Computational Intelligence, Volume 0, Number c9.tex, 0000

INTERPRETING PRESUPPOSITIONS USING ACTIVE LOGIC: FROM CONTEXTS TO UTTERANCES

John Gurney
Intelligent Systems Branch, Army Research Laboratory, Adelphi, MD 20783, USA
(301) 394-3920 gurney@arl.mil

Don Perlis and Khemdut Purang
Dept. of Computer Science, University of Maryland, College Park, MD 20742, USA
perlis@cs.umd.edu kpurang@cs.umd.edu

Presupposition is a pervasive feature of human language. It involves many interesting interactions between the utterances of a discourse and the *context* of the discourse. In this paper we focus on issues of logical form connected with the interaction of presupposition and discourse context, and illustrate our theory with some implementational work using the *active logic* framework.

After reviewing some of the major issues in presupposition theory we turn to a largely successful unified approach of Heim. We show how the main principles of this theory can be implemented in active logic. But we also find two serious difficulties. These consist in (a) a straightforward counterexample and (b) a type of discourse that we call a garden-path discourse.

We maintain that both the counterexample and the garden-path type of discourse can be handled by our active-logic version of Heim's theory. This requires us to reformulate and extend Heim's

Although this work is largely theoretical, both Heim's theory and ours have important things to say about the incremental processing of the utterances that make up discourse. And we present our theory as a specification of a processing device that takes logical form of a sentence along with current discourse context as input and delivers an updated discourse context as output. As an experiment, we have implemented portions of this device.

Key words: presupposition, discourse, context, accommodation, active logic.

1. INTRODUCTION

Presupposition is a pervasive feature of human language. It involves many interesting interactions between the utterances of a discourse and the *context* of the discourse. In this paper we focus on issues of logical form connected with the interaction of presupposition and discourse context, and illustrate our theory with some implementational work using the *active logic* framework.

After briefly reviewing two major types of presupposition theory (due to Gazdar and to Kartunen) we turn to the largely successful unified approach of Heim, where we find two serious difficulties. These consist in (a) a straightforward counterexample and (b) a type of discourse that we call a garden-path discourse. A garden-path discourse is one where a speaker first seems to presuppose some proposition with one utterance (e.g. that France has a King of that someone failed an examination) but then immediately pulls back, or cancels, that presupposition with the next utterance. We maintain that both the counterexample and the garden-path type of discourse can be handled by our active-logic version of Heim's theory. This requires us to reformulate and extend Heim's proposition-based logic-of-discourse framework into a syntactic mold.

Although this work is largely theoretical, both Heim's theory and ours have important things to say about the incremental processing of the utterances that make up discourse. And we present our theory as a specification of a processing device that takes logical form of a sentence along with current discourse context as input and delivers an updated discourse context as output. As an experiment, we have implemented portions of this device.

Our main contributions are (a) the reformulation and extension of Heim's theory into syntactic mode, and (b) the algorithmic details of that reformulation in terms of the active logic framework amenable to implementation. The latter contribution speaks to our larger ongoing project, in which active logic (already applied to real-time planning, common sense reasoning, as well as other issues in pragmatics of discourse) will be combined with companion work in Natural Language and Virtual Reality, in an effort to build a robust system that can deal with various effects of intra-sentential context, extra-linguistic context, and discourse context.

We begin with an informal discussion of discourse, utterance, context, and presupposition. Most of the examples we use are derived from the literature on presupposition found among the linguistic community.

As a monologue (a one-person sequence of utterances) or a dialog (a two-person sequence of utterances) unfolds, there is a context which influences the interpretation of each utterance of the discourse in turn. Most theories of discourse (ours included) (van der Sandt, 1992; Soames, 1989; Gazdar, 1979; Heim, 1982; Purang *et al.*, 1996a; Beaver, forthcoming) model a discourse *context* as including sets of propositions (Heim, 1982) or representations of such sets (van der Sandt, 1992).[1] There are two basic questions to be answered by any theory of discourse that has a place for discourse context:

1. What determines the contents of that context?
2. How does that context matter to the interpretation of the temporally unfolding sequence of utterances that comprise the discourse?

[1]Most theories also model context as having structure that is more complex than just a set of propositions or a list of representations (Kamp and Reyle, 1993). We discuss the structure of discourse context below.

In this paper we are concerned entirely with these two questions — but only as they apply to utterances that contain *presuppositional constructions* (Levinson, 1983; Beaver, forthcoming). Presuppositional constructions such as the aspectual verbs *stop* in:

- Have you stopped drinking rum?

set up, or trigger, *potential* presuppositions. In this case the potential presupposition is the proposition represented by:

- You have been drinking rum.

In other words, if someone said *Have you stopped drinking rum?*, he would normally be presupposing that you have been drinking rum.

In all cases, the potential presupposition is some proposition that can be determined from the sentence. These presuppositions are only potential because — depending on the context and depending on the form of the utterance itself — the presupposition may or may not be an actual presupposition. An actual presupposition is a proposition that is (or becomes) part of the discourse context.[2]

Presuppositional constructions include: (i) syntactic structures such as cleft sentences, definite noun phrases (*the red roses, that man in the wagon*), possessive noun phrases (*John's children, our house*), and wh-questions; (ii) sortal predicates such as *bachelor* and *widow*; (iv) aspectual verbs such as *stop* — in the above example — and *still*; and (v) factive verbs such as *regret* and *discover*. The odd-numbered examples among the following sentences exemplify these presuppositional constructions; the even-numbered examples are close cousins to their odd-numbered counterparts, but they lack the presuppositional constructions.[3]

Example 1. **It wasn't** some mice **that** ate the cheese.[4] (Cleft sentence.)

Example 2. Some mice did not eat the cheese. (Declarative sentence.)

Example 3. **Who** spilled the milk? (Wh question.)

Example 4. Did someone spill the milk? (Yes/no question.)

Example 5. Dole **regrets** that he has only one wife to give to his country. (Factive verb.)

Example 6. Dole said that he has only one wife to give to his country. (Non factive verb.)

[2]There are, of course many other ways in which discourse context influences the interpretation of the discourse utterances. See (Donnellan, 1978; Hirst, 1981; Grosz *et al.*, 1995; Crouch, 1995; Hahn *et al.*, 1996; Grice, 1989; Hirschberg, 1991; Green, 1990; Purang *et al.*, 1996a) for discussions of anaphora, centering, ellipsis, and implicature. These phenomena can be (and have been) studied and modeled computationally without dealing with the presupposition phenomena of this paper.

[3]The bold-face words and phrases in the following odd-numbered examples are the constructions that make these sentences presuppositional. They are the presupposition triggers that we discuss below.

[4]In this and other examples of negated verb phrases, we assume the reading where negation has wide scope — in this case over the whole sentence. We will return to this matter of scope in section 2.1 where will explain why this is an important consideration.

Example 7. Mary is a **widow**. (Sortal predicate.)

Example 8. Mary is happy. (Attributive predicate.)

Example 9. If he **stops** talking I'll ask my question. (Aspectual verb.)

Example 10. If he is talking I'll ask a question. (Temporal auxiliary verb.)

Example 11. I did not find **the bunch of roses**. (Definite noun phrase.)

Example 12. I did not find a bunch of roses. (Indefinite noun phrase.)

Each of the odd-numbered examples of utterances, above, depends on the discourse context (the context just prior to the utterance of the sentence) for its interpretation in the following ways: *1* presupposes that cheese was eaten; *EX3* presupposes that milk was spilled; *EX5* presupposes that Dole has only one wife to give; *EX7* presupposes that Mary is married; *EX9* presupposes that someone is talking; and *EX11* presupposes that there are roses. By contrast, the even-numbered examples (which have similar informational content to their odd-numbered counterparts) do not make these presuppositions. This difference is apparent in the following sample discourses which employ the first pair of examples:

Discourse 1.

