

Plan and Intention Recognition

Phillip Michalak

April 28, 2004

Contents

1	Motivation for Plans and Intentions	3
1.1	Plans and Intention	3
1.2	Plans and Intentions in Planning	4
1.3	Plan Recognition	5
1.4	Outline	6
2	Plan Recognition Foundations	8
2.1	Classes of Plan Recognition	8
2.2	Criteria for Assessment	9
2.3	Models of Plan Recognition	11
2.3.1	Logical Plan Recognition	11
2.3.2	Probabilistic Plan Recognition	12
2.4	Summary	17
3	Plan Recognition Systems	18
3.1	Logical Systems	18
3.1.1	Allen and Perrault	18
3.1.2	Kautz	23
3.2	Probabilistic Systems	29
3.2.1	Albrecht et al.	29
3.2.2	Charniak and Goldman	34
3.2.3	Bauer	42
3.3	Recent Work	45
3.4	Other Work	47

3.5	Summary	48
4	Problems	49
4.1	Narrowing Hypotheses	49
4.2	Noise	51
4.3	Knowledge Acquisition	54
4.4	Scale	56
4.5	Summary	58
5	Future Directions	59
5.1	Assumptions about Human Planning	59
5.2	Continuous Plan Recognition	60
5.3	Reasoning in Plan Recognition	62
5.4	Conclusion	64

Chapter 1

Motivation for Plans and Intentions

I'd like to begin this discussion about plan and intention recognition with some of the reasons that plans and intentions are interesting topics to study in their own right. One may claim that a key difference between humankind and other known animal species is our superior ability to form and execute plans. While other animals may display some rudimentary planning capability, the extent and depth of their ability seems to be extremely limited compared to our own. To acknowledge that other species have the same basic capability (in a limited form) begs the question of whether or not the ability to plan is one that can distinguish us from other species. Surely it is.

We consider ourselves to be the dominant species on the planet, but it is not because of physical size, speed, or strength (which seem factors in ordering the rest of the animal hierarchy). It is our ability to reason and to form plans that gives us the upper hand. If other species were able to form and execute plans as detailed and complex as our own, the case for our dominance would be less clear. Could it be the case that our ability to form plans is merely an artifact of the architecture of the human body and mind? This is a possibility and seems to me a likelihood, but what of it? This does not make planning any less important. Achieving our present state of affairs without the ability to plan would raise a question as to its importance, but it seems extremely unlikely that this could be the case.

1.1 Plans and Intention

It seems then, that our ability to plan is an important one. What can be said of the importance of plans and intentions? Before answering this question, we'll first define some terms. Intention is difficult to put into words, even though we seem to have a clear intuitive understanding of what it is. One can speak of intentional action, in which

case the action is being described as one that is performed purposefully. One can also talk about intending to act, which is a description of a state of mind. Throughout this paper we will refer to intention in this second sense, the state of mind in which one has a commitment to act. Bratman calls this the *inertia* or the *characteristic commitment* of intentions. They tend to resist casual change, making them useful for planning (described below).

Plans are slightly easier to define. They are an abstract representation of the knowledge required to achieve some goal. Some have used the term *recipe* to describe this sense of the word plan. A different use of the word is to talk of “having a plan to *A*”. In this case one has a *recipe* in mind for how to achieve *A* and one has the intention to achieve *A*. It is this sense of the word plan that we will refer to throughout the paper. When necessary to refer to the abstract data type, we will use the term *recipe*.

The Belief-Desire-Intention (BDI) community has chosen to represent intentions as “adopted” recipes. These adopted recipes have the characteristic commitment associated with intention, and for all intents and purposes can be thought of as intentions.¹ As such, the terms *plan*, *goal*, and *intention* will be used more or less interchangeably throughout this paper. The term *plan* will be used when we wish to emphasize the entire hierarchical intention structure, and the the terms *goal* and *intention* will refer to the states of affairs that the agent is committed to achieving.

1.2 Plans and Intentions in Planning

Coming back to our original question, what is the importance of plans and intentions to planning creatures like us? Bratman indicates that intentions play a central role in our reasoning capabilities because they are conduct controlling in a way that other attitudes are not ([Bra87]). He considers desire and intention to both be pro-attitudes, i.e. attitudes that can motivate behavior. But only intention, because of its characteristic commitment, can actually control behavior. He provides two main aspects in which intention is conduct controlling: as a *filter of admissibility* and through *means end coherence*.

A desire becomes an intention after it has been reasoned about, compared to other desires, and committed to, all against the framework of an agent’s beliefs. From this point until the intention is satisfied or revoked, the intention serves as a constraint on further reasoning by screening incompatible actions. In order to act rationally, an agent should have internally consistent plans. The intentions of the agent form the *filter of admissibil-*

¹The author prefers to maintain a distinction, if only mentally, between the two. In natural language understanding domains it is generally held that discourse intentions are not equivalent to plans ([LA90]). Since natural language discourse is not discussed in any depth in this paper and the majority of the systems we discuss bill themselves as plan recognizers, we will do the same.

ity that disallows those actions that would be inconsistent. The commitment inherent in intention is the reason why such filtering is reasonable. One can desire conflicting things and still be rational, so desire is clearly not suitable for this role. Intention, however, represents those states of affairs that the agent is committed to achieving. In order to be a consistent part of the agent's plan, the action must be consistent with the agent's intentions.

Another aspect of the way that intentions control conduct is the way that they pose subproblems to be solved. The high level intentions may not initially specify the means by which they will be achieved. As time progresses, the agent must fill the plans to contain at least the level of detail that he believes to be currently required. This is the *means end coherence* constraint exercised by intention; a plan not filled in to appropriate levels of detail will eventually become means-end incoherent.

So intentions (and synonymously plans) serve the important role of constraining future planning activities. This is a necessity because of the limited computational resources that we have available to us. Our inability to reconsider every possible option at every moment often constrains us to making decisions based on prior deliberation. But they also serve another fundamental purpose; they provide a framework for social interaction.

Bratman discusses both intra-personal (among one's own plans) and interpersonal (amongst agents) coordination. He argues convincingly that neither of these would be possible without the characteristic commitment associated with intention. Knowing that agent *A* desires to be at a rendezvous point at an appointed time does not provide sufficiently convincing evidence that agent *A* will be there. As we noted earlier, an agent can desire contradictory things, so there is no guarantee that agent *A* will do anything to achieve his desire to be at the rendezvous point; he may have some other conflicting desire that he acts upon. The agent may in fact do nothing at all because that is deemed a better option. In contrast, Agent *A*'s intention to be at the rendezvous point *does* allow us to expect that Agent *A* will attempt to be at the rendezvous point (since intention is conduct controlling). Furthermore, we can also expect that Agent *A* will be in a position to act successfully on that intention because of the *means-end coherence* pressure provided by intention. In effect, these two primary roles of intention both support the expectation that Agent *A*'s intention to do something will result in *A*'s doing that something.

1.3 Plan Recognition

We now turn our attention to plan recognition in order to motivate it as a topic of research. From a practical perspective, we want software to interact with users more intelligently. The extent to which this can be achieved by merely organizing the content presented to the user is limited. At some point software systems must begin to augment the user's efficacy by providing appropriate resources and services at the correct

time. More generally, software systems have more information available to them than the keystrokes, mouse-clicks, and commands that are explicitly represented. All of these are situated in patterns of behavior that suggest intention. This is a high level information source that has been largely ignored in simple user interfaces. As interfaces become more complex, as they begin to incorporate modalities of human communication, intention recognition becomes a key to understanding what the user is really doing in order that the system may collaborate.

A second independent motivation for plan recognition stems from the author's sentiment that any theory of intelligent agency should incorporate the ability to interact with other agents. Principled interaction requires coordination, and coordination requires that agents act in predictable ways. As discussed above, intentions provide this requisite predictability, but in order to capitalize on it, agents must first be able to recognize it. Only then can coordination efforts be successfully initiated based on mutual expectation.

Before moving on to the foundations of plan recognition, we clarify the use of the terms *plan recognition* and *intention recognition* throughout this paper. We mentioned previously that *plans* and *intentions* will be used interchangeably. We will also use *plan recognition* and *intention recognition* interchangeably. In instances where it is important to emphasize the high level intention structure we will use *intention recognition*, but we will use *plan recognition* throughout most of the paper because it is a more accurate description of the assumptions in plan recognizers: that the agent is executing a plan which the recognizer is trying to infer. What we really want to be able to recognize, however, is the high level conduct controlling intentions of an agent. The low level actions that we observe are not necessarily drawn from the observed agent's intentions and may be just an artifact of problem solving. This topic is addressed in more detail in Chapter 5.

1.4 Outline

This chapter has provided some basic motivation for research in intention and plan recognition. Humans are planning creatures, and computer systems that interact with them should take advantage of that fact. Intentions play a central role in framing the development of future plans, and as such will be good predictors of future behavior. Recognition of these will allow systems to interact more naturally and with more focus.

Chapter 2 provides some basic technical materials that serve as the basis for a number of plan recognition systems. Systems are broadly categorized as begin either logical or probabilistic. A set of criteria are outlined for assessing a representative sample of plan recognition systems.

In Chapter 3, five plan recognition systems are described and assessed according to the criteria laid forth in Chapter 2. The first two are logical systems developed by Allen

and Perrault ([AP80]) and Kautz ([Kau91]). The next three are broadly characterized as being probabilistic and are the systems of Albrecht, Zukerman, and Nicholson ([AZN98]), Charniak and Goldman ([CG93]), and Bauer([Bau94a]). The author's work in plan recognition is briefly discussed as well.

Chapter 4 discusses some of the problems that plan recognizers have faced as well as some of the research addressing them. Finally, Chapter 5 discusses some future directions for research that are both novel and interesting.

Chapter 2

Plan Recognition Foundations

Plan recognition is informally defined as the process of inferring the plans of an agent from observed behavior. Traditionally, observed actions are used as inputs to an inference process that selects from a list of domain-specific goals a set of goals (and the plans by which they are achieved) that the agent may be pursuing. The goals and/or plans generated by the system will in some sense explain the actions that have been observed. The motivation for developing such an inference procedure has already been addressed. In this chapter we'll address some of the basic techniques by which such inferences have been achieved. Before discussing these methods, we first divide plan recognition into three broad classes and discuss other criteria by which plan recognition systems (henceforth also referred to as *plan recognizers* or just *recognizers*) may be classified and compared. The basic mechanisms employed by the majority of plan recognizers are discussed in Section 2.3, which includes a brief introduction to the formalisms the under-pin many probabilistic systems.

2.1 Classes of Plan Recognition

The three major classes of plan recognition differ in the type of interaction that the recognizer has with the observed agent. The simplest case is that of *keyhole recognition*, in which the observed agent is unaware of the system performing the plan recognition. This case is the current paradigm for many software systems that perform plan recognition. However, it seems that this view of plan recognition becomes less applicable as the general public attributes more intelligence to computer systems. Eventually, users will attribute planning and plan recognition to the software system, yielding at least two other classes of plan recognition. The first is *intended recognition*, in which the observed agent is aware of the recognizer and is actively cooperating to make its plans inferable. This case arises naturally in the case of dialogue systems, which attempt to interact with human users via natural language. Chapter 5 will discuss dialogue systems, which attempt to recognize plans that are specific to discourse. The most difficult

- a 1. X purchases a gun
2. X goes to the bank
- b 1. X goes to the bank
2. X purchases a gun

Figure 2.1: The importance of event order

of the three is the case of *adversarial recognition*, in which case the observed agent is actively attempting to make the task of plan inference more difficult. This may be the most appropriate model for warfare, business interactions, and strategy games. These are typically situations where the agent is best suited by the ignorance of “opponents”. Very little research has been devoted to this topic; two of the relevant works are [Pol86] and [AFFH86]. Adversarial recognition will not be discussed further in this paper.

2.2 Criteria for Assessment

It makes sense to first delineate some criteria for evaluating different methods to plan recognition. The first criteria that we’ll use to compare systems is the complexity of goals that they can form. Some systems will be able to form only simple goals, while others will be able to form conjunctive goals. For example, the system of Albrecht et. al. [AZN98] is capable only of inferring single “quests” that the user may be engaged in. The system as described is unable to ascribe simultaneous pursuit of multiple quests. In contrast, the Kautz’s system [Kau91] is capable of deriving conjunctive goals. As an example, Kautz provides an example from the Unix domain whereby a sequence of three Unix commands results in a conjunction of two distinct goals. A special case of conjunctive goal recognition is recognition of goals that are temporally interleaved. Even more specific is the question of whether a single action can play a role in multiple plans. In some artificial domains, this distinction may not have much consequence. It appears, however, that interaction with human planners will require the capability to ascribe a single action to multiple plans. It is often the case in human planning that one decides to perform (a sequence of) actions because they contribute to the realization of multiple goals. The plasticity of intention poses another challenge to recognizers. As discussed previously, intentions involve commitment and thus are persistent. However, these intentions are also subject to reconsideration and revision. Goals may change depending on the circumstances that the agent encounters. We may also consider the ability of a system to deal with aborted or changing goals. Can the system make any predictions when the action sequences viewed are “noisy” in any of these regards (multiple/interleaved goals, aborted goals)?

We may also want to consider temporal aspects of the plans that can be recognized. Can the system recognize the importance of ordering in the observed actions? As an example,

consider the importance of order in the two event sequences shown in figure 2.2. While previous and subsequent event sequences will play a role in a subjective assessment of X's plan(s), it seems intuitively that the first event sequence is more ominous than the second. The order of the events plays a strong role in our assessment of an individual's plans. So it seems important that a plan recognition system be able to make distinctions based on the ordering of events. However, it is not always the case that the order of actions should affect inference. For example, the order in which one pours milk and cereal into a bowl during the preparation of breakfast is largely a matter of personal taste. In both cases, the same plan (perhaps a "prepare_cereal" plan) is being executed. So the plan recognizer should be capable of making distinctions based on temporal considerations, but also capable of ignoring temporal event order when it is appropriate to do so.

Another consideration is whether or not the system can rank plan alternatives. A system that indicates strength of preference among alternatives whenever it is unable to infer a single plan is likely to be more useful than one that merely presents the alternatives with no preference information. For example, a system may conjecture multiple plans for the first sequence in figure 2.2 above, two of which might be: 1) X is planning to rob a bank, 2) X is purchasing a gun for less nefarious purposes, perhaps to hunt. A system that infers these two plans but does not provide any preference information leaves the beneficiary of plan recognition in the position of choosing one plan from the available options as THE plan that is being enacted. While one can rely on prior knowledge about plan likelihoods to determine an appropriate course of action (phone the police? ignore the observations?), the plan recognizer could have done the same thing. Furthermore, the plan recognizer has the benefit of having examined the plan space, and thus has a better basis for evaluating the plans.

Another aspect of recognition systems that should be considered is the restriction that the mechanism places on the system knowledge. Most systems place constraints on the form of the knowledge or on assumptions underlying the domain. As an example of a constraint on knowledge form, consider Kautz's system ([Kau91]). His system requires (as do many) that a *plan library* be provided that provides plan decompositions and abstraction relationships for each of the plans that a user may pursue. Additionally, his system makes the domain assumption that the *plan library* is complete. If a plan is not present in the plan library or is achievable in some way that is not encoded, then it will not be recognized by the system. Constraints on the type and form of the knowledge can thus influence the applicability of a plan recognizer. Additionally, the knowledge required by a system can affect the feasibility of the system and the costs of its maintenance. If the system relies heavily on hand-coded knowledge, then the cost of initially specifying the system knowledge and subsequent knowledge management may be prohibitive in realistic applications.

A final consideration is whether or not the system is computationally feasible. While a system that considers all possible plans simultaneously may never make a wrong assess-

ment of an agent's plans, it will likely never make an assessment at all! Plan recognizers are desirable for their utility in practical application. Ignoring the computational feasibility is therefore absurd, as a system that is too computationally expensive is unlikely to be useful in practical applications.

These criteria should serve to illuminate the strengths and weaknesses of the systems described in Chapter 3. There are bigger issues, however, that have yet to be addressed in plan/intention recognition. Chapter 5 will address two issues that should, in the mind of the author, play a central role in future plan/intention recognition research efforts: continuous plan recognition and integrated plan recognition and inference.

2.3 Models of Plan Recognition

This section attempts to delineate some of the most prominent models in the construction of plan recognition systems. Most models to date can be categorized as either logical or probabilistic. This is not a clean-cut distinction, however. Many systems make use of both logic and probability, and other systems seem to be heuristic based. Still, for the purposes of this paper we will classify systems according to this distinction. A third classification has been suggested, that of *argumentative* systems or *abductive* systems. However, some have claimed ([May92]) that all plan recognition systems are in essence a form of abductive reasoning. In Chapter 3 we will present one system which adheres to an abductive philosophy more closely than most, but will again classify the system in the space of probabilistic and logical systems.

2.3.1 Logical Plan Recognition

The traditional approach to plan recognition is a logical one. In the words of Sandra Carberry ([Car01]),

Models of plan inference start with a set of goals that an agent might be expected to pursue in the domain and an observed action by the agent. The plan inference system then has the task of inferring the agent's goal and determining how the observed action contributes to that goal. To accomplish this, the system traditionally is provided with a set of actions that the agent might execute in the domain and a set of *recipes* that encode how an agent might go about performing these actions. These recipes constitute a *plan library* and include each action's preconditions, the subgoals that comprise performing the action, and the effects or goals of executing the action. The reasoning performed by the system uses this domain-dependent knowledge but is itself largely domain-independent.

To infer the agent's goal from the observed action, the plan inference system constructs a sequence of goals and actions that connect the observed action to one of the possible domain goals. This is accomplished by chaining from actions to goals achieved by the action, from these goals to other actions for which the goal is a precondition or subgoal, from these actions to their goals, etc.

2.3.2 Probabilistic Plan Recognition

One alternative approach is to base the recognizer on probabilistic reasoning. This approach has the advantage of having candidate hypotheses ranked by the probabilities assigned by the system, and this is usually cited as one of the reasons for choosing a probabilistic framework as the basis for plan recognition. However, the probabilistic framework that the recognizer uses may impose limits on the class of plans that can be recognized. For instance, the Dynamic Bayesian Networks discussed later in this chapter provide only a rudimentary basis for reasoning about temporal influence. As such, systems that use them as their primary inference mechanism may fail to distinguish between plans that are similar except for complex temporal interactions among their actions.

The general operation of probabilistic systems is to update the state of the system with new evidence at each observation. The probabilities for each internal structure are recomputed based on the new data, which will result in updated probabilities for the hypotheses. The relationship between the evidence presented to the system and the hypotheses resulting from the system are dependent on the internal structure of each system.

