Retroactive Recognition of Interleaved Plans for Natural Language Dialogue

Nate Blaylock

The University of Rochester
Computer Science Department
Rochester, New York 14627

Technical Report 761

December 2001

Abstract

State of the art plan recognition for use in natural language dialogue systems has progressed in coverage of discourse phenomena and plan navigation strategies. Most systems, however, suffer from several deficiencies, namely, they do not have a specific strategy for the (inevitable) case where they make an incorrect hypothesis inference and they cannot handle interleaved plan navigation, where a user jumps back and forth between several plans. In addition, most plan recognition systems cannot handle the rich variety of possible natural language utterances a dialogue system may receive as input, especially the case where a language utterance corresponds to several actions which the system considers to be atomic. We discuss previous work in plan recognition, especially in the area of dialogues systems. We then describe a plan recognition system which can recover from incorrect inferences, handles interleaved plan navigation, and handles several linguistic phenomena, including support for natural language multi-action paraphrase.

This material is based upon work supported by the Office of Naval Research grant number N00014-95-1-1088 and the Keck Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of ONR or the Keck Foundation.

Contents

1 Introduction

Recognizing the underlying plans of a user within a natural language dialogue system can make the difference between system that seems intelligent and helpful system and one that seems sarcastic or dumb. Consider the following input that a dialogue system of the future (embedded in a robot) might see (U1), together with several possible responses from the robot.

U1:	Do you have a watch?
S1:	Yes, it's 4:30.
S2:	Yes, it's a Rolex.
S3:	Yes.

Depending on the underlying plan of the user, any one of these responses could be considered to be appropriate, while the other two would be considered either rude or just plain dumb. We will look at three different contexts/user plans, each of which has a distinct appropriate response.

Context 1: A user looks at his arm, (sees nothing on it) and then approaches our robot and utters U1.

The user's plan here is to find out what time it is, and S1 is the appropriate response. A response of S2 would make our robot seem like a dummy and S3 would be considered a very rude response.

Context 2: The user and his buddy are arguing about what kind of watches robots tend to have. The user turns to the robot and utters U1.

In this case, the user's plan is to find out *what kind* of watch our robot has in order to help him win his argument. S2 is the most appropriate response to make. Uttering S1 would seem a very odd response and, again uttering S3 would seem very rude. Here again, it was very important for the robot to recognize what the user's underlying plans were.

Context 3: The robot is trapped in a burning room with MacGyver[1]. MacGyver looks around the room and tells the robot he has an idea. He grabs a roll of duct tape and a pencil and then looks at the robot and utters U1.

MacGyver's plan, of course, is to escape by building some sort of invention using duct tape, a pencil, and a watch. S3 is the most appropriate response for our robot this time. Responses S1 and S2 would be so inappropriate that they might make good lines in a comedy movie with our poor robot playing the not-so-intelligent character.

[1]MacGyver is a television character who always escapes from dangerous situations by combining everyday items into powerful inventions.

Plan recognition is the process where, given a set of observations of the actions of an agent (*the acting agent*), a second agent (*the recognizing agent*) tries to infer the goals and plans of the acting agent. Plan recognition is typically divided into two types, which reflect the acting agent's attitude towards the recognition. *Keyhole recognition* is where the acting agent is either unaware that the recognition is taking place, or is apathetic to it. In *intended recognition*, on the other hand, the acting agent is actively trying to make known his plans to the recognizing agent[2]. Both of these types of plan recognition are widely used by humans, and, as we will discuss below, useful in a variety of applications. Natural language dialogue systems (like our robot above) need to make use of both kinds of plan recognition, but rely more heavily on intended recognition. Communication is a shorthand for intentions. Usually, both parties desire for these intentions (i.e. plans) to be understood, which means that the speaker (i.e. the acting agent) tends to structure his utterances in ways that will make his intentions easily recognizable by the hearer (the recognizing agent).

In this paper we first discuss different applications which use plan recognition. We then explore previous research which has been done on plan recognition, especially as it is used in natural language dialogue systems. We discuss shortcomings of previous methods and then describe a plan recognizer for a natural language dialogue system which overcomes several of these shortcomings. We conclude by discussing our goals for future work.

