A NEAT[1] THEORY OF MARKER PASSING

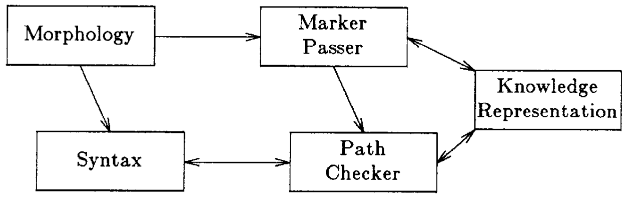
Eugene Charniak
Department of Computer Science
Brown University
Providence, Rhode Island 02912

Abstract

We describe here the theory behind the language comprehension program Wimp. Wimp understands by first finding paths between the open-class words in a sentence using a marker passing, or spreading-activation, technique. This paper is primarily concerned with the "meaning" (or interpretation) of such paths. We argue that they are best thought of as backbones of proofs that the terms (words) at either end of the paths exist in the story and show how viewing paths in this way naturally leads to the kinds of inferences which are normally thought to characterize "understanding." In a companion paper we show how this interpretation also accomplishes much of the work normally expected in the parsing of language (noun-phrase reference, word-sense disambiguation, etc) so we only briefly touch on this topic here. Wimp has been implemented and works on all of the examples herein.

I Introduction

This paper describes Wimp (Wholy Integrated Marker Passer), a program which understands simple stories in English. Wimp uses incoming words (in particular the open-class words) as input to a *marker passer* which finds connections between these words. These connections, or *paths* go to a *path checker* which makes sure that the paths "make sense" and extracts from them the facts which are needed to plausibly claim that the input has been "understood." (In particular we concentrate on questions of character motivation and causality.) The overall structure of Wimp is this:



To summarize how Wimp works, consider its operation on "Jack went to the supermarket" We simplify by assuming that Wimp is given some preparsed internal representation. (In section 5 we briefly consider how Wimp works when it starts directly off the input English.) The pre-parsed version looks like this:

```
(inst go1 go)                    ; There is a going event go1
(= (agent go1) jack1)            ; for which jack1 is the agent
(= (destination go1) smarket1); and smarket1 is the destination.
(name jack1 jack)                ; Jack1 has the name "jack"
(inst smarket1 smarket)          ; and smarket1 is a supermarket.
```

In line with previous work on story understanding and recognition of speaker intention [Sc77,Sc78,Wi78,Pe80,Wo81] we assume that a minimum "understanding" of the sentence would include the fact that Jack will be shopping at the supermarket mentioned in the sentence. Thus the result of the understanding process should be the addition of the following to the database.

```
(inst super-shop1 smarket-shopping)      ; A supermarket shopping
(= (agent super-shop1) jack1)            ; event for which jack1 is the
(= (store-of super-shop1) smarket1)      ; agent and smarket1 is the
(= (go-step super-shop1) go1)            ; store is the reason for going.
```

We call these the ***abductive assumptions*** for the sentence. Wimp makes these abductive assumptions because it finds a path between "went" and "supermarket" which goes through **smarket-shopping**. The Wimp's path checker considers this path to be a "proof" of the fact that "going" and "supermarkets" appear in the story. However, to make this proof go through it must make some assumptions – the abductive assumptions listed above. For Wimp to "believe" a path simply means to believe the abductive assumptions which are required for the path's proof to go through, and thus the assumptions are added to the database.

Wimp is related to several strands of work within AI, the most obvious being the language comprehension work of [Gr84,Al85], and [No86]. All three of these system use a marker passer to find paths which are then evaluated in some fashion to produce the inferences required for story comprehension. (This is also the model suggested in [Ch83].) The major differences between the work reported on here and these models are a) the current work, but not the others provides a formal basis for evaluating paths, and b) the current work uses the path finding and evaluation process not just for finding important inferences, but also for all the aspects of language parsing which require semantic or real-world knowledge. In this later regard it is somewhat like the work of [Ri85] and the early work of [Qu69]. Less obviously, Wimp is related to the ***resolution residues*** of [Ge84] in the use of resolution to produce explanations, and to ***connection graphs*** of [Ko75] in the use of graphs over first order formulas to find proofs. Lastly since Wimp (among other things) tries to determine character's plans, it is also related to work which has been done on this, such as [Wi78,Pe80,Wo81] although Wimp uses quite different methods.