Most probabilistic systems to date have been based on Bayesian Belief Networks ([Pea88]) or on Dempster-Shafer theory ([Sha76]). Bauer ([Bau94a]) and Carberry ([Car90]) have used Dempster-Shafer theory in their plan recognition work. Albrecht et. al. ([AZN98]) and Charniak ([CG93]) have based plan recognizers on variations of Bayesian Belief Networks. Since these formal probabilistic frameworks are used so frequently, we will briefly introduce them both here. System-specific applications of each formalism will be discussed in Chapter 3.

Bayesian Belief Networks

A Belief Network (BN) is represented as a directed graph, in which nodes represent random variables and edges represent causality (see 2.3.2). Roughly speaking, the values of a parent node have a causal influence on the values of the child node. By assumption, each node is conditionally independent of all other non-descendent nodes given the values of its parent nodes. This allows a specification of the full joint probability distribution (which normally requires a number of probabilities exponential in the number of nodes) to be specified via a conditional probability distribution (CPD) for each node. A CPD

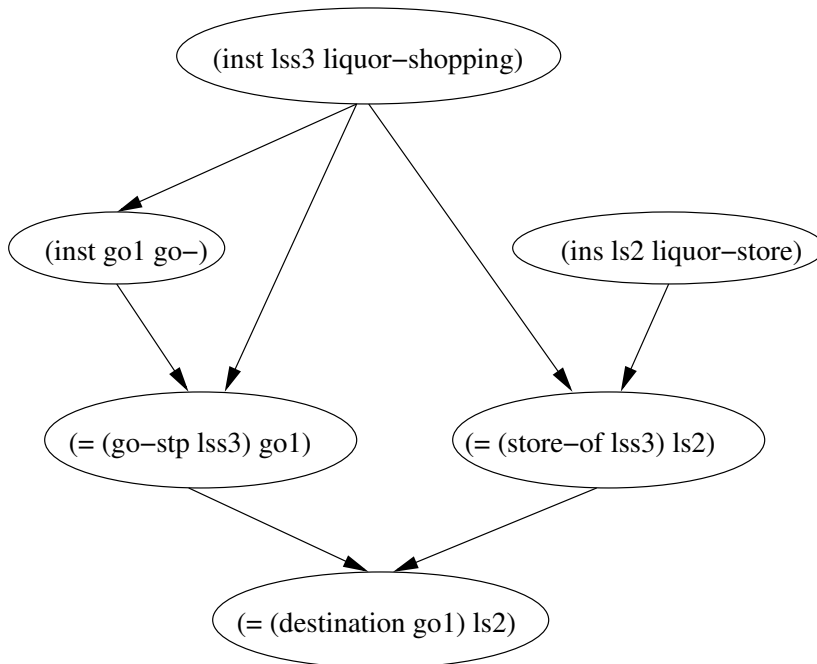


Figure 2.2: An example Bayesian Belief Network from [CG93]

specifies the probability of each value that the node may take for every combination of the parent node values. Root nodes (those with no causal parents) are simply assigned an a priori probability distribution over the values that the node may take. For sparsely connected networks, this results in the need for exponentially fewer probabilities.

The joint probability distribution specifies the relationship between any pair of random variables, and can therefore be used to compute the posterior probabilities of any set of nodes conditioned on any other set of nodes. Using the chain rule, the joint probability of a BN can be computed from the CPDs at each node. These posterior probabilities represent *beliefs* about the propositions contained at the nodes. Beliefs are updated by placing evidence on the network (fixing the value of some nodes) and re-computing the posterior probabilities. This is the method by which plan recognition occurs in systems based on belief networks. Domain goals are represented as top level nodes in the BN. These are causally linked to children nodes representing variables of interest that are affected by the goal. For instance, the goal node might represent a particular food cooking activity and a child node may represent the main ingredient used in the cooking activity. Clearly there is a causal relationship between the food being prepared and the main ingredient. However, information can also propagate from children to parent. If an observation is made for one of the child nodes (i.e. the main ingredient in the current example), then parent nodes are constrained by this observation, and their posterior probabilities will reflect the *evidence* placed on the child node. Plan recognition in systems based on BNs works in this fashion. Evidence due to observations constrain probabilities on hypothesis nodes elsewhere in the system.

BNs by themselves do not provide an explicit mechanism for dealing with temporal relationships between propositions. One technique introduces multiple nodes representing the same proposition at different instances in time. Nodes representing the value of the proposition at earlier times are generally causally related to the nodes representing the value of the proposition at a later time. This technique results in *Dynamic Belief Networks*, named so because new nodes are continually being introduced into the network as time progresses. This method introduces a linear notion of time into the system and allows the values of a proposition at various times to be considered as evidence. We will discuss Dynamic Belief Networks again when we discuss [AZN98].

Dempster Shafer Theory

One criticism of probability theory is that there is no provision for representing the degree of uncertainty about the probabilities involved. Consider the case of trying to predict the outcome of a future coin toss based on the outcomes of a sequence of coin tosses. After seeing three heads in four coin tosses, a maximum likelihood estimation of the probability that heads will be seen on the next coin toss is 75%. A different estimation technique may assign a different probability to the same event, but no estimation relying solely on probability theory will provide a measure of confidence in the estimate.¹ Consider now the case where 1000 coin tosses have been observed, with 750 heads having been observed in this sequence. Again, the maximum likelihood estimation technique predicts that a head will be seen on the next coin toss with a probability of 75%. This time, however, the probability is likely to be much more accurate. In both cases, the same probability is used to represent the probability of seeing a head on the next coin toss even though there is clearly some difference in the confidence that we would place in each of these estimates.

Dempster Shafer Theory (DST) is introduced in [Sha76], and provides a concise representation of both the uncertainty of an agent's knowledge as well as the degree of incompleteness or ignorance. DST has the notions of an *evidence space* (θ) and a *hypothesis space* (Ω). The evidence space represents the possible observations that can be made and the hypothesis space represents all of the possible explanations. The basic idea in DST is that observed evidence induces an evidence mass on supporting hypotheses. It is a generalization of probability because it allows probability to be assigned to subsets of the hypothesis space instead of limiting it to individuals (c.f. probability theory). The generalization from a probability function is a *basic probability assignment* (bpa). A bpa m maps any subset of the hypothesis space to the interval $[0, 1]$ with the constraint that $\sum_X m(X) = 1$. Probability mass assigned to a subset of the hypothesis space represents ignorance about the exact distribution of the probability among the members of the subset.

¹In general, probability theory tells us only how to compute exact probabilities. Confidence interval techniques are an extension of probability theory; the semantics of update are not specified within probability theory.

To clarify this description, consider the coin tossing experiment above. In this situation, the evidence space is the set of potential observations, namely that the outcome of a coin was either heads (H) or tails (T), so $\theta = \{H, T\}$. Since we are trying to predict the outcome of a future coin toss, the hypothesis space is the same: $\Omega = \{H, T\}$. Observing the outcome of a single toss gives only partial information about the nature of the coin. BPAs allow us to specify this uncertainty by assigning mass to subsets of the hypothesis space. The BPA h that assigns mass .5 to $\{H\}$ and .5 to $\{H, T\}$ has assigned mass to the hypothesis H and to the disjunctive hypothesis $H \vee T$. Intuitively this is reserving judgement about whether or not T is an actual possibility, and would be an appropriate evaluation after seeing a single coin toss whose outcome was H . Similarly, a BPA t may be defined for the situation where a single outcome T has been observed.

Given a BPA m , one can obtain upper and lower bounds on the probability of any subset of the hypothesis space according to the following *belief* and *plausibility* functions:

$$Bel_m(A) = \sum_{B \subseteq A} m(B)$$

$$Pl_m(A) = \sum_{B \cap A \neq \emptyset} m(B)$$

Belief represents the lower bound on the probability mass assigned to the set A and all of its subsets, while the plausibility represents an upper bound on the probability mass that could eventually come to rest on A and its subsets. Hence the range $[Bel_m(A), Pl_m(A)]$ represents the range in which the under-specified probability of the set A lies. The belief and plausibility bounds for heads and tails using the h BPA in the current example are:

$$[Bel_h(H), Pl_h(H)] = [.5, 1]$$

$$[Bel_h(T), Pl_h(T)] = [0, .5]$$

Finally, the Dempster rule of combination can be used to combine two BPAs to form another BPA. The rule is given by:

$$(m_1 \oplus m_2)(A) = \frac{\sum_{B_1 \cap B_2 = A} m_1(B_1)m_2(B_2)}{\sum_{B_1 \cap B_2} m_1(B_1)m_2(B_2)}$$

DST is often used as an update mechanism for probabilistic systems. As new evidence is obtained, a BPA (representing belief in hypotheses based on the evidence) is combined with the current BPA (representing current belief in hypotheses) using Dempster's rule of combination.

Let's consider how these elements of DST can be combined in the context of the coin tossing experiment. A reasonable starting BPA before any coin tosses have been seen might be m_0 such that $m_0(\{H, T\}) = 1$ and $m_0(X) = 0$ for all other subsets of the hypothesis space. The BPAs h and t defined above seem to reasonably represent the information

obtained by observing the outcome of a single coin toss. The information from observation of a single coin toss with outcome H is contained in h . We can use Dempster's rule of combination to obtain a new BPA that reflects this additional information:

$$m_1 = m_0 \oplus h = (\langle \{H\}, .5 \rangle, \langle \{H, T\}, .5 \rangle)$$

Observation of two more H outcomes would warrant two more applications of Dempster's rule of combination:

$$m_3 = m_1 \oplus h \oplus h = (\langle \{H\}, \frac{7}{8} \rangle, \langle \{H, T\}, \frac{1}{8} \rangle)$$

The belief and plausibility for H and T after these observations:

$$\begin{aligned} [Bel_{m_3}(H), Pl_{m_3}(H)] &= [\frac{7}{8}, 1] \\ [Bel_{m_3}(T), Pl_{m_3}(T)] &= [0, \frac{1}{8}] \end{aligned}$$

The important thing to notice here is that the probability mass is shifting from the $\{H, T\}$ hypothesis to the H hypothesis as we obtain more information. Next consider the observation of three T outcomes. The resulting BPA and belief and plausibility:

$$m_6 = m_3 \oplus t \oplus t \oplus t = (\langle \{H\}, \frac{7}{15} \rangle, \langle \{T\}, \frac{7}{15} \rangle, \langle \{H, T\}, \frac{1}{15} \rangle)$$

$$\begin{aligned} [Bel_{m_6}(H), Pl_{m_6}(H)] &= [\frac{7}{15}, \frac{8}{15}] \\ [Bel_{m_6}(T), Pl_{m_6}(T)] &= [\frac{7}{15}, \frac{8}{15}] \end{aligned}$$

DST is used in a similar manner for plan recognition. In this case, the observations that can be made for a domain (i.e. boil water, make spaghetti from a cooking domain) comprise the evidence space. The plans that the agent could be executing comprise the hypothesis space. A *historical* BPA is maintained that includes information from all observations. Additionally, each element of the evidence space has an associated BPA that indicates the mass which it assigns to each subset of the hypothesis space. These *evidence mass* BPAs are unchanging, but the historical BPA changes to reflect the sequence of observations. When a new observation is made, the corresponding evidence mass BPA is combined with the historical BPA and the historical BPA is updated with the new information. To be concrete, the BOIL-WATER action may have associated with it the BPA $bw := (\langle \{MAKE - SPAGHETTI\}, .5 \rangle, \langle \{MAKE - SPAGHETTI, MAKE - FETTUCINI\}, .5 \rangle)$ which indicates that the agent may be boiling the water in order to make spaghetti or to make fettucini, but admits some uncertainty. This information can be combined with the historical BPA to yield the new assessment of plan probabilities.

In practice, a more complex variant of DST is necessary to handle observation of abstract actions (uncertainty as to which specific action was performed) and to recognize abstract

plans. In addition, does not directly concern itself with temporal considerations, so plan recognition systems that use it as a hypothesis evaluation method must address these concerns in another manner. These modifications will be addressed in Chapter 3 with the discussion of [Bau94a].

2.4 Summary

This chapter has provided a brief description of some criteria for classifying and assessing plan recognition systems, and has reviewed some of the basic uncertainty models that underly the probabilistic reasoning systems that will be introduced in the next chapter.

Chapter 3

Plan Recognition Systems

This chapter discusses a number of plan recognition systems, beginning with some of the early logical foundations and then addressing more recent probabilistic systems (not necessarily chronologically). The systems of Allen and Perrault ([AP80]) and Kautz ([Kau91]) typify logical approaches and are discussed below in detail. Probabilistic techniques have been used independent of and in combination with logic. The system of Albrecht, Zukerman, and Nicholson ([AZN98]) is discussed as representative of the former approach and the systems of Charniak and Goldman ([CG93]) and Bauer ([Bau94a]) as representative of the latter.

3.1 Logical Systems

Most plan recognition systems have some logical component; all but one of the systems discussed in this chapter make use of a *plan library* which is invariably represented using predicate calculus or an equivalently expressive formalism. This is not surprising; inferring the structure of plans seems to require at the very least a knowledge of the actions that can be performed and possibly additional information regarding their interactions. This type of knowledge has long been represented in planning systems in a logical form. The expressiveness of logic and familiarity with planning machinery made the reuse of logically represented plans a natural choice. Thus, many systems have at least a logical representation of the plans that can be inferred; we characterize as *logical* those systems which rely primarily on logical reasoning mechanisms (i.e. deduction, circumscription, forward/backward goal chaining, etc.) in order to infer plans.

3.1.1 Allen and Perrault

Some of the earliest work on plan recognition is that of Allen and Perrault ([AP80]). Their system focused on reproducing a specific form of human question answering behavior. They observed that when answering questions, the human agents manning information booths in a train station often provided more information than they were asked

for. In order to account for this behavior, Allen and Perrault analyzed the structure of these dialogues and attributed plan recognition to the agents manning the booth. They argued that when a patron approached the information booth with a question, they were acting on a plan to obtain some required knowledge in order that they might carry out their travel plans. In order to facilitate the needs of the patrons, the information agents had to recognize these plans based on the questions that they were asked. Once they had inferred the patron’s plan, they could then perform *obstacle detection* to find out what obstacles might prevent the patron from performing the plan. In the train station domain, this typically meant that the patron did not have the requisite information regarding the particulars of the station operation. Once the obstacles (missing information) had been detected/inferred, the information agent could respond to the question by providing the information that the agent deemed necessary for fulfillment of the inferred plan. Often this meant providing more information than just the information specifically requested.

The system described in [AP80] used *defeasible inference* to infer a patron’s plan from a proffered question. These type of rules allow a reasoner to make a plausible but non-deductive inference. A sequence of defeasible inferences makes no guarantees of correctness. These inference rules were termed the *plan inference* rules for their role in determining the patron’s plan based on observations. The system also made use of *plan construction* rules which were similar, but allowed deductive inferences in the opposite direction. These rules were generally used to construct plans similar to the ones that the information agent hypothesized that the patron must be making. In the following rule descriptions, the terminology ΘB should be read “ Θ believes”, and ΘW should be read “ Θ wants. Thus, $SBAW(\Phi)$ should be read “S believes that A wants Φ . In this context, $AW(\Phi)$ (equivalently, “A wants Φ ”) indicates more than just a desire on A’s part for Φ to come about. It indicates A’s *intention* to bring about Φ , where intention takes the meaning described in Chapter 1.

Plan Inference Rules

Sample plan inference rules in the system were:

1. $SBAW(P) \Rightarrow SBAW(ACT)$, if P is a precondition of ACT.
 This rule allows the recognizer to make the inference that A wants to perform ACT based on the belief that A wants for proposition P to be true and the fact that P must be true in order for ACT to be performed. If ACT is heaping food on a plate and P is reaching the front of a queue, then the observation that A wishes to reach the front of the queue would warrant the conclusion that A wishes to heap food on a plate.
2. $SBAW(B) \Rightarrow SBAW(ACT)$ if B is part of the body of ACT.
 This rule allows the recognizer to infer the intention of A to perform a larger action based on the belief that A wishes to perform a smaller part of that action. If B is

piling food on a plate and ACT is eating dinner, then the observation that agent A wishes to pile food on a plate warrants (under this rule) the conclusion that A wishes to eat dinner.

3. $SBAW(ACT) \Rightarrow SBAW(E)$ if E is an effect of ACT.

This rule allows the recognizer to infer from belief that A wants ACT to come about that A wants E to come about, when E is an effect of ACT. If the effect E was the state of satiety achieved by executing the action ACT of eating dinner, then an observation of A eating dinner would warrant the conclusion that A wanted to bring about a state of satiety.

4. $SBAW(A \text{ KNOWIF } P) \Rightarrow SBAW(P)$.

This rule allows the recognizer to infer from the belief that A wants to know whether proposition P is true that A wants P to be true. If P were the proposition that lunch was free, and S believed that A wanted to know whether lunch was free, then S could use this rule to infer that A wanted lunch to be free.

5. $SBAW(A \text{ KNOWIF } P) \Rightarrow SBAW(\neg P)$.

This rule allows the recognizer to infer the opposite conclusion from the above rule. Intuitively, this is because an agent may be curious about the status of a proposition for one of two reasons. The agent either wishes the proposition to be true or to be false.

Examination of any these rules may cause the deductive reasoner to cringe. The conclusions (or rather hypotheses) reached by these rules are non-deductive and are not assumed to be entailed by the observations. They are meant only to be plausible causes for the observation.

Plan Construction and Heuristics

Examining the last two of the plan inference rules, it is clear that there are multiple ways to extend some of the observations. The single observation that A wants proposition P to be true can generate two contradictory hypotheses by applying different rules. The contradictory nature of the hypotheses is not a problem, as each is a plausible cause for A's wanting to know the truth of proposition P. The important fact is that one observation can be justified by multiple causes. In general this becomes a problem for all *backward chaining* algorithms. Such algorithms start with an observation and reason backward to all of the potential causes of the observation. For each of these potential causes, they reason backward to further causes, continuing until the search space has been exhausted or until some system imposed limit on the amount of reasoning has been reached. Two methods of dealing with the combinatorial search space are employed by this system.

The first technique is the use of expected plans to limit the search space. *Plan construction* rules were used to expand expected plans in a top-down fashion; presumably

these would intersect with the viable plan hypotheses, limiting the search space. The train domain for which this system was developed has a small number of top-level goals that will almost always drive the behavior of participants. Agents will likely approach an information booth to request train departure or arrival times and locations. Allen and Perrault encoded these top level plans as expectations, and favored plan hypotheses that tended to conform to these expectations. A “null” explanation was provided to handle the case that an agent was pursuing an unexpected goal. While this approach is somewhat limited, it seems to mimic a crucial facet of human reasoning. We often have expectations regarding the behavior of others, and reason according to these expectations. It should not be difficult for any reader to recall cases where behavioral expectations have lead to misinterpretation of another’s plans.