2 Applications of Plan Recognition

Plan recognition is used in a wide variety of applications. In this section we explore some of these areas and describe a few of the most prominent applications which use plan recognition. The main areas we explore are user modeling, multi-agent interaction, and natural language processing.

2.1 User Modeling

The field of user modeling has made wide use of plan recognition. We describe some of the most prominent user modeling applications that use plan recognition, specifically in the areas of operating systems, intelligent help systems and intelligent tutoring systems.

Operating Systems

Recognizing a user's plan in an operating systems environment allows the system to do many helpful things such as making suggestions, automatically fixing user errors and offering auto-completion of tasks.

Huff and Lesser [HL82] built an intelligent system which aided computer programmers in their day-to-day tasks.. The system watched user commands to UNIX, and, based on those commands, would determine the user's plan as it related to programming. It was

[2]A third type of plan recognition occurs when the acting agent is trying to thwart recognition of his plans. Pollack [Pol86a] calls this an *actively non-cooperating actor*. Very little research has been done for this third type of recognition, which may be why it is frequently not included in the typology.

built to recognize the plans of programmers such as browsing code, editing, compiling, and so forth. It would then use its knowledge of recognized plans in interpreting ambiguous commands and repairing commands that had errors in them.

Lesh [Les98, LE95b, LE96, Les97, LE95a] built a plan recognition system which he applied to user goals in the UNIX and WindowsNT environments. The system would watch user's successive commands and try to determine the goal they were pursuing. Although Lesh never applied this plan recognition system to an overall system, such as Huff and Lesser above, it could easily be applied to the same sort of activity.

Pachet and Giroux [PG95] describe a software engineering method that can be used with any existing object oriented application to 'spy' on a user in order to make observations for, or during plan recognition. This will allow systems like those mentioned above to be 'plugged into' any arbitrary system.

Intelligent Help Systems

Related to the work of plan recognition in operating systems, is that for intelligent help systems. Recognition of a user's current plan(s) can allow the system to give more directed help than normal help systems can.

The Berkeley UNIX Consultant Project (UC) [Wil82, WAC84, WCL$^+$89, May89, May92] also performed plan recognition within the UNIX operating system, but with a slightly different purpose. UC was a facility for helping beginning users learn to use UNIX. Users could ask questions in English and get responses back from the system about how to operate the system, accomplish different tasks, and so forth. UC would use plan recognition to determine the intentions behind a user's query, which would help it to generate a more helpful response. In many ways, UC could be considered a natural language question-answering system, but most of the plan recognition literature categorizes it here.

The WIZARD system [Fin83] also provided help in an operating system. This time in the VAX/VMS environment. WIZARD watched the user's actions and performed plan recognition. When WIZARD detected that the user was using an *inefficient* method (i.e. plan) to accomplish a task, (for example using copy followed by a delete instead of rename) it volunteered helpful hints on how to accomplish the task more efficiently.

Bauer [BBD$^+$93, BP93] implemented a intelligent help system for email. A plan recognition system would watch a user's actions and then offer advice or help in a context-sensitive way that would not have been possible without knowing the user's plan.

Intelligent Tutoring

Greer and Koehn [GK95] give several examples of how plan recognition relates to diagnosis in intelligent tutoring. This includes recognizing the (erroneous) plan the student was trying to use to construct a solution, and improving the model of a student's overall problem solving strategy.

Johnson [Joh95] suggests that, although plan recognition has a place in intelligent tutoring, it is not as important, since students may not have *any* plan that they are executing

to try to solve the problem. He suggests, however, that there are some domains in tutoring, as Greer and Koehn also suggest, do benefit from plan recognition.

2.2 Multi-agent Interaction

General research in multi-agent environments has also shown promise for using plan recognition. The systems in [HDW94] and [VW01] use plan recognition in multi-agent domains. In many domains, direct agent communication is either not possible, or is expensive, and agents must coordinate their activity instead by observing the actions of other agents and recognizing their plans. Both systems use this plan recognition to allow agents to coordinate without explicit communication.

