1-3) did, with the same result. The argument has been successfully parsed into the tree:



To illustrate adding missing premises to the speaker model, consider the following. Assume the speaker and the system have a common belief that all Greek men are mortal (represented by *mortal*(X) – *greek*$(X) \wedge$ *man*(X)). The speaker says "Socrates is Greek, and therefore mortal." The oracle can recognize *greek*(*socrates*) as evidence for *mortal*(*socrates*) with two missing pieces: the rule about the mortality of Greek men, and a belief that Socrates is a man. As long as the speaker's beliefs do not indicate that Socrates is not a man, we can accept that belief as plausible. The predicate *man*(*socrates*) would then be added to module *missing*.

2.5. Multiple evidence relations

Sometimes more than one frame of evidence may be appropriate for a given pair of predicates. Consider the following argument:

(2-1) Socrates is Greek,	*greek*(*socrates*)
(2-2) and, so, mortal.	*mortal*(*socrates*)

[2]This module could have been called "shared," since the beliefs here are shared between the oracle and the speaker (this may be seen as one-sided mutual belief, as in Clark and Marshall (1981)). But since these beliefs are meant to be shared with *any* speaker, the name "facts" was considered more descriptive.

[3]The implementation was tested on a number of examples, including several much longer than the ones presented in this paper for illustration.

TABLE 4. Relative plausibility

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

with shared beliefs

$mortal(X) - greek(X) \wedge man(X)$
$mortal(X) - greek(X) \wedge woman\ (X)$

There are two frames of modus ponens that allow (2-1) as evidence for (2-2). One has Socrates as a man, the other as a woman[4].

If one of these beliefs is plausible and the other not, then there would be no problem to choose between them. When both are plausible, however, we must make a decision. The simplest solution is to take whichever is more convenient — the first one generated, for example. Another solution is to refrain from choosing — simply report a relation without updating *missing*. The former solution could lead to many errors, while the latter results in lost information. Keeping disjunctive knowledge in *missing* is another possibility, but would require a more complex management of the *missing* module. We prefer to make some decision on the intended interpretation, in accordance with the general strategy of the AUS to incrementally reconstruct the representation for the input.

2.6. *Belief levels*

Our solution is as follows: when faced with multiple evidence relations the oracle will select the most plausible. A predicate is not plausible if its negation appears in any of *facts*, *explicit*, or *missing*. The beliefs considered plausible (in order of decreasing plausibility) are (*i*) beliefs the speaker has attributed to herself; (*ii*) beliefs the oracle has attributed to the speaker; (*iii*) beliefs typical to the speaker's stereotype; and (*iv*) beliefs not contradicted explicitly by (*i*) or (*ii*).

Table 4 gives numerical values to the various acceptable combinations. Conjunctions receive the value of the least plausible conjunct. (Note that 0 indicates the most plausible, and 11 the least. Only predicates without a value in Table 4 are implausible.)

A predicate is *true* in a module if it appears in that module, *false* if its negation appears there, and *unknown* otherwise. Based on the truth values found, the oracle assigns a numerical value representing plausibility. This internal representation allows comparisons to be made more easily.

Belief levels are presented as a solution to the problem of selecting between competing evidence relations, as described in Sect. 2.5. If the system is aware that Socrates was a man and is using its own beliefs as a stereotype of the speaker, then it will judge it more plausible that the speaker also believes Socrates to be a man than that she believes him to be a woman.

[4]Actually, this is a case where there are two possible major premises for the same frame of evidence. There are examples as well where two different frames may apply, for example, concession or refutation. See Young (1987) for more details.

hearer's beliefs: *man(socrates)*
shared beliefs: $mortal(X) - greek(X) \wedge man(X)$
$mortal(X) - greek(X) \wedge woman(X)$
considering greek(socrates) for mortal(socrates)

greek(socrates) is evidence for mortal(socrates)
with [man(socrates)] missing.
man(socrates) has belief level 9
The conjunction has belief level 9

greek(socrates) is evidence for mortal(socrates)
with [woman(socrates)] missing.
woman(socrates) has belief level 10
The conjunction has belief level 10

success greek(socrates) for mortal(socrates)
missing [man(socrates)]

FIG. 2. Sample session: multiple relations.