Heuristic ranking of partial hypotheses is the major technique that Allen and Perrault used to manage search complexity. Six domain-independent heuristics in three categories adjust the priority of each hypothesis as it is created. The control architecture for plan inference is based on a priority queue. The most promising task is pulled from the queue and is executed. Each task execution yields a new set of tasks with updated priorities, based on the heuristics below:

- H1 (*Action Based*) Decrease the rating of a partial plan if it contains an action whose preconditions are false at the time the action starts executing.
- H2 (*Action Based*) Decrease the rating of a partial plan if it contains a pending or executing action whose effects are true at the time that the action commences.
- H3 (*Expectation Based*) Increase the rating of a partial plan if it contains descriptions of objects and relations in its alternative that are *unifiable* with objects and relations in its expectation.
- H4 (*Search Based*) Increase the rating of a partial plan if the referent of one of its descriptions is uniquely identified. Decrease the rating if it contains a description that does not appear to have a possible referent.
- H5 (*Search Based*) Increase the rating of a partial plan if an intersection is found between its alternative and expectation, i.e., they contain the same action or goal.
- H6 (*Search Based*) Increase the rating of a partial plan each time an inference rule is applied.

The action based heuristics (H1 and H2) affect task priorities based on the likelihood of their hypothesized plan being true given the relationship action effects and preconditions to the state of the world at the time of action execution. Intuitively they capture the notion that an agent will tend to construct realistic/achievable plans; inefficient and unachievable plans are penalized by the action based heuristics. If the action preconditions are not met when the action is performed, then the hypothesized plan is unlikely to be correct, and hence it is penalized by (H1). If an action’s effects have already been

achieved at or before the time of execution, then the hypothesized plan is unlikely to be one that the agent is pursuing, and hence it is penalized by (H2).

The expectation based heuristic (H3) reduces search by focusing on expansion of those partial plans that seemed most likely to merge with expectations. Expected plans and inferred plans were considered to match to the extent that they referred to compatible objects and relations.

The search based heuristics attempted to increase search efficiency by considering specificity of plan hypotheses. A more specific plan that referred to a uniquely identifiable object was given higher priority than one that did not appear to refer to any domain object (H4). Similar to the expectation based heuristic (H3), heuristic (H5) gives higher priority to those plans that resemble expectations. Plans that share an action or goal with an expectation have some likelihood of having common references uncovered by expansion. Finding these commonalities may yield quicker termination of the search.

Evaluation

As one of the early works in plan recognition, this system began to address the difficulties inherent to the task. The logical approach to plan recognition carries the baggage of a combinatorial search space that must be efficiently searched in order to provide a computationally feasible system. While there is little mention of system performance in [AP80], it appears that the limited domain and the heuristic search made the system computationally feasible. There were an extremely limited set of top-level plans that could be recognized (meeting a train and catching a train), which bounded the size of the search space through the plan construction focusing mechanism. A more extensive library of domain plans would have substantially impacted the performance of the system.

The limited nature of the plan library recognized by the system limits its applicability. Plan libraries of this sort can be expanded, but at the cost of computational efficacy.

The system could make predictions based only on a single observation, and the prediction could be only a single plan/goal. Since it dealt only with a single utterance at a time, it did not have to address issues of interleaved, overlapping, or noisy observations. Likewise, issues of temporal orderings are moot. Clearly these are all very limiting to the applicability of the system, but this work serves largely as a jumping off point for exploration into models of dialogue and plan recognition. It serves as a basic foundation that has been extended and improved (e.g. [LA90], [Car90]). Particularly important are the introduction of heuristic search control and the effect of prior expectation as methods of constraining search in the plan space.

3.1.2 Kautz

One potential criticism of Allen and Perrault is the use of heuristic means to derive the single most plausible plan inference. Other things considered equal, we would generally prefer a more formal approach, one in which the underlying formal model provides guarantees about the validity of its results. Of course, other things are not often considered equal, making the heuristic approach a viable alternative.

System Overview

A particularly elegant formulation of the plan recognition problem is introduced by Henry Kautz ([Kau90], [Kau91]). Kautz leverages McCarthy's circumscription ([McC80]) as a basis for plan recognition. The generality of this approach makes it applicable to many different domains, whereas previous systems had been often designed with a specific domain in mind (story understanding for example). His motivation for such an approach was the lack of a theoretical basis for the various control decisions made in other plan recognition systems. As an example, he cites the work of Hobbs and Stickel ([HSME88]) and Allen ([All83]). Hobbs and Stickel use an abductive approach in which a plan is considered a plausible inference if the plan and any necessary domain assumptions entail the observed evidence. As Kautz points out:

Furthermore, even if the recognizer has complete knowledge and the agent makes no errors cases naturally occur where no plan actually entails the observations. ... In order to make abduction work, therefore, the plan (or explanation) must be able to also include almost any kind of assumption...yet the assumptions should not be so strong as to trivially imply the observations.

The theoretical justification for methods of assuming and incorporating these assumptions is missing. Allen's system, on the other hand, makes use of unsound inference rules. The system relies on theoretically unjustified (but intuitively plausible) heuristics to decide which rules to apply and at which point to stop applying them. Given this motivation, Kautz provides a theory based on circumscription that specifies "the assumptions the recognizer makes without recourse to a control mechanism lying outside the theory".

Kautz modeled the plan library as a set of first order logical formulas (which he termed the *event hierarchy*). Each event in the hierarchy represented a plan or an action that could be performed by an agent in a particular domain. The hierarchy provided all necessary knowledge about abstraction, decomposition, and temporal relations between events. In order to provide footing for a discussion about the assumptions made by the system, we provide the following description of the event hierarchy taken from [Kau91]:

An event hierarchy H contains the following parts, H_E , H_A , H_{EB} , H_D and H_G :

1. H_E is the set of unary event type predicates. H_E contains the distinguished predicates *AnyEvent* and *End*. Any member of the extension of an event predicate is called an *event token*. *AnyEvent* is the most general event type, and *End* is the type of all events which are not part of some larger event.
2. H_A is the set of abstraction axioms, each of the form:

$$\forall x.E_1(x) \Rightarrow E_2(x)$$
 for some $E_1, E_2 \in H_E$. In this case we say that E_2 *directly abstracts* E_1 . The transitive closure of direct abstraction is abstraction; and the fact that E_2 is the same as or abstracts E_1 is written E_2 abstracts= E_1 . *AnyEvent* abstracts= all event types. The inverse of abstraction is specialization.
3. H_{EB} is the set of basic event type predicates, those members of H_E that do not abstract any other event type.
4. H_D is the set of decomposition axioms, each of the form:

$$\forall x.E_0(x) \Rightarrow E_1(f_1(x)) \wedge E_2(f_2(x)) \wedge \dots \wedge E_n(f_n(x)) \wedge \kappa$$
 where $E_0, \dots, E_n \in H_E$; f_1, \dots, f_n are role functions; and κ is a sub-formula containing no member of H_E . The formula κ describes the *constraints* on E_0 . E_1 through E_n are called *direct components* of E_0 . Neither *End* nor any of its specializations may appear as a direct component.
5. H_G is the set of general axioms, those that do not contain any member of H_E . H_G includes the axioms for the temporal interval relations, as well as any other facts not specifically relating to events.

Assumptions

In order to perform plan recognition, Kautz's system makes a number of strong assumptions about the plan hierarchy, and hence about the domain in which it is reasoning. These assumptions turn out to yield exactly the same models as circumscription of certain parts of the hierarchy, and hence is equivalent to circumscription. We now briefly introduce each of the assumptions that Kautz made, and delay further discussion until the evaluation. We will use the cooking domain portrayed in figure fig:cookingworld as presented in [Kau91].

The first assumption is the Exhaustiveness Assumption (EXA), which states that the occurrence of each event is also an occurrence of at least one of its specializations. This assumption is achieved by adding a set of formulas of the form:

$$\forall x.E(x) \Rightarrow \bigcup_i E_i(x)$$

where each of the E_i are directly abstracted by E in H_A . Consider the observation of a *MakeSauce* event. Then the EXA warrants the conclusion that this event is an event of type *MakeMarinara*, *MakeAlfredoSauce*, or *MakePesto*. By the transitivity of

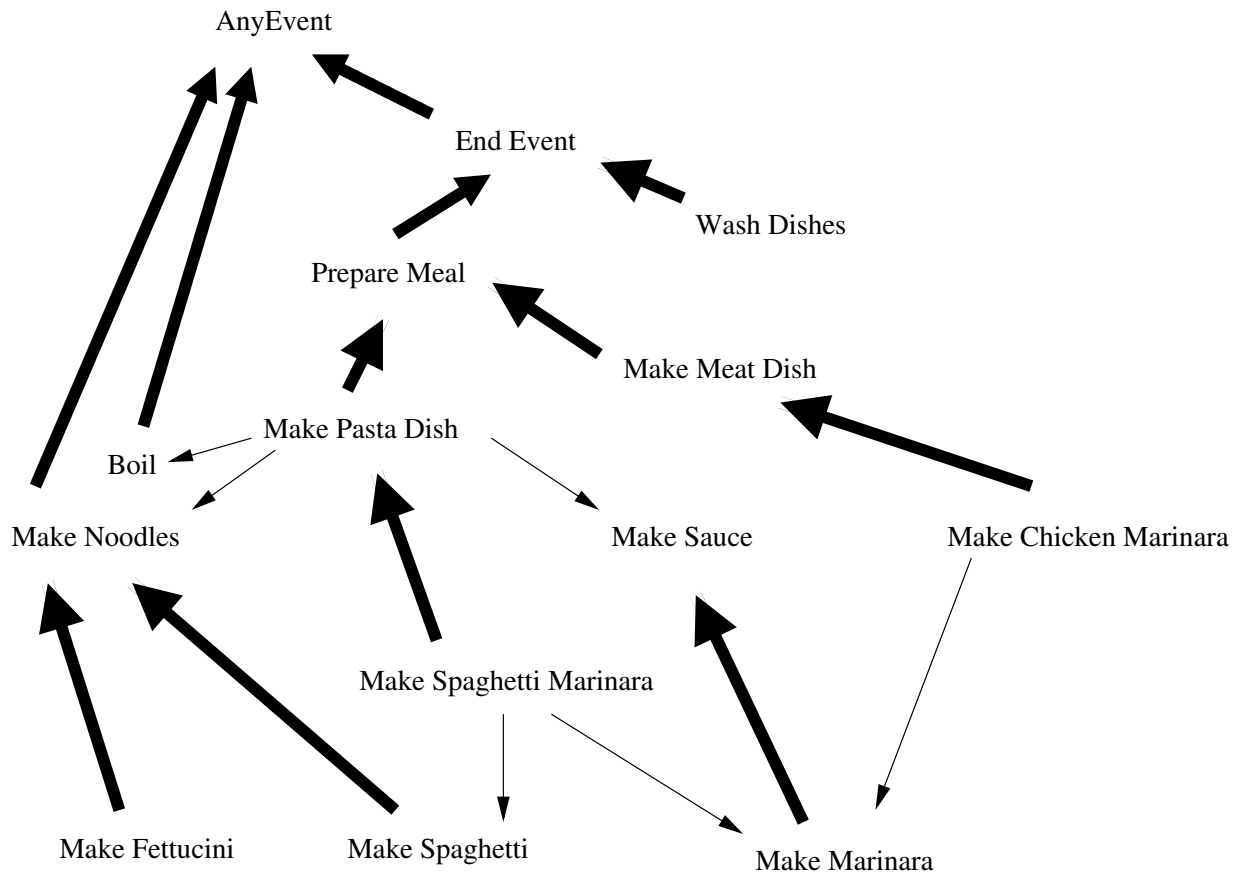


Figure 3.1: Part of the cooking microworld of [Kau91]

implication, this assumption states that every observed event will be an instance of one or more of the basic event types. Kautz shows that this is equivalent to minimizing the non-basic event types; formally stated:

$$H_A \cup EXA \Leftrightarrow Circum(H_A, H_E - H_{EB})$$

The circumscription formula $Circum(S, \pi)$ is the circumscription of the predicates in π relative to S .

The second assumption is the Disjointness Assumption (DJA), which states that all *incompatible* event types are disjoint. Two event types are considered to be incompatible when they do not share a specialization (either directly or via the transitive closure of specialization). This assumption is achieved by adding formulas of the form:

$$\forall x. \neg E_1(x) \vee \neg E_2(x)$$

where E_1 and E_2 are incompatible event types. Kautz shows that adding this assumption to the first assumption is equivalent to circumscribing all event predicates except for *AnyEvent*. Formally:

$$H_A \cup EXA \cup DJA \Leftrightarrow Circum(H_A \cup EXA, H_E - \{AnyEvent\})$$

This is simply placing each event into the extension of as few event type predicates as possible. An event will not be of two different types unless they are compatible, i.e. they share a specialization.

The third assumption is the Component Usage Assumption (CUA), which states that each non-end event must be part of the decomposition of some other event. Formally, this can be expressed using formulas of the form:

$$\forall x. E(x) \Rightarrow End(x) \vee \bigvee_i (\exists y. E_i(y) \wedge f_{i,j}(y) = x)$$

where each of the E_i are events in which an event of the type E is part of the decomposition. This assumption plays a primary role in implementing plan inference in this framework. It warrants the conclusion of a higher-level plan from observation of lower level actions. For instance, observing an instance of *MakeSauce*, it allows one to conclude $MakePastaDish \vee MakeChickenMarinara$. Kautz showed that this assumption is equivalent to the following circumscription:

$$H \cup EXA \cup DJA \cup CUA \Leftrightarrow Circum(H \cup EXA \cup DJA, H_E - \{End\})$$

The final assumption needed needed in order to produce reasonable plan inferences using this circumscriptive approach is the Minimum Cardinality Assumption (MCA). This

assumption states that the number to top-level *End* events should be minimized. Intuitively, this assumption prefers to ascribe as few simultaneous top-level plans to an agent as possible. Circumscription is formulated in terms of set-wise minimization and hence is not suitable for this type of minimization. Under circumscription, a model in which the extension/interpretation of *End* is $\{C\}$ is not preferable to one in which its extension is $\{A, B\}$ because the former is not a proper subset of the latter. In order to achieve cardinality minimization, Kautz outlined an approach in which a sequence of formulas admitting successively more top-level events are checked consecutively for consistency. The earliest in the sequence that is shown to be consistent (using some upper bound on the amount of computation) is used to determine the recognized plans. The formulas used to implement this operation take the form:

$$\begin{array}{ll}
 MA_0 & \forall x. \neg End(x) \\
 MA_1 & \forall x, y. End(x) \wedge End(y) \Rightarrow x = y \\
 MA_2 & \forall x, y. End(x) \wedge End(y) \wedge End(z) \Rightarrow (x = y) \vee (y = z) \vee (x = z) \\
 & \dots
 \end{array}$$

Evaluation

The formalism of circumscription provides an elegant framework for building a theory of plan recognition. It does however, require some strong assumptions about the domain in which the recognizer will be deployed. We will first discuss the limitations that arise from these assumptions, and then continue to discuss this theory of plan recognition in light of the other criteria laid out in the previous chapter.

The EXA formulas are essentially a closed world assumption, stating that any event recognized by the system will be one of the basic types (those that are members of H_{EB}). This assumption requires a domain in which this condition can formally be shown to hold, or in domains where incorrect or incomplete output (resulting from inferences based on incomplete hierarchy information) is acceptable. As such, this seems to limit the usefulness of the system to domains in which the set of available actions is relatively static. This problem is similar to one that will be shared by a number of plan recognizers; a problem of knowledge limitation. The biggest difference is that a system that makes closed world assumptions (i.e. the EXA and others) risks making erroneous inferences regarding plans as opposed to just being incomplete.

The DJA formulas are another type of closed world assumption, stating that event types cannot be of multiple types (multiple inheritance) unless specifically indicated in the abstraction hierarchy (H_A). Furthermore, the algorithms that Kautz outlines in [Kau91] will not handle multiple inheritance in the abstraction hierarchy. This seems quite a limitation, as most real-world applications would seem to require this capability. The hypothetical *MakeMeatRavioli* action can not be included in the hierarchy as a specialization of both *MakeMeatDish* and *MakePastaDish*, limiting the type of inferences

that the system could make. One alternative is to derive sets of mutually exclusive categories (*MakeMeatDish*, *MakePastaDish*, *MakeMeatPastDish*), but this can cause an explosion in the size of the hierarchy.

Perhaps the most constraining assumptions are the CUA and the requirement that certain top-level events be specified (which has been criticized by some researchers, e.g. [CG93]). These assumptions do not admit actions for the sake of action. The knowledge engineer has to decide whether the event/plan for eating dinner is a top-level event worthy of pursuit by itself, or whether it is a means to some end, perhaps a self-nourishment plan. Additionally, the decomposition portion of the hierarchy (H_D), seems to admit only a single decomposition for each action. This seems a difficult limitation to work with, especially for “creative” domains where it is possible to achieve a goal in many different ways.

The final assumption that the number of top-level events should be minimized has also been called into question ([CG93]). Charniak uses the medical domain as an example where multiple simple causes may be preferable to a single complex cause. Though the intuition behind this minimization criterion seems intuitively plausible, this criticism is also valid. Again, the assumption has limited the number of domains and situations in which the recognizer can be effectively used.

Another potential weakness is this method’s inability to rank the alternatives that are generated. Even though one alternative may be much more likely than others given the data, all are presented as being equally viable options. Any system which makes use of such output must be able to reason about the plausibility of each plan in deciding how best to proceed. This issue tends to be the major motivation for probabilistic systems.

A final criticism is the computational feasibility. Though Kautz provides an implementation of a subset of the theory, it is unclear how efficiently the algorithm that Kautz describes can perform.

The strength of this method is in its generality. It can recognize multiple plans enacted simultaneously (conjunctive goals), with interleaved actions. As soon as a contradiction is detected with the current assumption about the number of goals, another top-level event is postulated, allowing recognition of multiple goals. This mechanism also gracefully handles repeated actions (as might happen if the action fails to produce the desired effects), since they will be assumed to be part of the same plan so long as it is consistent to do so. Aborted plans will either be incorporated into an ongoing plan if it is consistent to do so, and will otherwise be reported as a separate, under-specified plan. Similarly for extraneous actions.

Temporal interactions between events is also handled gracefully. Any temporal restric-

tions between sub-events in the decomposition of a higher-level event (specified in the κ portion of the formulas in H_D) will be respected. If observations specify the temporal information that contradicts the restrictions in the decomposition, then the corresponding actions will be allocated to different higher-level events. If there are no temporal restrictions, then the events will be considered to be part of the same higher-level events as long as it is consistent to do so.

