Probabilistic Abduction for Plan Recognition

Eugene Charniak[1]
Robert Goldman[2]

Technical Report No. CS-91-12
February 13, 1991

[1]Department of Computer Science, Brown University, Providence, RI 02912
[2]Department of Computer Science, Tulane University, New Orleans, LA 70118

Probabilistic Abduction for Plan Recognition*

Eugene Charniak
Department of Computer Science Brown University

Robert Goldman
Department of Computer Science Tulane University

February 13, 1991

Abstract

Plan-recognition requires the construction of possible plans which could explain a set of observed actions, and then selecting one or more of them as providing the *best* explanation. In this paper we present a formal model of the latter process based upon probability theory. Our model consists of a knowledge-base of facts about the world expressed in a first-order language, and rules for using that knowledge-base to construct a Bayesian network. The network is then evaluated, to find the plans with the highest probability.

1 Introduction

Plan recognition is the problem of inferring an agent's plans from observations. Typically the observations are actions performed by the agent, and previous work in this area has limited itself to this case. We will do so as well, although the approach we suggest generalizes easily to other types of observations (states of affairs, actions of others, etc.). Plan recognition is abduction (reasoning from effects to causes) in that we reason from the actions (effects) to plans (causes). A plan-recognition system must be able to retrieve, or construct, possible explanatory plans, and also decide to what degree the evidence supports any particular plan hypothesis. In what follows we will concentrate exclusively on the second of these tasks.

*This work has been supported by the National Science Foundation under grant IRI-8911122 and by the Office of Naval Research, under contract N00014-88-K-0589.

Our particular interest is in plan-recognition as it is needed in text understanding: understanding the meanings of stories by understanding the way the actions of characters in the story serve purposes in their plans. Our work builds upon earlier work in script- and plan-based understanding of stories like that of Cullingford [7], Wilensky [17], Wong [18], Charniak [2] and Norvig [15].

We see plan-recognition and text understanding as a particular case of the problem of abduction.[1] In particular, for the case of simple, declarative text, we view the language user as a transducer. The language user observes some thing (event or object) in the 'real world', and translates this thing into language. Our task is to reason from the text to the intentions of the language user and thence to the thing described.

Because abduction problems involve uncertainty, we have adopted a probabilistic approach to the problem of story comprehension. In order to make such an approach feasible, a number of techniques have been used:

1. Simplifying assumptions
2. A graphical representation of the probabilistic model
3. Incremental construction and evaluation of the representation of this probabilistic model.

We have built a program, Wimp3, which has been used to experiment with our probabilistic framework for story understanding. We will not have time to discuss here the linguistic aspects of Wimp3. Here we will simply discuss plan-recognition. For more details on other aspects of Wimp3 see [8].

2 Previous Approaches to Plan-recognition

Probably the best-known work in the area is that of Kautz [12], [13]. Kautz provides a formal basis for the problem in terms of minimizing (in the circumscriptive sense) the number of "top-level plans." The idea is that every observed action is part of one or more top-level plans. When this restriction is taken into account, plan-recognition becomes a task of nonmonotonic deduction. For example, if action A_1 can only be part of P_1 or P_2, while A_2 can only be part of P_2 or P_3, after asserting that there is only one top-level plan, it follows deductively that A_1 and A_2 are both part of P_2. While this work is justly noted for the clarity with which it lays out its logical and algorithmic basis, as a theory of plan-recognition it suffers from three major flaws.

[1] See [11] or [5] for a statement of this position.

First, because this approach is, essentially, minimal set covering, it is incapable of deciding that a particular plan, no matter how likely, explains a set of actions, as long as there is another plan, no matter how *unlikely*, which could also explain the observed actions. Consider

> Jack packed a bag. He went to the airport.

Any normal reader would assume that Jack is taking a plane-trip. But Kautz's plan-recognition system would not be able to decide between plane-trip, and air-terrorist-bombing, since the latter also has the terrorist packing a bag (with a bomb) and going to the airport (to get the bag on the plane). Bayesians (such as your authors) would say that Kautz's program ignores the priors on the plans, and thus cannot make the right decision. Nor can one simply say that the program should defer the decision. There will be cases where a decision must be made. Imagine the following:

> Person1: Is Jack around?
> Person2: I saw him leave for the airport with his bag this morning.
> Person1: Yes, but is he around?

Person1 must ask this question because Jack might be bombing the plane, rather than flying on it, and if he is bombing the plane he will not have boarded, so Jack might still be in town.

Second, the distinction between top-level plans, which are minimized, and the rest, which are not, is problematic. While most of the time walking is in service of a higher plan (getting some place), occasionally we walk "because we feel like it." So sometimes walking is a top-level plan, and sometimes it is not. Nor can Kautz do away with the distinction. If one minimized over all actions, then one would never postulate a top-level plan at all.

Finally, set minimization as a principle for abduction (reasoning from effects to causes) is simply wrong. For example, in medicine it may be correct to diagnose two common ailments rather than a single uncommon one, particularly if the symptoms are strongly associated with the common ailments, but not with the uncommon one.