1. Someone ate all of our cheese and crackers!
   (A declarative sentence.)
2. *EX1.* But it wasn't mice that ate the cheese. (The cleft sentence.)

After the first utterance, the context includes the fact that someone or something ate the cheese. We will specify how this inclusion of a fact can be represented formally in section 2 below. For now we will say that in *Discourse 1*, after the first utterance, the context happens to entail the presupposition that will be comming with the second utterance. The first utterance of *Discourse 1* is a declarative sentence. So the proposition it asserts is most likely something like:

Proposition 1. $\exists$ x [(person(x) $\wedge$ past:eat(x, f)] $\wedge$ $\exists$ x [(person(y) $\wedge$ past:eat(y, p)][5]

This proposition is simply added to the existing context. On our reading — where negation always has wide scope — *1*, the second utterance does not *entail* its presupposition (that cheese was eaten). Presupposition is a fact about the relation of an utterance to the previous context, not a fact about what can be deduced from an utterance. We will represent the logical form of *EX1* as:

Proposition 2. *LF1.* $\neg$ $\exists$ x [mice(x) $\wedge$ past:eat(x, f)].

This, of course, does not entail the presupposition:

Proposition 3. *PR1.* $\exists$ x [past:eat(x, f)].

[5]The constant f stands for the cheese and p stands for the crackers.

Discourse 1 is normal and straightforward because the first utterance updates the context in such a way that the presupposition of the second utterance, *PR1*, is already at hand; the context entails the presupposition that cheese was eaten.

Now compare the above discourse to a similar discourse with a different first utterance:

Discourse 2.

1. Eat up, everybody; no one has even touched the cheese and crackers.
   (A declarative sentence.)
2. *EX1.* So, it wasn't mice that ate the cheese.
   (The cleft sentence.)

This discourse seems odd because the context (after updating by the first utterance) actually *contradicts* the presupposition of *EX1*, the second utterance. So the speaker of the second utterance would be presupposing something that cannot be presupposed. That would be worse than just contradicting the first speaker (assuming this is a dialog). This is a clear case where context constrains future utterances. But since *EX1* does not entail its presupposition, *PR1*, there is no problem here. Presupposition is defeasible; the current context can block a potential presupposition from occurring. This defeasibility of presupposition has been modeled using the machinery of nonmonotonic logic.[6]

The difference between entailment and presupposition is apparent when we compare *Discourse 2* to the following sample discourse. Here we substitute the declarative sentence *EX2* for the cleft sentence *EX1* as the second utterance. *2* is the counterpart of *EX1* and both seem to assert the same thing — only using a different choice of words.

Discourse 3.

1. Eat up, everybody; no one has even touched the cheese and crackers.
   (A simple declarative sentence.)
2. *EX2.* So, mice did not eat the cheese.
   (The non-cleft sentence.)

Here there is no presupposition to worry about.

These and similar observations can be repeated for all of the other odd-numbered examples above. In each case, there is some proposition asserted *and* there is some other proposition presupposed. However, the logical forms of some of the odd/even pairs of examples look very much alike. The pair [*EX1*, *2*] have identical logical forms:

Proposition 4. *LF1.* $\neg \exists$ x [mice(x) $\wedge$ past:eat(x, f)][7]

Proposition 5. *LF2.* $\neg \exists$ x [mice(x) $\wedge$ past:eat(x, f)]

And the pair [*EX7*, *EX8*] have the same logical structure:

[6]See (Gurney and Morreau, 1995; Mercer, 1992).
[7]The constant f stands for the cheese.

Proposition 6. *LF7.* widow(m)[8]

Proposition 7. *LF8.* happy(m)

But these logical forms do not capture the presuppositions of *Example 1* and *Example 7.* The logical forms represent what is *asserted*, not what is presupposed. Yet we need to detect and properly deal with any presuppositions as they arise in a discourse.

1.1. Contexts for Presupposition and Accommodation

The triggers for presuppositions are (as we observed above) certain syntactic forms and certain words. This means that a discourse processing device must have access to both the syntactic forms of utterances and to information about their lexical items (as well as the logical forms). All of this can be managed by storing a feature +presupposition with the lexical entries of certain words, which can then be accessed after the utterance is parsed. This is an obvious technique for factive verbs and sortal predicates.[9] But (as we noted above) the proper processing of an utterance that includes a presuppositional construct also depends on the context of the utterance. There are three possibilities here:

Remark. (i) The context might entail the presupposition.

Remark. (ii) The context might entail the contradiction of the presupposition.

Remark. (iii) The context might entail neither the presupposition nor its contradiction.

Discourse 1 was an example of the first possibility and *Discourse 2* was an example of the second possibility. We observed that processing *Discourse 1* is unproblematic and straightforward.[10] We noted that *Discourse 2* also has an unproblematic interpretation, where the presupposition is defeasible with respect to the context. So far, we have not faced any problems regarding the first two possibilities — other than the computational problem of checking for inconsistency between a potential presupposition and a context.[11]

When (iii), the third possibility, arises, the context is neutral with regard to the potential presupposition. To illustrate this possibility, assume the following discourse begins with a context that is neutral with respect to the presupposition of *Example 9.*

Discourse 4.

1. Have you seen the senator? (A yes/no question.)
2. *EX9.* If he stops talking I'll ask my question. (The aspectual verb.)

The second utterance, *EX9*, is a conditional. It adds the following proposition to the context:

[8] The constant m stands for Mary.

[9] Storing +presupposition with the words *the*, *that*, *this* and some others in the lexicon will also be all that is needed to detect the definite noun phrase type presuppositional construction.

[10] That is, straightforward with regard to the presupposition.

[11] Our processing model deals with this problem using active logic in section 3 below.

Proposition 8. *LF9.* $[\exists$ x [talking(x, s) $\wedge$ stop(s, x)] $\rightarrow$ [ask(u, q)]

This says that if there is a talking, x, by the senator, s, and the s stops that x, than the utterer will ask q, his question. The scope of the quantifier, ($\exists$ x), is within the antecedent of this conditional. Hence *LF9* does not entail that the senator *is* talking. That is a presupposition in this discourse but the context, so far, does not include that presupposition. We have seen that: (a) if the context already entails a presupposition, *PR* then it need not be added to the context, and (b) if the context contradicts *PR* then it *should* not be added. But in the present case the first speaker should realize that the second speaker assumes the presupposition:

Proposition 9. *PR9.* $\exists$ x [talking(x, s)]

must hold. Dealing with presuppositions in this way is called *accommodation*. The rationale for performing accommodation is that: (a) the speaker said something that has a presupposition; (b) what was said would be inappropriate if the presupposition did not hold; (c) so, barring any reason to the contrary, one should just assume that the context really does entail the presupposition.[12] Accommodation is not presupposition. The latter is a property of utterances. The former is an action the hearer (or some simulation of the hearer in the form of a natural language interpreter) takes. So there are two tasks regarding the presuppositions of utterances:

1. The Detection Task: This answers the question, *What are the presuppositions of this utterance in this context?*
2. The Action Task: This answers the question, *Given a presupposition PR, what should be done?*

Based on the discussion so far, discovering presuppositions is not completely straightforward; Task 1 depends on presuppositional constructs, the entailments of the context, and the logical form of the utterance. Below we will raise further empirical problems in the discovery of presuppositions. Then, in section 2.2 we will show that accommodation can also be more complex than in *Discourse 4*. In section 3 we will specify our approach to processing assertion, presupposition, and accommodation that gets all of the empirical facts we have raised right.