TABLE 5. Laughing at geniuses

Belief-believer	Facts	Speaker	Hearer
genius(columbus)	Unknown	Unknown	True
genius(bozo)	False	Unknown	Unknown

Consider the sample session in Fig. 2, which shows the analysis of the argument about Socrates, with the modules for the hearer's beliefs (then attributed to *stereotype*) and shared beliefs (equivalent to *facts*) initialized as indicated. The modules for *explicit* and *missing* are currently empty. When the evidence relation is tested, there are two plausible bases for confirming the relationship. Filling in that Socrates is a man has belief level 9, since it is unknown in *explicit* and *missing*, but true in *stereotype* (which is taken by default from the hearer's beliefs). That Socrates is a woman is simply unknown everywhere, resulting in a value of 10.

2.7. *Sample arguments*

2.7.1. *Concessions and refutations*

Consider the following argument[5]:

> Not everyone who is laughed at is a genius. Sure, they laughed at Columbus; but they also laughed at Bozo the Clown.

This would translate to our notation as:

(3-1) $\neg(genius(X) - laughed\_at(X))$
(3-2) $laughed\_at(columbus)$
(3-3) $laughed\_at(bozo\_the\_clown)$

Suppose the system believes that Columbus is a genius. Suppose further that it is common knowledge that Bozo the Clown is a fool (that is, a non-genius).

In this argument, (3-2) and (3-3) are given as evidence for (3-1)[6]. However, the relationship between (3-2) and (3-1) is not the same as that between (3-3) and (3-1). Columbus appears in the argument for rhetorical purposes. The speaker does not mean to say that Columbus is not a genius; rather, he is a genius who was laughed at. The speaker does, on the other hand, intend to say that Bozo the Clown is not

[5]Paraphrased from an argument by Carl Sagan.

[6]See Sect. 3.2 for sample runs showing how the AUS determines this.

a genius. These two relations are captured by the concession and refutation frames of evidence (Table 1, Sect. 2.1).

Note that the argument is understandable even without the clue words "sure" and "but" (though not so easily as with them). When processing this argument, there is no syntactic difference between (3-2) and (3-3). The oracle must consider both concession and refutation for both (3-2) and (3-3).

If the speaker means there to be a concession relation between (3-2) and (3-1), then he believes that both he and the listener can recognize Columbus as a genius. The oracle doesn't yet know what the speaker believes, but it does consider Columbus a genius. It assigns a belief level of 9 to the speaker believing the same. If, instead, the speaker is using a refutation frame, then Columbus should not be a genius. This is contrary to what the oracle believes. Since we have no knowledge yet of the speaker's beliefs, the oracle assigns a belief level of 11 to the speaker believing Columbus to be a non-genius. The preferred explanation, then, is that Columbus is offered as a concession.

For Bozo the Clown, the same two relations must be considered. The concession frame requires that the speaker think Bozo a genius. But Bozo is a well-known non-genius, as represented in the *facts* modules above. Thus this frame is rejected as implausible. The refutation frame, however, is acceptable, since it contradicts no known beliefs of the speaker. Thus the refutation frame is selected for (3-3), as intended.

Note that the oracle employs a liberal attribution of beliefs to the speaker, even with the distinctions in likelihood of belief introduced by belief levels. A statement is considered to be a plausible belief of the speaker as long as it is not explicitly contradicted in the speaker model. The speaker's beliefs are allowed to contradict those of her stereotype, as well as those of the oracle.

2.7.2. Conflicting beliefs between speaker and system

While the oracle will use its own beliefs when it doesn't know the speaker's (by virtue of attributing hearer's belief to *stereotype*), it will prefer the speaker's beliefs where available. Say we have the following pair of statements:

(4-1) Mark went to the CSCSI conference.
(4-2) He enjoyed it.

Further, the speaker and the listener share beliefs that

(R-1) Studious people enjoy conferences.
(R-2) Party animals enjoy conferences.

If the system knows that the speaker believes Mark to be a party animal, then the oracle will find (4-1) to be evidence for (4-2) by rule (R-2). This will be so even if the system believes Mark to be studious.