Indeed, because of such objections, and particularly because of the need for priors, we doubt that any model which does not incorporate some theory of reasoning under uncertainty can be adequate. The only other such model that we are aware of is Carberry's [1], which uses Dempster-Shafer (D-S) belief functions. We applaud her decision to confront the issues of reasoning under uncertainty and her use of numerical measures for belief, and thus the criticisms we have of her work are of a much narrower sort. These criticisms have to do with our preference for probability theory over D-S for this problem, and the architecture of her approach.

We prefer Bayesian probability theory to D-S on general principles. It is instructive to note that many attempts to give a semantics to D-S belief functions do so by relating them to probabilities, thus admitting that probability is on a much firmer footing than "beliefs." Given this, it seems to us that it is incumbent on anyone using D-S to justify why probability theory would not do. Carberry has not.

Carberry makes two arguments against probability theory. First she claims that probability theory requires "a great deal of time-consuming and complicated computation." In response we point out that while evaluating Bayesian networks is NP hard, so are D-S calculations. In fact, since point-valued probability theory is a limit case of the D-S belief calculus, D-S calculations are in practice *at least as* expensive as probability updating. Furthermore, Carberry's updating problems involve extensive independence assumptions, and she assumes that each action plays a part in only one plan, and that no action plays a part in more than one plan. If these computations were rephrased in terms of Bayesian probability, they could be computed in linear time.

Carberry's second complains is that "probability computations are extremely difficult to explain and justify to a lay person." We have yet to see an argument that Dempster's rule of combination is more intuitively comprehensible than Bayes' rule.[2] In any case, one does not explain the computations, one explains the *results* of the computations. This is what Carberry does with her D-S computation, and we would do likewise.

We have been able to subsume much more of our system's reasoning within our uncertainty calculus than Carberry has done with hers. In particular, several of the heuristic rules that Carberry puts forward are not needed in ours.[3] Finally, because our model of plan recognition is embedded in a more general probabilistic model of language understanding, it is capable of explicating how plan-recognition interacts with other language tasks like pronoun resolution, word-sense disambiguation, etc., something Carberry has yet to attack.

3 Our Plan Representation

Our model of plan recognition consists of a knowledge-base K of facts about the world, which is used in the production of a set of Bayesian networks, $\mathcal{P}$, called "plan-recognition Bayesian networks." K is expressed in a first-order notation. It

[2]In fact, there has recently been very promising work on developing explanations of probability updating in networks [10].

[3]Rules D4,D5, and D6, for those familiar with her paper. They all deal with mediating conflicting evidence.

is a first-order *notation*, and not a first-order *theory*, because the semantics of the formulas are given by the probabilistic model, to be described in the next section.

The language consists of terms denoting actions and physical objects (which will have names consisting of a type name, followed by a distinguishing number e.g., buy43, milkshake2), terms denoting sets of action and object natural kinds (type name followed by a hyphen, e.g. buy-, milkshake-), functions from terms to other terms, typically used to indicate roles, (either common case names e.g., agent, patient, or ending with "-of", e.g., straw-of, so we might have (agent buy43) denoting the agent of the event buy43), the three predicates ==, inst, and isa; and the usual truth-functional predicates.

The predicate inst (set membership), as in (inst buy43 buy-) says that the first term is a member of the set of buying events buy-. The predicate isa, (subset) as in (isa buy- obtain-) says that the first term is a subset of the second.

The predicate == (the better name relation) as in (== (agent buy43) jack2) says that the second argument is a better name for the first. This is implemented in the system by copying facts known about the first argument (the 'worse name') as facts known about the second ('better name'). This does not lead to cycles because we can impose a total order on the names in the system. The better name relationship is used for noun-phrase reference, as in (== it22 milk8) would be the proposition that milk-8 is a better name for it22. This would be true if the pronoun "it" which gave rise to it22 referred to the milk in question. We abbreviate == to = when space is important.

We make no distinction between plans and actions. Plans are complicated actions — those with sub-actions. The sub-actions of a plan relate to the plan in the same way that roles do, as functions from the plan-instance to the sub-action. So (== (pay-stp shop55) pay76) indicates that the paying event, pay76, was the "pay step" of the shopping event, shop55. Thus, in this notation, recognizing a plan p from sub-actions $a_1 \ldots a_n$ is inferring the following:

$$(\text{inst } p \textit{ plan-type}) \; (== (stp_1\ p)\ a_1) \ldots (== (stp_n\ p)\ a_n)$$

Making such inferences requires generic information about plans and their parts. This information can be represented using the above predicates, functions and terms. For example, to state that actions of type A can fill the A-*of* slot in of plans of type P, and that if it does then the A_1-*of* slot of A must be filled by the entity which fills the P_1-*of* slot of P, and the A_2-*of* by P_2-*of* etc would come out as this:

$$(\text{inst } ?P\ P) \rightarrow (\text{and } (\text{inst } (A\text{-}of\ ?P)\ A) \\ (== (A_1\text{-}of\ (A\text{-}of\ ?P))\ (P_1\text{-}of\ ?P)) \ldots$$

For example:

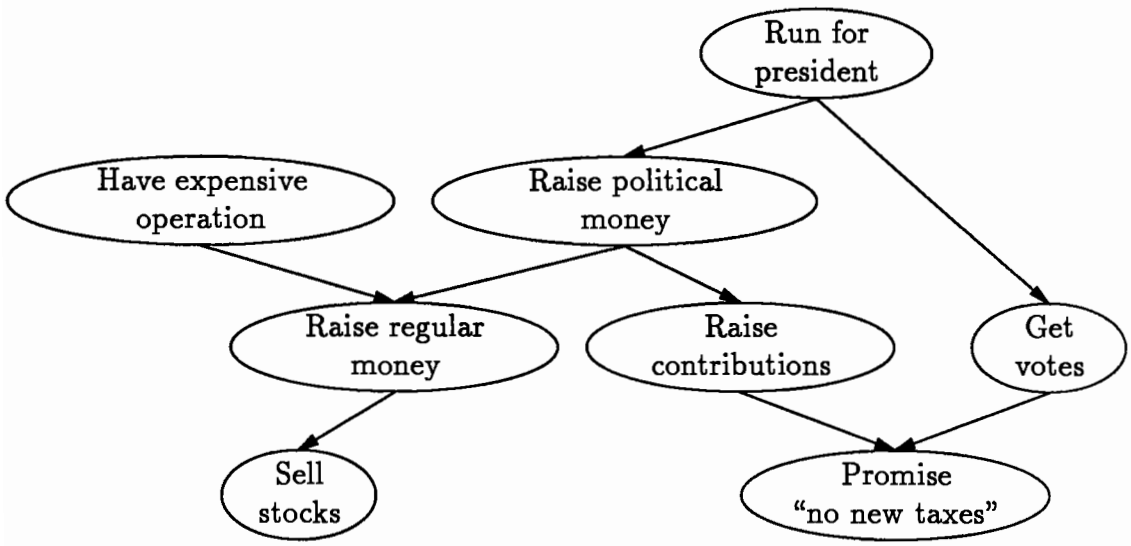

Figure 1: Example Bayesian network

```
(inst ?shop shopping-) → (and (inst (go-stp ?shop) go-)
                              (== (agent (go-stp ?shop)) (agent ?shop))
                              (== (destination (go-stp ?shop))
                                  (store-of ?shop)))
```

4 The Probabilistic Model

Given this model of plan schemas (as actions which can be broken down into distinct sub-actions), and plan recognition (finding the plan instances for which the given actions are sub-actions) we can now talk about how a probabilistic model can make choices between competing hypotheses.

Before we do so, however, we digress to give a brief introduction to Bayesian networks. We cannot do justice to this fascinating topic in only a short space, however. The interested reader is directed to [16] or [14] for a thorough discussion.

Bayesian networks are a directed acyclic graph representation of probability distributions. An example is given in Figure 1. Formally, a belief network is a pair $< V, E >$, where V is a vertex set and E is a set of directed edges, (v_i, v_j). The nodes of a Bayesian network correspond to a set of random variables, $v_0, v_1, \ldots$ In Figure 1, the variables are *Run for President*, *Get Votes*, *Sell Stocks*, etc. Each random variable has a state space ω_v, of mutually exclusive and exhaustive states. There is a probability distribution over the *configuration space*

$$\Omega = \omega_{v_0} \times \omega_{v_1} \times \cdots$$

For example, the state space for *Run for President* might be { *Run for President*, *Don't Run for President* }, or { *Don't Run for President*, *Run for President in 1992*, *Wait and run for President in 1996* }. An element of the configuration space might be:

> Run for President, Don't have expensive operation, Raise political money, Don't raise regular money, Raise contributions, Get votes, Promise "no new taxes" and Don't Sell Stocks.

The joint distribution over the state space can be factorized into a local representation: we need only specify the probability of each node given the values of its parents. In our example, we would have to specify unconditional distributions for *Run for President*, because it has no parents: $P($ *Run for President* $)$ and $P($ *Don't run for President* $)$ We would have to specify a posterior distribution for *Raise regular money*:

$P($ *Raise regular money* $|$ *Have expensive operation, Raise political money* $)$
$P($ *Raise regular money* $|$ $\neg$ *Have expensive operation, Raise political money* $)$
$P($ *Raise regular money* $|$ *Have expensive operation,* $\neg$ *Raise political money* $)$
$P($ *Raise regular money* $|$ $\neg$ *Have expensive operation,* $\neg$ *Raise political money* $)$

$P($ $\neg$ *Raise regular money* $|$ *Have expensive operation, Raise political money* $)$
$P($ $\neg$ *Raise regular money* $|$ $\neg$ *Have expensive operation, Raise political money* $)$
$P($ $\neg$ *Raise regular money* $|$ *Have expensive operation,* $\neg$ *Raise political money* $)$
$P($ $\neg$ *Raise regular money* $|$ $\neg$ *Have expensive operation,* $\neg$ *Raise political money* $)$

Less formally, a Bayesian network is a graph which depicts relevance relations between a set of random variables. The edges in a Bayesian network represent *direct* influences. Every random variable in a Bayesian network is conditionally independent of the rest of the random variables, conditional on what Pearl calls its *Markov blanket*. This is the set of its descendents, its parents, and the parents of its descendents. In our example network, the Markov blanket of *Raise contributions* is (*Raise political money*, *Promise "no new taxes"*, *Get votes*)

A second advantage is the compactness of the local representation of the probability distribution. If the graph is sparsely connected, we realize considerable savings over a full joint probability table of size 2^N. The local distribution also allows for efficient computation of posterior distributions, given some set of evidence, what is the probability distribution of the remaining random variables. While in general this problem is NP-hard [6], there has been a great deal of successful research on algorithms for finding posterior distributions.