II Marker Passing

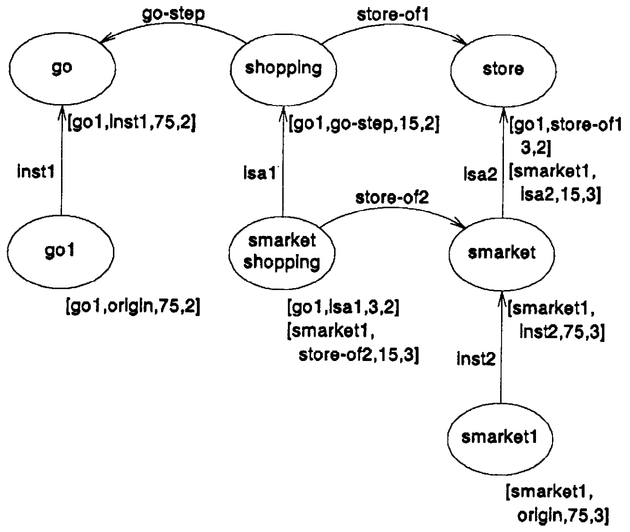
We shall assume a database of first-order predicate-calculus formulas indexed by the terms which appear in them. (This is what our knowledge representation language Frail (FRame-based AI Language) [Ch82] gives us.) So terms have pointers to all formulas in which they appear, and the formulas point to their other terms. Thus we have a network where nodes are terms, and links are first-order formulas. For example:

```
(if (and (inst ?x smarket-shopping)  ; The store-of a
         (= (store-of ?x) ?str))     ; supermarket-shopping is
    (inst ?str smarket))             ;  a supermarket.
```

(In Frail this rule would actually look like **(inst (store-of ?x:smarket-shopping) smarket)** where **?x:smarket-shopping** says that **?x** can be bound to any instance of a **smarket-shopping**, and the equality would be handled automatically. We ignore this here and use the more bulky traditional representation for such facts.) This rule would

[1] "Neat" here is as opposed to "scruffy" – Abelson's terms for the two styles of cognitive science research. This work has benefited from conversations with Lawrence Birnbaum, Tom Dean, Jim Hendler, and Drew McDermott, and has been supported in part by the National Science Foundation under grants IST 8416034 and IST 8515005 and Office of Naval Research under grant N00014-79-C-0529.

form a link between the tokens **smarket-shopping** and **smarket** and would be part of the path which Wimp uses to establish Jack's motivation for his action. Until section 5 we assume that the input comes pre-parsed, so Wimp passes marks from each predicate calculus term as it comes in. Marks are 5-tuples, containing the origin of the mark, the node and link from whence it came, the "date" of creation, and a positive integer called "zorch" which is a rough measure of the "strength" of the mark. When marks are passed from a new origin there is a *initial zorch* and each time a mark is passed from one node to the next the zorch is divided by the branching factor at the node just left. The zorch still available is recorded in the mark at the new node. Should it fall below 1 no further marks are propagated. Here we show what a portion of a Frail database would like like after begin marked from **go1** at time 2 and **smarket1** at time 3 with an initial zorch of 75. (Prior nodes were deleted from marks to improve legibility.)



The date part of the mark allows marks to "decay" exponentially over time. Dates are measured by how many times the marker passer has been called to pass marks from a new origin. After a certain half life (currently 4) the zorch at a node is divided in half. Should this cause it to fall below 1 the mark is removed. If in the course of marking a node is found with marks from origins other than the one from which marks are currently flowing the marker passer reports an intersection and the link portion of the mark is used to reconstruct the path from the two origins. Zorch on marks also allows for a crude indicator path "strength." This *path zorch* is the zorch on the mark, times the zorch of the incoming mark, divided by the branching factor on the node where they meet. This is, in fact equal to the following:

$$\textit{path-zorch} = \frac{\textit{initial-zorch}^2}{\prod_{i=1}^{i=n} \textit{brach}_i}$$

where $\textit{brach}_i$ is the branching factor at the *i*th node in the path. Paths are reported back in decreasing path-zorch order. In the example network the marker passer would find two paths, of which we concentrate on this one:

(go1 inst1 go gostep shopping isa1 smarket-shopping store-of2 smarket inst2 smarket1)

The atomic names on links are the names of formulas, the content of which we describe later.