Plan recognition has also been used by the military for tactical decision making [AFFH86, AFH89]. Enemy plans are recognized from observations of individual enemy ship and airplane activity.

2.3 Natural Language

One of the largest areas of plan recognition research has been in the field of natural language processing. Human communication is filled with intended recognition, which allows a more compact form of communication to happen. Any application which needs to understand language needs to be able to recognize the intentions of the speaker.

We discuss here just a few of the many natural language applications which have utilized plan recognition in areas of story understanding, machine translation, and, of course, dialogue systems, which is the main focus of this work.

Story Understanding

Rumelhart [Rum75] noted that stories have structure, much in the same way that sentences do. Several story understanding systems have employed plan recognition to try to extract that structure. Two examples are the systems SAM (Script Applier Mechanism) and PAM (Plan Applier Mechanism), which both came out of Schank and Abelson's work on scripts [SA77].

SAM [SA77, SR81] used scripts to understand various types of stories, including newspaper articles. It could then use its analysis to summarize or paraphrase the story. It could also answer questions about the story.

PAM [SA77, Wil78, SR81, Wil83] performed similar functions, using knowledge of both scripts and plans to analyze plan-based stories.

Machine Translation

A good machine translation system needs to understand the speaker's plans, goals, and intentions. The Verbmobil system [KGN94] is a speech-to-speech natural language translation system for English, German, and Japanese. The system translates for businessmen making appointments.

Plan recognition in Verbmobil [Ale95] allows the system to track the discourse goals [JKM+95, ABWF+98] of the dialogue participants. Knowledge of the current plans of the participants allows for more robust and accurate translation.

Dialogue Systems

We now come to the application which is the main topic of this paper, that of dialogue systems. Most recent work in plan recognition in natural language processing has focused on its use in dialogue systems. Although there are potentially many systems we could mention here (some of which we mention later), we only mention two. Carberry's student advisor system, and the TRAINS/TRIPS systems at the University of Rochester.

Carberry [Car90b] used plan recognition in a dialogue system which advised college students about taking classes, getting degrees, and so forth. Recognizing the user's plans allowed the system to give appropriate and helpful responses to the student.

At the University of Rochester, research over the last decade has produced first the TRAINS system [AS91, ASF+94], and now the TRIPS system [FA98, ABD+00, ABD+01, AFS01].

The TRAINS dialogue system allowed the system and user to work together to try to route trains to make deliveries across the eastern United States. We discuss plan recognition in TRAINS in more detail in Section 4.6.

TRIPS is the next generation of the TRAINS project. TRIPS is a domain independent, mixed-initiative dialogue system core, which can be easily ported to new domains. It has successfully been ported to such domains as emergency evacuation, disaster relief, and military resource allocation.

We are currently working to expand the TRIPS system capabilities and port it to a 911 operator domain. This domain, set in Monroe County, New York, allows the system and user to work together to try to respond to 911-type emergencies. This is a much richer domain which requires more sophistication from a plan recognition system. The work reported in Section 7 was developed to serve as the plan recognition component in TRIPS.

3 Foundations of Plan Recognition

In this section we present two early pieces of research which are considered to be the foundations of plan recognition. Although there were many other early general plan recognition systems, almost all subsequent work on plan recognition has roots in these two systems. We first discuss the BELIEVER system and then Kautz' work on generalized plan recognition.

3.1 BELIEVER

One of the first plan recognition projects was the BELIEVER system [SSG78], which showed that humans do indeed use plan recognition. In building BELIEVER, Schmidt, et al conducted psychological experiments to see how humans do plan recognition. They then

built the BELIEVER system Using these results. We will first describe their experimental results and then the workings of their system.

The Experiment In the experiment, subjects were given a description of a sequence of actions. At certain times between actions, the subject would be asked to summarize what he thought the actor was trying to do[3].

In their summarizations, the subjects would attribute beliefs and intentions to the actor, particularly about his plans and goals. The subjects were doing plan recognition on the observed actions.