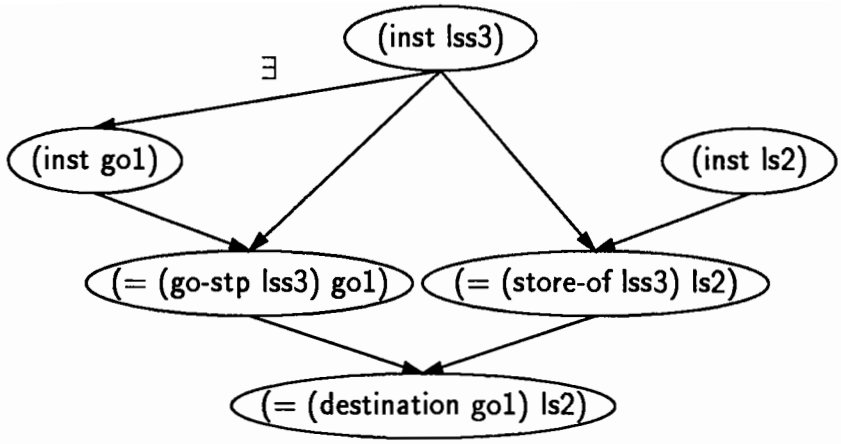

Figure 2: A Bayesian network for a plan-recognition problem

Finally, Bayesian networks make it easier to assess the quantities necessary to build a stochastic model. Conditional probabilities over a small set of relevant random variables are easier to assess subjectively. Pearl suggests some heuristics to further simplify this process by drawing on experts' causal intuitions.

Figure 2 shows a Bayesian network for plan recognition. In the rest of this section we first informally explain what is going on in this network, and then formally describe the class of Bayesian networks which we use.

The random variables in our network are of two sorts. Most of them denote propositions, i.e., boolean variables. The only exception to this are the random variables which indicate the type of a thing. In Figure 2, (inst go1) is one such variable. The sample space for this type of variable is the set of primitive types (types with no sub-types) in the isa hierarchy. Thus the possibility that, go1 is a driving event would be expressed as (inst go1) = drive-.

Figure 2 is the network constructed to represent the possibility that someone's going to a liquor store (ls2) is part of a liquor-store shopping event (lss3). At the top of the network is the hypothesized high-level plan itself. Our belief in this plan-recognition hypothesis will be expressed by the probability that the value of this random variable is liquor-store-shopping-. The hypothesized plans are typically root nodes in the Bayesian network so we need to specify their prior. This is done according to the semantics outlined in [4] by which lss3 is a random variable (in particular, a Bernoulli trial) with a sample space consisting of all entities in the world. We assume a large, but finite domain D of equiprobable elements. Thus the prior of (inst i) = *type* is $\frac{|type|}{|D|}$ If an inst statement is not a root node (e.g., (inst go1)) in Figure 2 its distribution is assigned according to the "existential" rules described below.

Below the node denoting the plan are propositions describing the entities in the input which are to be "explained" by fitting them into the high-level plan as slot fillers. In Figure 2 these are

(inst go1), (== (go-stp lss3) go1), (inst ls2), (== (store-of lss3) ls2).

That is, we have two entities, go1 and ls2, which the two slot-filler statements (the two equality statements) fit into the lss3 plan. As we noted earlier in our discussion of the basic model, the first of these, (== (go-stp lss3) go1) constitutes the *direct* explanation of the going event in terms of the higher-level plan of shopping at a liquor store.[4]

Next, note the arcs from the inst statements to the slot-filler statements involving the instances (e.g., the two arcs into (== (store-of lss3) ls2)). These arcs are there to capture the fact that the probability that two entities ((store-of lss3) and ls2) are the same is zero if they are of different types, and $\frac{1}{|t|}$ $(= \frac{P(=)}{P(t)})$ if they are both of type t. (Remember that we are assuming a set of equiprobable elements.)

There is an extra arc (marked with a $\exists$ in Figure 2) from the plan to the inst node of one of its slot-fillers, go1. We call this the existential slot-filler. It is distinguished from the rest of the plan's slot fillers by having a different probability distribution at its inst node and slot-filler proposition, (== (go-stp lss3) go1). To see the need for this, consider the difference between how this going event fits into lss3, and how, say a subsequent paying event might fit into it. When we see the going event we have no shopping hypothesis. Thus when we create it, the going event serves to "define" it in the sense that if there is a shopping event here, it is the one into which this going event fits (as the go-stp). The random variable lss3 is akin to an existential variable scoped within go1. Thus, the probability of the inst being the appropriate type, and filling the slot, given that lss3 is a shopping event is 1: the probability that go1 fills the go-step slot of the liquor-shopping plan which explains go1. We call this the *existential* slot filler because of the *exists* in the interpretation, and the arc is called an "up existential" arc. (We will shortly encounter "down existentials " as well.)

