Updating Discourse Context with Active Logic

John Gurney
Software and Intelligent Systems
Army Research Laboratory, Adelphi,
MD 20783
gurney@assb01.arl.mil

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

Abstract

In this paper we present our implementation of a system of active logic that processes natural language discourses. We focus on problems that involve presupposition and the associated well–known problems of the projection of presupposition. We discuss Heim's largely successful theory of presupposition and point out certain limitations. We then use these observations to build our discourse processor based on active logic. Our main contributions are the handling of problems that go beyond the scope of Heim's theory, especially discourses the involve cancellation of presupposition. Ongoing work suggests that conversational implicature and the cancellation of implicature can also be treated by our methods.

Keywords: presupposition, discourse, context, accommodation, active logic, implicature.

1 Introduction

Discourses that involve presupposition present important challenges for research into the computational treatment of the interplay between discourse context and the flow of utterances that comprise a discourse.

Traditional studies of presupposition [2, 7, 15] have focused on characterizing the "right" final context resulting from a discourse while paying less attention to a realistic model of the intermediate processes involved in achieving the final context (or final interpretation of what was said). We think that there is much to be gained from a more fine-grained approach, modeling the underlying step-by-step reasoning a listener may perform during the unfolding of an ongoing conversation. Active logic [13, 1] seems to us to be a potentially powerful tool for modeling such behavior, allowing contradictory beliefs to trigger *reasoned* belief revision.

In this paper we will examine a well–known theory of context updating proposed by Heim [5]. This theory is largely successful in accounting for much of the complexity of processing discourses that involve presupposition. We will point to some problems with theory. Following this we will present our implementation of an active logic system that is based largely on Heim's theory but uses the unique machinery of active logic to deal with cases of presupposition cancellation and other related problems. The complexity of discourse context updating can be demonstrated by considering some of the canonical examples of presuppositional discourses.

1.1 Canonical Examples of Presupposition Discourses

The presupposition phenomena we will discuss occur in the following discourses, which we will represent as sequences of utterances like this:
$D_n = \langle u_1, u_2, u_3$, etc. $\rangle$.
We begin with a straightforward example which we assume is thoroughly unproblematic.
$D_1 = \langle$ There are roses and tulips. But the roses are not yellow. $\rangle$
Here the second utterance u_2 presupposes that there are roses. That is, even though u_2 is a negation, the speaker is presupposing that there are roses to which he is referring. He means only to say something about the color of those roses. The presupposition arises from the speaker's use of a definite description "the roses". Now since u_1 has already introduced roses into the discourse context, the presupposition "there are roses" is unproblematic. We will say that u_1 makes u_2 felicitous.

Now consider a more interesting, less straightforward discourse.
$D_2 = \langle$ There is no king. So the king is not in hiding. $\rangle$
Here u_2 has the same essential form as the u_2 of the previous discourse; there is a definite description "the king" which could (in some cases) suggest a presupposition that there is a king. However, in this case, u_1 did not introduce a king into the discourse context; rather u_1 asserts there is no king. So the context now entails that there is no king. Yet u_2 is felicitous. In this case the definite description does not ultimately project a presupposition.

The next discourse shows a different way that context influences presupposition.

$D_3 = \langle$ If John has a son, then his son is not here.$\rangle$
Here the possessive noun phrase "his son" in u_2 gives us the potential presupposition that John has a son. But in this context (as the consequent of the if/then sentence) that presupposition does not project. That is, this single utterance discourse has no presuppositions about the existence of John's sons.

Note that the next, similar, discourse *does* presuppose that John has a son.
$D_4 = \langle$ If John has a car, then his son is not here. $\rangle$

The next discourse displays what appears to be yet another way that a potential presupposition can be stopped from projecting.
$D_5 = \langle$ If I discover that Bill is in New York there will be trouble. $\rangle$
Here the factive verb phrase "discover that Bill is in New York" has the potential presupposition that Bill is in New York. This becomes apparent when we compare D_5 to D_6. Yet there is no presupposition in D_5.

$D_6 = \langle$ If John discovers that Bill is in New York there will be trouble. $\rangle$
Here the potential presupposition that Bill is in New York survives. It projects up to sentence level and the speaker is making that presupposition.