III Path Checking and the Meaning of Paths

We said that a path is the backbone of a proof that the terms at either end exist in the story. To be a bit more precise, it is a proof that the **inst** statement associated with the term is true. For the supermarket we want to prove **(inst smarket1 smarket)** is true. It may not be obvious why this is a reasonable thing to do, so an explanation is in order.

A standard platitude is that understanding something is relating it to what one already knows. The exact nature of such "relating" is not obvious, but one extreme example would be to prove that what one is told must be true on the basis of what one already knows. To a first approximation that is the view we take here, but qualified in two important ways. First, most of what one hears (e.g., "Jack went to the supermarket.") is new information, and thus not deducible from previous knowledge. Rather, we want to prove what one is told *given certain assumptions*. That is, Wimp's path checker tries to create a conditional proof, where the conditions are the abductive assumptions.

The second constraint on our "proof" comes from the marker passer. Since Wimp is designed to help with parsing, it must work prior to syntactic analysis, and therefore works directly off the words. Thus there is no possibility of passing marks from the propositional content of the sentence – only from terms denoted by words in the content. To do otherwise would require postponing marker passing until after parsing. Furthermore marker passing requires having a pre-existing network. Thus to pass marks based upon the propositional content, from, say, "Jack went to the supermarket" to "Jack is at the supermarket" would require that such facts already be in the database, which they are not.

Therefore, if our "proof" is to prove anything, it can only prove the **inst** propositions which started it all, since those are the only ones which can be deduced directly off the incoming terms (or later words). Thus the path checker tries to prove the terms at the ends of the paths and uses the path as the backbone of the proof. In general the proof is a conditional one where the conditions are the abductive assumptions we have been talking about. One major problem in all of this is that if we are allowed to make assumptions we can prove anything at all (if only by assuming something false). Therefore there must be constraints on the assumptions we allow, a topic which we discuss shortly.

First let us make more precise the idea of treating the path as the backbone of a proof. The easiest way to do this is to treat this as a proof by contradiction using resolution. We therefore assume that all of our formulas are in conjunctive normal form, and the proof starts out by negating the conjunction of the **inst** formulas at either end. Wimp does not actually use resolution, but it is close, and it is easiest to see Wimp from that vantage point. Starting from the negated **inst** conjunction each formula in the path is resolved against the remaining disjuncts, starting with the formulas at the two ends, and working toward the middle. For reasons discussed later, Wimp may clash against the converse of the formula in the path. For the moment we ignore this.

The resolvents in the procedure are those (and only those) found in the path. Thus there is no combinatorial explosion since there is no search for resolvents. (The search is effectively done by the marker passer.)

Let us consider the example of the path between **go1** and **smarket1** shown in the example network. The formulas used are these:

```
go-step    ¬(inst ?shp shopping) V
               ¬(= (go-step ?shp) ?go) V (inst ?go go)
isa1       ¬(inst ?shp smarket-shopping) V (inst ?shp shopping)
store-of2  ¬(inst ?shp smarket-shopping) V
               ¬(= (store-of ?shp) ?str) V (inst ?str smarket)
```

We then start with

¬(inst go1 go) V ¬(inst smarket1 smarket)

This is then unified with go-step giving

¬(inst smarket1 smarket) V ¬(inst ?shp shopping) V
¬(= (go-step ?shp) go1)

We then resolve this against isa1 giving

¬(inst smarket1 smarket) V ¬(inst ?shp smarket-shopping) V
¬(= (go-step ?shp) go1)

This is resolved against store-of2 to give us:

¬(inst ?shp smarket-shopping) V ¬(= (go-step ?shp) go1) V
¬(= (store-of ?shp) smarket1)

At this point we have resolved against everything in the path. If this is a path that the system chooses to believe (more on why and how particular paths are chosen later) then the system must renegate the remaining clauses, and add them to the database.

Note that in renegating the clauses the universally quantified variable ?shp gets flipped to an existentially quantified variable, which then is turned into a skolem constant, giving this:

(inst supershop1 smarket-shopping)
(= (go-step supershop1) go1)
(= (store-of supershop1) smarket1)

These are, of course, just the abductive assumptions we suggested would be the minimum for claiming to understand the sentence.