On the other hand, consider the possibility that a subsequently mentioned paying event fills the pay-stp in lss3. Here lss3 already exists, and thus cannot be interpreted as, by definition, the shopping plan into which the paying fits. Rather we must interpret this as a "random" paying event which might not fit into lss3, even if lss3 is of the correct type. Thus the probability that the paying event fills the (pay-stp lss3) is not the probability that such a paying event exists (given lss3 is a shopping), but rather the probability that it exists, *and that the one mentioned*

[4]This direct explanation might be part of a more complicated explanation such as the agent's plan to buy refreshments for a party.

is the one for lss3. We did not have this second part for go1 because it was true by definition of lss3.

While we explained the difference between existential and non-existential slots by contrasting the go-stp with a subsequent pay-stp, the other slot filled in Figure 2, store-of, would have done just as well since it too is a non-existential slot. That is, once we have defined lss3 in terms of go1, the probability that ls2 is the store-of is the probability that there exists such a store, *and that* ls2 *is it.* The distinction between the two cases is reflected in our model by an extra arc between the high-level plan and its existential slot-filler (see the arc between (inst lss3) and (inst go1)). This arc is necessary if we are to state that the probability of the latter is 1 given the former.

We would like to point out two complications. First, we do not allow up-existentials from objects to actions. For example, we could not define lss3 as the liquor-store shopping which occurred at ls2 since there are many such events. Second, in some more complicated cases we also allow "down existential" arcs from a plan down to an otherwise undefined sub-part. This *may* be between actions and objects, since (in our axiomatization) there is only one filler for each slot. The precise constraints here will be given in our formal description of the networks, below.

Now let us consider evidence other than the presence of objects. Items of evidence for a hypothesis are facts in the input which are (probabilistically) implied by the hypothesis.[5] For example, in Figure 2 the statement (== (destination go1) ls2) is evidence for the shopping hypothesis. It is evidence because if one is shopping, then one will go to the store at which one is shopping. Note that this fact would also constitute evidence in favor of any other hypothesis which predicted going to a liquor-store. In Section 6 we will address the issue of how the model handles competing hypotheses.

We will now give a formal definition of our networks. In this description we will use e_b^a to denote the edge from a down to b. In what follows we will only give the probabilities for the case where there is only a single proposed plan in the network. In the case of more than one the joint distribution will be that specified by the "noisy or" gate [16]. (Those unfamiliar with this can simply think of it as normal or-gate, i.e., the probability is 1 iff at least one of the plans is true.)

The set of "Plan-recognition Bayesian networks" ($\mathcal{P}$) for a knowledge-base K, and a set of instances I, are Bayesian networks (which are pairs (N, E) of nodes and edges) defined as follows:

1. (Basis) $(\{\}, \{\}) \in \mathcal{P}$

[5] I.e., the propositions E such that $P(E|hypothesis) > P(E)$.

2. (Object Evidence) Let $(N, E) \in \mathcal{P}$, and $b = (\textsf{inst}\ j)$ $(j \in I)$, then $(\{b\} \cup N, \{E\}) \in \mathcal{P}$. Any formula introduced this way is part of the evidence.

3. (Up-Existential) Let $(N, E) \in \mathcal{P}$. If $b = (\textsf{inst}\ j) \in N$, $(\textsf{inst}\ ?\textsf{i}\ t_1) \rightarrow (\textsf{inst}\ (\textit{slot}\ ?\textsf{i})\ t_2) \in K$, $i, j \in I$, and either t_2 is an event or t_1 is an object, then $(N \cup \{a, c\}, E \cup \{e_c^a, e_c^b, e_b^a\}) \in \mathcal{P}$, where $a = (\textsf{inst}\ i)$, $c = (= (\textit{slot}\ i)\ j)$, and $a, c \notin N$. The probability of $b = t_2$ given that $a \subset t_1$ is 1. The probability of $b = t_2$ given that $a \not\subset t_1$ is $\mathrm{P}(t_2) - \mathrm{P}(t_2 | a \subset t_1) \cdot \mathrm{P}(a \subset t_1)$. The probability of c given that $a \subset t_1$ and $b \subset t_2$, is 1, else 0.

4. (Down-Existential) Let $(N, E) \in \mathcal{P}$. If $a = (\textsf{inst}\ i) \in N$ $(\textsf{inst}\ ?\textsf{i}\ t_1) \rightarrow (\textsf{inst}\ (\textit{slot}\ ?\textsf{i})\ t_2) \in K$, and $i, j \in I$, then $(N \cup \{b, c\}, E \cup \{e_c^a, e_c^b, e_b^a\}) \in \mathcal{P}$, where $b = (\textsf{inst}\ j)$, $c = (= (\textit{slot}\ i)\ j)$, and $a, c \notin N$. The probabilities are defined as in the up-existential case.