These examples D_1 through D_6 are among the canonical cases that any theory of presupposition and context should predict. It is clear that presuppositional behavior is complex — even for relatively simple discourses. Apparently, certain syntactic forms (such as definite descriptions, possessives, and factive verbs) give rise to potential presuppositions which may or may not produce a presupposition in the total discourse.[1] One of the best theories designed to account for this complexity was proposed by Heim [5] in the form of three rules for incrementally updating discourse context. These rules account for all of the presupposition facts right in the above examples. We will discuss Heim's rules in the section 2.

These are the rules that we will use as a basis for our implementation of discourse context updating which we will present in section 4. We will motivate our implementation as one which (a) preserves the correct results of Heim and also (b) deals with other examples such as D_7, D_8, and D_9 below that Heim's rules do not account for. These three examples are superficially similar to those above. However they present problems for a computational theory of presupposition and context that have not been widely discussed (but see the section 6 where we mention [9]).

[1]For our purposes here we will only work with examples where presupposition is caused by the use of a definite description. What we say could also be said for some other sources of presupposition, including factive verbs as in D_4, possessives as in D_3 and cleft sentences.

$D_7 = \langle$ The King is not in hiding, Because there is no king. $\rangle$
Here after u_1 there is a presupposition that there is a king. Then after u_2 that presupposition is *withdrawn*. We call this "presupposition cancellation". This phenomenon is essentially linked to the flow of time in discourse. Heim's rules (as well as many other theories) do not address this phenomenon. They deal only with cases where potential presuppositions may not project due to the existing context – cases like D_2 above. They do not distinguish these cases from those where a presupposition that was projected previously must later be withdrawn. We will call these cases "garden path discourses". First the hearer goes down a path assuming some presupposition; then he finds that he shouldn't have. Now he must retract that assumption. This is a place where discourse updating is nonmonotonic.

The next discourse invokes *two* potential presuppositions.
$D_8 = \langle$ There are no roses. So the roses are not in the fridge. $\rangle$
Here, u_2 creates two potential presuppositions "there are roses" and "there is a fridge". But only the latter projects because the context after u_1 entails that there are no roses. It turns out that Heim's rules cannot account for this example – even though this is not a garden path discourse. We will examine this case below in section 4.

Here is the garden path version of D_8.
$D_9 = \langle$ The roses are not in the fridge. Because there are no roses. $\rangle$

We will use this discourse to demonstrate our system for discourse processing in section 4.

2 The Theory of Context Updating

Heim's theory of context updating has two parts:

(1) a set of update rules

(2) a mechanism for accommodation of presuppositions (which we will explain below).

2.1 Heim's Rules

Heim's rules for updating a discourse context are based on a function called +. This is partial function that takes as input a discourse context c an utterance u. It outputs a new context c'. Although we have no rigorous specification of what makes up a discourse context we can say that it is a set of propositions (or one inclusive proposition) that should include the content of what is mutually accepted by the discourse participants.

Many of the complex presupposition projection cases, including the discourses D_1 through D_6, are predicted by Heim's three rules of context updating, the Context Change Potential Rules, the CCPs, to which we have added a basis rule CCPB.

CCPA: $c + (u \text{ and } v) = (c + u) + v$. [conjunction]
CCPN: $c + (\text{not } u) = c \setminus (c + u)$.[2] [negation]
CCPC: $c + (\text{if } u \text{ then } v) = c \setminus ((c + u) \setminus ((c + u) + v))$. [conditional]
CCPB: $c + u = c \cap [[u]]$. [atomic basis]
The first three rules are rewrite rules for complex utterances. Repeated application of these rules will reduce a complex utterance to a formula containing atomic utterances where rule CCPB can be applied. These rules account for the complexity of presupposition by systematically reducing complex presupposition problems to simple ones.

In the theory, propositions are taken to be sets of possible worlds, while the utterances that give rise to and effect these propositions are tokens of natural language. In our implementation (in section 4) we will represent contexts and propositions syntactically (as logical formulas) but for the present we will stay with the possible world interpretations. So adding a new proposition to a context amounts to the intersection of two sets of possible worlds. This accounts for the notation in the basis rule CCPB which says that for a simple utterance take its propositional content [[u]] and intersect that set of possible worlds with the context c. Updating reduces the size of the set of possible worlds that constitute the evolving context.

The first rule CCPA is just what we would expect for straightforward discourses like D_1 above. Think of the two utterances of that discourse as forming a conjunct u_1 *and* u_2. First we add the first conjunct to the existing context to produce a new context. Then we add the second.