There are two complications to clear up before we move on to judging the relative "believability" of paths as interpreted by the proof process. First, note that some of the abductive assumptions left by the procedure could already be provable from the database.. For example, if we already knew that Jack had a plan to shop at the supermarket, say supershop-22 then one of the clauses left at the end of the path proof could be removed by clashing it against (inst supershop-22 smarket-shopping). This, of course, changes the remaining clauses (which still become abductive assumptions), and they would now become:

(= (go-step supershop22) go1)
(= (store-of supershop22) smarket1)

That is, the going and the supermarket would be linked to the previously known supermarket shopping plan, rather than a newly minted one. In fact, Wimp tries to prove all of the abductive assumptions, and when this is possible it creates alternative versions of the path proof, one for each way of binding variables in the proofs, plus one where the initial abductive assumption is left unproved. Which of the alternatives is actually believed is decided by the mechanisms described in the next section.

The last thing which needs clearing up is why we sometimes resolve against the converse of the formula in the path. To see how this could arise, consider this path from smarket1 to milk1

```
store-of2     ¬(inst ?shp smarket-shopping) V
                 ¬(= (store-of ?shp) ?str) V (inst ?str smarket)
purchased2    ¬(inst ?shp smarket-shopping) V
                 ¬(= (purchased ?fd) ?str) V (inst ?fd food)
isa3          ¬(inst ?mlk milk) V (inst ?mlk food)
```

Intuitively this corresponds to the chain of reasoning that supermarket shopping predicts the existence of supermarkets (as the store) and food (as the purchased) and milk is food. When we try to apply the path-proof procedure we get stuck after this:

```
¬(inst smarket1 smarket) V ¬(inst milk1 milk) ; initial disjunction
                                               ;
¬(inst ?shp smarket-shopping) V                ; after store-of2
   ¬(inst milk1 milk) V ¬(= (store-of ?shp) smarket1)
```

At this point nothing else can resolve. The problem is that the formulas are not sufficient to prove (inst milk1 milk) but at best only (inst milk1 food) since shopping at the supermarket does predict that there is food involved, but not necessarily milk. (Another example of this would be the path from "restaurant" to "hamburger" in "Jack went to the restaurant. He decided on a hamburger.") Suppose we resolve against the converse of isa3.

(inst ?mlk milk) V ¬(inst ?mlk food)

Then we can carry this forward as follows:

```
¬(inst ?shp smarket-shopping) V  ; after converse of isa3
   ¬(inst milk1 food) V ¬(= (store-of ?shp) smarket1)

¬(inst ?shp smarket-shopping) V  ; after purchased2
  V ¬(= (store-of ?shp) smarket1)
  V ¬(= (purchased ?shp) milk1)
```

We get similar assumptions, although using a converse counts as an extra.

IV Selecting the Best Paths

We have seen how paths can be interpreted as proofs, and how the assumptions needed to make the proofs go through are the abductive assumptions required by story comprehension. We also noted that some of the paths are "believed" which means adding their abductive assumptions to the database. We now look at how the "believable" subset of the paths is singled out.

Roughly speaking, a path-proof goes through three stages in its route to belief. First its abductive assumptions must be internally consistent, and consistent with what is already known. This is handled by trying to prove each assumption false, and if this fails adding it to the database and trying the next one. Second, the assumptions must have *predictive power*, in that they must make enough true predictions about the text to warrant making the assumptions. And finally, there cannot be any other path which is equally good, but which makes incompatible assumptions. (By incompatible we mean to include "contradictory", but allow for other forms of incompatibility as well. More on this later.)

Returning to the second stage, the basic idea is a crude approximation of the justification of scientific theories. A scientific theory is good to the degree it makes true predictions about the world, and bad to the degree that new assumptions are needed to make such predictions. To a first approximation the rule we use is that the number of true predictions minus the number of assumptions (a number we call the path's *figure of merit*) must be greater than or equal to zero. For example, in the "go to the store" example we have already seen that there are three assumptions:

```
(inst supershop1 smarket-shopping) ; Call this shop-assum
(= (go-step supershop1) go1)       ; this go-assum,
(= (store-of supershop1) smarket1) ; and this store-assum.
```

There are, as well, three true predictions which follow from these assumptions, so the figure of merit is zero.

```
(inst smarket1 smarket)
   ; From shop-assum, store-assum, and store-of2.
(inst go1 go)
   ; From shop-assum, go-assum, and go-step.
(= (destination go1) smarket1)
   ; From shop-assum, store-assum, go-assum, plus a rule (not given)
   ; that destinations of go-steps are the store-of shopping events.
```