5. (Slot-filler) Let $(N, E) \in \mathcal{P}$. If $a = (\textsf{inst}\ i)$, and $b = (\textsf{inst}\ j) \in N$, $(\textsf{inst}\ ?\textsf{i}\ t_1) \rightarrow (\textsf{inst}\ (\textit{slot}\ ?\textsf{i})\ t_2) \in K$, and $i, j \in I$, then $(N \cup \{c\}, E \cup \{e_c^a, e_c^b\}) \in \mathcal{P}$, where $c = (= (\textit{slot}\ i)\ j)$. The probability of c given $a \subset t_1, b \subset t_2 = \frac{P(=)}{P(t_2)}$, else 0.

6. (Other Evidence) if $(N, E) \in \mathcal{P}$, and $\{a_1, \ldots, a_i, =_1, \ldots, =_n\} \subset N - E$ where E is the evidence introduced by rule 2, all $=_i$ are equality statements, and $A_1, \ldots, A_i \rightarrow C$ is a member of K, such that after using the equality statements, $a_1, \ldots, a_i$ unifies with the antecedent of the rule, then the network created by adding C plus edges from all of $a_1, \ldots, a_i, =_1, \ldots, =_n$ to C is a member of $\mathcal{P}$, provided that if C is a member of N, then it was introduced by this rule. The probabilities are given by the rule.

7. Nothing else is in $\mathcal{P}$.

5 Creating Hypotheses

We have built a program, Wimp3, which automatically creates the networks we have described. Wimp3 deals with more than just plan recognition. It produces such networks directly from English narratives, and thus must (and can) deal with other problems such as pronoun reference, word-sense ambiguity, case ambiguity, and syntactic ambiguity. In what follows we will concentrate on how Wimp3 goes about plan recognition and ignore these other issues. It should be noted, however, that the machinery we describe for plan recognition is also used in these other tasks.

The structure of Wimp3 is shown in Figure 3. Words from a narrative are fed

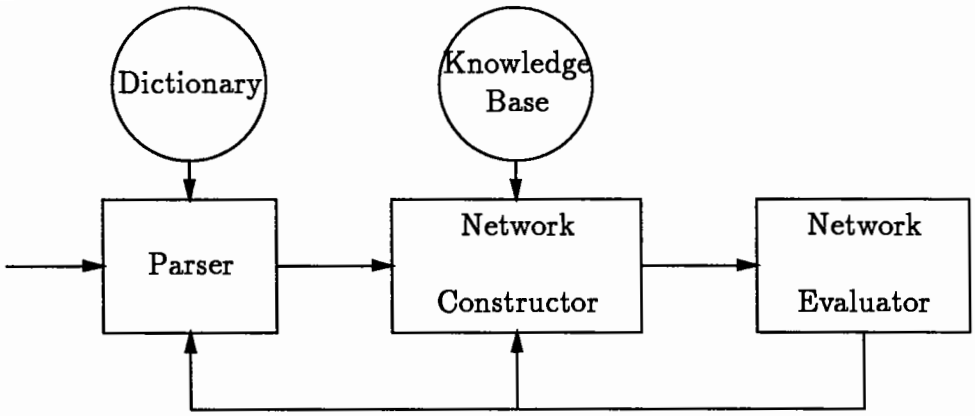

Figure 3: Structure of Wimp3

to the parser one at a time, and a syntactic parse is produced (as a series of first-order formulas) which are then fed to "Frail," our inference engine. Frail produces the Bayesian network. Directly responsible for this are a set of forward chaining rules which produce not simply propositions, but Bayesian networks [9]

The forward chaining rules pretty much follow the formal description of $\mathcal{P}$.[6] The major difference between the forward-chaining rules which *construct* $\mathcal{P}$s and the rules which *define* them is that the former contain restrictions on when to apply them. These restrictions are there to avoid the construction of useless sections of the network. In principle they should have no effect on what plans are actually selected.

For example, consider the rule for up-existentials. In the description of $\mathcal{P}$ it says that if we have a token j, and a rule stating that things of type t_2 can fill slots in things of type t_1, then propose a thing of type t_1 of which j fills the appropriate slot. In the corresponding rule we also require that there is some suggestion that j is, in fact, of type t_2, and also the approval of a marker passer [3] which is responsible for proposing hypotheses. In much the same way, the forward chaining rule which corresponds to clause 5 of our definition of $\mathcal{P}$'s has some added preconditions so that it requires more evidence that the consequent is true before it is added to the Bayesian network in Wimp3.

6 Competing Hypotheses

In a story like "Jack went to the liquor store. He pointed a gun at the owner," Wimp will believe with high probability that Jack is shopping at a liquor store after

[6] Although currently Wimp3 does not have down-existentials, they would be easy to add.