Rule CCPN is an analog of a rule for logical conjunction $[[P \wedge \neg Q]] = [[P]] \setminus [[Q]]$. In effect, Heim can be assumed to be proposing that, for discourse updating where the utterance is a negation, *not u*, we do the following: (i) Start with the context c and then imagine that the discourse had been u (the positive form) rather than the actual negation not u — but in the same context. Compute the updating of this imagined discourse. Next (ii) subtract the final proposition that processing this imagined discourse would yield from the initial context.

A naive rule for updating contexts with negated utterances such as
BADN: $c + (\text{not } u) = c \setminus [[u]]$
cannot be made to work properly. This rule can account for D_1 but it cannot account for D_2. On the other hand CCPN can account for both of these discourses – as we will explain below.

The third rule CCPC can be derived from CCPA and CCPN.

Now take discourse:

[2] In general, $p \setminus q$ is the set theoretic intersection of p with the complement of q, i.e., the subtraction of q from p.

$D_1 = \langle$ There are roses and tulips. But the roses are not yellow. $\rangle$ First we apply rule CCPA to update the context with u_1.
$c_2 = (c_1 + (\text{ there are roses})) + (\text{there are tulips})$
Then we apply rule CCPN to update the new context with u_2.
$c_3 = c_2 \setminus (c_2 + (\text{the roses are yellow}))$.
But u_2 has a presupposition, namely, [[there are roses]]. This plays a role in these rules because the function + is subject to the restriction that $c + u$ is undefined unless c is a subset of (that is, entails) every presupposition of u. Thus the context for an utterance does, in fact, entail all presuppositions of the utterance. In the present case, we can see that c_2 must entail [[there are roses]]. Therefore, the updating can proceed.

2.2 Accommodation

Context updating becomes interesting when we consider discourses where $c + u$ is undefined but where things can be made right through adjustments to the existing context. For example, assume that the initial context is something irrelevant like
$c_1 = [[\text{grass is green}]]$[3]

Assume a very simple discourse like
$D = \langle$ John's son is here. $\rangle$
The applicable rule is CCPA
$c_2 = c_1 + u_1$
But the use of + would be undefined here because c does not entail [[John has a son]] which is the presupposition of u_1. On the other hand, it seems that u_1 should be perfectly understandable in this context. The story of the reason why goes like this: The person attempting to understand this discourse can easily just assume that c_1 should have entailed that John has a son. When a hearer makes such an assumption he is "accommodating" the speaker [16]. In this case the hearer accommodates to transform c_1 into
$c_1' = c_1 \cap [[\text{John has a son}]]$,
thus making the updating defined after all. As a default, accommodation is permitted when required to make a + operation in a CCP rule defined (as it was in the above discourse D). Clearly accommodation is another source of nonmontonicity in discourse updating.

2.3 Global and Local Accommodation

Accommodation is also, however, a source of trouble for the theory of context updating. The trouble is that, just as presupposition projection seems to be a complex function of various kinds of context, accommodation is also variable. While the empirical facts of the matter are fairly clear, just when and how accommodation can be applied is difficult to specify.

Consider the discourse:

[3] We choose [[Grass is green]] as a stand in for a context that is irrelevant to the discourse.

$D_2 = \langle$ There is no king, So the king is not in hiding. $\rangle$
Here, after c_1 is updated with u_1, we have a c_2 that entails [[there is no king]]. This is interesting because u_2 has [[there is a king]] as a potential presupposition! But we cannot simply accommodate by adding [[there is a king]] to c_2! That would make the context a contradiction.

However, recall that the applicable rule CCPN analyses the updating process as follows.
c_2 + not (x is the king and x is in hiding) =
$c_2 \setminus ((c_2$ + x is the king) + x is in hiding).[4]
Here accommodation is required because the first + operation to the right of the back slash is not defined; c_2 does not entail the presupposition [[there is a king]]. If we accommodate as in the previous discourse we would back up and add [[there is a king]] to the original c_2. We would then go forward again using $c_2{}'$ in place of c_2. This type of accommodation is known in the theory as global accommodation. However this would yield a contradiction in the final updated context.

As a way out, Heim suggests using what she calls local accommodation [5]. Rather than back up to accommodated c_2 and then go forward again to update with u_2, we accommodate only the instance of c_2 in the imagined discourse, to the right of the back slash.
$c_3 = c_2 \setminus ((c_2{}'$ + x is the king) + x is in hiding).
Here c_2 entails [[there is no king]] and $c_2{}'$ entails [[there is a king]] as well as [[there is no king]]. Perhaps it is worrisome that $c_2{}'$ is a contradiction. On the other hand it *does*, of course, make the first + here defined. And since this contradiction appears to the right of a back slash, it will have no effect on that context c_2 to the left. This procedure ultimately leaves the proposition
[[there is no king]]
as the final interpretation of discourse D_2 – which we take to be correct.