One interesting observations made by the experimenters was that the subjects reported their beliefs about the plans of a user as a single hypothesis, even when much ambiguity existed. When details were not available, subjects would give a "sketchy hypothesis" for the plan, such as "John is getting *something* to eat." When further observations invalidated this single hypothesis, a new hypothesis would be proposed which fit the new data.

In other words, these experiments showed that humans make the best guess they can as to the plans of the actor they are observing, instead of keeping a disjunctive list of possible user plans. As new data arrives, humans revise this hypothesis as necessary. As Lesh points out [Les98], despite these findings, it is interesting that most plan recognition systems since have done just the opposite, proposing many different possible hypotheses, and then trimming them down as more data comes along.

The System The BELIEVER system was built to try to mimic the human plan recognition behavior found in the experiments described above. BELIEVER was given a priori knowledge of the world and about specific plans. It would then be given an observation, from which it would build a plan hypothesis for the actor.

BELIEVER matched incoming observations into a pre-built *expected plan structure*. The expected plan structure was a graph of actions which connected different enables/in-order-to relations among the preconditions and results of the different actions. A set of *propositions* existed which asserted the relationship between actions. A violation of a proposition would result in that hypothesis being thrown out. Matched observations would then be put into the *grounded plan structure*. Once all actions had been observed/grounded, the plan was considered fully grounded.

When a hypothesis was deemed incorrect, (upon receiving another observation) the hypothesis revision process would run. Depending on the class of revision (i.e. how the hypothesis was deemed unacceptable), different rules would apply, which would generate a set of possible revisions. These revisions were then checked and one selected. This checking/selection algorithm was neither implemented nor defined in BELIEVER, however.

BELIEVER is probably the most similar system to our work. Our work differs, however in several ways. First, BELIEVER performed generalized keyhole recognition, while our system assumes the intended recognition setting of natural language dialogue. Secondly,

[3]Other experiments also asked subjects to recall the events or to predict what the next action would be. This particular experiment, however, is the most relevant to our discussion.

BELIEVER did not allow for hierarchical plans. Thirdly, our hypothesis revision process is quite different from that of BELIEVER. BELIEVER's revision process was never completely defined, however was specified to generate and explore all revision possibilities. Our system uses an undo mechanism to minimally change hypotheses and does not generate and consider all possibilities. Lastly, BELIEVER's plan recognition did not support interleaved plans as ours does.

3.2 Kautz' Generalized Plan Recognition

The other foundation work in plan recognition was that of Kautz [KA86, Kau87, Kau90, Kau91]. Kautz cast plan recognition as the logical inference process of circumscription. This logical cast on plan recognition allowed him to use the a very rich knowledge representation (essentially that of first order logic), as well as to use temporal logic to represent actions and time.

Kautz represented the space of possible plans as an event hierarchy, which included both abstraction and decomposition (subaction) relations. Certain actions were labeled as *end* actions, meaning that they were an end unto themselves, or an ultimate goal of an actor.

By making a few assumptions (such that this event hierarchy was complete), plan recognition became a problem of circumscription. Observations (also representable in first order logic), were used to infer plans non-monotonically. Minimalization was then used to further reduce the deductions.

Although this model is able to account for interleaved plans, it suffered from most of the problems we will discuss in later sections, including an exponential runtime and problems with the assumptions made in the model. It also was not developed for the intended recognition of dialogue.

4 Applying Plan Recognition to Natural Language Dialogue

We have already mentioned above that plan recognition is a vital part of natural language understanding. It should come as no surprise, then, that a lot of work has been done on using plan recognition in natural language dialogue systems. In this section we first explore the theory of speech acts which ties dialogue and planning together. We then look at several of the most notable computational systems and discuss their strengths and weaknesses.

4.1 Speech Acts

Austin [Aus62], Grice [Gri57, Gri69, Gri75], and Searle [Sea70, Sea75] all noted that human utterances can actually cause *changes* in the world. The utterance "I pronounce you man and wife" said by the right person in the right context actually causes two people to be married. More subtly, the utterance of "John is in the kitchen" may have effect of causing the hearer to believe that John is in the kitchen.