In fact, the oracle will recognize the speaker's use of rule (R-2) even if it knows that Mark is not a party animal. By maintaining the speaker's beliefs in a module separate from its own, the system can deal with this conflict of belief with no problem. Once the oracle has recognized the evidence relation, a *response unit* for the hearer may react to indicate rejection of the validity of the rule. Still, the argument is parsed correctly, so an appropriate response may be generated. If the system believes Mark to be studious, this response may be of the form "Yes, he did enjoy the conference, but not for the reasons you think."

2.8. Inconsistent beliefs

The EO must often judge whether it is plausible that the speaker hold a given belief. We have two opposing concerns as we try to reach the decision. We want to recognize many different kinds of argument. At the same time, we do not want to attribute beliefs to the speaker that she does not hold. It would be nice if we could assume the speaker to be ideally rational. Yet, it is not often we meet someone who knows everything that follows from her beliefs. We need to know what we can deduce about the speaker's beliefs based on what she says.

Our solution is based on Fagin and Halpern (1985), which is in turn based on Levesque (1984). Levesque divides the speaker's beliefs into explicit and implicit parts. Explicit beliefs are those that the speaker is aware of. Implicit beliefs are those that merely follow from what is explicitly believed. Fagin and Halpern suggest that reasoning agents would not believe both F and $\neg F$ explicitly. Implicit contradictions, however, are allowed. Thus an agent may explicitly believe that A implies C, that B implies $\neg C$, and that both A and B. Beliefs C and $\neg C$ are merely implicit, and so do not cause the agent discomfort.

In our system, this notion of explicit contradiction is represented by requiring the actual logical negation of the predicate under consideration to be present in the speaker model. No chains of inference are followed while trying to detect a contradiction.

In our model of the speaker we have three modules. One of these, *stereotype*, consists of statements we think it likely the speaker believes, but which have not yet been attributed to her. Since these may not be beliefs of the speaker, we do not look for contradictions here. The other modules, *explicit* and *missing*, consist of beliefs that have been attributed to the speaker by herself and by the system, respectively. In the case of the former, we can be sure that the speaker holds them explicitly; she has stated them openly. These beliefs we will not allow to be contradicted. In the case of the latter, it is less obvious. We make a design decision to not allow these beliefs to be contradicted, either. (Note that we are not saying that these "missing" beliefs are also explicitly held, only that they are so obvious that any contradiction should have been pointed out already. If an undetected contradiction does underlie an intended relation, then our analysis of the argument should fail completely, and the speaker's error be discovered in a dialog with the response unit.)

Our system also maintains a set of beliefs, *facts*, which it holds to be shared with the speaker. We will not allow beliefs in this module to be contradicted.

2.9. Chaining beliefs

While contradictions in the speaker model must always be explicit, other operations do look for implicit beliefs. In particular, when searching for missing premises in a putative evidence relation, belief chains are followed. So, if we are trying to find a missing major premise to link P to Q, it need not appear as $P \rightarrow Q$. The rules $P \rightarrow R$ and $R \rightarrow Q$ would do nicely. Beliefs are also chained when looking for examples and counterexamples to possible generalizations (see below).

When belief chains are being followed, it is important not to end up going in circles. The implemented oracle makes sure that it does not go into infinite loops by including a simple ancestor-check in its search. Before we expand G in

any search, we check to see that G has not already been expanded (all searching is depth first). This simple test prevents most forms of infinite regress, but at the cost of completeness. Thus, the oracle may fail to find some solutions.

2.10. Missing major premise

When one of the missing premises of an argument is the major premise, the judgement of plausibility is carried out somewhat differently from that described above. Clearly, any two premises can fit the *modus ponens* frame with an appropriate missing major premise. For this reason, no such major premise is generated unless it appears somewhere in the system's knowledge of the world.

On the other hand, we must allow for the chance that the intended rule has not been previously encountered by the system, and yet is still valid. Therefore, if no other frame of evidence is found to fit the given predicates, the oracle will generate a *generalization* major premise and test that premise for plausibility. This is the case refered to as "partial evidence" in Sect. 2.1.

The following two tests then measure the plausibility of $P \rightarrow Q$:

1. 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.
2. There are more instances of $\neg P \wedge \neg Q$ than of $P \wedge \neg Q$. When Q is false, P is more likely to be false than true.

Both must be true for the generalization to be established.