1.2. Logical Form also Constrains Presupposition

The logical structure of the utterance is the remaining constraint on presupposition that we will consider. Logical structure was not a relevant factor for presupposition in the previous examples, *1*, *EX3*, *EX5*, *7*, *EX9*, and *EX11*. In these examples, any presupposition within any part of the sentence was also a presupposition for the complete sentence. This is not the case with the following examples:

Example 13. If he is talking he will soon **stop** talking . (Aspectual verb.)

This example is like *Example 9* but with the aspectual verb in the consequent rather than in the antecedent of the conditional. The logical form looks like:

Proposition 10. *LF13.* $\exists$ x [talking(x,s)] $\rightarrow$ [$\exists$ y [talking(y, s) $\wedge$ soon: stop(s, y)].

[12](Stalnaker, 1973) is the first clear statement of the concept of accommodation.

Example 14. If John has children then **his children** are very quiet. (Definite noun phrase.)

This example is like *Example 11* but with the definite noun phrase in the consequent of a conditional rather than in the outermost clause. The logical form looks like:

Proposition 11. LF14. $[\exists\ x\ [\text{children}(x) \wedge \text{has}(j, x)] \rightarrow [\exists\ y\ [\text{children-of}(j, y) \wedge \text{very:quiet}(y)]]$

Both *Example 13* and *Example 14* have presuppositional constructions that are in the consequent of a conditional (**stop talking** and **his children**). But, in neither case does the complete sentence have any presupposition. No one would think that anyone uttering either sentence is presupposing anything. For these examples, it is a good hypothesis that the semantic content of the antecedent prevents the potential presupposition in the consequent from becoming an actual presupposition. To save space we will not discuss other types of example that yield more facts about presupposition.[13]

2. RULES FOR CONTEXT UPDATING

In the field of linguistics, there are theories that are designed to account for the effects of context on presupposition; (Gazdar, 1979) is an example. There are also theories that are designed to deal with the effects of the internal sentence structure; (Kartunnen and Peters, 1979) is an early example. More recently, (Heim, 1983a) has proposed a unified theory of context updating that employs very reasonable underlying principles that can account for *all* of the cases so far. Thus, in place of a complex system of rules about the various cases of presupposition and accommodation, there are more convincing, simpler, rules that, by the way, apply to all types of discourse, not just the presuppositional cases. These rules can also be implemented as the interpretation module of a natural language discourse processor. In section 3 we will present our implementation of the discourse updating portion of such a module.

Heim's theory of discourse context has two parts:

1. There are four rules that employ a function, called +, for updating a discourse context, C, with a new utterance, U.
2. There is a mechanism for using accommodation for cases like *Discourse 4*, above, where the expected presupposition of an utterance, U, is not already entailed by the context.

The four rules will handle all of the problems in determining just what the presupposition of an utterance is — for any context and for any utterance (no matter what its logical form). The accommodation mechanism handles cases like *Discourse 4* above and also handles more complex cases (which were first investigated by Heim). We will discuss these cases and other problems with accommodation below in section 2.2. In the next section we discuss the four rules. In section 2.2, we discuss the accommodation mechanism, along with some new problems that must be addressed.

[13]The other cases include disjunction sentences, psychological verbs, and modal verb phrases (Levinson, 1983; Heim, 1992), as well as more blocking presuppositions through world knowledge (Gurney and Morreau, 1995).

2.1. Heim's Four Rules

The first rule is a basis rule that applies only to atomic sentences. Its purpose is to get a new context from the old context by adding the proposition that the utterance *asserts* to the old context:

Algorithm 1. *CCPB.* $C + U = C \cap [[U]]$.

Heim's formulation treats propositions as sets of possible worlds rather than as the logical forms we employ.[14] Thus C is a set of possible worlds in which the proposition or propositions that make up the context are all true. U is simply the sentence of the utterance and [[U]] is the proposition (set of possible worlds) asserted by that utterance. This and the other three rules only apply where the function + applies (that is, where + is defined). That function only applies where *all* of the presuppositions of U are already entailed by C. This is the case (i) of section 1.1, above. To use CCPB, we must assume there is a process that has access to the syntactic form of U in order to detect any presuppositional constructions like those discussed in the Introduction. This not problematic because potential presuppositions that originate from constructs in *atomic* sentences are always potential presuppositions of the complete sentence. If C entails [[woman(m)]] then this rule would apply to interpreting *Example 7.*

The next three rules deal with the non-atomic sentences. These are cases where the logical form of a sentence determines whether a potential presupposition is really a presupposition of the complete sentence (section 1.2 above).

Algorithm 2. *CCPA.* $C + (U \wedge V) = (C + U) + V$.

This is an obvious extension of *CCPB*. If [[U]] happens to entail the presupposition of V then this rule will apply even where C does not already entail that presupposition. By the time V comes up for interpretation, the new context has U in it.

The formulation of the next rule is crucial to Heim's theory. This is the rule for negated sentences:

Algorithm 3. *CCPN.* $C + \neg U = C \setminus (C + U)$.[15]

This rule tells us to first drop the negation on U and update C with U, as if U were being asserted, not denied. This will give us some context, say CN. Then subtract that context from the original C. This rule applies to *Discourse 1* where we want to update with the secon utterance:

- *EX1.* It wasn't some mice that ate the cheese.

The net result is that, presuppositions on negated utterances are the same as the presuppositions of the non-negated counterparts. This empirically correct result follows from *CCPB* and *CCPN.*

The fourth rule, for material implication, can be derived from the others:

Algorithm 4. *CCPC.* $C + (U \rightarrow V) = C \setminus ((C + U) \setminus ((C + U) + V))$.

[14] This makes for an elegant formulation of the rules. We will address the problem of implementing these rules using logical forms in section 3.

[15] The operator \ performs set-theoretic intersection of two sets of possible worlds. A naive rule for updating contexts with negated utterances such as: $C + (\neg U) = C \setminus [[u]]$ cannot be made to work properly.

CCPC applies to sentences like *Examples 13* and *EX14*. The four rules, taken together, account for the variety of facts about presupposition by systematically reducing complex presupposition problems to simple ones.[16]

2.2. Accommodation

In Heim's system, Context updating becomes interesting when we consider cases like *Discourse 4* where accommodation is appropriate. This was a case where the context was neutral regarding the presupposition: *The senator is talking*. Heim's rule for accommodation applies whenever some C does not entail a presupposition.

Algorithm 5. *ACC.* Where C + U is not defined; $C + U = (C \cap PR(U)) + U$.

This rule, of course, works immediately for *Discourse 4*. The was a case (ii) type context (of section 1.1). However + is also undefined for a case (iii) type context — where C entails the contradiction of the presupposition. That means that the rule *ACC* is intended to apply the type (iii) contexts as well as type (ii). We will examine how this works next.