The idea of requiring paths to have explanatory power solves a puzzle which crops up in previous work. For example, [Al85] has a rule that prevents the marker passer from finding a path of roughly the following form:

origin1 . . . plan1 planpart plan2 . . . origin2

where the connections between *planpart* and the two plans say that *planpart* is a substep of both *plan1* and *plan2*, or an object used in both. Alterman justifies this rule because one action or object is seldom used in more than one plan at a time. But killing two birds with one stone is generally considered a good thing to do, so it is hard to see the logical basis for such a rule.

The thing to notice from our point of view is that such paths typically have a large number of assumptions. For example, if we know of two reasons for getting a rope, say, making a noose and jumping rope, then a path that went from **make-noose** to **rope** to **jump-rope** would usually require at least the following assumptions

```
(inst make-noose1 make-noose)
(= (patient make-noose1) rope1)
(inst jump-rope1 jump-rope)
(= (instrument jump-rope1) rope1)
```

plus whatever else was required for the path. Unless noose making and jumping rope account for a great deal of evidence such a path could not have sufficient explanatory power. Of course, if there were such predictions then this would be exactly the case where Alterman's rule would break down. Most obviously, if we already knew that the agent was planning one of these activities then some of the above would not have to be assumed and a two birds path would come to mind. For example "Jack decided to jump rope and then kill himself." Thus we see that Alterman's rule is a first approximation to a more complicated reality, one which the sufficient-explanatory-power rule captures much more accurately.

It has also been noticed [Ch83, Al85] that *isa plateaus* – paths which meet due to two objects being in the same class (e.g., a path from "boy" to "dog" meeting at **animal** because they are both animals) – are essentially useless and have to be pruned out. Similar arguments show how this rule also follows from our theory.

We said that the rule of at least as many predictions as assumptions is an approximate one. The actual rule is considerably more complicated, with what appear at this point to be special cases. There is, as well, a complication dealing with subsidiary evidence which is also rather ugly. In both cases we take the uglyness as an indication that the theory is inadequate at this point, and the section on future work spells out some of the problems, and what we see as the best bet for improving things.

It is possible that there be several paths, each of which has sufficient predictive power by the above criteria, but which are not compatible. Thus all path-proofs with sufficient predictive power are compared by comparing their assumptions and predictions. For example, if two paths have the same assumptions they are equivalent, so obviously there is no point in believing both. We can arbitrarily ignore one or the other. Similarly if two paths are contradictory Wimp should only believe one or the other, but here the choice is not arbitrary. Wimp chooses the path with the highest figures of merit. If two or more are tied for highest, then none are believed and all computations are thrown away in the hope that later evidence allows a decision.

There is one other case of some interest, and that is where the paths are incomparable since there are no contradictions in believing both, and they make different assumptions and predictions. Here we distinguish two cases: those which are truly compatible and those which are covertly incompatible. The basic idea is that for two path proofs to be truly compatible their conjoined mert (the number of predictions they make jointly, minus the number of assumptions they require jointly) must be greater than or equal to zero. Note that two path-proofs can individually be explanatory, while jointly they are not, if they have, say, disjoint assumptions, but share some predictions.

This comes up in the cases where an action can be explained in more than one way. Both explanations by themselves might be explanatory, but together they are no good because they predict essentially the same facts. So a sentence like "Jack got a rope" could be explained by assuming he is jumping rope or hanging up laundry, but not both because they share same predictions (there is a **get** and a **rope** and the rope is the patient of the get) yet have different assumptions, so together they are not explanatory. In such cases Wimp believes only the better of the two. If neither is better than neither is believed. So after "rope" in "Jack wanted to kill himself. He got a rope." a path through making a noose is preferred because it explains more, while in "Jack got a rope. He wanted to kill himself." the explanations are initially equal, and it is not until the second sentence with "kill" that a path to rope (through making a noose) eventually finds an explanation of the action in the first sentence.

For the most part all paths reported by the marker passer are judged by the path checker. However, as a minor efficiency measure if a believable path is found which path zorch p and no believable but conflicting paths are found with path zorch greater than $p/10$ then no further paths are considered. This eliminates from consideration many marginal paths.