For example, if the oracle is asked whether *shark(joey)* might be meant as evidence for *dangerous(joey)*, and it finds no other frames of evidence appropriate, it will generate the rule *shark*$(X) \rightarrow$ *dangerous*(X), and test it for plausibility. Assume the system has the following beliefs[7]:

shark(joey),	dangerous(joey),
shark(fred),	dangerous(fred),
whale(louis),	dangerous(louis),
whale(carl),	lnot(dangerous(carl)),

lnot(shark(X)) $\leftarrow$ whale(X)

When *shark*(X) is true, we have two examples of *dangerous*(X) true, and none for *dangerous*(X) false. When *dangerous*(X) is false, we have one example of *shark*(X) false, and none of *shark*(X) true. Thus our rule is deemed plausible, it is added to module *missing*, and the oracle reports an evidence relation found.

3. Related work

3.1. Plan inference

The general problem of plan inference has been addressed recently by several researchers, including Kautz (1987), Pollack (1986), and Carberry (1988). Pollack states that it would be useful to have a model of plan inference that distinguishes the beliefs of the planner from those of the observer. The reason for this distinction is to allow the observer to recognize incorrect plans and generate appropriate responses. The EO makes such a distinction in the context of argument understanding, and can be applied to plan inference as well.

[7]The term "lnot" is used to stand for logical negation.

By identifying claim with plan, and evidence with subplan, the oracle can detect when one stated goal of the user is a subgoal of another. With the distinction between *explicit* and *missing* in the speaker model, the oracle can help isolate the cause of the user's error, especially when more than one explanation of the error exists. The oracle can also detect plans from questions regarding the conditions on a plan (Young 1987, p. 50).

To elaborate on the potential use of the stratified speaker model of the *evidence oracle* for plan inference, consider the following example, from Pollack (1986):

> I want to prevent Tom from reading my mail file. How can I set the permissions on it to faculty read only?

with the translation into predicates as follows:

(5-1) *prevent*(*mmfile, read, tom*)
(5-2) *set_permission*(*mmfile, read, faculty*)

and a starting set of rules in the system of the following form:

prevent(F, P, U) $\leftarrow$ *set_permissions*(F, P, G)
lnot(*member*(U, G))
lnot(*system_mgr*(U))
set_permissions(F, P, G) $\leftarrow$ *valid_permission*(P, G)
type("*set protection* (G : P) F")

Using its own beliefs as a basis for the stereotype of the user, the system infers that she intends to achieve (5-1) by doing (5-2). Missing beliefs that Tom is not a member of the faculty and that he is not the system manager may be filled in. If one of these beliefs is wrong (in the system's view), then the plan is, in Pollack's terminology, ill-formed. The *response unit* could provide a reply by using the general rules above. Now, the division of beliefs into *explicit*, *missing*, and *stereotype* may provide some more information. 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, however, all missing beliefs are also believed true by the system, then the plan may still be incorrect — for example, *valid_permission*(*read, faculty*) could be false, and the plan thus unexecutable. The response unit could provide a different kind of response, in this case.

Carberry (1988) considers assumptions often made by plan recognition systems that she thinks ought not to be made. The oracle takes these issues into account to greater and lesser extents, as described below.

- *The user does not have incorrect beliefs about the domain.* By dividing the domain beliefs into those shared with the speaker and those not, the oracle allows for incorrect beliefs of the speaker. The oracle will attribute incorrect beliefs only when they are required to make the argument coherent.
- *The user's statements are correct and not misleading.* Again, the speaker can make statements that the oracle disbelieves because the two sets of beliefs are kept separate. Misleading statements, on the other hand, are those that cause the listener to infer incorrect beliefs. The oracle cannot guard against these, but will be able to recognize them when the erroneous beliefs contradict something that the oracle already knows.
- *The user's queries always address aspects of the task within the system's knowledge of the domain.* The oracle

will try to recognize novel methods, and to follow new premises laid out by the speaker. It does, however, appeal to its own knowledge to judge plausibility. In this respect, it works best in limited domains where the relevent rules (and common misconceptions) are likely to be already stored.