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 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 produce the correct result for D_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.

We take (2) to be the more serious problem. The discourse
$D_6 = \langle$ There are no roses, So the roses are not in the fridge $\rangle$
in which there are two potential presuppositions is a case in point.

A speaker of this discourse could reasonably be taken to be asserting that there are no roses while also presupposing that there is a fridge. The final context c_3 should entail both [[there are no roses]] and [[there is a fridge]]. The other potential presupposition [[there are roses]] should not project. After processing u_1 we have this updating calculation to perform.
$c_3 = c_2 \setminus ((((c_2{}'$ + x are the roses) + y is the fridge) + in(x, y)).
$c_2{}'$ is the accommodated c_2 which is now a contradiction, so the first + is defined. But the second +, which will update with [[y is the fridge]] is now also defined. So there is clearly no need for any further accommodation. That means that we never have to accommodate the instance of c_2 to the left of the back slash. As before, when all the + ing is over we have simply
$c_3 = c_2 \setminus$ [[contradiction]] $= c_2$.
So the potential presupposition [[there is a fridge]] never makes it into the final context (although, of course, it should).

Our diagnosis 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 the theory accounts for nonmonotonic facts 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.

2.5 Garden Path Discourses and Cancellation

We have used discourse D_6 to uncover a subtle problem with, what we think, is one of the best theories of context updating. Recall that D_6 was not a garden path discourse; it was very similar to the canonical examples of the set of presupposition problem cases that any theory of presupposition and context updating should account for. On the other hand, the discourse

[4] From this point forward we follow Heim by using free variable syntactic forms of sentences.

D_7 = ⟨ The King is not in hiding, Because there is no king. ⟩
is a garden path discourse. Given our previous discussion, it should be clear that Heim's system will not handle this or any other garden path discourse without modification. These discourses first posit a presupposition and then withdraw it. It may be conjectured that there is an easy fix – using backup and restart operations that could handle these garden paths. The reason we discussed the subtle problem with the non garden path D_6 first was to suggest that there may be no such easy fix for all cases.

3 Active Logic Compared to Nonmonotonic Logic

Active logic [13, 1] is a family of formalisms developed for the purpose of 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.

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 way.

In this paper we will present a treatment of the "fridge and roses" problem D_9, 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 formulae that represent the discourse context, that is, the record of what has been said up to the current step. At step n the information state might look something like

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

[...] is an ordered list of logical formulae of 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.

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. For example, just after u_1 of discourse D_1 we would have something like update([c_1, assert(+, assert(exists(x, roses(x))), +, assert(exists(x, tulips(x))))], 3).[5]

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.

[5]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.

1. i: ut('X', i)
   i+1: parse(Y, i+1)

   where Y 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.

2. i: ctxt(C, i) parse(X, i)
   i+1: update(Z, i+1)

   This is where a syntactic parse of an utterance gets translated into a form ready for the application of Heim's CCP rules. For example, given ctxt(c_1, 1) and parse(and(dfnt(roses(x)), red(x)), 1) at time 1, we get update([c_1, +, dfnt(roses(x), +, red(x)], 2) at time step 2.

3. i: update(X, i)
   i+1: update(Y, i+1)

   where Y is the result of executing 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.

4. i: update(X, i)
   i+1: ctxt(X, i+1)

   This rule is a sub case of the previous and is applied when all context updating is complete for one particular utterance. Once the update is complete, the new context is put back into the set of beliefs of the system.

5. 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 decides which member of the pair can be inherited to the next step.

6. i: ctxt([..., kill(foo(X), ...,
   kill(bar(not(Y))), ...,
   contra(foo(X), bar(not(Y)))], 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.[6] In our system, an assertion is always preferred for inheritance over a presupposition.

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 while choosing which contradictand to kill.

4.3 Output Trace for Discourse D_1

We now present some of the steps of the output trace for D_1. Some details are not shown, for example the argument representing time in the predicates.

D_1 = ⟨There are roses and tulips. 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 = [].[7]

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.

[6] See Miller [13] for more on contradiction resolution in active logic.

[7] 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.