Utterances, then, can have preconditions and effects, the same as other non-linguistic actions. Thus we can build plans that contain utterances as well as other actions. A full

discussion of speech acts is beyond the scope of this paper. What is important to realize, however, is that treating utterances like actions (speech acts) allows us to use our theory of plan recognition within dialogue systems. A speaker constructs a plan, say to cause us to believe that John is in the kitchen. His plan is to utter the words "John is in the kitchen" to us. We (the hearer) then must be able to recognize what the speaker's original goal was (whether to inform us that John is in the kitchen, to get us to go into the kitchen (to see John), to hint that we should *not* go into the kitchen (because John is in there), etc.) Different planning contexts will result in different interpretations for the same utterance.

4.2 Allen, Cohen, and Perrault

Allen, Cohen, and Perrault were the first to computationalize the theory of speech acts. They showed [CPA82] that, in question answering systems, users expected the system to recognize their unstated goals in order to provide more helpful responses to questions.

Cohen [Coh78, CP79] concentrated on using plan synthesis together with speech acts for natural language generation. Allen [All79, AP80, All83], on the other hand, used plan recognition of speech acts for natural language understanding. We will concentrate here only on Allen's work.

Allen studied transcripts of actual interactions at an information booth in a Toronto train station. A typical exchange was something like this [All83].

patron: When does the Montreal train leave?
clerk: 3:15 at gate 7.

Note that, although the patron only requested the departure time, the clerk also volunteered information about the departure gate as well. Presumably, the clerk *recognized* the plan of the patron (to board the train), and realized that the patron would also need to know where the train departed and volunteered that information as well. Allen called this behavior *obstacle detection*.

Allen's system took the direct speech act of the utterance, and, using certain inference rules and heuristics to apply them, did backward chaining in order to infer the user's true plan. Heuristics included things such as, if a person wants P, and P is a precondition of action ACT, then the person may want to perform ACT; or if a person wants to know if P is true, they may want P to be true (or false).

Using these inference rules, the system was able to recognize not only indirect speech acts, but also the user's domain plan. However, the large search space and large number of potential hypotheses made this approach unworkable in larger domains. Moreover, Allen's system only worked for one-utterance dialogues.

4.3 Multiple Utterance Plan Recognition

Carberry [Car87, Car90b] extended previous work to incrementally (utterance by utterance) account for multiple utterances. She used a plan hierarchy similar to Kautz' (above)

which held information about decomposition of plans in the domain. Her system explicitly kept track of this hierarchy and filled it in as the user navigated and expanded different nodes. This type of plan recognition system accounted for both bottom-up (talk about actions first) and top-down (talk about goals first) dialogue.

Carberry's system did plan recognition in a two-phase manner. The first phase did *local analysis* on input, which tried to identify the speaker's *immediate* goal, regardless of context. Afterwards, one of these goals was chosen based on *global analysis* (the second phase), which tried to fit one of these goals into the context of previous utterances.

Focus was an important part of this system. Carberry found that speakers usually navigate and fill in the tree in a predictable way. Thus, if focus were on a certain node, the system could calculate the most likely shifts of focus from that node. This allowed the system to use context as well as dialogue behavior to filter among the possible immediate goals from local analysis and place them into the context.

Our system uses focus much like Carberry's, although, as we discuss below, we allow a wider range of tree navigation techniques based on things like interleaved plans, which Carberry does not account for. Also, Carberry's system first generated all possible interpretations and then filtered among them, making it vulnerable to a possible runtime explosion in a larger domain. Our system uses context first to decide which options to explore.

4.4 Dialogue and Domain Plans

Litman and Allen [Lit85, Lit86, LA87, LA90] extended Carberry's and Allen's work to better account for various dialogue phenomena. Although a dialogue's focus is on the domain, there seem to be several meta-layers which help ensure robust communication.