2.3. Global and Local Accommodation

Discourse 2 is a type (iii) case; the context for the second utterance entails that the cheese was not eaten, but the utterance presupposes that it *was* eaten. There is a way that the accommodation rule *ACC* and the update rule *CCPN* work together to (perhaps surprisingly) correctly interpret discourses like *2*. *CCPN* applied to the second utterance looks like this:

- $C + \neg \exists x\ [\mathrm{mice}(x) \wedge \mathrm{past{:}eat}(x, f)] = C \setminus (C + \exists x\ [\mathrm{mice}(x) \wedge \mathrm{past{:}eat}(x, f)])$

Given that the operation to the right of the \ must be performed first, if the accommodation rule applies only here, and only to this instance of C, the discourse will be interpreted properly. This has been called local accommodation; only the local instance of C is accommodated with a presupposition called for by the utterance. Since C already entails $\neg\ [\exists x\ [\mathrm{past{:}eat}(x, f)]]$ adding the presupposition of the second utterance to (this instance of) C will, of course produce a contradiction, but only to the right of the backslash. So that will have no affect on the original C, the one to the left of the backslash. This means that in *Discourse 2* the second utterance does not update the context at all. We take this to be the correct result. This is another case where Heim's rules work together to to handle some of the complexities of interpreting presuppositional utterances.

2.4. Some Problems with Accommodation

Obviously contradictions and their avoidance (where necessary) play an important role in context updating – along with the need to know (or deduce) what a

[16]Some investigators have proposed that we can view the difference between cases where potential presuppositions become actual and cases where they do not as differences scopal reading. This is antithetical to pragmatic approaches to presupposition (such as Heim's and ours). Since, it does not distinguish between presupposition and entailment, Heim's theory would be vacuous. For an explanation why these scopal theories will not work for the general case see (Beaver, forthcoming).

context entails. However, there are two problems in using accommodation that remain unsolved.

(1) Given both global and local accommodation, is there a principled way to choose in every case?

(2) Although local accommodation produced the correct result for *Discourse 2*, it allowed a contradiction to appear in the calculation. There are other discourses where this fact ensures that the method will give the *wrong* results. The following is that sort of discourse:

Discourse 5.

1. There are no roses today.
2. So, **the roses** are not in **the fridge** as I had thought.

The second utterance has two presuppositional constructions — both definite noun phrases. It's logical form is:

- $\neg \exists$ x y [roses(x) $\wedge$ fridge(y) $\wedge$ in(x, y)]

This non-atomic logical form can be rewritten as a quantifier-free formula so that the CCP rules can be applied to it.[17] Then rules *CCPA* and *CCPN* will apply:

- C + U = C $\setminus$ (((C + roses(x)) + fridge(y)) + in(x, y))

Now the problem with local accommodation can be exposed. C does not entail $\exists$ [roses(x)], which is the presupposition of the clause roses(x). This clause is our logical form for **the roses** in the utterance. So, it's presupposition is $\exists$ [roses(x)]. That means *ACC* should be used on (C + roses(x)). Now adding the presupposition, $\exists$ [roses(x)] to C produces a contradictory context for the next clause, fridge(y), because the first utterance placed $\exists$ [roses(x)] in C:

- C + U = C $\setminus$ (($\emptyset$ + fridge(y)) + in(x, y))

The correct interpretation for the second utterance in the discourse about the fridge is that it actually presupposes there is a fridge. However, once the context becomes contradictory, $\emptyset$, everything on the right of the backslash collapses to $\emptyset$. There is no more accommodating and we are left with the original C. The CCP rules cannot get the right interpretation. This is one problem for the theory. In the next section we briefly mention another, related problem.

2.5. Garden Path Discourses and Cancellation

We have used discourse *Discourse 5* to uncover a fundamental problem with, what we think, is one of the best theories of context updating. There is another type of discourse that the CCP rules will get wrong. We call these garden path discourses,

[17] To simplify the discussion in this paper, we have presented a propositional-logic version of the CCP rules. For logical forms whose quantifiers scope over $\wedge$, the techniques of discourse representation theory (Heim, 1982; Chierchia, 1995) — which delay binding of quantifiers — can be used to give the CCP rules access within the scopes of quantifiers.

because the interpreter first accommodates some presupposition, adding it to C but then has to withdraw it from the context when the next utterance explicitly denies that presupposition. If we reverse the order of the two utterances in *Discourse 2* we have such a discourse.

Discourse 6.

1. *EX1.* It wasn't mice that ate the cheese.
   (The cleft sentence.)
2. In fact, no one has even touched the cheese and crackers.
   (A declarative sentence.)

Here the cleft sentence come first, leading to the addition of the presupposition that the cheese was eaten to the context. Next, the second utterance contradicts that presupposition. Thus the presupposition must be withdrawn.

2.6. Contradictions

Our diagnosis for both of the above problems is that something has gone wrong in the handling of contradictions. The method of local accommodation allows contradictions to appear in the formulae. In some cases they are harmless, in others not. The trouble was that once a contradiction appeared there was no way to remove it. Although Heim's theory accounts for facts about presupposition in some sense, the logic employed is in one sense monotonic. Once a proposition is incorporated into a context it cannot be removed. We see that, in this system, contexts always "increase" monotonically; thus the sets of possible worlds they represent always shrink monotonically.

An active logic, by contrast, is one that will allow propositions to be both added and later withdrawn from the evolving context. It also allows contradictions to appear. In our implementation any explicit contradictions are promptly removed. This kind of growing and shrinking of the context as well as the harmless appearance of contradictions require principled management. Active logic achieves this by an explicit ordering of steps along with rules that may refer to previous steps. None of this was envisioned in Heim's system. Our hypothesis is that we can implement most of Heim's system in active logic and thereby properly manage the troublesome aspects of context updating.

3. ACTIVE LOGIC COMPARED TO NONMONOTONIC LOGIC

Active logic (Miller, 1990; Elgot-Drapkin and Perlis, 1990) is a family of formalisms developed for the purpose of computationally modeling the reasoning process in a way that respects the passage of time as reasoning proceeds. These formalisms have been applied to a number of domains, from multi-agent interaction to deadline-coupled planning, from fully-decidable default reasoning to reasoning in the presence of contradictions, from correcting misidentification errors to perceptual reference.

The language of active logics is first-order, and there is a standard first-order semantics, except that the predicate expression "Now(x)" is true iff the time is x: i.e., there is an external clock. Since contradictions are tolerated (see below) then there is no need for inference rules to be sound[18]; moreover, inference rules can explicitly make use of the current time.

As an illustration of an active logic rule of inference, consider:

i: Now(i)
i+1: Now(i+1)

This indicates that from the belief $Now(i)$ at time i the agent infers the belief $Now(i+1)$ at time $i+1$, and no longer believes $Now(i)$. This further illustrates that beliefs are not held indefinitely. Some are inherited from step to step and some are not.

Rather than proceeding from one nonmonotonic theory (with one set of axioms) to another nonmonotonic theory (with an updated set of axioms) there is one evolving theory in active logic. It models a process of thinking that takes a reasoner from one belief state to the next. As a default everything believed at step n would be inherited to step n + 1. But there are various rules that modify this blanket inheritance. For example, if p and not(p) appear at step n then the belief contra(p, not(p)) appears at step n + 1. Then both p and not(p) are blocked from inheriting to step n + 2. For our present task, this is perhaps the most important characteristic of active logic. It works by forward chaining from step to step allowing contradictions to appear as they will. It uses detection of explicit contradictions to disinherit propositions from the belief set. In this way active logic achieves some of the effects of various nonmonotonic logics but in a different manner.

A traditional nonmonotonic approach to contradiction is the truth maintenance system (Doyle, 1979) which initiates a backtracking process on finding a contradiction. This examines the causes of the contradiction, chooses some assumption to reject, and repeats this process if necessary until the contradiction is not supported. By contrast, although an active logic can be structured so as to do the same thing, this is not required. Rather, at a minimum two direct contradictands P and $\neg P$ are simply disinherited and not allowed to serve as antecedents for further inferences. This can proceed in parallel with the other inferences in the active logic and therefore the inference process does not seize up when a contradiction is discovered.