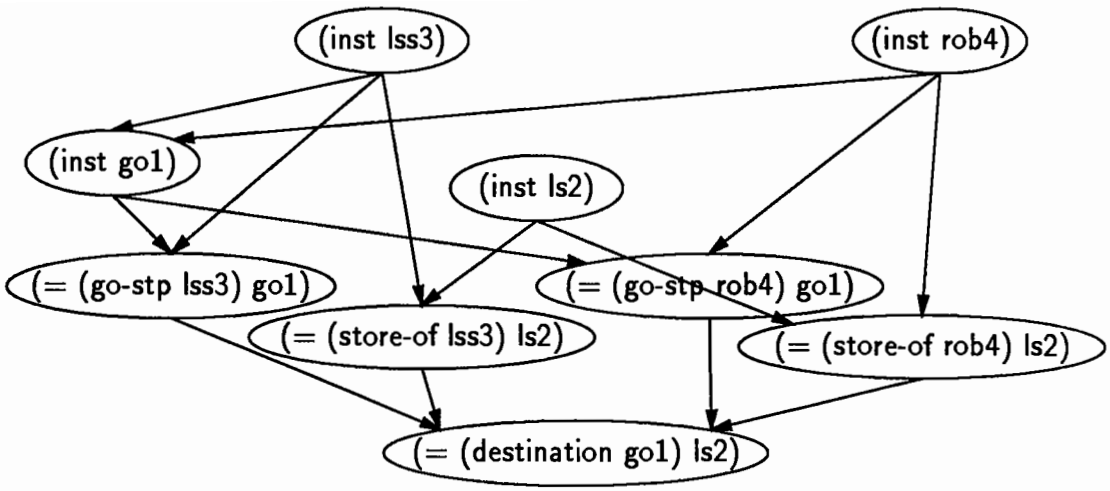

Figure 4: Competition between shopping and robbing

reading just the first line.[7] It does this even though it has in its knowledge-base a plan for robbing stores. After reading the second sentence, Wimp will "change its mind," and decide that the robbery hypothesis is more likely. It is examples such as this that to us most decidedly argue against a non-monotonic logic approach, and for a system which uses numbers to encode the prior probabilities of events. In this section we will examine this example in more detail, to show how Wimp "changes its mind."

Given this story, and a knowledge-base containing both liquor-shop and rob- the competition between them would be expressed by the Bayesian network shown in Figure 4. The two explanations, lss3 and rob4, compete for belief because they both serve to explain the same evidence, (= (destination go1) ls2). The node for this latter proposition is, as we have said above, a noisy-or node. That is to say, this proposition can be explained either by the hypothesis lss3 or[8] the hypothesis rob4. Evidence at or-nodes in Bayesian networks gives support to all their possible causes roughly in proportion to how probable these explanations are based on any other evidence observed.[9] In the case at hand this means that the evidence (= (destination go1) ls2) will support lss3 and rob4 roughly in proportion to the prior probabilities of shopping and robbing liquor stores, since there is no other evidence. Since shopping is much more common, we find that the probability assigned to it is quite high (about .8 in our system) while that for rob4 is only slightly above its prior probability (currently about 10^{-6}).

[7] We apologize for the anthropomorphic terminology; it is simply more succinct.

[8] *inclusive*

[9] This follows simply as a consequence of Bayes' law.