Essentially, Litman and Allen added a new layer to the plan recognition system, that of problem-solving[4] plans. Previous plan recognition systems had only accounted for domain-level plans (as well as speech act level plans). Litman and Allen's system was able to account for a number of problem-solving phenomena, including that of clarification subdialogues. For example, consider the following dialogue [LA90].

teacher: OK the next thing you do is add one egg to the blender, to the shrimp in the blender.
student: The whole egg?
teacher: Yeah, the whole egg. Not the shells.
student: Gotcha. Done.

The domain-level plan here one of cooking. The student's first utterance, however is caused by confusion in the dialogue. Instead of replying directly to the teacher's instruction, the student asks a clarification question, which is a problem-solving plan (i.e. that of making sure he understands correctly). The teacher responds to this question and then the student 'pops' back down to the domain level again and says "Done."

[4]Litman and Allen actually called these *discourse* plans. In light of subsequent work, however, these are better characterized as problem-solving plans.

Litman and Allen allowed this stack-like structure of dialogues and recognized not only domain-level plans, but also the domain-independent problem-solving plans. This allowed systems to more fully cover dialogue phenomena.

4.5 Other Plan Levels in Dialogue

There have been several efforts to extend the work of Litman and Allen, further categorizing the types and levels of plans that can exist in dialogue. We first discuss the work of Lambert and then that of Ramshaw (and the extensions by Ardissono).

Lambert Lambert [LC91] proposed a three-level model of dialogue, consisting of the domain and problem-solving levels of Litman and Allen as well as a level of discourse, which specifically handled recognition of multi-utterance speech acts. Consider the follow utterances, which, only taken together constitute a warning [LC91].

U1: The city of xxx is considering filing for bankruptcy.
U2: One of your mutual funds owns xxx bonds.

Their separation of discourse plans allows them to be able to recognize such speech act like phenomena at a multi-utterance level.

Ramshaw During the same period of Lambert, Ramshaw [Ram89b, Ram89a, Ram91] proposed a differently separated model (which was later extended by Ardissono [ABL96]). Instead of a discourse level, Ramshaw proposed an *exploration* level. The intuition is that some utterances are made simply in the attempt to *explore* a possible course of action, whereas others are explicitly made to attempt to *execute* a plan.

This model allows the system to distinguish between the two cases, since, presumably, different responses from the system would be required, although neither Ramshaw or Ardissono make this point. This approach seems more adapted to domains where execution of plans happens as the dialogue progresses. We discuss this issue in more detail in Section 6.3.

4.6 Plan Recognition in TRAINS

The TRAINS system [AS91, ASF+94] was an end-to-end speech dialogue system. The real-time nature of the system presented a need a more tractable form of plan recognition. Plan recognition is done in two stages. The first stage uses linguistic information to compute and filter possible indirect speech acts. This filtered set is then sent to the main plan recognition system. We discuss just the preprocessing phase below. The plan recognition system is similar to that described by Carberry (above), enriched with a problem solving model.

Linguistic Preprocessing The linguistic preprocessor is the result of the work of Hinkelman and Allen [HA89]. They noted that previous plan recognition techniques, especially for speech acts, disregarded many linguistic clues in the utterance, which can actually constrain interpretations. Consider the difference between the following utterances [HA89].

U1: Can you speak Spanish?
U2: Can you speak Spanish, please?

At a direct speech act level, these two utterances look the same. However, the possible interpretations differ. Whereas U1 can either be interpreted as a question of ability ("Do you know how to speak Spanish?") or a request ("Will you speak Spanish?"), U2 is only interpretable as the later. The presence of "please" constrains the interpretation.

Hinkelman built a set of rules for English based on these principles that would generate and constrain possible indirect speech act interpretations of utterances. This process greatly speeds up interpretation.

5 Problems with Prototypical Plan Recognition and Extensions to Solve Them

The plan recognition systems that we have described so far are, in a way, considered to be 'prototypical' within the field. However, several researchers have pointed out problems with these models. This section presents work which has been done to try to correct some of these problems, but, which has not really been integrated into the prototypical model in the field.