• *No errors are introduced into the user model.* The oracle cannot prevent incorrect inferences from being made, any more than people can. However, it does try to keep the speaker consistent, and keeps what the speaker said separate from what it infers she believes. It also saves its work, which should make it easier to go back and fix the model when an error is detected.

A summary of the comparison between plus recognition and the oracle's processing is as follows.

Plan recognition requires the storing of all plan configurations ahead of time, and recognizing one of these configurations in the input. If a "match" cannot be produced, this can be as a result of misconceptions on the part of the speaker as to what can legitimately be executed in the domain. We would want to at least recognize the expected faulty plan, and the operations of the oracle can accomplish this by accepting relations believed to be held by the speaker, even if these are not part of the hearer's beliefs.

Another possibility, if a match cannot be found to an existing plan library, is that the speaker's proposed plan is correct, but merely novel, that is, the system is not yet aware of the possibility. This may occur, for example, when parts of the real world knowledge have simply not been configured as a plan in the library, that is, the relations between actions have not been fully realized[8].

The procedures of the oracle allow recognition of novel plans. If a relation between two propositions is deemed plausible, by virtue of lack of contradictory evidence in the system's beliefs, then this novel plan can be accepted[9]. The clean separation of speaker model and system beliefs facilitates this recognition.

One feature of the oracle's processing which distinguishes it from plan recognition is the absence of a stored plan library. Relations between propositions (or actions) are individually listed, to be recognized. This is not dissimilar to the proposal of Goodman and Litman (1989) and Litman and Goodman (1989) for constructive plan recognition, to accept novel plans in the domain of CAD design.

3.2. Argument understanding

This work on the evidence oracle can be merged with existing implementations of the AUS (Smedley 1986, 1987), providing a prototype working version of the overall model of Cohen (1983).

In Smedley (1986), the basic proposition analyser is implemented. The evidence oracle is replaced by an "ask-the-user" facility. Smedley (1987) augments this basic analysis algorithm to include the acceptance of connective clues, to help restrict the processing and detect incoherent arguments (according to the algorithm of Cohen (1987)).

In Song (1988), a fully integrated implementation of the AUS is constructed, using the proposition analyser and clue interpreter of Smedley (1987), and a slightly modified version of the code in Young (1987) for the evidence oracle.

There are two options for someone interested in examining the analysis of a sample argument. "Parse" (option one) adopts the procedures of Young (1987) and assumes claim-first input only. It is convenient to use "parse" to examine raw interpretations of evidence relations in simple arguments, to facilitate understanding of the *evidence oracle* operations. "Analyse" (option two) allows other transmission orderings and reacts to the presence of clues according to the algorithms implemented in Smedley (1987).

The input in Smedley (1987) is unsophisticated — a list of words in each sentence, with cue phrases clearly labelled — because no semantic processing was required. (The user would supply the answers regarding intended evidence relations.) The input for Young (1987) is a predicate encoding of each individual sentence, to facilitate processing in Prolog and chaining on inferences when necessary. This predicate form is therefore also the required input form in Song (1988).

For the analysis, the user must supply

1. the argument to be analysed, with each statement coded in predicate form; and
2. beliefs initially held by the hearer (the user), and those believed shared by hearer and speaker ("facts"). The speaker model may also be initialized if there are known beliefs recorded from an earlier session or by a stereotype.

Session one.

To show the integrated version at work, we return to the example from Sect. 2.7.1.

(3-1) $\neg(genius(X) \leftarrow laughed\_at(X))$
Not everyone who is laughed at is a genius.
(3-2) *laughed_at(columbus)*
Sure, they laughed at Columbus,
(3-3) *laughed_at(bozo_the_clown)*
but they also laughed at Bozo the Clown.

```
Waterloo Unix Prolog [Version 3.1a — Beta Release
March 16, 1988]
?setup;
? add_beliefs(facts, [lnot(genius(bozo_the_clown))]);
? add_beliefs(hearer, [genius(columbus)]);
? 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)) is not believed by the model
laughed_at(columbus) is evidence for
  lnot(for_all(genius(columbus),
```

[8]Creating the initial plan library often requires freezing the description at one specific level of detail; if a plan is input at a more specific level of detail, the matching algorithm may fail. Kautz (1987) and others have investigated the organization of plan libraries along specialization links, which begins to address the difficult problem of allowing varying levels of detail in the input.