The elegance and generality of this method are very appealing, addressing a number of problems present in previous systems. However, the strong assumptions about the domain and computational seem to limit the practical usefulness of the system.

3.2 Probabilistic Systems

The next set of systems addressed in this paper use probabilistic and uncertain reasoning methods in their approach to plan recognition. These methods seem to be motivated strongly by the desire to incorporate *prior* likelihoods of various plans. Some plans are much more plausible than the alternatives at explaining a sequence of observations, and these systems make an attempt to differentiate between likely and unlikely alternatives. The first two rely on Bayesian Belief Networks and the third relies on Dempster-Shafer theory (both introduced briefly in Chapter 2).

3.2.1 Albrecht et al.

Albrecht et al. discuss keyhole plan recognition in the context of a multi-user dungeon (MUD) adventure game ([AZN98]). Their work is part of a trend to leverage machine learning techniques to alleviate some of the difficulty of hand-constructing domain-specific plan libraries. Some other work in this vein will be discussed in a subsequent chapter (4). The aspect that will receive most of our attention in this section is the probabilistic framework used in [AZN98].

MUD Domain

One of the goals of this work is to implement goal and plan recognition in an environment that resembles the complexity of real-world systems. The “Shattered Worlds” MUD is a text-based virtual reality game containing over 4700 locations and 20 different quests (goals). While the number of locations seems to approximate real life scenarios, the number of quests seems to be somewhat smaller than what one would attribute to human agents in general. However, limiting the scope of a plan recognizer to a specific domain (virtually mandatory with current state of the art) makes this number of goals

seem more reasonable. One would still like to see results with a larger number of potential goals. One feature of real-world systems is the noise that may be introduced into observations. This system must contend with noise that comes in the forms of spelling mistakes, typographical errors, conversation between users (which is not goal oriented), newly defined commands, and abbreviations of commands. Typical users execute somewhere between 25 and 500 actions in the course of completing a quest, based on the difficulty of the quest and proficiency of the user. Often, only a fraction of these actions actually contribute to the success of the quest since users may not know what actions are required to complete the quests. Thus, long-term and incremental evidence must be effectively handled in order to effectively predict user goals.

There are a number of other domain features with which this system must deal, some of them not addressed by the systems discussed previously. Many of the quests may be achieved in multiple fashions, and some sequences of actions may accomplish more than a single goal. The system of Allen and Perrault did not have to deal with sequences of events, and thus avoided these types of problems. Kautz's system assumed a single method of achieving each goal, but could recognize the solution of multiple goals from a sequence of actions. An issue particular to this system is the limited interface with the game; however, the information provided to the recognizer (the agent's current location, actions performed, and system notification of question completion) is more than the other systems have assumed. This system does have the added complexity of predicting the user's location and next action in addition to the current quest. Some goals have ordering constraints on the actions necessary to complete them while others do not. Finally, the outcome of a user's action is uncertain; the performance of the action is not sufficient to guarantee the intended effect.

Implementation

Albrecht et al. use an extension of Bayesian Belief Networks (BN) called Dynamic Belief Networks (DBN), which allow for temporal reasoning to accommodate a constantly changing world. Change is modeled by letting the BN grow over time, representing the state of each domain variable by a series of DBN nodes (see Figure 3.2). A typical assumption is that such variable nodes are dependent only on the previous state of the variable (the Markov assumption); furthermore, only some limited number of variable states are maintained (a window) in order to reduce the state space.

Three domain variables are modeled in the DBN's used by Albrecht et al for maintaining state and prediction:

1. *Action (A)* This represents the possible actions that the player may perform in the MUD. There were 4904 unique actions present in the data that this system was trained on. This variable may also have the *other* action as a value, representing all actions that were not observed in the training data.

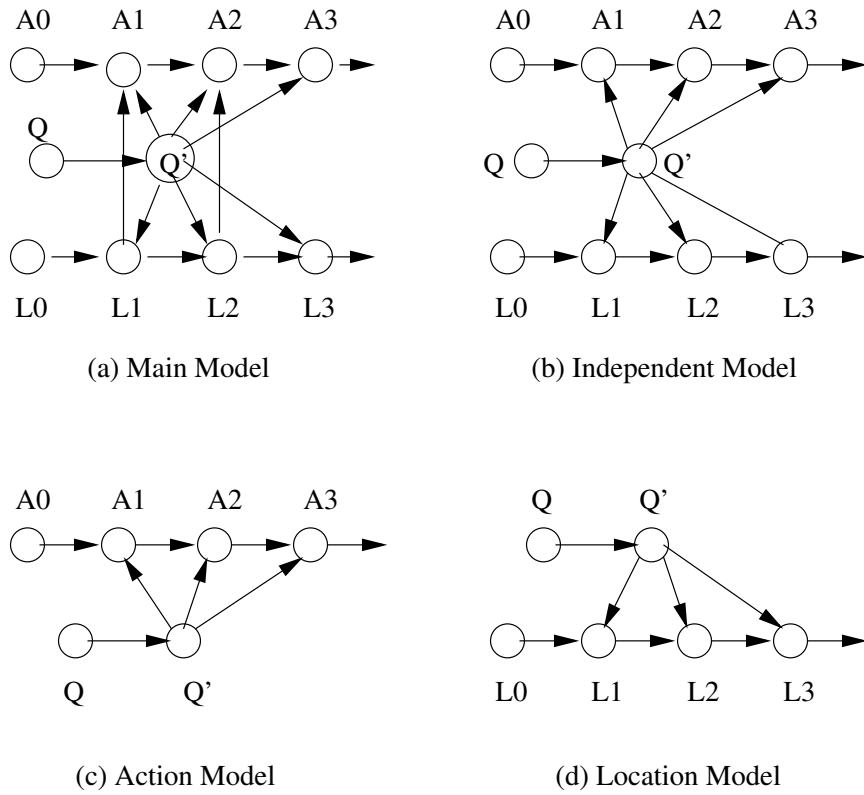


Figure 3.2: Dynamic Belief Networks for the MUD

2. *Location (L)* This represents the location of the player in the MUD. The system was trained using data that contained 3369 unique locations. There is also an *other* location representing all locations that were not present in the training data..
3. *Quest (Q)* This represents the 20 different quests that a player can undertake in the MUD, plus an *other* quest which represents all quests that were unseen in the training data.

Several models were evaluated in order to find the best tradeoff between complexity and performance. These models are shown in Figure 3.2. The *main model* is the most complex due to its causal influences. In this model, the location at time i depends on the value of the current quest Q' and on the location at the time $i - 1$. The action at time i depends on the previous action, the current location, and the current quest. The location dependencies are based on the assumptions that agents are constrained to move in physically contiguous regions and that certain regions are associated with different quests. The action's causal dependency on the previous action models the assumption that sequences of actions are important and will tend to be repeated. Its causal dependency on the current location models the assumption that certain actions are specific to certain locations (i.e. eating in a house or inn, purchasing goods at a shop, etc.). Its causal dependency on the quest models the assumption that some actions are specific to the current quest.

Simpler models include the *independent model*, the *action model*, and the *location model*. The *independent model* removes the dependence of actions on location, while the location and action models try to predict based on only a single aspect of the state information.

The probabilities for each of the models are calculated from previously obtained data that is preprocessed to yield values for the tuple: (previous quest, current quest, current action, current location, next action, and next location). Conditional joint probability tables are constructed for each of the nodes based on the frequency with which each tuple value occurs. Good's *flattening constant* is used to assign a small amount of probability to the *other* events so that the system will be able to predict during testing when encountering events unseen in the training data.

As actions and locations are observed, they are placed as evidence on the DBNs and used to update belief regarding the user's next location, next action, and current quest. Formulas for updating belief provided in [AZN98] rely on the concept of *d-separation* for proof of independence and justification of simplicity. The updated beliefs specify a probability distribution over the potential actions, locations, and quests. The analysis provided by Albrecht et al. shows the *main model* to be the most effective for predicting quests and locations, but the *independent model* performs better on actions and almost as well on the others with the additional benefit of being more efficient to train.

Evaluation

Albrecht et al. have used a probabilistic approach to predict user actions, movements, and quests in a text-based MUD. Their approach is particularly well suited to the domain, as the number of potential actions and locations is far greater than those that have been considered in the systems previously discussed. However, their approach has a number of drawbacks, the principle one being the complexity of the plans and goals that it can infer.

Only very high-level *quests* and individual actions are predicted instead of providing a rich plan structure. This weakness seems reasonable because of the domain in which the system has been applied, but there are a number of domains where detailed plans are required. Automated help systems may require a detailed description of the user's plan in order to provide meaningful help. It is unclear that a probabilistic system such as this one could provide complicated enough plan descriptions to accurately model a user's thought process.

On the other hand, this system does handle noise in a robust manner. Due to the large amount of data that is available to the system for making predictions, extraneous patterns of behavior are ignored. The system allows users to proceed in any manner that they desire, and therefore must be capable of predictions when novel actions and locations are encountered. Additionally, users may attempt to achieve multiple goals simultaneously. Though these characteristics contribute negatively to the system performance, it was able to show reasonable predictive capability despite them.

This system seems to lack a reliable method of capturing temporal ordering constraints. Though frequent repetition of the correct order will increase the probability of predicting that sequence, the system can't reliably determine to what extent the order is important. Part of the issue is the scope of the top level quests that this system is predicting. They often require hundreds of actions for completion, so that it is not feasible to track temporal ordering constraints between individual actions.

As mentioned previously, the desire to rank alternatives is one strength of probability based systems. The probability of the current quest, next action, and next location are computed at each time step. Though a host of possibilities are conceivable, in practice only a few are likely. Another benefit is computational feasibility. The majority of the computation for a system such as this is performed during training. At prediction time, a small amount of computation is required to update the belief network given new evidence.

The domain assumptions leveraged by this system are captured in the construction of the DBN that is used for reasoning. Assumptions about physical limitations and the influence of the current quest, actions, and locations are encoded as causal links in the DBN. Using this system in another domain would require the creation of a DBN struc-

ture that accurately modeled domain interactions. Though the structure of a BN can be arbitrarily complex, efficient update mechanisms are known only for certain classes of BNs. Thus, it seems BN based systems suffer two weaknesses in this regard: the granularity at which the domain can be modeled competes against computational considerations, and the domain must be modeled in the somewhat limited representation of a BN. It is only fair to note that logical systems suffer from the first weakness as well.

In summary, this system is one of the first to address the scale issues that plan recognizers must address in order to be applicable in real world situations. This system handles such a large number of potential actions and locations primarily because of the simple structure that can be used to characterize agent behavior in the domain, and the high level at which goals are predicted.

3.2.2 Charniak and Goldman

As mentioned in the discussion of Albrecht et al., the use of BNs as the primary tool for modeling a domain sacrifices the richness of the plans that can be inferred. The work of Charniak and Goldman ([CG93]) on the WIMP3 story understanding system leverages the strengths of both logic and BNs. Their work stemmed from criticism of purely logical approaches. Charniak and Goldman offer the following three criticisms of Kautz's work (which they took to be the best known representative of the logical camp):

While Kautz's 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. 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, that could also explain the observed actions...

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 someplace), occasionally we walk "because we feel like it". So sometimes walking is a top-level plan and sometimes it is not. Unfortunately, Kautz cannot do away with the distinction and simply minimize all actions. If one minimized the total number of actions, then one would never postulate a top-level plan at all.

Finally, set minimization as a principle for abduction is simply wrong. For example, in medical diagnosis, it is often the case that a two-disease diagnosis is better than one which reports only a single disease. Patients often have two common ailments rather than a single uncommon one which accounts for all of the symptoms.

The first criticism motivates the use of probability theory in their own work, especially for the role that prior probabilities play in determining the likelihood of a plan. The second criticism is based on an assumption that Kautz made regarding the structure of plan hierarchies, namely that there are certain top-level events that cannot be included as a part of another plan. This assumption guarantees that the plan hierarchy will be *acyclic*. The limitation that this imposes on a real-world recognizer is demonstrated with the plan of *going* somewhere via airplane. In order to fulfill this plan, one must execute a sub-plan of *going* to the airport. Plans that cannot be represented in an acyclic plan hierarchy cannot have all of their constituent parts used effectively for recognition. The price paid for more expressiveness, is of course the complexity of the resulting search space. The final criticism mentioned above is a just criticism of the principle on which Kautzian plan recognition is based. While this assumption may often hold, there are cases in most every domain where it will be violated. Notably absent from these criticisms is Kautz's restriction that the abstraction hierarchy be tree-shaped (an implementation restriction); Charniak and Goldman maintain the same assumption in their own system.

System Overview

This system maintains two separate types of knowledge, plan hierarchy and probability information. Plan hierarchy information is stored in a knowledge base of predicate calculus formulas that resembles the decomposition and abstraction hierarchies of Kautz. The biggest difference is that the plan hierarchy is not constrained to be acyclic (there are no top-level events). The probability information is maintained in specially formed BNs called *plan recognition networks* (PRNs), which are dynamically constructed based on observations and the plan structures in the plan hierarchy (the mechanism will be described later). The probability associated with a node that represents a plan hypothesis in the PRN indicates the system belief that the plan is an explanation for the event and object observations. The PRNs may be updated as additional observations are made or additional plans are hypothesized; these changes result in a dynamically evolving structure and corresponding changes in hypothesis beliefs. Charniak and Goldman note that this very desideratum, that predicted plans change accordingly with new evidence, is a strong argument for a non-monotonic approach to plan recognition.

In essence, each node of a PRN represents a random value for the truth of a certain proposition. Thus the random variable may take as its value one of {TRUE, FALSE}, and the probability at each node represents the probability that the proposition is true. Examples of propositions placed at the nodes in a PRN are shown in Figure 3.3 and described below:

1. (inst lss3 liquor-shopping) - This represents the fact that constant lss3 refers to a liquor-shopping event. The \exists labeled edge in the diagram indicates that the liquor-shopping event *lss3* is hypothesized as a plan to explain related observations.

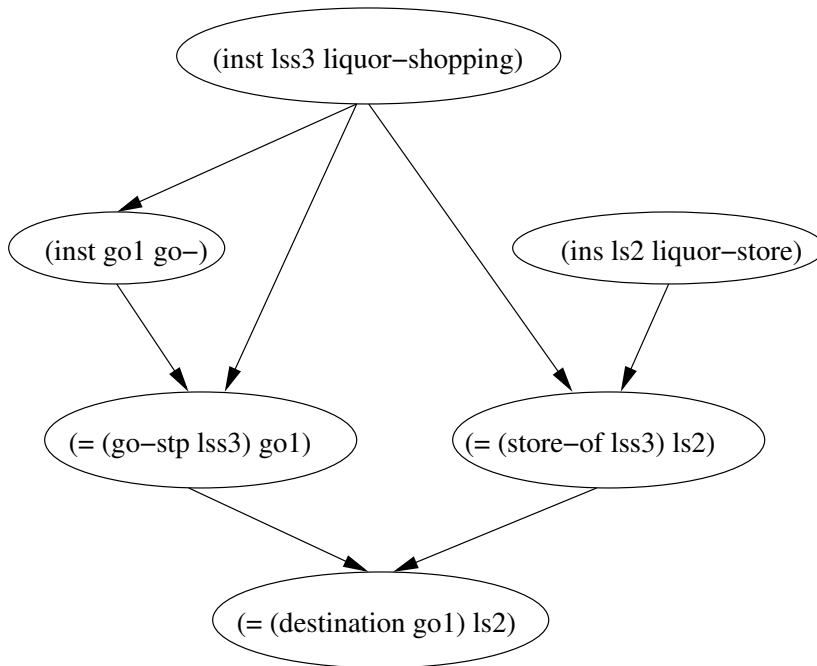


Figure 3.3: A Bayesian network for a plan recognition problem

2. (inst ls2 liquor-store) - This represents the fact that constant ls2 refers to a liquor-store object. This represents an observation.
3. (= (store-obj lss3) ls2) - This represents the fact that liquor-store ls2 is the store object of liquor-shopping event lss3. This is an instance of a *slot-filler* hypothesis.

As discussed in the section about BNs, the arrows in the PRN indicate causal influence. In this system, hypothesized plans exert causal influence on observations, i.e. they *explain* the presence of the observed *events*. Consider the liquor shopping event lss3 in Figure 3.3. It has been hypothesized to explain the presence of the go1 event observation. Hypotheses can perform the role of explanation for multiple events and objects, and a single event or object can play the role of evidence for multiple hypotheses.

Each PRN can be described by a set of nodes and edges (N, E) and a corresponding set of conditional probability distributions. In the following abstract rule descriptions, we ignore the probability distribution in the representation of existing PRNs. The PRN construction rules are used to recursively generate the set of valid PRNs P for a given knowledge base K and observation set I . Note that naive application of the following rules may result in infinite regression or extremely large networks. As such, these rules must be judiciously applied; the focusing mechanism of *marker passing* will be discussed shortly.

PRN Construction Rules and Marker Passer Focusing

1. *Basis* The empty PRN $(\{\},\{\})$ is in P .
2. *Object evidence* If an object or action is observed, then it is included in the evidence. Hence the observation o will be added to $(N, E) \in P$ to yield $(N \cup \{o\}, E) \in P$. The prior probability assigned to such a node comes from a corpus estimation and is equal to the ratio of number of events of that type to the number of all events in the corpus.
3. *Up-existential* If existing observations are consistent with and explained by some higher level plan in the knowledge base, then add a node to the PRN representing degree of belief in this hypothesis and add causal edges from this node to the evidence. More formally, if there is a proposition o of the form $(inst\ it_1\ t_1)$ in a PRN $(N, E) \in P$, a formula in K of the form $(inst\ ?i\ t_2) \rightarrow (inst\ (slot\ ?i)\ t_3)$ such that type t_3 is consistent with t_1 , and either t_2 is an object type or t_3 is an event type, then a new hypothesis node A of the form $(inst\ it_2\ t_2)$ and a new slot filler node B of the form $(= (slot\ it_2)\ it_1)$ will be added to N . Additionally, three edges are added to the PRN: an edge from the explanatory hypothesis node A to the observation node o , and edges from each of the explanation node and observation node to the slot filler node B .

The prior probability of the node A is specified from a corpus as in the previous rule.

The CPD of the node o is specified as follows: $P(o|A) = 1$, $P(o|\neg A) = P(t_3) - P(t_3|A)P(A)$.

The CPD of the node B is specified as follows: $P(B|A, o) = 1$, and 0 otherwise.