V Parsing with Wimp

We have now done what we set out to do, explain the meaning of the paths found in our marker-passing approach to language comprehension. While we could now stop, we cannot resist the opportunity to show the elegance of this theory by indicating how it solves problems found in parsing language – in particular those where one's understanding of the story is required to aid in the disambiguation of the input (e.g., noun-phrase reference, word-sense disambiguation etc). (For a better description, see [Ch86].)

Syntax is still separate from Wimp, so now Wimp gets, in effect, the phrase marker from the syntactic parser. So it is told that a certain word is the main verb, and that certain head nouns stand in various relations to it. For example "Jack went to the supermarket" would be given to Wimp like this:

```
(syntactic-rel subject* went1 jack1) ; Jack is the subject of went.
(syntactic-rel to went1 smarket1)    ; He went "to smarket1."
(word-inst jack1 "jack")             ; These relate particular
(word-inst went1 "go")               ; instances of the word to the
(word-inst smarket1 "supermarket")   ; dictionary entry
```

These formulas are now used by Wimp as things to predict. Thus predictions can be either state of affairs in the story, or descriptions of what is in the sentence.

Note that a path must go from a word to one of the concepts which that word could denote before finding a path to another word. Thus all paths have in them a choice of word meaning, and this choice becomes explicit as an abductive assumption in the course of doing the path proof. In this way word-sense disambiguation is automatically done in the course of path proofs. For example, in "Jack went to the restaurant. He decided on a milkshake. He got the straw." a path is found from "straw" to **drink-straw** and then to **drink** ending up at **milkshake**. This disambiguates "straw ."

We handle noun-phrase reference in much the same way. Each new noun phrase is initially assumed to refer to a newly minted internal object. (We distinguish between "denote" which we take as a relation between a word (or symbol) and an object in the world, and "refer" which we use as a relation between a word and an object in the first-order language.) Wimp decides that this noun-phrase refers to an already present term by including an equality statement in the abductive assumptions of a path it believes. For example, in "Jack went to the supermarket. He found the milk on the shelf. He paid for it." the "it" at the end is assumed to refer to **milk1**, (a term created during the second sentence), because of the abductive assumption **(= it1 milk1)**. This assumption is required because the best path proof for **pay1** saw it as the paying step of the shopping already created in line one, and **milk1** was already established as the **purchased** in this shopping event, and thus had to be the same as the **it1**. (Currently our knowledge representation language only allows single objects as the fillers of the "slots" (first order functions). If this were not the case this example would be more difficult, but the same reasoning should apply.)

Lastly Wimp gives semantic guidance to its ATN parser. As each open class word is parsed the ATN finds all possible parses up to that point and they (in the form given above) are handed off to

Wimp. Then each path evaluates itself with respect to each of the possible parses. Since the parses produce formulas which are used as predictions, a path may predict the formulas in one parse but not another. For example, in "Alice killed the boy with poison" the "poison/kill" path predicts that "poison" modifies "kill" and not "boy" (the other possible parse). Thus each path selects the parse which maximizes its figure of merit. When a path is selected as the best, it in turn selects a parse (the one it used) and this parse is passed back to the ATN which then follows up on this parse and kills off the rest. (A less extreme version might relegate the others to a "back burner" queue.)

VI Problems and Future Research

There are many areas where Wimp needs more work, from its basic knowledge representation to its ability to syntactically parse English. We concentrate here on the areas of direct relevance to this paper, namely marker passing and path checking.

We noted earlier that the actual algorithm for determining sufficient explanatory power (as represented by the figure of merit) is more complex than we let on. For example, currently a "prediction" that there is a physical object in the story, or a person, is not counted as a prediction at all. This implemented as a check on those candidates put forward as a prediction.

This is not unreasonable. After all, given the ubiquity of people in stories, predicting that there is a person is no big deal, as opposed to predicting a supermarket. What is unreasonable is that this is implemented as a special-case exception and that it is an all or nothing affair. Much better would be to somehow note that predicting a physical object is no prediction at all, a person only a little better, a parent still better, a telephone – not bad, while a computer dial-up device is a pretty good prediction.

Another problem with the current scheme is what would happen in our "go to the supermarket" example if it knew that someone would go to the supermarket if he or she were dateing someone who worked there. Depending on the exact axiomitization Wimp might not be able to rule this possibility out, even if no mention of the date had been made.