[9]The current oracle is quite generous in its acceptance of new relations; this can be adjusted in the implementation with stricter rules on what is considered "acceptable." Table 4 of Sect. 2.6 shows all acceptable relations, when filling a missing proposition. To fill a missing major premise, the rules of Sect. 2.10 apply.

```
    [laughed_at(columbus)])) with
      [genius(columbus)] missing.
  genius(columbus) has belief level 9
The conjunction has belief level Max:9, Sum: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 Max:0, Sum: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)]))

The argument tree is:

lnot(for_all(genius(X), [laughed_at(X)]))
    laughed_at(columbus)
    laughed_at(bozo_the_clown)

? print_beliefs(missing);

module: missing

    lnot(genius(bozo_the_clown))
```

Session two.

Similar to the example above, but this time, the argument is given in claim-last order and is described by the integrated program using Analyse.

```
Waterloo Unix Prolog [Version 3.1a — Beta Release
March 16, 1988]
?setup;

? add_beliefs(facts, [lnot(genius(bozo_the_clown))]);

? add_beliefs(hearer, [genius(columbus)]);

? analyse([laughed_at(columbus),
           laughed_at(bozo_the_clown),
           lnot(for_all(genius(X), [laughed_at(X)]))
           ]);

The argument is:
laughed_at(columbus)
laughed_at(bozo_the_clown)
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(columbus)
  for laughed_at(bozo_the_clown)
failure laughed_at(columbus)
  for laughed_at(bozo_the_clown)

considering lnot(for_all(genius(X), [laughed_at(X)]))
  for laughed_at(bozo_the_clown)
failure lnot(for_all(genius(X), [laughed_at(X)]))
  for laughed_at(bozo_the_clown)

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 Max:0, Sum: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)]))

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)) is not believed by the model

laughed_at(columbus) is evidence for
  lnot(for_all(genius(columbus),
    [laughed_at(columbus)])) with
    [genius(columbus)] missing.
  genius(columbus) has belief level 9
The conjunction has belief level Max:9, Sum:9
success laughed_at(columbus) for
    lnot(for_all(genius(X), [laughed_at(X)]))

Argument tree:

lnot(for_all(genius(X),[(laughed_at(X)]))
    laughed_at(columbus)
    laughed_at(bozo_the_clown)

? print_beliefs(missing);

module: missing

    lnot(genius(bozo_the_clown))
```

The predicate "analyse" can deal with not only claim-first order but also claim-last order and hybrid order or both. The resulting argument tree is the same as above, but the intermediate outputs are different from the previous example.

In the example sessions above, the argument tree shows the final structure of the given argument. In each consideration, "success" confirms the claim–evidence relation between two propositions with the case having smallest Max or Sum as the result. The predicate *print-beliefs*(*M*) is used to show how modules have been updated. Note that, in this example, some missing evidence is said "not believed by the model." This means that it contradicts some existing beliefs in one of the four modules. Of course, this case is eliminated from consideration.

The integrated system handles clues, and different transmission strategies. Incorporating clue interpretation saves calls to the oracle. The acceptance of various orderings of propositions and the analysis of clue words in the input are discussed in greater detail in Cohen (1987*a*, *b*); since these topics are not the focus of this paper, we will not include further discussion here.

4. Conclusion

In this paper we have described an implementation of a module for deciding a question of evidence in the context of argument understanding. In particular, the module answers the question "Does the speaker intend her statement *P* to be evidence for her statement *Q*?"

To decide the question, the oracle uses frames — prototypes of evidence relations. Each relation found must match one of these frames. If more than one relation is found, then considerations of user and system beliefs, with a ranking provided by "belief levels," provide an indication of the most likely intention of the speaker.

The implemented oracle thus makes possible a full implementation of an argument understanding system based on the model of Cohen (1983), critical to the advancement of practical processing models of discourse.

As it answers the questions posed to it, the oracle takes note of what beliefs are required to support the relations found, and ensures that these are plausible. If the relation is accepted, this "keyhole recognition" is used to expand the speaker model. The system does not, however, give as much weight to these inferred beliefs as to explicit statements by the speaker.