Nevertheless the general approach to discourse we are advocating here could be carried out by other frameworks as well. The key element here is that to handle discourse updating, one needs a framework that takes account of time passing.

In this paper we will present a treatment of the "fridge and roses" problem of *Discourse 5*, as a key illustration of our ideas. First, however, we provide some material to orient the reader to our system.

4. CONTEXT UPDATING IN ACTIVE LOGIC

Here we present our implementation of context updating in active logic for the purpose of understanding discourse. For this we will be concerned primarily with lists of formulae that represent the discourse context, that is, the record of what has been said up to the current step. We operationalize Heim's account of discourse context by replacing propositions by sentential beliefs.[19] These include the record of

[18]Nevertheless some work has been done on possible world semantics in an attempt to capture a version of soundness and completeness (Nirkhe *et al.*, 1995).

[19]There are well known paradoxes of self-reference associated with syntactic approaches to propositional attitudes such as belief (Montague, 1963; Thomason, 1980). For more discussion see (Perlis, 1988). These

utterances of the discourse up to the present moment and also any presuppositions generated from that discourse.

At step n the information state might look something like

Step n: ctxt([...], n)

where [...] is an ordered list of logical formulae representing the discourse context. Although some of the hearer's other beliefs will normally change as the discourse unfolds we will ignore this possibility and only represent beliefs that concern what was said in the discourse. Here we introduce some of the predicates and rules used in our system.

4.1. Predicates Used

We now list ten predicates relevant to our formalism. Note that while most of them have a time argument, some do not. This is simply because the latter always appear inside other predicates that do have a time argument.

1. *ctxt(c, t)* represents that the context at time *t* consists of the list *c* of formulae. For example, ctxt([assert(exists(x, king(x))), assert(hiding(x))], 3) could be the context at time 3 in the mind of a hearer. It will become apparent that, in general, there are many more active logic steps than utterances in a discourse.

2. *dfnt(X)* represents a definite description in the utterance. This is a piece of syntax produced by the parser. An example would be dfnt(king(x)).

3. *ut('X', t)* represents that *X* has been uttered at time *t*.

4. *parse(X, t)* is the parse obtained at time *t* by processing an utterance at the previous step, time *t – 1*. If the previous step had a new utterance such as ut('The roses are red', 5) then we would find parse(and(dfnt(roses(x)), red(x)), 6) at the next step. The potential presuppositions in most of our examples arise from the speaker using selected syntactic forms such as definite descriptions. Therefore it is essential to parse utterances in such a way that exhibits this syntax.

5. *update(X, t)* represents at time *t*, elements of the discourse that still need to be incorporated into the context according to Heim's rules. X is a list of contexts, atoms from the inputs and the + and \ operators.[20]

6. *presup(X)* marks *X* as a presupposition in the context.

7. *exists(x, P(x))* indicates that an object with property *P* exists. This is the typical presupposition, for example, presup(exists(x, king(x))).

8. *assert(X)* marks *X* as having been asserted by an utterance.

issues do not affect the present considerations.

[20]In the code for our implementation we use a postfix ordering of the operators + and \. This facilitates parsing formulae according the CCP rules. In this paper we leave those operators in their places (as infix operators as they appear in Heim's CCP rules) for better readability.

9. *contra(X, Y, t)* indicates that there is a contradiction between the formulae X and Y in the context at time $t - 1$. In our implementation, only explicit contradictions can be detected. An example is contra(assert(not(exists(x, king(x)), presup(exists(y, king(y))), 4))). Here an assertion is found to contradict a presupposition at time 4.

10. *kill(X)* indicates that formula X has been marked for killing. It will not be inherited to the next step. In our system both members of a contradiction are marked kill, so neither will be straightforwardly inherited to the next step.

4.2. Rules of inference used.

The rules will be presented in the form:

i: X
i+1: Y

If X is believed at step i, then Y is added to the beliefs at step i+1. Nothing else is added to the beliefs that is not mentioned by these rules.

0 i: ut('X', i)
i+1: parse(P(X), i+1)
where P(X) is a parse of X. If X is heard as an utterance at step i then the parse of X appears at the next step. This "rule" is not actually in our system and ideally would be handled by a parsing module.

1 i: ctxt(C, i) parse(X, i)
i+1: update(Heim(C, X), i+1)
Given a syntactic parse X, this rule initiates the process of updating the context C by recursively applying Heim's CCP rewrite rules (see section 2.1) to C and X. The result, $Heim(C, X)$, is a sequence of basic context update operations involving only + and \. For example, if the parse is parse(and(dfnt(roses(x)), red(x)), 1), and the context is ctxt(c_1, 1), $Heim(c_1, X)$ will be [c_1, +, dfnt(roses(x), +, red(x)] which does not involve "and".

The next task is to successively apply the basis rules from left to right to obtain the updated context. This is done in several steps using the two rules below.

2 i: update(X, i)
i+1: update(first(X), i+1)
first(X) is the result of applying a basis rule to the first operation in the list X. There are several cases depending on the operator and on the form of the operands. For example, given update([c_1, +, dfnt(roses(x), +, red(x)], 2) at time 2, we will have update([c_1 +, red(y)], 3) provided that exists(y, roses(y)) appears in c_1. This case is an illustration of the rule CCPA where no accommodation was needed because c_1 already entailed the presupposition of dfnt(roses(x)). These active logic rules are the ones that implement the CCP rules along with global accommodation where necessary as described in section 2.

3 i: update(X, i)
i+1: ctxt(X, i+1)

Once we have exhausted the operations in the update using the above rule, the resulting context is added to the set of beliefs of the system.

4 i: ctxt([..., foo(X), ..., bar(not(Y)), ...], i)
i+1: ctxt([..., kill(foo(X), ..., kill(bar(not(Y))), ..., contra(foo(X), bar(not(Y)))], i+1)
This rule detects direct contradictions in the context. Here, X and Y are unifiable and *foo* and *bar* are either *assert* or *presup*. Note that both members of the contradicting pair *foo(X)* and *bar(not(X))* are tagged for killing at *i+1*. The next rule which is applied at step $i+2$ decides which member of the pair can be inherited to step $i+3$.

5 i: ctxt([..., kill(foo(X), ..., kill(bar(not(Y))), ..., contra(foo(X), bar(not(Y)), i+1)], i+1)
i+1: ctxt(Z, i+1)
Z is the context resulting from resolving the contradiction flagged at step i. The contradiction can be resolved by using various additional sources of information.[21] In our system, an assertion is always preferred for inheritance over a presupposition.

6 i: ctxt(X, i), ctxt(Y, i)
i+1: ctxt(X ∪ Y, i+1)
Rule 4 results in adding a freshly updated context to the beliefs of the agent. If the beliefs contain the context before the update, we need to union the 2 contexts. Note that this could introduce contradictions in the resulting context. That will be detected at the step $i+2$ using rule 4 above.

7 i: ctxt(X, i)
i+1: ctxt(X, i+1)
We simply inherit the context to the next step we have just one context (that is, if rule 6 does not apply).

8 i: now(i)
i+1: now(i+1)
This is the "clock rule". Time does not stand still while we are reasoning.

All rules are active at all times. That is, if a rule applies at a step, it always fires at that step. There is no need to employ resolution between conflicting rules. Systems of nonmonotonic logic often resort to conflict resolution and prioritizing of default rules. These measures are applied to avoid the appearance of contradictions. In our system we can manage contradictions. We let them arise at one step whereupon we disinherit them at the next step, allowing time in further steps to decide which contradictand (if any) to accept.