4. *Slot-filler* If there is an existing event or object represented in the PRN that could fill the slot of an existing plan hypothesis, then add a node to the network that asserts this relationship. More formally, for some $(N, E) \in P$, if $(inst\ it_1\ t_1), (inst\ it_2\ t_2) \in N$ and $(inst\ ?i\ t_3) \rightarrow (inst\ (slot\ ?i)\ t_4) \in I$, t_1 is a subtype of t_3 , t_2 is a subtype of t_4 , and it_1 is a hypothesized plan, then $(= (slot\ it_1)\ it_2)$.
5. *Other evidence* If the antecedent of a formula in the knowledge base can be unified with propositions from the PRN, then a node representing the proposition that the consequent of the implication is true may be added to the PRN. Formally, for some $(N, E) \in P$, if nodes $\{a_1, \dots, a_n, =_1, \dots, =_m\} \subset N$ and a_1, \dots, a_n after application of the equality statements unify with the A_1, \dots, A_n of an implication $(A_1, \dots, A_n \rightarrow C) \in K$, then C plus edges from all of $\{a_1, \dots, a_n, =_1, \dots, =_m\}$ may be added to P . The probability distribution that is applied to the new node for the proposition C is not clearly specified in [CG93].

Before further clarification of the PRN construction rules via an example of their use, we first describe the focus mechanism of marker passing in order to provide a coherent

description of the system. The problem of infinite regress was mentioned earlier. Since event hierarchies are permitted to have cycles (recall the example of *going* to the airport in service of *going* to some distant destination), it is possible to fall into cyclic application of the up-existential rule. Furthermore, as plans are hypothesized the tree grows bushier, providing more opportunity for application of the construction rules. In practice, some mechanism is needed to restrict the growth of the PRNs so that they remain reasonable to evaluate and focused on reasonable hypotheses; this is the role of the marker passer.

The marker passer algorithm works by spreading marks from activated propositions (activated by observation, for instance) in the knowledge base to “nearby” propositions (propositions related by the abstraction relationship, propositions that fill a slot in the given proposition, or propositions in which the given proposition plays a role). Data obtained from corpora determine the probabilities of each of these relationships, which are used to determine at what point the spreading should stop. Whenever the spreading activity from two distinct nodes overlaps, a marked path is found between the nodes that initiated the spreading. Such paths are compared to the PRN construction rules to see if the appropriate relationships are contained in the path, and if so they are applied to the existing PRN. Intuitively, this mechanism finds the most plausible interconnections in the knowledge base that accounts for the data. Very low probability paths (corresponding to events and observations that are not likely to be related) are not considered because the marker passing is cut off by the probabilities before these paths are completed. Such unlikely relationships probably shouldn’t be considered without further evidence. Charniak and Goldman claim that their marker passing algorithm is *sound* “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 about the evidence)” ([CG93]). In other words, an unlikely hypothesis may sometimes be recommended, but a likely one will never be ignored.

The marker passer thus locates plausible hypotheses for a given set of observations. These potential hypotheses and the corresponding observations are then matched against the up-existential construction rule, and if successful, the appropriate nodes and edges are added to the PRN. The marker passer mechanism may also discover relationships between existing hypotheses and observations. Such paths are matched against the slot-filler rule in order to incorporate existing evidence into existing hypotheses.

Example

We now continue with an example and further discussion of the construction rules. Figure 3.3 and the text “John went to the liquor store” will be used as an example to illustrate the purpose and application of each rule. As the sentence is parsed, various propositions are generated regarding the semantic content of the sentence. One of these corresponds to an observation that there was a *going* event, another is the presence of a liquor store. Using the *object evidence* rule described above, nodes (*inst go1 go-*) and

$$\begin{aligned}
(inst?shopshopping-) \rightarrow (and \quad & (inst(go - stp?shop)go-) \\
& (= (agent(go - stp?shop))(agent?shop)) \\
& (= (destination(go - stp?shop))(store - of?shop)))
\end{aligned}$$

Figure 3.4: A plan for liquor shopping

$(inst\ ls3\ liquor - store)$ are added to the PRN to reflect the observations.

After the PRN has been updated with these additions, the marker passer algorithm is run to see if the observations suggest any plausible hypotheses. Assuming that there a liquor shopping plan similar to the one in Figure 3.4, the path between the liquor store observation and the going observation would be located by the marker passer and then matched against the up-existential rule. As a result of a successful match, the nodes $(inst\ ls3\ liquor - shopping)$ and $(= (go - step\ ls3)\ go1)$ would be added to the PRN, representing respectively the liquor shopping hypothesis and the role played by the observed *going* event. In addition, edges from the hypothesis to the observation, and from both the hypothesis and observation to the slot-filler nodes are added to the PRN.

The marker passer algorithm will at this point also notice a likely path between the liquor store observation $ls3$ and the previously hypothesized liquor shopping plan $ls3$. The *slot-filler* rule matches just this case, and the slot-filler proposition $(= (store - of\ ls3)\ ls3)$ is added to the PRN along with edges from the observation and the plan hypothesis to the new slot-filler node.

At this point (and it is unclear in [CG93] exactly when this occurs), an application of the *other-evidence* rule can be performed. The liquor shopping plan of 3.4 indicates that the destination of the *going* event and the store of the *shopping* event are equivalent. The equality propositions $(= (store - of\ ls3)\ ls3)$ and $(= (go - step\ ls3)\ go1)$ and the liquor shopping plan provide the appropriate components. Application of the rule adds another node $(= (destination\ go1)\ ls3)$ with edges to this evidence from the other equality propositions.

The plan hypothesis node in the resulting PRN has an associated probability that indicates the system belief in this plan as an explanation of the observations. Other plan hypotheses may be postulated if further information is provided. For instance, subsequent processing of the sentence “He pointed a gun at the clerk” would result in the addition of a robbery hypothesis that explained the same set of observations. The robbery hypothesis would initially have been filtered by the marker passing algorithm because of its low prior probability, but the additional evidence involving the gun and the threat would have completed the path in the plan hierarchy. The best hypothesis is the one with the higher probability; in this case the low prior probability of the robbery

hypothesis combined with its explanation of the *going* event as well as the gun and the threat will make the robbery hypothesis the most probable.

Evaluation

Having outlined the major operation of the plan recognition system in WIMP3, we now turn to evaluation. The combination of logical and probabilistic techniques is most certainly a step in the right direction. Simple probabilistic approaches akin to the one taken by Albrecht et al. lack the ability to recognize rich plan structure. On the other hand, purely logical approaches (ala Kautz) lack a formal basis for preferring one hypothesis over the competition. These problems found in earlier systems are addressed by the combination of logical and probabilistic approaches in this system. As far as showing the need for probabilistic information in plan recognition was the main goal of this work, it was a success. It is unclear, however, to what extent the system was a success in practical terms. The author has been unable to find any information regarding its performance. The rest of this section will address the capabilities of the system according to the criteria laid out in Chapter 2.

This approach allows recognition of richly structured plans like the logical approaches and in contrast to some other probabilistic approaches. A plan can be hypothesized as an explanation of observed events and objects. By the same token, a higher level plan can be hypothesized as an explanation for a previously hypothesized plan. The example given in [CG93] is that a *party* plan may be hypothesized as explanation for the *liquor shopping* plan.

Interleaved and overlapping plans, as well as conjunctive goals can be recognized in this framework. The example discussed in the previous section involved two hypothesized plans (shopping and robbery). Though these plans were “in competition”, it should not be hard to imagine cases where multiple plans are hypothesized that could co-occur in an interleaved fashion. Furthermore, the decision criteria (unspecified in [CG93] but discussed briefly in [CC91]) may attribute multiple plans to the agent, i.e. a conjunctive goal composed of interleaved plans. A single action may play multiple roles in interleaved plans in a manner similar to that of the *going* event in the shopping and robbery plans.

Two distinct processes address “noise” in an agent’s behavior. The first is the marker passer algorithm which will only consider hypotheses for actions that are consistent with other observations. The second is the decision criteria for selecting from among competing hypotheses. A spurious action that is closely enough related to observations to be considered by the marker passer and introduced into the PRN will still have contribute enough evidence to an incorrect hypothesis in order to have any negative effect on the system’s predictions.

Aborted plans pose a problem for this system. The system may recognize a plan while it is being enacted, but there is no mechanism described that would recognize cessation of the plan. Furthermore, the system is not equipped to perform plan recognition for extended periods of time. One source ([CC91]) indicates that it recognizes “stories” that are 1 to 4 sentences long. The growing complexity of PRNs as observations are made and the restriction that at most one instance of a plan type be hypothesized in a PRN are likely contributors to the limitation on story size.¹ Most other systems discussed in this paper (with the exception of Albrecht et al.) also lack the ability to perform continuous plan recognition. This issue is discussed in more detail in Chapter 5.

As described, this system does not provide a means to enforce temporal order in actions. The marker passer does not address this issue, and the PRN takes no notice of the order of observations. In one regard, this is due to an incomplete specification of the actions that comprise a plan. As presented, there is no allowance for ordering constraints in the operation of the system even though the predicate calculus representation of plans would allow it. It is therefore conceivable that such constraints *could* be addressed, for instance, by the marker passer. A simple filtering procedure could prevent a sequence of actions that violates an ordering constraint from triggering a hypothesis that would otherwise have explained the sequence.

A more general criticism is that this system does not take advantage of contextual information. It considers neither temporal information nor action preconditions and effects. As such, it is constrained to recognizing the plans that are present in its plan library. This restriction is admissible only to the extent that the plans in a domain can be exhaustively cataloged. In addition to requiring a detailed plan library, this system also requires that the abstraction hierarchy be tree shaped and that probabilistic data be compiled for all event types and inter-relationships. It is not implausible that humans have access to the same type of information and therefore not unrealistic for a system to require it. It is however, an expensive requirement if it must be manually compiled for each new domain.

It is difficult to assess the computational feasibility of the system, as there is little description of its performance. What is more important is the use of a focusing mechanism in order to make its computation feasible. The marker passer algorithm is an intuitively pleasing way to restrict the amount of computation performed by restricting the hypotheses under consideration. Humans seem to quickly come to a set of reasonable explanations for a given set of observations. It is highly unlikely that a search of the entire space of possibilities has been conducted to reach these explanations. Somehow the more likely ones are quickly retrieved. The marker passer algorithm quickly locates a set of plausible hypotheses without searching the entire space. Incremental use of *spreading activation* narrows the search possibilities.

¹It is somewhat amusing that Charniak and Goldman have chosen to illustrate their criticism of assuming top-level plans with an example that their own system also fails to handle. Since only a single instance of each plan type may be hypothesized in a PRN, the example of needing a *travel* plan to get to the airport in service of *traveling* to a distant destination would not be recognized.

The biggest contribution of [CG93] is its argument for a probabilistic basis to plan recognition and the resulting combination of logical and probabilistic approaches. While it does address a number of the issues that we laid down as evaluation criteria, this system does fall short in some; notably the restrictions placed upon it by its assumptions. In addition, we have seen that this, like the others, does not support continuous plan recognition and does employ a general inference procedure in tandem with its plan recognition procedures.

3.2.3 Bauer

The last of the systems that we will discuss is that of Bauer ([BP93], [Bau94a]). This system, like Charniak and Goldman’s, is a hybrid system. It uses a modal logic to encode hypotheses, a generalized form of *abduction* to recognize potential hypotheses, and a variant of Dempster-Shafer Theory (DST) as a means of choosing between them. The domain of the plan recognizer is a Unix mail reader, but the DST-based incremental preference update mechanism is a general technique that may be incorporated into other plan recognizers. Subsequent systems have used DST in a similar manner to decide when enough is known to make a prediction ([BA03]).

System Overview

In an effort to be brief, some of the implementation details will be omitted, but we hope to concisely convey fundamentals of the system. We focus first on the variant of abduction that Bauer and Paul propose in [BP93]. The basic theory of abduction warrants inference of the explanation ϕ from observation ω under theory \mathcal{T} if:

1. $\mathcal{T} \cup \phi$ is consistent
2. $\mathcal{T} \cup \phi \models \omega$
3. ϕ is a ground instance from some pre-specified set of *abducible* formulas

In other words, so long as ϕ is of the correct form, and ϕ together with the theory explains ω , then abduction warrants the inference of ϕ .

Bauer and Paul extended abduction to work for the LLP modal logic in which they encoded plans for their mail-reader domain. The abductive theory described above turned out to have a correctness criterion that was too stringent (item 2) for the temporal facets of the hypotheses that they encoded. Before an example, it is worth mentioning that LLP interprets formulas with respect to an interpretation and a subinterval. For example, the formula $\diamond\phi$ (read “sometimes ϕ ”) is true on σ precisely when there is some suffix sub-interval of σ for which ϕ is true under interpretation \mathcal{I} . $\diamond\phi$ cannot

be inferred as an explanation for ϕ because there are some models of $\diamond\phi$ (namely the one where ϕ is not true in the initial half sub-interval but is true for the second half) are not models of ϕ . They relaxed the correctness criterion by requiring only that some variation of the explanation be compatible with the observation. The existence of such a *refined explanation* would then warrant the inference of the original. The formal condition for \mathcal{P}' to be a refined explanation for \mathcal{P} with observation ω is that $\text{MOD}(\mathcal{T} \cup \mathcal{P}') = \text{MOD}(\mathcal{T} \cup \mathcal{P}) \cap \text{MOD}(\omega)$. Then for a logical theory \mathcal{T} defined over LLP, \mathcal{P} an LLP formula and ω an observation, \mathcal{P} is considered an explanation for ω if:

1. $\mathcal{T} \cup \mathcal{P}$ is consistent
2. There is a refined explanation \mathcal{P}' of \mathcal{P} for which $\mathcal{T} \cup \mathcal{P}' \models \omega$
3. \mathcal{P} is a ground instance of some pre-specified set of abducible formulas

The system appears to be initialized with a small set of predetermined candidate hypotheses. These serve as the set of initial candidate hypotheses. As each action is observed, the candidate hypotheses are checked via the generalized abduction rule specified above ² to make sure that the hypothesis could still be an explanation. After the consistency check eliminates impossible hypotheses, a variant of DST is used to estimate the likelihood of each remaining hypothesis.

Bauer and Paul describe a variant of DST that they use to value the likelihood of various hypotheses. We introduce the major components of the modified DST below, and then discuss how the components are used to evaluate hypothesis likelihoods.

evidence space (θ) This is made up of the various *basic actions* in the mail domain, those that do not abstract any others. i.e. READ, SAVE, WRITE. An abstract action is considered to be the union of all the basic actions that it abstracts, and hence is a subset of 2^θ .

hypothesis space (Ω) This is made up of the various *basic plans* in the mail domain, those that do not abstract any others. i.e. READ-AND-WRITE-ALL, READ-AND-SAVE-ALL, READ-AND-STORE-ONE. An abstract plan is considered to be the union of all the basic plans that it abstracts, and hence is a subset of 2^Ω .

evidence bpa ($c : 2^\theta - \emptyset \rightarrow [0, 1]$) This bpa describes an observation. It takes as input all subsets of the evidence space (so that abstract actions may be observed) and maps each to the likelihood that it was the observed action. The sum over all subsets must be 1. Note that this need only be defined for certain subsets of 2^θ , namely those that correspond to the basic and abstract actions. A typical example of an observation in this domain would map READ to 1.0 and everything else to 0, because observations are perfect in this domain.

²Bauer and Paul indicate that the regular abduction rule (instead of the generalized abduction rule) is applied to abstract plans in the hierarchy

hypothesis bpa ($m_i : 2^\Omega - \emptyset \rightarrow [0, 1]$) This bpa describes the system's belief in each of the candidate hypotheses. It takes as input all subsets of the hypothesis space (so that abstract plans may be assessed) and maps each to the likelihood that it is the explanation for the observed actions. The subscript indicates that this is updated each time that a new observation is made.

rule frame ($R = \langle \theta, \Omega, \Delta \rangle$) This contains the rules for converting input observations into update information for the hypothesis bpa. $\Delta : (2^\theta - \emptyset) \rightarrow 2^{2^\Omega \times [0,1]}$ maps an observation to the hypotheses that the observations supports, indicating the degree of support for each hypothesis. The sum of likelihoods from each input must sum to 1. This is called the rule frame because the Δ function can be thought of as rules of the form $E_i \rightarrow A_{E_i,1} \langle S_{E_i}(A_{E_i,1}) \rangle, A_{E_i,2} \langle S_{E_i}(A_{E_i,2}) \rangle, \dots$. A typical rule frame entry might map SAVE to ¡READ-AND-STORE-ALL, 6_i , ¡READ-AND-STORE-ONE, 4_i .

When an observation is made (in the form of a bpa c), the rule frame is used to construct a new bpa r which represents the *hypothesis strength* induced by the rule frame R and evidence c using the hypothesis strength function:

$$r(X) = \sum_{\emptyset \neq E \subseteq \Theta} s_E(X)c(E),$$

where $s_E(X)$ is one entry in the value of $\Delta(E)$. Once the hypothesis strength r for an observation has been computed, it is combined with the current belief bpa m_i using Dempster's rule of combination to yield m_{i+1} , the updated belief in candidate hypotheses. In essence, an uncertain observation (c) induces an evidence mass on the hypothesis space (r) which is then used to update the system beliefs.

Evaluation

The major contribution of this work is a recognizer independent method for assessing hypothesis likelihoods. To use the DST variant outlined in [Bau94a], a recognizer must have a complete plan hierarchy (an assumption made by the system) and be able to describe its domain in terms of an *evidence space*, *hypothesis space*, and a *rule frame* of the same kinds that Bauer provided.

The plan recognizer described in [BP93] should in principle be capable of recognizing arbitrarily complex plans. They have, however, implemented a rather restrictive set of heuristics in order to reduce the search space. The most limiting restriction is the a priori restriction of the generalized abductive inference procedure to a few static plans. The DST based ranking scheme cannot handle arbitrary plans; it is constrained to evaluating those that exist in the plan hierarchy.

Neither component of this system is capable of handling interleaved goals. Since actions are used to filter hypotheses in the recognizer as they occur, actions from interleaved plans may be incompatible and result in an empty hypothesis set. In the hypothesis

valuation, the sequence of actions is assumed to be from a single plan. It may be the case that the plans from which the actions are drawn will both increase in likelihood, but it is not clear that this will be the case. In particular, there are situations for which the Dempster rule of combination is undefined, namely those situations where the bpas are have no overlapping mass.

The recognition component is not robust in the face of noise. All of the inputs are perfectly specified because of the domain, so it doesn't have to deal with uncertain input. Unintended or erroneous actions would likely eliminate valid hypotheses due to its inconsistency filter. The hypothesis valuation portion is more robust to noise, but there are no guarantees that an update bpa won't be inconsistent with the current hypothesis beliefs.