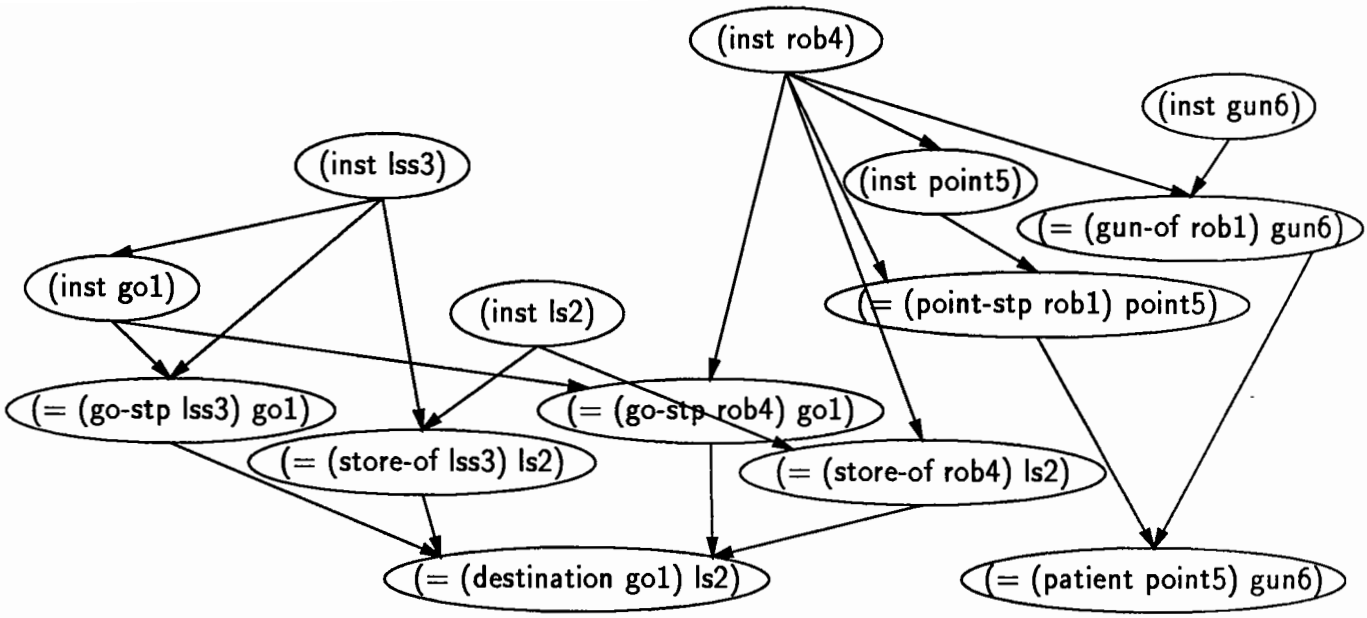

Figure 5: Competition after pointing a gun

Actually, Wimp does not construct the network shown in Figure 4 after seeing "Jack went to a liquor-store." Rather the network it constructs is the simpler one shown in Figure 2, without an explicit robbery hypothesis at all. The reason for this is that Wimp uses a marker passing scheme to find potential explanations, and the scheme weeds out the robbery hypothesis. In [3] it is shown that the marker passer is sound with respect to the probabilistic evaluation mechanism in that the number it computes as a cut-off mechanism for its search is an upper bound on the probability of the hypothesis (given certain reasonable assumptions above the evidence). Thus while the marker passer will sometimes suggest an hypothesis which does not pan out, if it does not find an hypothesis, it means that the hypothesis would have been rejected even it if had been proposed. This is what happens to robbery.

The situation, however, is quite different after reading "He pointed a gun at the owner." Figure 5 shows a part of the network which Wimp constructs after "pointed a gun."[10] From an intuitive point of view, the fact that Jack pointed a gun suggests robbery, and does so even more strongly after we learn that he is pointing it at the owner of the liquor store. This tips the balance in the competition over the evidence that (= (destination go1) ls2) so that now this evidence is seen

[10]We have omitted the section of the network concerned with the pronoun reference of "He" and the evidence provided by the fact that Jack is the agent of both activities.

to support rob4, not lss3, and the posterior probability of lss3 now goes down.

7 Conclusion

A crucial component of any plan-recognition system is the ability to decide between competing hypotheses. We have presented a model of this process in terms of a Bayesian network which allows us to evaluate the conditional probability of the competing hypotheses given the evidence. Currently this model has been implemented within the Wimp3 story understanding system. Furthermore the model is completely compatible with the other decision processes of Wimp3, such as pronoun referent resolution, word-sense disambiguation, case-determination, and syntactic ambiguity resolution. Wimp3 as been subjected to a single-blind test in which it had to pair-up stories which had the same plans after being debugged on half of each pair. The results of this test are reported in [8]. To summarize these results, the program correctly paired 19 out of 25 after 3 lexical items were added to its lexicon, and 24 out of 25 after the further addition of 4 formulas to the knowledge base. The remaining example generated a network too large to evaluate, pointing to what remains the most important impediment to the scheme we have proposed.

References

[1] S. CARBERRY, *Incorporating default inferences into plan recognition*, Proceedings of the Eighth National Conference on Artificial Intelligence, 1990.

[2] E. CHARNIAK, *A neat theory of marker passing*, AAAI-86, 1986.

[3] E. CHARNIAK AND G. CARROLL, *A Probabilistic Analysis of Marker-Passing Techniques for Plan-Recognition*, Department of Computer Science, Brown University, Technical Report, 1991.

[4] E. CHARNIAK AND R. P. GOLDMAN, *A semantics for probabilistic quantifier-free first-order lanuages, with particular application to story understanding*, IJCAI-89, 1989.

[5] E. CHARNIAK AND D. MCDERMOTT, *Introduction to Artificial Intelligence*, Addison Wesley, 1985.

[6] G. F. COOPER, *The computational complexity of probabilistic inference using Bayesian belief networks*, Artificial Intelligence, 42 (1990), pp. 393–405.

[7] R. E. CULLINGFORD, *Script Application: Computer Understanding of Newspaper Stories*, Yale University Department of Computer Science, Report 116, 1978.

[8] R. GOLDMAN, *A Probabilistic Approach to Language Understanding*, Department of Computer Science, Brown University, Technical Report, 1991.

[9] R. GOLDMAN AND E. CHARNIAK, *Dynamic construction of belief networks*, Proceedings of the Conference on Uncertainty in Artificial Intelligence, 1990.

[10] M. HENRION AND M. J. DRUZDEL, *Qualitative propagation and scenario-based approaches to explanation of probabilistic reasoning*, Proceedings ofl the Sixth Conference on Uncertainty in Artificial Intelligence, 1990.

[11] J. HOBBS, M. STICKEL, P. MARTIN, AND D. EDWARDS, *Interpretation as abduction*, Proceedings of the 26th Annaual Meeting of the Association for Computational Linguistics, 1988.

[12] H. KAUTZ, *A Formal Theory of Plan Recognition*, University of Rochester, Technical report, Rochester N.Y., 1987.

[13] H. KAUTZ AND J. ALLEN, *Generalized plan recognition*, AAAI-86, 1986.

[14] E. E. NEAPOLITAN, *Probabilistic Reasoning in Expert Systems*, John Wiley & Sons, New York, N.Y., 1990.

[15] P. NORVIG, *Inference in text understanding*, Proceedings of the Seventh National Conference on Artificial Intelligence, 1987.

[16] J. PEARL, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, Los Altos, Calf., 1988.

[17] R. WILENSKY, *Why John married Mary: Understanding stories involving recurring goals*, Cognitive Science, 2 (1978).

[18] D. WONG, *Language comprehension in a problem solver*, Proc. Ijcai, 7 (1981).