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

This step readies the information from the utterance for application of the CCP rules.

7 c_3

At the end of processing the first utterance, the context contains the assertions that there are both roses and tulips in the discourse context. We now 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 sets things up for the application the rule for negation CCPN. Since there is a definite description of roses, the system first looks for exists(y, R(y)) in c_3 which does in fact include it. Thus updating can proceed normally.

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

Here everything from u_2 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)))))

4.4 Output Trace for Discourse D_7

Below we will display some of the output from our system processing D_7 This is the garden path version of the discourse involving two potential presuppositions that we discussed earlier.

D_7 = ⟨The roses are not in the fridge. Because there are no roses.⟩

Here we have a case where something 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 nor its cousin D_6. 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 accommodated globally. The significance of this will become clear where we discuss steps 3 and 4 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 D_6 as well as D_7.

For D_7 we can see that Heim's strategy of choosing local accommodation to avoid global contradiction (which worked for D_2) is not even applicable. After u_1 the context should contain two presuppositions, that roses exist and that a fridge exists. Then, after u_2, 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 renders the parse of u_1 into the proper form for the application of the CCP rules.

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

The next applicable rule is CCPA which applies to
[c_1, +, dfnt(R(y)]
at step 2. 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. Thus c_2' here at step 3 includes the information 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. We don't have to worry about using local accommodation (in which only the instance of c_2 after the backslash would be accommodated) because our system can eliminate contradictions (in a controlled way). Of course, in this example, Heim's system would also have used global accommodation because there was no threat of a contradiction arising.

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 u_2 which should cancel one of the presuppositions in the current context. Since u_2 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 u_2 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_7, 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.

5 Conversational Implicature

We believe that presupposition and conversational implicature [3], illustrated by the two examples below, are distinct discourse phenomena. This is a point we will develop on another occasion. However, both phenomena are nonmonotonic in that both can be either blocked or cancelled as a discourse progresses. We have modelled and implemented this aspect of implicature using active logic. In this section we will briefly discuss the current state of our implementation.

Using a simple example of a dialog with an implicature that arises part way through and then is later retracted, we have modelled in active logic how Gricean maxims and nonmonotonicity may relate to each other and to a computational treatment of implicature. In effect we seek to track reasoning along Gricean lines over time.

In this work we wish to seriously consider how cancellation of implicatures might work and how to implement the actual positing and withdrawal of implicatures *in real time*. Our hypothesis is that the same underlying framework of active logic that we have applied to *presuppositional* inference in real–time (evolving) dialog–processing also is applicable to inference of *implicatures*.

We have been concerned with two dialogs that require the hearer, Kathy, to figure out an implicature in order to realize the import of the speaker's answer to her questions. In the first example Bill has given an indirect answer to her question followed by an explicit cancellation.

(A) Kathy: Are the roses fresh?
(B) Bill: They are in the fridge.
(C) Bill: But they're not fresh.

In the first example Bill has given an indirect answer to her question followed by an indirect cancellation.

(A) Kathy: Are the roses fresh?
(B) Bill: They are in the fridge.
(D) Bill: But they are old.

For our implementation we will appeal to three of Grice's maxims. These are usually stated as rules for a speaker in a cooperative conversation. They come into play in discourse processing when the hearer makes essential use of one or more of the maxims in his reasoning about what the speaker means by an utterance. That is, the hearer in some way assumes the speaker is adhering to a maxim an uses that assumption to figure out something that should follow from an utterance. We have modelled how this works for certain yes–no dialogs like those above.

The maxims that play a part in out system are the maxims of quality, Quantity, and Relevance:

The Maxim of Quality: Always make your contribution to the conversation truthful; or don't say something for which you do not have adequate evidence.

The Maxim of Quantity: Always convey as much and no more than is required for the purposes of the conversation.

The Maxim of Relevance: Always make your contribution relevant to the purposes of the conversation.

In our implementation which we discuss in more detail in [4], we have not represented any of the Gricean maxims explicitly. We regard them as specifications for building a discourse participant. Each of our discourse rules articulates one or more of the maxims. The system consists of (a) rules representing three kinds of knowledge and (b) an inference procedure that applies these rules repeatedly taking us from one step to the next. The three kinds of knowledge are: the active logic meta–theory, general beliefs about dialogs involving questions and answers, and background beliefs about refrigerators, roses, food, and so on.