The LLP logic is certainly capable of representing ordered and partially ordered plans, but it appears that the recognizer implementation will only be capable of recognizing strict order. The rule frames will decide how temporal order affects hypothesis valuation. It is unlikely that these will contain history information, and the hypothesis valuations will therefore order independent.

The recognizer component of this system should be tractable for the domain described because of its small size and the heuristics implemented. In general, though, it will face the same problems that other logical systems face, a huge search space. The DST calculations performed for the hypothesis valuation component are tractable because arbitrary subsets of the evidence and hypothesis space are not considered. Only those highly structured subsets that correspond to abstract actions and plans need to be considered in computing the new bpas.

3.3 Recent Work

The author has performed some recent work on goal recognition, examining the benefit of incorporating type information into an existing goal recognizer. This work builds on the work of Blaylock and Allen [BA03]. In this work, Blaylock and Allen use statistical methods to quickly and incrementally recognize the goals of users in the Linux domain. They formulate the problem of goal recognition as that of finding the most probable goal given an observed action sequence. Formally, the problem is defined as finding the goal g such that:

$$g = \operatorname{argmax}_G P(G^S, G^{1,q} | A_{1,n}^S, A_{1,n}^{1,r}) \quad (3.1)$$

where G^S represents the goal schema, $G^{1,q}$ represent the goal parameter values, $A_{1,n}^S$ represent the action schemas for the observed actions, and $A_{1,n}^{1,r}$ represent the parameter values for the observed actions. This probability can be decomposed into forms that are

easier to approximate and work with via the following sequence of transformations:

By the chain rule, the probability in (3.1) becomes:

$$P(G^S | A_{1,n}^S, A_{1,n}^{1,r}) P(G^{1,q} | G^S, A_{1,n}^S, A_{1,n}^{1,r}) \quad (3.2)$$

At this point, a simplifying assumption is made in [BA03]: that goal schemas are conditionally independent of the observed action parameters given the observed action schemas. This assumption neglects to take advantage of the additional information provided in the action parameter values, so I do not make this assumption. The probability in (3.2) suggests a two step approach to goal prediction: first predict the goal schema, then predict the parameters based on the prediction of the goal schema. This project concerns itself solely with improving the performance of predicting the goal schema with the additional parameter information (since parameter value information is already incorporated into the parameter recognizer described in [BA03]). Focusing on the goal schema recognition problem, we continue with first term of (3.2):

$$P(G^S | A_{1,n}^S, A_{1,n}^{1,r}) \quad (3.3)$$

By Bayes Rule,

$$P(G^S) P(A_{1,n}^S, A_{1,n}^{1,r} | G^S) \quad (3.4)$$

With a bigram assumption made on the action sequence,

$$P(G^S) \prod_i P(A_i^S, A_i^{1,r} | G^S, A_{i-1}^S, A_{i-1}^{1,r}) \quad (3.5)$$

This term is completely analogous to the one derived in [BA03] except that there are parameter values in the conditional probability of (3.5). It is computed using the same machinery but with a different distribution.

Action Signatures

Define the *action signature* for action A with parameters $P_{1,n}$ to be the sequence $(A, \text{type}(P_1), \dots, \text{type}(P_n))$, where the types are obtained by inspection of the parameters and come from some fixed type hierarchy. For example, the action signature of the linux command 'ls *.txt' would be (ls extension), assuming that the string '*.txt' is determined to be an extension.

Action signatures abstract away from individual parameter values, and allow us to approximate the conditional probability in (3.5) from a plan corpus. Abstracting from the actual parameter values is necessary to avoid the pitfalls of data scarcity. Even with this abstraction, it is still possible that the space of action signatures for a particular action schema be large. When estimating the conditional probability in (3.5) it is sometimes necessary to backoff to the action schema bigram, and sometimes even further to the

N-best(τ)	1 (0.3)	2 (0.7)
Precision	31.5%	63.6%
Recall	18.5%	31.5%
Convergence	36.5%	54.7%
Convergence Point	3.8/5.9	3.7/6.7

Figure 3.5: Original Recognizer Results

N-best(τ)	1 (0.3)	2 (0.7)
Precision	37.9%	66.2%
Recall	22.7%	34.3%
Convergence	43.5%	54.3%
Convergence Point	3.3/5.8	3.7/6.9

Figure 3.6: Modified Recognizer Results

unigram probability. I used a backoff strategy that backs off from an action signature bigram to the action bigram. If further backoff is required, it looks at smaller n-grams, preferring action signature n-grams when enough data is available.

Using this approach I was able to improve upon the results originally found in [BA03]. See Figure 3.5 for the original results and Figure 3.6 for the modified results.

Goal recognizers perform, in some senses, a simpler task than plan recognizers, by ignoring the plan structure and directly proceeding to infer the top level goals of the user. This is often a useful task as a first step in filtering options for further inference. Treated as a complete system, however, it suffers from many of the same faults as the system of Albrecht et al. It can only infer a single goal at a time, it does not infer a rich plan structure, it does not take temporal relations among actions into account, etc.

3.4 Other Work

In the discussion of Charniak and Goldman’s system, we saw a “successful” combination of logical and probabilistic inferences for achieving plan recognition. There is a body of literature concerned with integrating BN style reasoning and first-order logic (e.g. [Poo93], [Poo97], [Kol98], [Pfe00], [Poo03]). In fact, any formal theory of first order probabilistic reasoning would be beneficial for plan recognition, which until now has had to use ad hoc methods to combine the two types of inference.

3.5 Summary

This chapter has discussed a number of plan recognition systems with the goal of providing a good sampling of the approaches that have been applied to plan recognition. While the recognizers were divided into logical and probabilistic categories, many contained elements of both. We saw a number of problems that many of the systems shared, including search space size and noisy data. We turn now to a survey of these and other common problems that plan recognition systems face.

Chapter 4

Problems

We now recapitulate some of issues that plan recognition systems have had to address, using the systems surveyed in Chapter 3 as a guide. These issues include managing the search space, handling noise in the data or recognition process, acquiring the knowledge required for system operation, and dealing with increasingly complex domain requirements. We examine these in more detail below, and additionally suggest two topics for future research: continuous plan recognition and integration of plan recognition and reasoning.

4.1 Narrowing Hypotheses

Of the systems surveyed, the system with the potential for the largest search spaces is that of Allen and Perrault ([AP80]). Recall that their recognizer uses backward goal chaining to infer potential plans. Like some of the others, this system may hypothesize that a user wishes to perform a goal G when it recognizes that the agent wishes to perform one of G 's sub-goals. But it may infer that the agent wishes to achieve G from other observations as well. In particular, observing that the agent wishes to achieve one of G 's preconditions is sufficient to hypothesize that the agent may wish to achieve G . Modeling the preconditions and effects of actions in the plan hierarchy thus allows novel plans to be inferred in exactly this manner. The ability to detect novel plans is not present in any of the other systems, and adds significant complexity to the search space. As such, narrowing the hypothesis space is of paramount importance to their system.

Allen and Perrault addressed this problem using heuristics, a prominent technique in plan recognition ([Car01]). Some of their heuristics were based on rationality and coherence assumptions. One such heuristic downgraded the valuation of a hypothesis whenever it was one of multiple alternative inferences. The underlying motivation for such devaluation was that a speaker would not have intended an ambiguous plan. This assumption is especially appropriate for discourse situations, where ambiguity reduces conversational

efficacy. Another heuristic was based on on expectation; a small number of high level goals were postulated and used to constrain the search space. While this particular implementation was limited in scope, the general technique of constraining inference based on expectation is a useful one.

By narrowing consideration from many possible alternatives to a few *expected* ones, a recognizer limits the amount of inference that must be performed. Expectations embody some knowledge that allow an agent to focus inference. How this knowledge is obtained is unimportant from the perspective of the plan recognizer so long as it does an effective job of narrowing the search space.¹ In Allen and Perrault’s system, the knowledge was provided by humans and was hard-coded into the framework of the system. But it could just as easily have been provided as part of a knowledge base. Furthermore, instead of being required to use a fixed set of goals as expectations, a system could reason from its current knowledge of the situation in order to come to a set of expectations. This notion of incorporating more general inference with plan recognition will be discussed further in Chapter 5.

Charniak and Goldman ([CG93]) use domain specific expectations to limit the amount of inference required. Their algorithm constrains itself to hypotheses that can be found along paths connected by the marker passing algorithm. The probabilities used by the marker passer algorithm are the instrument of expectation. Recall that each observed action or entity begins a round of marker passing. Marks are passed from an observation proposition P to other propositions in the knowledge base (treated as a semantic network) that are related in one of the following ways: the proposition plays a role in P , P plays a role in the proposition, or the proposition and P are related by abstraction. Each relationship in the semantic network has a probability associated with it which is estimated from corpora. Mark propagation continues until the “probability” along a path (determined by the probabilities of the relations in that path) fall below a certain threshold. Thus, the probabilities determine which propositions will be considered as hypotheses. The probability associated with each relationship can be thought of as quantifying how expected the relationship is, and the combination of probabilities along a path as quantifying how likely (expected) a relationship between the start and end nodes is.

The methods described above are applicable to any user. It is possible however, to use information that is specific to a group of users or to an individual. Carberry ([Car01]) cites the work of Gertner ([GW96], [Ger97]) as examples of work that use knowledge about specific groups of users. In Gertner’s system, physicians’ plans were inferred during emergency trauma situations. Since the physicians are generally considered to be experts and therefore unlikely to make mistakes, the system preferred hypotheses with fewer “incorrect” goals. Systems that interact frequently with specific users may acquire

¹Effectiveness in this context has two distinct components: the degree of search space reduction, and the degree to which the results approximate the results that would have been attained without expectation-guided search.

enough data to learn individual preferences. Carberry indicates that Ardissono and Sestero ([ABL96]) use beliefs about the user in conjunction with stereotypes and analysis of previous user actions in order to filter out unlikely hypotheses. Bauer ([Bau94b], [Bau95a], [Bau96a]) treats repeated patterns of user action as preferences, and uses these preferences to help choose from among competing hypotheses.

Lesh ([Les97]) takes user-specific reasoning one step further by considering how adapting the recognizer affects user-specific recognition. Lesh discusses a recognizer independent algorithm to improve performance for individual users. The algorithm maintains the notion of recognizer specific adaptations that can be used to improve performance. In the case of the BOCE recognizer on which this algorithm was tested, adaptation was performed by adding or removing goals in the set of *potential goals* and the set of *background goals*. The potential goals are those that could be attributed to the user and the background goals are those whose observation can be ignored. The algorithm uses hill-climbing in conjunction with a recognizer specific scoring function (based on *accuracy* and *coverage*) to select from among the set of possible adaptations based on a set of observed action sequences.

Both of these user-centric approaches exploit sources of knowledge that domain specific expectations (like those of Charniak and Goldman) and domain independent expectations (like those of Allen and Perrault) do not, namely information regarding individuals or groups of individuals. Bauer's approach is perhaps the nicest, because it learns during operation, and does not require offline modification of the recognizer (c.f. Lesh), or hard-coded preferences (c.f. Gertner).

TIE IN?? Again, it seems that this is an opportunity where the ability to incorporate general inference could help. Being able to reason about individual agent preferences as interaction takes place would provide the benefits of

4.2 Noise

It is arguable that plan recognizers have not yet become as effective as they can be because of limited capacity to deal with noise. Systems that interact with users in realistic environments have to be able to handle noise robustly in order to be useful. In this section we examine some of the various types of noise that make the plan recognition process more difficult than it is under ideal circumstances. Noise in the observed actions, in the knowledge base, or in the hypothesized plans of the recognizer cause difficulties in recognizing user plans. We'll address issues from each of these categories and indicate what steps have been taken to remedy them.

Noise in the observations can take multiple forms. An agent attempting to achieve a

goal may perform some actions that are irrelevant to that goal, and can therefore be considered a form of noise. This type of behavior was an issue with the system of Albrecht et al. ([AZN98]). Users performed many actions that were unnecessary to their current quest whether out of ignorance or choice. The technique of relying on statistical correlations between actions and goals as learned from data proved to be useful in the MUD domain. Individual actions and patterns of action are associated strongly with a quest when many users performed them in pursuit of the quest. Those actions that occur frequently in many quests distribute the probability among all of the quests in which they occur; as a result they have lesser discriminatory power than those action (sequences) that are correlated with fewer quests because they contribute less to each individual action. Noisy actions, those that don't provide much information about the quest, tend to fall into this category. One of the examples reported as noise was user conversation. Conversation is not likely to play a role in the user's quest because it is an event that is likely to occur frequently during all quests. For this reason it will hold little discriminatory power and little effect on predicted quest probabilities.

A second form of noise in the observations is user misconception. The observed agent may intend to achieve some goal G and be executing a plan to do so, but fail to achieve G because part of the plan is misconceived. A plan recognizer observing an agent executing such a faulty plan may not be able to infer the agent's goal because there is no logical plan to achieve G which involves the observed sequence of actions. Pollack ([Pol90]) addresses this issue by treating plans as mental attitudes instead of as data structures. In this framework, an agent believes that an action has certain preconditions and effects, and thus may construct a plan that conforms to their beliefs yet does not accomplish the goal because of misconceptions on the agent's part. This treatment allows the plan recognizer to systematically alter its own beliefs about the preconditions and effects of actions (in pre-specified ways) in order to account for otherwise unexplainable behavior. This approach is superior to the inclusion of *buggy plans* ([BB78]) because of its generality. This approach allows the plan recognizer to perform some set of inferences about the way that action preconditions and effects interact with each other in formulating plans, and thus is another instance of incorporating general reasoning into plan recognition. Carberry ([Car01]) suggests that extension of Pollack's simple variations requires extensive research. As more variations are allowed, the search space becomes more and more complex. Thus, it is unrealistic to consider all possible variations. Perhaps Pollack's methods can be extended by ideas borrowed from the continuous planning field ([PH99], [dDCLOW99]), where it is important to limit inference for effective performance. Similar methods might strike a good balance between reasoning about plan variations (representing misconceptions of the observed agent) and reasoning about other explanations for the action. Again it appears that more general reasoning capabilities offers opportunities to generalize existing plan recognition capabilities.

A third source of noise that arises in the observations is caused by the interaction between the user and the environment. Sometimes an agent's action will not have the desired effect because of the situation at the time that it was performed. In these cases, the user

may attempt an alternate method of achieving the same goal that the original action was meant to achieve. This “reactive” behavior can be interpreted as a form of noise because the reactive portion of the agent’s behavior was not part of the agent’s original plan. The original and modified plans may diverge significantly, causing difficulties for recognizers to combine the observation sequences from before the error corrections and afterward. Though this type of behavior appears to be noisy behavior, it is possible that more might be extracted from such observations based on principled assumptions about the underlying human behavior. The next chapter will discuss this topic, assumptions about the structure of human planning behavior, as a potential topic for future work in plan recognition.

The knowledge base may be noisy in the sense that the plan hierarchy is incomplete; i.e. it does not contain all possible goals or all possible ways of achieving each goal. In this regard, backward chaining algorithms such as the one presented in [AP80] are better prepared to recognize plans that do not occur in the plan library. Other systems (c.f. [Kau91], [CG93]) do not have this capability, and can thus recognize only those plans that are explicitly encoded in the plan library. Carberry ([Car01]) indicates that the inference of novel plans is a difficult problem which has not received much attention. She cites Cohen et al. ([CSSvB91]) as providing a means of updating the system’s plan library with novel *recipes*, but still lacking a means of inferring them. Still, this provides a crucial facility for systems that will change their performance over time.

Finally, the plan recognizer may introduce noise by attributing incorrect plans to the user. Maintaining only a single best hypothesis of an agent’s current plan is risky for this very reason. Probabilistic systems may provide some relief by simultaneously considering many hypotheses, however they still face the same issues regarding user misconceptions and noise due to the knowledge base. Eller and Carberry ([EC92]) address the issue of misconception on the part of the plan recognizer, and its effects on subsequent plan recognition. They propose an approach that considers possible misconceptions on the part of the observed agent and on the part of the plan recognizer. As with Pollack’s work, these variations are limited in scope.

In summary, the noise introduced by agent actions, incorrect inference and incomplete data causes serious problems for plan recognition systems. These sources of error cause drastic expansion of the search space, especially when the number and complexity of principled variations for misconceptions increase. The problems briefly mentioned in this section are by no means solved, and are still areas of active research.

4.3 Knowledge Acquisition

Many plan recognizers make use of a plan library as an input to the plan recognition process (all of the systems discussed in Chapter 3 except for that of Albrecht et al.). Recall that plan libraries contain the recognizer's knowledge about how goals can be achieved. Specifying a plan hierarchy is a time consuming task that has traditionally been the responsibility of the system designers. Recent work has begun to move offload some of this burden onto machine learning techniques. In this section we briefly review the work of Bauer ([Bau98b], [Bau98a], [Bau99]) in this area.

In [Bau98a], Bauer addresses the issue of creating a plan library from observed action sequences. His primary motivation is the “nontrivial task” of creating an adequate representation of plans in a given domain, particularly for situations where the actions are known but it is difficult to come by plan decomposition information. As a sample domain, he cites intelligent application help systems that are developed after the software system has been implemented. In these situations, the command semantics may be unknown, and observation provides a tool to learn them. His method is based on the assumption that actions which are *necessary* for the completion of a plan will be present in all action traces pertaining to that plan. As input, his system takes a set of action sequences that correspond to a single plan, as well as some domain specific information regarding domain object structures and type/action abstraction information. Any or all of the pieces of domain information may be empty and the algorithm will still produce results, albeit of lower fidelity.

This system considers plans to be action sequences as a set of actions with associated temporal and structural constraints. A temporal relationship between two actions indicates that one occurred before the other. A structural constraint is an arbitrary proposition relating the objects manipulated in actions. An example structural constraint is that the first argument of action A_1 and the third argument of action A_3 be equal.

The algorithm works for a pair of action sequences by pairwise *joining* of the actions in each sequence. The *join* operator yields the *most specific subsumer (mss)* of the entities or a null value; the null value results when this information is unknown (limited domain information)² or when the *mss* is the root of the action hierarchy. In this case, the most that one can say is that both are members of the action hierarchy, but it is impossible to specify a more specific category that includes both. The *join* operator is defined similarly for object types, and it is used to abstract the parameter types as actions are joined. The final components to consider are the temporal and structural edges of the action sequences. Two edges can be *joined* if the relationship on each edge is compatible, and the sources and destinations of each edge can be *joined*. This operation produces a potentially large set of abstractions that must somehow be interpreted as an abstracted

²equality of types will yield the type of both actions even when no domain abstraction information is provided

action sequence. Maximal subsets of these abstractions subject to some consistency constraints are called *valid joins*, and represent the set of abstractions that can be obtained from the initial action sequences. Bauer indicates that the *join* operator is commutative and associative, and hence it can be incrementally applied to a set of action sequences. The resulting abstractions can be considered plan decompositions for the goal that the original action sequences were meant to achieve.