The oracle might also be useful in recognizing plans, particularly those that are only hinted at by the speaker. The two problems are similar in that some hierarchy applies, and that not all relevant components are mentioned. Therefore, the general proposals for belief recognition and updating of the speaker model have useful extensions for other natural language understanding tasks.

Acknowledgements

Many thanks to Fei Song, Paul Kates, Bruce Spencer, and our anonymous reviewers for their assistance with this paper. This research was partially supported by the Natural Sciences and Engineering Research Council of Canada.

CARBERRY, S. 1988. Modeling the user's plans and goals. Computational Linguistics, **14**(3): 23-37.

CLARK, H.H., and MARSHALL, C.R. 1981. Definite reference and mutual knowledge. *In* Elements of discourse understanding. *Edited by* A. Joshi, B. Webber, and I. Sag. Cambridge University Press, New York, NY.

COHEN, R. 1983. A computational model for the analysis of arguments. Technical Report CSRG-151, University of Toronto, Toronto, Ont.

_____ 1984*a*. A theory of discourse coherence for argument understanding. Proceedings of the Canadian Society for Computational Studies of Intelligence Conference, Saskatoon, Sask., pp. 6-10.

_____ 1984*b*. A computational theory of the function of clue words in argument understanding. Proceedings of the International Conference on Computational Linguistics, Stanford, CA, pp. 251-258.

_____ 1987*a*. Analyzing the structure of argumentative discourse. Computational Linguistics, **13**(1-2): 11-24.

_____ 1987*b*. Interpreting clues in conjunction with processing restrictions. Proceedings of the National Conference on Artificial Intelligence, Seattle, WA, pp. 528-533.

FAGIN, R., and HALPERN, J.Y. 1985. Belief, awareness, and limited reasoning: preliminary report. Proceedings of the 9th International Joint Conference on Artificial Intelligence, Los Angeles, CA, pp. 491-501.

GOODMAN, B.A., and LITMAN, D.J. 1990. Plan recognition for intelligent interfaces. Proceedings of the IEEE Conferences on Artificial Intelligence Applications, Santa Barbara, CA, pp. 297-303.

KAUTZ, H. 1987. A formal theory of plan recognition. Technical Report TR 215, Department of Computer Science, University of Rochester, Rochester, NY.

LEVESQUE, H.J. 1984. A logic of implicit and explicit belief. Fairchild Technical Report No. 653, and FLAIR Technical Report No. 32, Palo Alto, CA.

LITMAN, D.J., and GOODMAN, B.A. 1989. A knowledge-based interface for process design. Proceedings of Third International Conference on Expert Systems and the Leading Edge in Production and Operations Management, Hilton Head Island, SC, pp. 675-688.

POLLACK, M. 1986. A model of plan inference that distinguishes between the beliefs of actors and observers. Proceedings of 24th Annual Meeting of the Association for Computational Linguistics, New York, NY, pp. 207-214.

POOLE, D., GOEBEL, R., and ALELIUNAS, R. 1987. Theorist: a logical reasoning system for defaults and diagnosis. *In* The knowledge frontier: essays in the representation of knowledge. *Edited by* N. Cercone and G. McCalla. Springer-Verlag, New York, NY. pp. 331-352. Also appears as Research Report CS-86-06, University of Waterloo, Waterloo, Ont., February 1986.

REITER, R. 1980. A logic for default reasoning. Artificial Intelligence, **13**(1&2): 81-132.

SADOCK, J. 1977. Modus brevis: the truncated argument. *In* Papers from the 13th Regional Meeting, Chicago Linguistic Society. pp. 545-554.

SMEDLEY, T.J. 1986. An implementation of a computational model for the analysis of arguments. Research Report CS-86-26, University of Waterloo, Waterloo, Ont.

_____ 1987. Integrating connective clue processing into the argument analysis algorithm implementation. Research Report CS-87-34, University of Waterloo, Waterloo, Ont.

SONG, F. 1988. An implementation of the argument understanding system (AUS) — integrating the proposition analyser with the clue interpreter, and the evidence oracle. Research Report CS-88-40, University of Waterloo, Waterloo, Ont.

YOUNG, M.A. 1987. The design and implementation of an evidence oracle for the understanding of arguments. Research Report CS-87-33, University of Waterloo, Waterloo, Ont.