We are currently looking at the use of probabilities to solve these and other problems. While we would keep the basic idea of path proofs, we would replace the idea of explanatory power by the probability that the abductive assumptions are true given the evidence from the story. So, for example, this would solve the first problem because in normal bayesian updating the posterior probability of a proposition given some evidence is inversely proportional to the prior probability of the evidence (given various independence assumptions, which typically have to be made to keep the number of statistical parameters in reasonable bounds). This would exactly capture the gradation we suggested in how much various predictions should count.

However, probably the most controversial aspect of this work is the use of marker passing in the first place. The problem with marker passing is that it is not obvious if it can do the job of finding important inferences in a very large and interconnected database. Or to be more precise, can it find the important inferences without finding so many unimportant ones that it becomes useless as an attention focusing device? Since Wimp to date has used a very small database (about 75 nodes and 225 facts) it provides no test. Indeed, Wimp finds a lot of garbage. For the simple examples we have run (the above examples, plus similar ones like "Jack put some ice in a bowl. The bowl was wet." and "Jack wanted to use the stereo. He pushed the on-off button.") an average call to the marker passer returns about 40 paths, of which 20 are quickly eliminated by a check for *isa plateaus* (paths which go up to meet at a common **isa** ancestor) and similar (but more technical) garbage. Of the remaining about one out of ten (2 on the average) is actually a good path. At least one other researcher (Norvig, personal communication) has found about the same one out of ten ratio. Nevertheless, the fear is that the ratio will worsen as the size of the database increases

Thus, while we intend to keep exploring the use of marker passing (there is no obvious alternative at this point), we surely intend to keep an open mind on its long-range utility. Interestingly the major results of this paper, the idea of path-proofs and their relation to story understanding, aid us in thinking about a possible liberation from marker passing. The interpretation for paths we have suggested is independent of how those paths are discovered.

VII References

Al85. Alterman, Richard., "A dictionary based on concept coherence," *Artificial Intelligence* **25**(2) pp. 153-186 (1985).

Ch82. Charniak, Eugene, Gavin, Michael, and Hendler, James, *Frail 2.1, Some Documentation,* Brown University Department of Computer Science (1982).

Ch83. Charniak, Eugene, "Passing markers: A theory of contextual influence in language comprehension," *Cognitive Science* **7**(3)(1983).

Ch86. Charniak, Eugene, "A Single-Semantic-Process Theory of Parsing," Technical Report, Department of Computer Science, Brown University (1986).

Ge84. Genesereth, Michael R., "The use of design descriptions in automated diagnosis," *Artificial Intelligence* **24** pp. 411-436 (1984).

Gr84. Granger, Richard H., Eiselt, Kurt P., and Holbrook, Jennifer K., *Parsing with parallelism: a spreading-activation model of inference processing during text understanding,* Irvine Computational Intelligence Project (1984).

Ko75. Kowalski, Robert, "A proof procedure using connection graphs," *Journal of the Association for Computing Machinery* **22**(4) pp. 572-595 (1975).

No86. Norvig, Peter., "Inference processes and knowledge representation for text understanding," Ph.D. Thesis, Department of Computer Science, University of California at Berkeley (1986).

Pe80. Perrault, C. Raymond and Allen, James F., "A plan-based analysis of indirect speech acts," *American Journal of Computational Linguistics* **6**(3-4) pp. 167-182 (1980).

Qu69. Quillian, M. Ross, "The teachable language comprehender: a simulation program and theory of language," *Communications of the ACM* **12**(8) pp. 459-476 (1969).

Ri85. Riesbeck, Christopher K. and Martin, Charles E., "Direct memory access parsing," 354, Department of Computer Science Yale University (1985).

Sc77. Schank, Roger C. and Abelson, Robert P., *Scripts, Plans, Goals, and Understanding,* Lawrence Erlbaum, Hillsdale, N.J. (1977).

Sc78. Schmidt, C.F., Sridharan, N.S., and Goodson, J.L., "The plan recognition problem: an intersection of psychology and artificial intelligence," *Artificial Intelligence* **11** pp. 45-83 (1978).

Wi78. Wilensky, Robert, "Why John married Mary: Understanding stories involving recurring goals," *Cognitive Science* **2**(3)(1978).

Wo81. Wong, Douglas, "Language comprehension in a problem solver," *Proc. Ijcai* **7**(1981).