At this point Bauer introduces the notion of *abstraction bias* as a way to select between multiple *valid joins*. He discusses two possible biases, and describes good approximations that can be easily computed. The first bias is toward restrictive plan decompositions, which allow little deviation. Such a bias may be justified when a recognizer wishes to capture deviation from expected behavior quickly in order to remedy suboptimal behavior. This bias can be heuristically implemented by selecting those decompositions that contain larger numbers actions and constraints. A second bias is toward less restrictive plan decompositions, which will be more lenient in tolerating spurious actions. This bias may be justified in a help system context where it is desirable to recognize user plans so long as their behavior is consistent with the currently hypothesized plan. In this case, we wish to be as lenient as possible. A similar heuristic regarding the number of actions and constraints can be used in this situation as well. In the previous case, we wish to maximize the heuristic value, and in this case we wish to minimize it. This heuristic yields plan decompositions that are nearly optimal based on the informally discussed criteria of restrictive and lenient recognizers; see [Bau98a] for details.

This algorithm can find and remove spurious actions and temporal/structural constraints by evaluating multiple action sequences for a particular goal. One consideration is that the fidelity of plan decompositions are drastically reduced when hierarchy knowledge is unavailable (recall that null values are produced by *join* if no abstraction information is present and the items are not equal). As a result, this system is most effective when that information is present. As Carberry notes in [Car01], determining this taxonomy information is often closely related to creating a plan hierarchy in the first place. Furthermore, plans that admit varying means of being accomplished will have very different action sequences for which it may be the case that little or no information can be extracted. To deal with this, Bauer uses a heuristic based on the number of *joinable* actions in two action sequences to determine whether or not the algorithm should be applied. Different categories distinguished by the heuristic correspond to different ways of achieving the same plan/goal. Still, Carberry seems justified in saying that “this work seems most appropriate for domains where data can be easily collected and where goals are achieved by a small set of alternatives”.

Other work in automated knowledge acquisition is that of Lesh and Etzioni ([LE96]). They address the same issue of attempting to learn the structure of plan hierarchies; their method of using goal and plan biases (principles of preference) to construct a plan library still requires that basic domain knowledge be encoded beforehand ([Car01]). Probabilistic approaches may also benefit from machine learning techniques. Automati-

cally learning the structure of Belief Networks removes some of the burden from system designers, but as Carberry indicates, “it is unclear how applicable this might be for large-scale plan recognition with complex influences among the actions and how extensive a training set will be required”.

4.4 Scale

Issues of scale must be addressed in order for plan recognizers to achieve ubiquity. New techniques are often developed in conjunction with small problem spaces in order to demonstrate their usefulness, and as such, much of the relevant literature has implicitly ignored scale issues even though most researches acknowledge their importance. Making the transition to practical systems that operate in real-world settings necessitates attention to issues of problem size. These issues include the number of goals that the system can recognize, the number, granularity, and relationships between actions in the domain, and the length of time for which the recognizer must operate.

There is a clear relationship between the number of goals that a recognizer is capable of recognizing and its applicability. A system capable of recognizing more goals will be applicable to more complex domains than a system capable of recognizing fewer goals. Multiple factors may affect the number of goals that a system can recognize; the complexity of the plans that achieve the goals, the number of ways to achieve goals, the complexity of the computations performed by the recognizer, etc. For the purposes of this discussion we assume that a representation for each domain is fixed, thus avoiding differences introduced by domain models. We are interested in the intrinsic differences between recognizers instead of variations introduced by representations.

It becomes more difficult for recognizers to infer goals when the number of possibilities increases because there are more explanations to sift through. The logical approach must explicitly consider each of these additional possibilities or filter them via some mechanism. Due to the nature of backward chaining algorithms, these additional possibilities may be considered many times in the context of different plans hypothesized to explain the observed actions. This results in a computational expense as well as necessitating more evidence to select from the additional possibilities. In probabilistic systems, the probability mass must be distributed among a larger number of possibilities. Having more goals in the system will in general provide more ways of explaining individual actions; this has the effect of distributing the evidential contribution of each action among more goals, and hence decreasing its net effect on any particular goal. Thus, reaching decision thresholds may take longer, requiring more observed behavior before being able to make a prediction. Additionally, increasing the number of goals requires an increase in the amount of data required to initialize the system probabilities. Similarly, an increase in the number of observable actions can make the recognition process more difficult. Backchaining algorithms suffer from the increased number of paths that can be traced

from observed actions to potential goals. Probabilistic systems suffer from increased data requirements and additional entries in the conditional probability tables.

Albrecht et al. ([AZN98]) handle a large number of possible actions and allow extended periods of observation. Almost 5,000 unique actions were observed in their training data, and typical observation sequences ranged between 25 and 500 actions. This is a significant step forward in addressing larger domains (c.f. the cooking domain of Kautz [Kau91]). However, this achievement does not come without cost. The system was able to recognize only simple top level quests and could not recognize simultaneous quests.

Lesh and Etzioni ([LE95]) discuss *version spaces* as a way to deal with large goal sets. The version space is a potentially compact way of representing the entire goal space. Two sets of goal schemas are maintained, the set S of the most specific goals and the set G of the most general goals. In this context, a goal G_1 is said to be more general than a goal G_2 if every action sequence that is consistent with G_1 is also consistent with G_2 . Then every goal in a domain is a member of G , a member of S , or somewhere *between* G and D . A goal is said to be between G and S if it is more general than some goal in S and less general than some goal in G .

The intuition behind version spaces is that it is not important to enumerate every element of a very large set. In many cases it is sufficient to be able to determine when the set has collapsed to zero elements (no goal is consistent with the observed action sequences), one element (a unique goal is consistent), or the case where all elements share a common subgoal. Version spaces provide a framework that allows these conditions to be checked more efficiently than if all goals were explicitly represented. The first condition (no consistent goals) can be checked by testing to see if the sets S and G are empty. The second condition (a unique consistent goal) occurs when S and G are both singleton sets and contain the same element. The third condition (a common subgoal exists) occurs when S is a singleton set and not equal to the set G . In this case, all goals that are still consistent with the observed action sequence are more general than the unique member of S , and hence this goal can be considered a subgoal of all remaining goals.

This treatment of goal recognition makes some strong assumptions about the plans of agents that it will observe. It assumes that the agents will not make any mistakes and will not pursue multiple goals simultaneously. Both of these situations could generate action sequences for which no single goal is consistent, and hence the algorithm presented by Lesh and Etzioni would not be able to find a goal to explain the agent's behavior. Additionally, it is unclear how to efficiently compute whether one goal is more specific than another in the general case. The specific class of goals that they represented was rather limited. Further work would be required to make this method amenable to real-world plan recognition problems.

In the next chapter, we will address one further issue of scale: the ability of a recognizer

to operate over an extended sequence of observations. The majority of systems known to the author are ones that can operate on only a small number of observations. The one exception that we have examined is the system of Albrecht et al. ([AZN98]), which can handle extended observation sequences.

4.5 Summary

All of the issues described above are still open issues; any plan recognition system must address them in some fashion. Solutions to the complexity problem for logical systems have thus far been based on heuristic methods of filtering less desirable plan hypotheses. Probabilistic systems have tended to maintain probabilities for just the set of high level goals instead of for all possible ways of achieving those goals, and thus have avoided some of the difficulties experienced by backward chaining systems. Hybrid systems can use either or both approaches. Complexity problems become even worse when additional considerations of noise (observer misconception, observed agent misconceptions, unnecessary actions, etc.) are taken into account. The sources of knowledge for recognition systems can be costly to obtain and to maintain; machine learning techniques have been applied to automatically building plan hierarchy information from some more primitive domain information, and methods of automatically updating plan hierarchies with new information have been explored. As plan recognition becomes more mature, it is applied to ever more complex domains. In order to perform well, issues of scale must be addressed.

In this discussion of plan recognition problems, we've mentioned a few important ideas or observations that will be topics for the next chapter. The first is the notion that there are multiple places in the plan recognition process where general reasoning can play a role. Recognition systems augmented with general reasoning can be more flexible in application of filtering strategies and in handling novel situations. The second is the observation that most plan recognition systems to date have been limited in the length of observation sequences that they can handle. Finally, we've noted a number of times throughout this paper that recognizing the structure of an agent's plans is in many regards a harder task; specifying all the details and fitting all of the observed actions into a coherent pattern can be tricky and sometimes impossible. Analyzing some of the underlying assumptions about user plans may shed light on new roads to addressing some of the problems mentioned above.

Chapter 5

Future Directions

The problems discussed in the previous chapter are still open, and as such, provide fertile ground for future research. Some of these problems have had more research devoted to them than others; methods to address the complexity of the search space have been necessary from the beginning. Others have seen less research; issues of scale and dealing with noise have seen comparatively little. This chapter introduces some areas for research that fall into the latter category, and are of particular interest to the author. These topics will of necessity relate to one or more of the previously described problems, but each introduces an element that has not previously been addressed. Topics regarding assumptions in plan recognition, continuous plan recognition, and the incorporation of general inference into plan recognition are motivated below.

5.1 Assumptions about Human Planning

As planning agents, we make plans to a level of detail that is sufficient for our purposes. Plans for the future that are too detailed result in wasted thought as we are forced to respond to the circumstances surrounding the planned activity. Plans that are under-specified may not be capable of being fulfilled because the means for accomplishing the goals have not been reasoned about in a timely manner. Bratman discusses the need for partial and hierarchical plans in [Bra90]:

The strategy of settling in advance on such partial, hierarchically structured plans, leaving more specific decisions till later, has a pragmatic rationale. On the one hand, we need to coordinate our activities both within our own lives and socially, between lives. And we need to do this in ways compatible with our limited capacities to deliberate and process information. This argues for being planning creatures. On the other hand, the world changes in ways we are not in a position to anticipate; so highly detailed plans about the far future will often be of little use and not worth bothering with. Partial, hierarchically structured plans for the future provide our compromise

solution.

Turning to a concrete example of our tendency to plan partially, take the example of an individual that is planning the activities of the day. The agent may plan to “go to work” and then “pick up the children” and then “return home”, without ever explicitly making a plan for the actual routes that will be traveled that day. One might argue that even in this case the details of the plans are being considered. So consider a hypothetical world for this agent, a world where every vehicle is each day assigned a set of roads that it is unable to travel on. Furthermore, this assignment can not be known in advance. The only way to find whether or not a road is open to the vehicle is to attempt to drive on it. In this case, it is futile for the individual to plan a route to work, to pick up his children, or for any other task. All traveling plans (at the beginning of the day) will necessarily be at the level of “go to X”. While this example may be far-fetched, it is certainly not beyond our capabilities to reason in such a situation. Should we ever find ourselves in such a city, we would make do by planning a route reactively. The fact that we can so naturally adapt our thinking to function in this scenario suggests that we are comfortable with planning at an abstract level of detail and interactively generating the rest of the plans as required.

So we are capable of planning partially, and this may even be our preferred mode of operation. Indeed, it seems that this is almost certainly the case. Planning every detail of every plan in advance would require more time and patience than most are capable of. A natural question to ask is whether or not the assumption made by traditional plan recognition models (that the agents being observed are enacting complete plans) has any adverse consequences. Two observations can be made immediately. The first is that ignoring the partial and hierarchical facet of human planning behavior leads to inaccurate plan recognition models. What types of error are introduced by this type of erroneous assumption? Will more accurate models, ones that acknowledge the incomplete and reactive nature of human planning, yield better predictive capabilities? Further research is required to illuminate the consequences, and to develop theories of plan recognition that formally account for reactive agent behavior.

The second observation is that models which don’t incorporate knowledge about the partial and hierarchical nature of human planning are ignoring a potential source of information about “noisy” behavior. We observed that “reactive behavior” could be interpreted as a form of noise in the previous chapter. Systems that ignore the source of this noise deprive themselves of a principled way of making sense of it.

5.2 Continuous Plan Recognition

Another issue that has received relatively little attention is the size of the observation sequences that a plan recognizer can handle. Most systems have been limited to a small

number of input observations. Others that are capable of handling extended sequences make strong assumptions about the goals that they will recognize (the system described in [AZN98] recognizes one goal at a time and the system is informed when the goal has been accomplished). In order for the potential of plan recognition to be realized, recognizers must be able to deal with complex observation sequences for extended periods of observation, i.e. continuously. When this mode of operation has been achieved, then the output of the plan recognition process may be used as input to other continuous processes that occur in tandem.

As motivation for this capability, we turn to recognizing the plans of a collaborator in task-oriented dialogue. Consider the TRIPS system, which in one application must interact with human users in order to develop emergency evacuation plans. In this situation, the system must track an extended conversation and be able to recognize multiple distinct and hierarchically related user goals. The user may have a top-level intention of rescuing all people in the vicinity of some danger, and many subgoals for achieving it. Throughout the course of the dialogue, the user may formulate new goals and abandon existing ones as options are discovered or eliminated. A plan recognizer in this situation must be able to robustly determine shifts in topic corresponding to the different goals that the user has. Additionally, the user input may sometimes provide ambiguous input, causing the recognizer to make an incorrect inference. The system must be able to recognize as quickly as possible that it made an incorrect inference and be able to recover gracefully from that mistake with as little waste of human effort as possible.¹ This example suggests two important lines of research for continuous plan recognition: distinguishing transitions in user goals and ability to perform robustly in the face of noise.

Recognizing multiple goals through an extended interaction is a difficult task. To date, most recognizers have ignored this issue completely by limiting the input to problems of the appropriate size ([AP80], [CG93]) or have been explicitly told when each goal had been achieved ([AZN98]). Once techniques have been developed that manage the complexity, the context provided by an extended interaction may actually provide better ability to focus further inference. Furthermore, failure to explain a new observation via the history of expectations provides evidence that a new goal should be recognized. Research by Lambert and Carberry ([LC91]) regarding incremental recognition of domain, problem-solving, and communication plans may serve as a starting point for this research. Other research by Rose et al. ([RELED95]) regarding complex focus shifts is also relevant. The literature on continuous planning may provide additional insight on mechanisms for balancing resources (time spent in inference) and recognizing cues that signify further inference is necessary (i.e. expectations are no longer met and a new goal must be inferred).

Perfect plan recognition is not possible because there is not a one to one correspondence

¹Wu ([Wu91]) contends that plan recognition systems should not be passive in this regard, that they should actively address this situation by querying the user. While this principle may be appropriate when applied judiciously, excessive use may hinder the problem solving process instead of helping it.

Agent1 What are you doing this weekend?

Agent2 My brother graduates.

Figure 5.1: A sample conversation

between action sequences and intention hierarchies. In many cases, multiple plans will justify a particular sequence of observed actions. So plan recognizers will necessarily make wrong predictions at some point. Recovering from these mispredictions is a capability that must be included in any long-term recognition process. Work by Eller and Carberry ([EC92]) provides an initial investigation into the problem of recovering from incorrect hypotheses about the agent's plan, but it provides only simple variations of the recognition system's beliefs about the agent's plan.

If plan recognition is performed continuously in real-time, then it can be used as a constraining mechanism for other processes. Borrowing again from the field of Natural Language, plan recognition could be used to disambiguate word meanings during parsing. The current set of recognized intentions can act as a *filter of admissibility*,² rejecting those interpretations that are inconsistent or less compatible.

5.3 Reasoning in Plan Recognition

Conspicuously absent from plan recognition systems to date is the use of general reasoning capabilities. It is likely that this omission is an intentional one, owing to the complexity of incorporating another inference process. This however, should not be reason enough to prevent its inclusion; we examine briefly some arguments for adding reasoning to plan recognition systems and suggest exploration into the nature of the relationship between general reasoning and plan recognition as a potential research direction. As a vehicle for exploring this relationship, we will examine the conversation fragment in Figure 5.3.

Plan recognition and general reasoning faculties are at the very least two components that are tightly integrated; perhaps the coupling is so tight as to make the two indistinguishable. First consider the ways in which general reasoning and plan recognition interact. Plan recognition in humans makes use of general inference in a number of roles. One of these is to evaluate hypotheses (if not to propose them in the first place). In the example conversation above, Agent2 will have to recognize Agent1's discourse intention of responding to the question. Furthermore, in order to make sense of the answer, Agent1 will have to infer Agent2's intention to participate in graduation activities and hence to have plans for the weekend. This involves reasoning about knowledge of graduations

²the phrase *filter of admissibility* has been used by Bratman to describe the role of intentions in constraining the planning process to yield plans consistent with the intentions

and the events that they entail. Agent2 must reason that Agent1 is likely to go to these events since the graduation is for Agent2's sibling and because Agent2 has responded to the question in this manner. This example is complex enough that we can see the reasoning processes that Agent1 must employ in order to infer Agent2's intentions. But just as reasoning plays a role in plan recognition, so too does plan recognition play a role in reasoning. In reasoning about what to do over the weekend, Agent1 may propose the question observed in the conversation above. Plan recognition must be employed in order to understand and reason about Agent2's response. Thus it appears that there is at the very least a tight coupling between the plan recognition and reasoning.

Further evidence is the fact that the requirements of reasoning drive the level of detail at which plan recognition is performed. Consider the case that Agent1 wishes to know whether or not Agent2 is available to go on a camping trip for the duration of the weekend, and has thus posed the question of Figure 5.3. In this case, merely knowing that Agent2 has plans to attend a sibling's graduation festivities is enough to terminate plan recognition at the level of "intends to go to graduation activities". However, if Agent1 wishes to know the exact whereabouts of Agent2 over the weekend, he may reason from the knowledge of Agent2's intention to attend graduation festivities and the knowledge that Agent2's sibling lives in another states to the conclusion that Agent2 intends to travel in order to attend the graduation festivities. In this case, Agent1's requirements for knowledge (posed by a reasoning process) have influenced the level of detail included in the plan recognition. It is this type of example that shows just how closely related the processes of plan recognition and reasoning are related. Perhaps they are so tightly bound as to be the same process. There is some research literature that seems amenable to this line of thought. Cañamero poses plan recognition as a problem solving activity in [Cn95].

Now that the plausibility of a close relationship between plan recognition and general reasoning has been established we spend the remainder discussion discussing one possible application of reasoning in plan recognition systems: control of plan recognition inference. A number of systems in the literature have hard coded methods of controlling inference that could be replaced with a general reasoning procedure and knowledge specifying how to control the search. We address specifically the expectations that a system may employ to constrain reasoning, augmented backward chaining, and a mechanism for controlling the amount of reasoning performed.