4.3. Output Trace for a Case (i) Discourse

We now present some of the steps of the output trace for a simple case (i) discourse. Some details are not shown, for example, the argument representing time in the predicates; and various steps are not shown.

[21] See Miller (Miller, 1990) for more on contradiction resolution in active logic.

Discourse 7.

1. There are roses and tulips.
2. But the roses are not yellow⟩

We assume the initial context is null, containing no information.

Step
0 ctxt([], 0),ut('There are roses and tulips')

Let c_1 = [].[22]

1 c_1, parse(and(exists(x,R(x)),exists(y,T(y))))

This is the result of parsing the utterance and inheriting the previous context.

2 c_1, update([c_1,+, exists(x,R(x)), +, exists(y,T(y))])

This step applies the CCP rewrite rules in preparation for applying the basis rules.

7 c_3

After a few steps, the updated context c_3 contains the assertions that there are both roses and tulips in the discourse context. c_3 is ctxt([assert(exists(x, R(x))), assert(exists(y, R(y)))]). We then add the next utterance.

8 c_3, ut('But the roses are not yellow')

9 c_3, parse(not(and(dfnt(R(z)),Y(z))))

The new utterance has been parsed and we now need to incorporate it into the context. For this exercise, we are ignoring rhetorical words like 'but' and 'because'.

10 c_3, update([c_3, \, c_3, +, dfnt(R(z)), +, Y(z)])

This results from application of the rule for negation, CCPN. Since there is a definite description **the roses**, the system first looks for exists(y, R(y)) in c_3 which does in fact include it. Then, updating can proceed normally.

14 c_3, update([c_4, \, c_6])

Here everything from the second utterance has been absorbed into c_6. All that remains is to combine c_4 with c_6 by set difference, \.

The final context for D_1 is

ctxt([assert(exists(x,R(x))),assert(exists(y,T(y))), assert(not(Y(x)))])

[22]We will use c_i for both the list of formulae in the context and for the predicate ctxt(c_i, j). Which is meant will be evident from the context.

4.4. Output Trace for a Garden Path Discourse

Below we will display some of the output from our system processing a garden path version of the discourse involving two potential presuppositions that we discussed earlier.

Discourse 8.

1. The roses are not in the fridge.
2. Because there are no roses.

Here we have a case where a presupposition (that there are roses) is first added to the discourse context, only to be later removed. Based on our analysis above, Heim's system cannot deal with this discourse. Our diagnosis was that since Heim had to avoid a contradiction in the final context, accommodation of the presupposition of the definite description "the roses" was done locally. In our system we can manage contradictions. Therefor we can always accommodate presuppositions globally. The significance of this will become clear where we discuss step 3 below. As predicted by Heim's analysis, global accommodation for examples like these will lead to unwanted contradictions. In active logic, if a contradiction arises we simply disinherit it at the next step. In this way our system can produce the correct results for this discourse.

Heim's strategy of choosing local accommodation to avoid global contradiction (which worked for *Discourse 2*) is not even applicable. After the first utterance, the context should contain two presuppositions, that roses exist and that a fridge exists. Then, after the second utterance, the first presupposition should be withdrawn. We will show that this is a fairly straightforward process in our system.

Step
0 ctxt([],0) ut('The roses are not in the fridge')

Let the initial context be null, $c_1 = []$.

1 c_1, parse(not(and(dfnt(R(x)),dfnt(F(y)),in(x, y))))

This is the result of parsing the utterance u_1 and inheriting the previous context.

2 c_1, update([c_1, \, c_1, +, dfnt(R(x)), +,dfnt(F(y)), in(x,y)])

The update predicate applies the CCP rules to the parse of the first utterance in preparation for the application of the basis rules. The next applicable rule is CCPA which applies to:

[c_1, +, dfnt(R(x))]

Since we have a definite descriptor, we first search the previous context c_1 for a previous mention of roses. As there is none, we accommodate (globally) the context with the presupposition that there are roses.

3 c_1, update([c_2, \, c_2, +, dfnt(F(y)), +, in(x,y)]

Thus c_2 here at step 3 includes the information that it is presupposed that there are roses. Since we began with a null context we have a very small context at this point:

c_2 = [presup(exists(x, R(x)))]

In our system we always use global accommodation. That means that both instances of c_1 in step 2 get accommodated with the presupposition.

At step 11 (below) all of the first utterance has been processed and the next utterance is perceived.

11 c_4, ut('Because there are no roses')

Here

c_4 = [presup(exists(x,R(x))),presup(exists(y,F(y))), assert(not(in(x, y)))])

and we are ready to process the second utterance which should cancel one of the presuppositions in the current context. Since this utterance itself has no presuppositions it will be added to the context c_4 in a straightforward way, using the rule CCPA. We skip down to step 21 where the utterance is fully incorporated into the context.

21 ctxt([presup(exists(x,R(x))),presup(y,F(y)), assert(not(in(x,y))), assert(not(exists(z,R(z))))

We now have a context which presupposes that there are both roses and a fridge but which also asserts that there are no roses. At the next step the contradiction is found.

22 ctxt([kill(exists(x,R(x))),presup(y,F(y)), assert(not(in(x,y))), kill(not(exists(z,R(z))))
contra(presup(exists(x,R(x))), assert(not(exists(z,R(z))))]

The formulae that caused the contradiction appear at this step flagged for possible killing. One or both will not inherit to the next step. Nor will the contra formula inherit to the next step.

Spreading the reasoning over steps is necessary to properly manage all this. The system can reason at one step on the basis of something that appears at a previous step, even though that something does not appear at the current step. This ability is important to the proper management of contradiction.

23 ctxt([NULL(exists(x,R(x))),presup(y,F(y)), assert(not(in(x,y))),assert(because),
assert(not(exists(z,R(z))))

The contradiction has disappeared. Using the fact that one of the contradictands was a presupposition and the other an assertion we disinherit the presupposition and we reinstate the assertion that roses do not exist.

24 ctxt([kill(exists(x,R(x))),presup(y,F(y)), assert(kill(in(x,y))),assert(because),
assert(not(exists(z,R(z))))

Since we are asserting the roses do not exist, we have to mark any formulae about roses for killing.

At the end of processing D_9, we have the following context:

ctxt([presup(y,F(y)), assert(not(exists(z,R(z))]])

Note that, even though other things were said in the discourse, the final context includes only two items. There is no information about roses not being in a fridge. The fact that the speaker *said* the roses were not in the fridge is part of the meta-linguistic information about the discourse. In the canonical presupposition examples we are treating, meta-linguistic information is, of course important. We represent and use this information via our ut predicate. However there is a discernable concept of the content of the discourse that is separate from the linguistic events and facts. This is what we have been calling the context and representing with our ctxt predicate. The other facts (ut, parse, etc.) are, however, still available. They inherit through all steps but we have only shown them where they play a role in reasoning from one step to the next.

4.5. Output Trace for an If-then sentence

In the case of if-then sentences, the potential presuppositions in the consequent will not always be presuppositions of the whole sentence.

Consider for example "If there are roses, then the roses are fresh". Here, the consequent has the potential presupposition that there are roses. However that presupposition is *not* a presupposition of the whole sentence. On the other hand, in the sentence "If there are tulips, then the roses are red", the presupposition of the consequent (that there are roses) does survive to become a presupposition of the whole sentence.

The output trace below show the behavior of our system in the first case.

Step
0 ctxt([],0) ut('If there are roses, then the roses are fresh')

Here too, the initial context is null, $c_1 = []$.

1 c_1, parse(if(exists(x, R(x)),and(dfnt(R(y)),F(y))))

This is the result of parsing the utterance and inheriting the previous context.