5.1 User's Incomplete/Erroneous Plans

This section could possibly be better labeled "when users make mistakes". Most systems described above contain the assumption that both the system and the user have *exactly* the same knowledge about plans in a domain (and non-domain plans as well). Both Pollack [Pol86a, Pol86b, Pol87, Pol90] and Quilici [Qui89] show that this assumption is not well-founded, especially in situations where the user is a novice, and is interacting with the system (an expert). In these cases, as well as in everyday interaction, users may have either incomplete or incorrect knowledge.

Pollack's solution suggests extending what is meant by the concept of 'plan' in the normal literature. A plan, she suggests, can be divided into two concepts, a recipe (which is a 'plan' of how to do something, which is what is typically thought of as a plan) and then, what she calls a *plan*, which is a set of beliefs and intentions with respect to a recipe. The additional model of beliefs and intentions about plans allow us to separately model what we know versus what we believe the user knows (especially about plans). It also allows us to be able to not only detect an erroneous plan, but also to detect, *why* the plan was erroneous, which allows us to respond to the user in a more natural way, such as correcting the user's incorrect knowledge.

5.2 System's Incomplete/Erroneous Plans

This section speaks of the flip side of the section above. Another fixed assumption from prototypical plan recognition systems is that the system's knowledge is both correct and complete. While this may be a reasonable, even desirable property in certain domains, it is generally undesirable for several reasons. First, it places the *great* burden on the system designer to enumerate and expand every possible recipe for every possible goal that the system will have to recognize, adding possibly to engineering time, required storage space for knowledge, and/or preexisting system limits. Secondly, in the *rare* event that the system actually *didn't* have all of the knowledge it needed (or had some incorrect knowledge), it would be desirable for a user to be able to add or correct (planning) knowledge within the system.

Robin Cohen's group [CSSvB91, CSH94] has looked a little at this problem. Their work allows a user to add either temporal details (via tense information) or novel plan knowledge to the system. When a new plan recipe is added to the system, the system uses its knowledge of action preconditions and effects to actually validate that the user has entered a valid recipe. Their research only concentrated on the simple task of where the user is describing his plan to the system. Work is still needed to see how to incorporate this kind of system behavior into a normal dialogue setting. This includes work on how the system can determine that *it* is wrong, and not the human, and thus correct its knowledge.

5.3 Ambiguity

One of the biggest problems with prototypical plan recognition is that there can be a lot of ambiguity (especially as the domain gets bigger). Two approaches to this problem have been taken. The first, somewhat unique approach to natural language dialogue, is the use of clarification, i.e. querying the user to help resolve the ambiguity. The second is to use uncertainty to allow the system to reason with the likelihood of any given interpretation. We discuss these here.

Solution: Clarification Dialogue when Necessary

The most straightforward solution to the problem of ambiguity, which humans have been shown to employ, is to just "wait-and-see" [SSG78]. If there is no need to disambiguate at the present state, then there is no need to worry. Humans often infer as much as is "reasonable" and when too much ambiguity is met, they just wait for more input to help them disambiguate [SSG78].

Van Beek and Cohen [vBC91] suggest that, in dialogue systems, it is not always necessary to resolve ambiguity to give a proper response. Oftentimes, the response would have been similar regardless which plan (among the ambiguities) the user has. Alternatively, the system can generate a reply which 'covers' all the ambiguous cases. This is especially useful if the number of ambiguities is small.

When ambiguity *does* matter, Van Beek and Cohen suggest the other method which humans use to resolve ambiguity, that of clarification. Clarification dialogue simply (directly

or indirectly) solicits information from the user for disambiguation. In other words, if an ambiguity exists, one simply needs to ask. This method is also used in [LRS99].

One downside of the clarification approach, however, is that, if done too often, it can quickly throw a dialogue into disarray. Also, clarification is usually more effective if there are only a few ambiguities to choose among. Clearly, if clarification is to be used, it must be used sparingly. Other methods are also needed.

Solution: Using Uncertain Inference

The second approach to the problem of ambiguity is to use uncertain inference. This at least gives each ambiguity some measure of probability which the system can use to further weight its options. We discuss here some of the many approaches which have been used.