As an example of incorporating expectation into a more general framework, consider the expectation mechanism in [AP80]. It will only consider the plans MEET-TRAIN and CATCH-TRAIN (and the null hypothesis) as possible explanations of user behavior. Instead of being hard-coded into the system structure, these plans could be specified in a knowledge base as likely intentions for train-station visitors. More generally, one could postulate that in situations of interaction between an agent and a service provider, the agent will generally intend to "consume" one or more of the services provided. Since the train-station provides "arrival" and "departure" services (knowledge in the knowledge base), this postulate would allow one to infer that agents generally intend to consume

an instance of an arrival or a departure. While the system may require additional facts about how such services are consumed, it is reasonable to think that there would be a MEET-TRAIN plan associated with the “consumption” of an arrival and a TAKE-TRAIN-TRIP plan associated with “consumption” of a departure. In this manner, general expectations could be derived from common-sense or domain knowledge.

Backward chaining algorithms, as described in Chapter 2, operate by extending hypotheses to include higher level goals that explain the existing goals. The basic reasons for chaining backward from a goal G_1 to a higher level goal G_2 are that the effects of G_1 satisfy a precondition of G_2 (and thus G_1 can be explained as making G_2 possible) or that G_1 is a subgoal of G_2 . This process can be augmented by allowing generic reasoning to infer these relationships instead of requiring them to be explicit in the representation. The benefit that this provides is more flexible plan recognition and the cost is an expanded search space.

Incorporating reasoning capabilities introduces computational complexity, but it is possible that the field of continual planning may provide some ideas for managing it. Continuous planning systems make use of *monitors* to track changes that could potentially affect the reasoning process. In [Mye98], *failure*, *knowledge*, and *assumption* monitors are defined to alert the system of changes in the environment that need to be addressed. Perhaps related constructs could be used to indicate the introduction of new facts that conflict with the currently hypothesized intentions. Additionally, techniques use in the PRS system described in [IGR92] specifically seek to limit the amount of time spent reasoning.

5.4 Conclusion

This paper began with discussion of why it is important to study plan and intention recognition, and how the two are related. Humans are planning creatures of necessity; so too are software agents. Computer systems that interact with other agents, be they human or otherwise, can improve cooperation by understanding the plans and intentions of collaborators. The transition of computer systems from tools to aids and eventually to autonomous collaborators requires the capability to recognize plans.

In Chapter 2 we discussed some of the formalisms that have been used by plan recognition systems and laid the groundwork for an analysis of some influential and representative systems presented in the literature. Chapter 3 presented a survey of five systems including two logical systems, two probabilistic systems, and a third that combined elements of logical and probabilistic approaches. This survey presented a wide variety of techniques that have been employed in plan recognition systems.

Chapter 4 addressed some of the problems that all plan recognition systems must address, using the selected systems as a basis for motivating and explaining the issues. In this final chapter, some new directions for research in plan recognition have been suggested. These combine new assumptions and ideas with elements of the problems that have been addressed. These directions seek to address the assumptions that current plan recognition models are based on, improve the practicality of plan recognition systems, and to generalize their capabilities. The author hopes that research on any or all of these topics will move us toward the goal of computer systems that can act proficiently as autonomous collaborators.

Bibliography

- [ABL96] Liliana Ardissono, Guido Boella, and Leonardo Lesmo. Recognition of problem-solving plans in dialogue interpretation. In *Proceedings of the 5th International Conference on User Modeling*, pages 195–197, Kailua-Kona, Hawaii, 1996.
- [ABS95] Liliana Ardissono, Guido Boella, and Dario Sestero. Recognizing preliminary sentences in dialogue interpretation. In Marco Gori and Giovanni Soda, editors, *AI*IA*, volume 992 of *Lecture Notes in Computer Science*, pages 139–144. Springer, 1995.
- [AFFH86] J. Azarewicz, G. Fala, R. FInk, and C Heithecker. Plan recognition for airborne tactical decision making. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 805–811, Philadelphia, Pennsylvania, 1986.
- [Ale95] J. Alexandersson. Plan Recognition in VERBMOBIL. In Bauer [Bau95b], pages 2–7.
- [All83] James Allen. Recognizing intentions from natural language utterances. *Computational Models of Discourse*, 1983.
- [All90] James F. Allen. Two views of intention: Comments on bratman and on cohen and levesque. In Cohen et al. [CMP90], pages 71–76.
- [AP80] James F. Allen and C. R. Perrault. Analyzing intention in utterances. *Artificial Intelligence*, 15(3):143–178, 1980.
- [AZN98] David W. Albrecht, Ingrid Zukerman, and Ann E. Nicholson. Bayesian models for keyhole plan recognition in an adventure game. *User Modeling and User-Adapted Interaction*, 8:5–47, 1998.
- [BA03] Nate Blaylock and James Allen. Corpus based, statistical goal recognition. In Georg Gottlob and Toby Walsh, editors, *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI03)*, pages 1303–1308, Acapulco, Mexico, 2003.

- [Bau94a] Mathias Bauer. Integrating probabilistic reasoning into plan recognition. In A. Cohn, editor, *Proceedings of the 11th European Conference on Artificial Intelligence*, pages 620–624, Amsterdam, Netherlands, 1994. John Wiley & Sons.
- [Bau94b] Mathias Bauer. Quantitative modeling of user preferences for plan recognition. In *Proceedings of the Fourth International Conference on User Modeling*, pages 73–78, 1994.
- [Bau95a] Mathias Bauer. A dempster-shafer approach to modeling agent preferences for plan recognition. *User Modeling and User-Adapted Interaction*, pages 317–348, 1995.
- [Bau95b] Mathias Bauer, editor. *IJCAI 95 Workshop on The Next Generation of Plan Recognition Systems: Challenges for and Insight from Related Areas of AI (Working Notes)*, Montréal, Canada, 1995.
- [Bau96a] Mathias Bauer. Acquisition of user preferences for plan recognition. In *Proceedings of the Fifth International Conference on user Modeling*, pages 105–112, 1996.
- [Bau96b] Mathias Bauer. Approximations for decision making in the dempster-shafer theory of evidence. In E. Horvitz and F. Jensen, editors, *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, 1996.
- [Bau96c] Mathias Bauer. Justification of plan recognition results. In W. Wahlster, editor, *Proceedings of the 12th European Conference on Artificial Intelligence*, Budapest, Hungary, 1996.
- [Bau98a] Mathias Bauer. Acquisition of abstract plan descriptions for plan recognition. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 936–941, 1998.
- [Bau98b] Mathias Bauer. Towards the automatic acquisition of plan libraries. In *European Conference on Artificial Intelligence 1998*, pages 484–488, 1998.
- [Bau99] Mathias Bauer. Machine learning for plan recognition. Workshop on "Machine Learning for User Modeling", Online Proceedings, 1999.
- [BB78] J. Borwn and R. Burton. Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science*, 2(2):155–192, 1978.
- [Bla02] Nate Blaylock. Managing communicative intentions in dialogue using a collaborative problem-solving model. Technical Report 774, University of Rochester, 2002.
- [BP93] Mathias Bauer and Gabriele Paul. Logic-based plan recognition for intelligent help systems. Technical Report 93-43, German Research Center for Artificial Intelligence (DFKI), 1993.

- [Bra87] Michael Bratman. *Intentions, Plans, and Practical Reason*. Harvard University Press, Cambridge, Massachusetts, 1987.
- [Bra90] Michael Bratman. What is intention? In Cohen et al. [CMP90], pages 15–32.
- [Car90] Sandra Carberry. Incorporating default inferences into plan recognition. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 471–478, Menlo Park, California, 1990. AAAI Press.
- [Car01] Sandra Carberry. Techniques for plan recognition. *User Modeling and User-Adapted Interaction*, 11(1-2):31–48, 2001.
- [CC91] Glenn Carroll and Eugene Charniak. A probabilistic analysis of marker-passing techniques for plan recognition. Technical Report CS-91-44, Brown University, 1991.
- [CF95] C. Castelfranchi and R. Falcone. From Single-Agent to Multi-Agent: Challenges for Plan Recognition Systems. In Bauer [Bau95b], pages 24–28.
- [CG93] Eugene Charniak and Robert P. Goldman. A bayesian model of plan recognition. *Artificial Intelligence*, 64:53–79, 1993.
- [CL90] Philip R. Cohen and Hector J. Levesque. Persistence, intention, and commitment. In Cohen et al. [CMP90], pages 33–70.
- [CMP90] Philip R. Cohen, Jerry Morgan, and Martha E. Pollack, editors. *Intentions in Communication*. The MIT Press, Cambridge, Massachusetts, 1990.
- [Cn95] D. Cañamero. A Knowledge-Level Approach to Plan Recognition. In Bauer [Bau95b], pages 18–23.
- [CSSvB91] R. Cohen, F. Song, B. Spencer, and P. van Beek. Exploiting temporal and novel information from the user in plan recognition. *User Modeling and User-Adapted Interaction*, 2(1):125–148, 1991.
- [CV98] Michael T. Cox and Manuela M. Veloso. Goal transformations in continuous planning. In Marie E. desJardins, editor, *Proceedings of the 1998 AAAI Fall Symposium on Distributed Continual Planning*, Menlo Park, CA, 1998. AAAI Press/The MIT Press.
- [dDCLOW99] Marie E. desJardins, Edmund H. Durfee, Jr. Charles L. Ortiz, and Michael J. Wolverton. A survey of research in distributed, continual planning. *AI Magazine*, 20(4):13–22, Winter, 1999.
- [DE95] B. Di Eugenio. Plan Recognition and Natural Language Understanding. In Bauer [Bau95b], pages 42–47.

- [DL03] Carmel Domshlak and James H. Lawton. On planning for multi-agent opportunistic execution. In *IJCAI-03 Workshop on Issues in Designing Physical Agents for Dynamic Real-Time Environments: World modeling, planning, learning, and communicating*, 2003.
- [EC92] R. M. Eller and S. Carberry. A meta-rule approach to flexible plan recognition in dialogue. *User Modeling and User Adapted Interaction*, 2(1-2):27–54, 1992.
- [Ger97] A. Gertner. Plan recognition and evaluation for on-line critiquing. *User Modeling and User-Adapted Interaction*, pages 107–140, 1997.
- [GGM99] Robert P. Goldman, Christopher W. Geib, and Christopher A. Miller. A new model of plan recognition. In Kathryn B. Laskey and Henri Prade, editors, *UAI*, pages 245–254. Morgan Kaufmann, 1999.
- [GHK99] Barbara J. Grosz, Luke Hunshberger, , and Sarit Kraus. Planning and acting together. *AI Magazine*, 20(4):23–24, Winter 1999.
- [GI89] Michael P. Georgeff and Francois F. Ingrand. Decision-making in an embedded reasoning system. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, volume 2, pages 972–978, Detroit, Michigan, 1989.
- [GK98] Barbara J. Grosz and Sarit Kraus. The evolution of sharedplans. In Anand Rao and Michael Woolridge, editors, *Foundations and Theories of Rational Agencies*. 1998.
- [GPP⁺99] Michael Georgeff, Barney Pell, Martha Pollack, Milind Tambe, and Michael Woolridge. The belief-desire-intention model of agency. In *Proceedings of Agents, Theories, Architectures and Languages (ATAL)*, 1999.
- [GW96] A. Gertner and B. L. Webber. A bias towards relevance: Recognizing plans where goal minimization fails. 1996.
- [Hon95] J. Hong. Different Roles of Deductive and Uncertain Reasoning in Plan Recognition. In Bauer [Bau95b], pages 134–135.
- [HSME88] Jerry R. Hobbs, Mark E. Stickel, Paul Martin, and Douglas Edwards. Interpretation as abduction. In *Proceedings of the ACL*, pages 95–103, 1988.
- [IG90] Francois F. Ingrand and Michael P. Georgeff. Managing deliberation and reasoning in real-time ai systems. In *Proceedings of the 1990 DARPA Workshop on Innovative Approaches to Planning*, San Diego, California, 1990.
- [IGR92] Francois F. Ingrand, Michael P. Georgeff, and Anand S. Rao. An architecture for real-time reasoning and system control. *IEEE Expert*, 7(6):33–44, December, 1992.

- [JNTM03] Hyuckchul Jung, Ranjit Nair, Milind Tambe, and Stacy Marsella. Computation models for multiagent coordination analysis: Extending distributed pomdp models. In M. Hinchey, J. Rash, W. Truszkowski, C. Rouff, and D. Gordon-Spears, editors, *Formal Approaches to Agent-Based Systems*. Kluwer Academic Publishers, Norwell, Massachusetts, 2003.
- [JT03] Hyuckchul Jung and Milind Tambe. Composing pomdp-based building blocks to analyze large-scale multiagent systems. Technical Report SS-03-02, AAI Spring Symposium, 2003.
- [JTBC02] Hyuckchul Jung, Milind Tambe, Anthony Barrett, and Bradley Clement. Enabling efficient conflict resolution in multiple spacecraft missions. In *Proceedings of the International NASA Workshop on Planning and Scheduling for Space*, 2002.
- [Kau90] Henry Kautz. A circumscriptive theory of plan recognition. In Cohen et al. [CMP90], pages 105–134.
- [Kau91] Henry Kautz. A formal theory of plan recognition and its implementation. In J. F. Allen, H. A. Kautz, R. N. Pelavin, and J.D. Tenenber, editors, *Reasoning About Plans*. Morgan Kaufmann, Los Altos, California, 1991.
- [Kol98] Daphne Koller. Structured probabilistic models: Bayesian networks and beyond. In *Proc. AAAI-98*, pages 1210–1211, Menlo Park, CA, 1998.
- [KP93] Kurt Konolige and Martha E. Pollack. A representationalist theory of intention. In *Proceedings of the 13th IJCAI*, pages 390–395, 1993.
- [LA90] Diane J. Litman and James F. Allen. Discourse processing and common-sense plans. In Cohen et al. [CMP90], pages 365–388.
- [LC91] L. Lambert and S. Carberry. A tripartite plan-based model of dialogue. In *Proceedings of the 29th Annual Meeting of the ACL*, pages 47–54, Berkeley, CA, 1991.
- [LE95] N. Lesh and O. Etzioni. Insights from Machine Learning for Plan Recognition. In Bauer [Bau95b], pages 78–83.
- [LE96] N. Lesh and O. Etzioni. Scaling up goal recognition. In *Proceedings of the 5th International Conference on Knowledge Representation and Reasoning*, pages 178–189, 1996.
- [Les97] Neal Lesh. Adaptive goal recognition. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, pages 1208–1214, Nagoya, Japan, 1997.
- [LM95] J.J. Lee and R. McCartney. Plan Recognition in Human Computer Interaction. In Bauer [Bau95b], pages 136–137.

- [May92] James Mayfield. Controlling inference in plan recognition. *User Modeling and User-Adapted Interaction*, 2(1-2), 1992.
- [McC80] John McCarthy. Circumscription – a form of non-monotonic reasoning. *Artificial Intelligence*, pages 27–39, 1980.
- [McR95] S. McRoy. The need to address plan misinference during dialogues and why abduction might help. In Bauer [Bau95b], pages 90–95.
- [Mye98] Karen L. Myers. Towards a framework for continuous planning and execution. In *Proceedings of the AAAI Fall Symposium on Distributed Continual Planning*, 1998.
- [Mye99] Karen L. Myers. Cpef: A continuous planning and execution framework. *AI Magazine*, 20(4):63–69, Winter, 1999.
- [NTM03] Ranjit Nair, Milind Tambe, and Stacy Marsella. Integrating belief-desire-intention approaches with pomdps: The case of team-oriented programs. In *Proceedings of the AAAI Spring Symposium on Logical Formalisms for Commonsense Reasoning*, 2003.
- [Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Mateo, California, 1988.
- [Pfe00] A. J. Pfeffer. *Probabilistic Reasoning for Complex Systems*. PhD thesis, Stanford University, 2000.
- [PH99] Martha E. Pollack and John F. Horty. There’s more to life than making plans: Plan management in dynamic, multi-agent environments. *AI Magazine*, 20(4):71–84, Winter, 1999.
- [Pol86] Martha Pollack. *Inferring Domain Plans in Question-Answering*. PhD thesis, University of Pennsylvania, Philadelphia, Pennsylvania, 1986.
- [Pol90] Martha E. Pollack. Plans as complex mental attitudes. In Cohen et al. [CMP90], pages 77–104.
- [Poo93] David Poole. Probabilistic horn abduction. *Artificial Intelligence*, 64(1):81–129, 1993.
- [Poo97] David Poole. The independent choice logic for modelling multiple agents under uncertainty. *Artificial Intelligence*, 94:7–56, 1997.
- [Poo03] David Poole. First-order probabilistic inference. In *Proc. IJCAI-03*, pages 985–991, Acapulco, Mexico, 2003.
- [RELED95] C. P. Rose, B. Di Eugenio, L. Leven, and C. Van Ess-Dykema. Discourse processing of dialogues with multiple threads. In *Proceedings of the 33rd Meeting of the Association for Computational Linguistics*, pages 31–38, 1995.

- [RG91] Anand S. Rao and Michael P. Georgeff. Deliberation and intentions. Technical Report 10, Australian Artificial Intelligence Institute, 1991.
- [RG95] Anand S. Rao and Michael P. Georgeff. Bdi agents: From theory to practice. In *Proceedings of the First International Conference on Multiagent Systems (ICMAS'95)*, pages 312–319, 1995.
- [RG99] Anand S. Rao and Michael P. Georgeff. Asymmetry thesis and side-effect problems in linear-time and branching-time intention logics. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, volume 1, pages 498–505. Morgan Kaufmann Publishers, 1999.
- [Sea90] John R. Searle. Collective intentions and actions. In Cohen et al. [CMP90], pages 401–416.
- [Sha76] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, Princeton, New Jersey, 1976.
- [Wae97] Annika Waern. Local plan recognition in direct manipulation interfaces. In *Proceedings of the 1997 International Conference on Intelligent User Interfaces*, pages 7–14, Orlando, Florida, 1997.
- [Wei95] R. Weida. Knowledge Representation for Plan Recognition. In Bauer [Bau95b], pages 119–123.
- [Woo95] S. Wood. The Role of Plan Recognition in Reducing Situational Uncertainty and Directing Attention in Planning. In Bauer [Bau95b], pages 124–128.
- [WS95] A. Wærn and O. Stenborg. Recognizing the Plans of a Replanning User. In Bauer [Bau95b], pages 113–118.
- [Wu91] D. Wu. *User Modeling and User-Adapted Interaction*, 1(2):149–172, 1991.