2 c_1, update([c_1, c_1, exists(x, R(x)), +, exists(x, R(x)), +, dfnt(R(y)), +, F(y),\, \])

The update predicate applies the CCP rewrite rules to the parse in preparation for the application of the basis rules. The first few updates are as we have seen before, and we get:

4 c_1, update([c_2, c_3, +, dfnt(R(y)), +, F(y),\, \])

Here $c_2 = c_3 =$ [exists(x, R(x))].
Since we next have a definite descriptor, we first search the previous context c_3 for a previous mention of roses. We find one and therefore we *do not need* to accommodate the potential presupposition of the definite descriptor. Instead, we equate the roses in the definite descriptor to the roses mentioned previously. The state now is:

6 c_1, update([c_2, c_4, +, f(y),\, \])

where c_4 = [exists(x, R(x)), exists(y, R(y)), equal(x, y)]
The next few steps proceed as in the preceding examples and the context we end up with is:
[assert(not(and(assert(R(x)),not(and(and(assert(R(y)),equal(x, y)),assert(F(y)))))))]
where we express if-thens in terms of ands and nots. Note that there is no presupposition present in this case.

Let us now consider the second case: "If there are tulips then the roses are fresh". Here, contrary to the previous case, we expect the presupposition that there are roses to be a presupposition of the whole sentence.

Step
0 ctxt([],0) ut('If there are tulips, then the roses are fresh')

Here too, the initial context is null, c_1 = [].

1 c_1, parse(if(exists(x, T(x)),and(dfnt(R(y)),F(y))))

This again is the result of parsing the utterance and inheriting the previous context.

2 c_1, update([c_1, c_1, exists(x, T(x)), +, exists(x, T(x)), +, dfnt(R(y)), +, F(y),\, \])

The update predicate applies the CCP rewrite rules to the parse in preparation for the application of the basis rules. The first few updates are as we have seen before, and we get:

4 c_1, update([c_2, c_3, +, dfnt(R(y)), +, F(y),\, \])

Here c_2 = c_3 = [exists(x, T(x))].
Since we next have a definite descriptor, we first search the previous context c_3 for a previous mention of roses. In this case, we do not find one, and therefore we add the presupposition that there are roses:

6 c_4, update([c_5, c_6, +, f(y),\, \])

Here, c_4 = [presup(exists(y, R(y)))], and
c_5 = c_6 = [assert(exists(x, T(x)), presup(exists(y, R(y)))].
Note that we have global accommodation here.

The rest of the process proceeds as usual, and we end up with the final context:
[presup(exists(y, R(y))), assert(not(and(assert(T(x)),not(fresh(y)))))]

This presupposes that there are roses in addition to asserting the conditional.

5. RELATED RESEARCH

There are numerous theories of presupposition, accommodation, and the projection of presupposition. There are fewer computational implementations. And of these most do not discuss or attempt to treat the cases of actual cancellation of presuppositions. We have chosen to study and adopt Heim's theory because it covers many of the problem cases and it also suggests the kind of step by step, forward chaining reasoning of active logic. Ours is an approach appealing to nonmonotonic reasoning. Other nonmonotonic approaches to presupposition include those of Mercer (Mercer, 1988), Marcu and Hirst (Marcu and Hirst, 1994), and McRoy and Hirst (McRoy and Hirst, 1995; McRoy and Hirst, 1993).

Mercer employs a system of default rules to model the presuppositions arising from syntactic forms that appear in utterances. In (Mercer, 1988) he deals with adverbial presuppositions such as the following:

If John kicked the ball, then Bill kicked the ball too.

If Fred called yesterday, then he will call again today.

In these cases the adverbs "too" and "again" give rise to potential presuppositions; that someone else kicked the ball and that Fred called before. But in each case the potential presupposition does not project. The examples we have been discussing are mostly cases of existential presupposition triggered by definite descriptions. We do not think that this is an important difference from Mercer's examples for the phenomena under study and we believe that we could in the future bring adverbial and other sources of presupposition into our system. The important similarity between Mercer's paper and ours is the concern with the complexity of presupposition. Mercer's if/then sentences block presupposition just as the if/then utterance in discourse D_3. Now Heim's CCP rules which we implement are intended to account for projection in if/then sentences in a well–founded, uniform way. Therefore we expect that our system can deal properly with Mercer's examples. A major difference between Mercer's approach and ours is that he does not address the time evolving positing and cancellation of presupposition. This is a constant theme in the comparison of our approach with others.

Marcu and Hirst (Marcu and Hirst, 1994) present a system designed to handle cancellation of presuppositions. But they take an approach quite different from our approach. They do not model the step by step incremental reasoning about context. Rather they compute an entire new theory after each utterance. Although we have not verified this, their system may be able to get the correct results for most if not all of our examples. It appears that they would deal with a discourse like D_9 by first computing the two presuppositions after the first utterance. Then, after the second utterance they would discard all beliefs and compute a fresh set of beliefs consistent with the entire discourse. They also develop an ontology based on Meinong's theory of objects. They use this ontology to deal with discourses about fictional entities and discourses that involve presupposition. We believe, along with others (Gazdar, 1979; Heim, 1983b; Kartunnen, 1973; Soames, 1982; Kay, 1992) that presupposition can be treated separately from fictional discourse and that we can achieve this without a Meinongian ontology. The ultimate success of our approach would bear out this claim.

McRoy and Hirst (McRoy and Hirst, 1995; McRoy and Hirst, 1993) present an abductive treatment of misunderstanding in dialogs. By way of contrast we use a largely deductive (though time–situated) inference engine. As McRoy and Hirst note, a deductive approach leads to contradictory beliefs and the need for belief revision. However, in our approach, belief revision is handled as part–and–parcel of the inference process; it does not require an additional module or phase of processing[23]. Moreover, contrary to (McRoy and Hirst, 1995), we do not need to assume there are

[23]Traditionally such an additional module might be treated as a truth maintenance system (Doyle, 1979)

no "abnormalities"; or rather any abnormality is easily retracted later in the dialog when new evidence is heard.

Thus our approach is an exploration of the utility of largely deductive methods in natural language processing; when contradictions arise, our logic engine applies the applicable rules.[24] As shown in our output traces in section 4 and in Miller (Miller, 1990), active logic engines are often able to reason quite effectively with contradictions. It is this fact that provides the underlying framework that we are exploiting.

Ballim and Wilks (Ballim and Wilks, 1991) provide another treatment of belief and inference, essentially context-based, that perhaps could be marshalled in similar ways to our use of active logics here. However, their treatment does not appear to be contradiction-tolerant, and their use of time is much less explicit than in active logics, and in particular the reasoning done with "viewpoints" (as their contexts are called) does not reflect a notion of current evolving time as in active logics. Instead they would apparently utilize a back-and-forth juggling of viewpoints to keep contradictions from surfacing within the same viewpoint.

6. CONCLUSION AND FUTURE WORK

In conclusion, we have shown that active logic can be applied to the problem of updating according to the + function in Heim's system of rules for discourse context. Heim's rules account for important effects of complex structure in utterances. And active logic accounts for the problem of how to alter a given context by both expanding and contracting contexts as required. In this way, the resources of active logic can be brought to bear on an important class of problems in natural language discourse processing. Well–known problems of presupposition projection can be accounted for as well as new problems exemplified by cancellation of previously inferred presupposition.