The meta–theory rules say things like: *If there is a contradiction don't inherit either alternative*, *Otherwise inherit anything that you can*, and *Update the time by 1 from one step to the next*. The discourse rules say things like *Believe anything that the speaker informs you*, *Believe any direct response to your question*, and *Try to figure out what an indirect response to your question means*. The background beliefs say things like *Things in fridges are cold*, *Cold roses are fresh*, and *Things in fridges are edible*. These last three beliefs happen to be defeasible; the rules will only fire if the right hand sides of these rules cannot be proven false.

The relation of our rules to Grice's maxims can be explained as follows: The rules about believing the content of any utterance implement the maxim of quality. The rule dealing with indirect responses relates to both the maxim of quantity and the maxim of relevance. For it is this rule that produces the relevance beliefs like rel(fresh(r1)):infridge(r1). These are the beliefs that begin a search for relevant rules to fire that may lead to an answer to one's question. In our model for the first example (above) Kathy begins looking for and answer immediately after hearing (B) *They are in the fridge*. There is no waiting to hear what comes next. We can view this as following the maxim of quantity; Kathy assumes for the moment that Bill has said all that is relevant. From this she infers the implicature that the roses are fresh; that is she thinks that Bill has answered her question at (B). But no harm was done, since implicatures can be cancelled. Our system performs this kind of generation and, where appropriate, cancellation of implicature.

6 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 as happens in D_5 and D_6. We have chosen to study and adopt Heim's theory because it covers many of the problematic 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 [12] and Marcu and Hirst [9].

Mercer employs a system of default rules to model the presuppositions arising from syntactic forms that appear in utterances. In [12] he deals with adverbial implicatures 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 [9] 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_7 by first computing the two presuppositions after u_1. Then, after u_2 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 [2, 5, 6, 14, 8] 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 [11, 10] presents 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. Moreover, contrary to [11], we do not need to assume there are 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. As shown in our output traces in section 4 and in Miller [13], active logic engines are often able to reason quite effectively with contradictions. It is that fact that provides the underlying framework that we are exploiting.

7 Conclusion

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.

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 the 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 facility treatment of both of 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 Betsy Klipple, Jean Braithwaite, Michael Miller, and Michael Morreau for helpful discussion.

References

[1] J. Elgot-Drapkin and D. Perlis. Reasoning situated in time I: Basic concepts. *Journal of Experimental and Theoretical Artificial Intelligence*, 2(1):75–98, 1990.

[2] G. Gazdar. *Pragmatics, Implicature, Presupposition and Logical Form*. Academic Press, New York, 1979.

[3] P. Grice. *Studies in the Way of Words*, chapter Presupposition and Conversational Implicature. Harvard, Cambridge, Ma., 1989.

[4] J. Gurney, D. Perlis, and K. Purang. Active logic applied to cancellation of gricean implicature. AAAI Spring Symposium on Implicature, 1996.

[5] I. Heim. On the projection problem for presuppositions. In S. Davis, editor, *Pragmatics*. Oxford, 1983.

[6] L. Kartunnen. Presuppositions of compound sentences. *Linguistic Inquiry*, 4:167–193, 1973.

[7] L Kartunnen and S Peters. Conventional implicature. In Choon-Kyu On and Dinneen David, A., editors, *Presupposition*, volume 11 of *Syntax and Semantics*. Academic Press, Orlando, 1979.

[8] Paul Kay. The inheritance of presuppositions. *Linguistics and Philosophy*, pages 333–379, 1992.

[9] Daniel Marcu and Graeme Hirst. An implemented formalism for computing linguistic presuppositions and existential commitments. pages 141–150. International Workshop on Computational Semantics, 1994.

[10] S. McRoy and G Hirst. Abductive explanations of dialogue misunderstandings. pages 277–286. Association for Computational Linguistics, 1993.

[11] S.W McRoy and G Hirst. The repair of speech act misunderstandings by abductive inference. *Computational Linguistics*, 21(4), 1995.

[12] R. E Mercer. Solving some persistent presupposition problems. *COLING*, pages 420–425, 1988.

[13] M. Miller. *A view of one's past and other aspects of reasoned change in belief*. PhD thesis, Department of Computer Science, University of Maryland, College Park, Maryland, 1993. (Directed by D. Perlis.).

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

[15] S. Soames. *Handbook of Philosophical Logic*, volume IV, chapter Presuppositions. Reidel, 1989.

[16] R. C. Stalnaker. Presuppositions. *Journal of Philosophical Logic*, pages 447–457, 1973.