Elsewhere, we have looked at the problem of computational implicature as a kind of inference that bears on context (Purang *et al.*, 1996b). The methods employed there are similar to those described here for presuppositions. We are working on a uniform treatment to cover both cases. Another line of work we have pursued is a virtual reality–natural language interface (Gurney *et al.*, 1996). We plan to combine that work with our context updating work.

Our long-range goal in this work is the design and implementation of a time-situated natural-language discourse-understanding system based on a formal theory of pragmatic reasoning. Among the issues for future research there is the following question: At any time (or step) in the discourse process there can be implicit contradictions – ones that have not yet been detected. Our current system only detects explicit contradictions. A question is: could this lead to trouble in a discourse? One answer is Perhaps not; perhaps a feature of a coherent discourse is that the speakers quickly say things to prevent such problems. If so, the fact that our system may be vulnerable to this kind of bad discourse may indicate that we are on the right track. This is a matter for empirical investigation. One immediate goal is to unify our algorithms for presupposition and implicature, to facilitate treatment of both of

[24]We do not argue that active logic is the only viable framework for this; truth-maintenance systems may work equally well. But that conclusion awaits either an implementation of an appropriate modification of Heim's theory or a implementation that solves the same problems.

these in the same discourse.

ACKNOWLEDGEMENTS

This research was supported in part by the Army Research Laboratory through a contract from the Army Research Office and in part by the National Science Foundation. We thank Elizabeth Klipple, Jean Braithwaite, Michael Miller, and Michael Morreau for helpful discussion of the syntax, semantics and pragmatics of natural language discourse.

REFERENCES

AFZAL BALLIM and YORICK WILKS. 1991. Artificial Believers: The Ascription of Belief. Lawrence Erlbaum Associates.

DAVID IAN BEAVER. forthcoming. Presupposition. In J. VAN BENTHAM and A. TER MEULEN, editors, The Handbook of Logic and Language. Foris.

G. CHIERCHIA. 1995. Dynamics of Meaning. University of Chicago Press, Chicago, Illinois.

RICHARD CROUCH. 1995. Ellipsis and quantification: a substitutional approach. In Proceedings of the European Chapter of the Association for Computational Linguistics, Dublin, Ireland. EACL.

K. DONNELLAN. 1978. Speaker reference, descriptions and anaphora. In PETER COLE, editor, Pragmatics, volume 9 of Syntax and Semantics. Academic Press, New York.

J. DOYLE. 1979. A truth maintenance system. Artificial Intelligence, **12**(3):231–272.

J. ELGOT-DRAPKIN and D. PERLIS. 1990. Reasoning situated in time I: Basic concepts. Journal of Experimental and Theoretical Artificial Intelligence, **2**(1):75–98.

G. GAZDAR. 1979. Pragmatics, Implicature, Presupposition and Logical Form. Academic Press, New York.

N. GREEN. 1990. Normal state implicature. In Proceedings of the Association for Computational Linguistics. ACL.

P. GRICE. 1989. Studies in the Way of Words. Harvard, Cambridge, MA.

B. GROSZ, A. JOSHI, and S. WEINSTEIN. 1995. Centering: A framework for modeling the local coherence of discourse. Computational Linguistics, **21**:203–226.

J. GURNEY and M. MORREAU. 1995. Presupposition and the concept of a nonmonotonic discourse. In Proceedings of the Second Dutch/German Workshop on Nonmonotonic Reasoning, Utrecht, Netherlands. University of Utrecht.

J. GURNEY, E. KLIPPLE, and C. VOSS. 1996. Talking about what we think we see: Natural language processing for a real-time virtual environment. In Proceedings of the IEEE International Joint Symposia on Intelligence and Systems, Washington, DC. IEEE.

UDO HAHN, KATJA MARKERT, and MICHAEL STRUBE. 1996. A conceptual reasoning approach to textual ellipsis. In Proceedings of the 12th European Conference on Artificial Intelligence, Budapest, Hungary. ECAI.

I. HEIM. 1982. The Semantics of Definite and Indefinite Noun Phrases. PhD thesis, University of Massachussets, Amherst, Ma.

I. HEIM. 1983. On the projection problem for presuppositions. In Proceedings of the West Coast Conference on Formal Semantics, volume 2, Palo Alto, CA. Stanford University.

I. HEIM. 1983. On the projection problem for presuppositions. In S. DAVIS, editor, Pragmatics. Oxford.

I. HEIM. 1992. Presupposition projection and the semantics of attitude verbs. Journal of Semantics, **9**:183–221.

J. HIRSCHBERG. 1991. A Theory of Scalar Implicature. Garland.

G. HIRST. 1981. Anaphora in Natural Language Understanding: A Survey. Springer-Verlag, Berlin.

H. KAMP and U. REYLE. 1993. From Discourse to Logic. Kluwer, Dordrecht, Netherlands.

L KARTUNNEN and S PETERS. 1979. Conventional implicature. In CHOON-KYU ON and DINNEEN DAVID, A., editors, Presupposition, volume 11 of Syntax and Semantics. Academic Press, Orlando.

L. KARTUNNEN. 1973. Presuppositions of compound sentences. Linguistic Inquiry, **4**:167–193.

PAUL KAY. 1992. The inheritance of presuppositions. Linguistics and Philosophy, pages 333–379.

S. LEVINSON. 1983. Pragmatics. Cambridge University Press.

DANIEL MARCU and GRAEME HIRST. 1994. An implemented formalism for computing linguistic presuppositions and existential commitments. In Proceedings of the International Workshop on Computational Semantics, pages 141–150.

S. MCROY and G HIRST. 1993. Abductive explanations of dialogue misunderstandings. In Association for Computational Linguistics, pages 277–286.

S.W MCROY and G HIRST. 1995. The repair of speech act misunderstandings by abductive inference. Computational Linguistics, **21**(4).

R. E MERCER. 1988. Solving some persistent presupposition problems. COLING, pages 420–425.

R. MERCER. 1992. Default logic and presuppositions. Journal of Semantics, **9**:223–250.

M. MILLER. Reasoning about appearance and reality. Manuscript, 1990.

R MONTAGUE. 1963. Syntactical treatments of modality, with corollaries on reflection prinnciples and finite axiomatizability. In Modal and Many-Valued Logics (Acta Philosophica Fennica, vol. 16). Academic Bookstore, Helsinki. Reprinted in R. Montague (1974). *Formal Philosophy,* New Haven, pp. 286-302.

M. NIRKHE, S. KRAUS, and D. PERLIS. 1995. Thinking takes time: A modal active-logic for reasoning *in* time. In Proceedings of BISFAI-95, Israel.

D. PERLIS. 1988. Languages with self reference II: Knowledge, belief, and modality. Artificial Intelligence, **34**:179–212.

K. PURANG, , D. PERLIS, and J. GURNEY. 1996. Active logic applied to cancellation of Gricean implicature. In AAAI Spring Symposium on Implicature.

K. PURANG, D. PERLIS, and J. GURNEY. 1996. Active logic for cancellation of implicatures. In Proceedings of the AAAI Spring Symposium, Palo Alto, CA. AAAI.

S. SOAMES. 1982. How presuppositions are inherited: A solution to the projection problem. Linguistics Inquiry, **13**:483–545.

S. SOAMES. 1989. Presuppositions. In D. GABBAY and F. GUENTHNER, editors, Handbook of Philosophical Logic, volume IV. Reidel.

R. C. STALNAKER. 1973. Presuppositions. Journal of Philosophical Logic, pages 447–457.

R THOMASON. 1980. A note on syntactical treatments of modality. Synthese, **44**:391–395.

ROB A. VAN DER SANDT. 1992. Presupposition projection as anaphora resolution. J. of Semantics, **9**:333–377.