Dempster-Shafer Carberry [Car90a] showed how to incorporate Dempster-Shafer probabilities to the focusing heuristics in her plan recognition system [Car90b] (described in Section 4.3 above). Bauer [Bau94] described a plan hierarchy closure method to attach Dempster-Shafer probabilities to any plan hierarchy.

Belief Networks Several systems have used Bayes Nets to integrate uncertain inference into plan recognition. In Albrecht, Zukerman and Nicholson [AZN98], dynamic belief networks were trained to predict a user's goal based on observed actions in a multi-user dungeon video game. Horvitz and Paek [HP99, PH00, HP00, PHR00] use dynamic Bayesian Networks to recognize user intentions in several dialogue domains. Charniak and Goldman [Gol90, CG91, CG93] built an entire natural language understanding system, including plan recognition, in a unified dynamic belief network. Plan hypotheses were generated by piecing together evidence (previous utterances, plan roles of items in the current utterance, etc.) A priori probabilities of the likelihood of observations were then percolated through the network to determine the overall likelihood of the hypothesis. As is discussed in Section 6.4 below, this system may be adaptable to the problem of incremental plan recognition.

Probabilistic Plan Libraries Calistri-Yeh [CY91] suggests a simpler system of placing probabilities on edges in a Kautzian plan hierarchy. Combining this evidence allows an overall likelihood to be computed for a hypothesis within the plan space. Calistri-Yeh's system also supports recognition of a range of types of erroneous plans, with likelihoods computed for these erroneous plans as well.

6 Problems with Current Plan Recognition in Natural Language Dialogue

In the previous section we described several problems with prototypical plan recognition and some proposed solutions. Here we discuss outstanding problems with current plan recognition systems, especially as they are used in the domain of natural language dialogue

systems. In the next section we describe a system which provides a preliminary solution to several of these problems, namely recognition of interleaved plans, repairing mistaken inferences, and integration with linguistic representation.

6.1 Scalability and Runtime

Lesh [LE96] points out that previous plan recognition work has typically only been investigated in domains where the number of goals were less than one hundred. This brings up the question of whether these plan recognition techniques can be scaled up to domains with thousands or even hundreds of thousands of goals and plans. Due to the exponential runtime of most plan recognition systems, this is very doubtful. In fact, even in small domains, plan recognition is still a very costly subroutine.

Although we consider this problem to be very much an open issue still, there have been a few attempts to try to confront it. Mayfield [May92] suggests a very pragmatic approach. His plan recognition system uses a utility function to determine if it should continue to search for a solution. Thus his system weighs the tradeoff between accuracy and utility. Because of the backward-chaining nature of typical plan recognition, a system finds intermediate goals before finding an 'ultimate goal' (and END in Kautzian terms). These partial results may be useful enough to a plan recognition system.

Of course, as Mayfield points out, utility can only be measured in an environment where plan recognition is being used within a larger system. His plan recognizer would do the best it could within a reasonable amount of time and then send the results to the next phase of processing in the larger system. This component (which used the results of plan recognition) would then know if the results it received from the plan recognizer were good enough or not. If a higher-level goal, or more accuracy was needed, the component would send the results back to the plan recognizer, which would continue its search until the time factor outweighed utility. The process would continue in this manner until the consuming component was satisfied.

Although this is not a cure-all for the scalability problem, Mayfield's approach is definitely a step in the right direction.

Lesh [LE96, Les97, Les98] has taken plan recognition research to the next level of scalability. Using automatic plan library construction techniques (see below), he has constructed plan libraries in several domains which contain up to 500,000 goals.

Lesh's research has been on *goal recognition*, which is a special case of plan recognition where only the user's goal is recognized, not the plan by which the user is trying to accomplish the goal. Lesh points out that goal recognition has several potential applications, including that of intelligent interfaces. We also believe goal recognition can be used to improve plan recognition. A fast goal recognizer could be used to sufficiently narrow the search space to allow a slower plan recognizer to be applied afterwards.

Lesh's goal recognizer performs in time logarithmic to the number of goals in the plan library. It does this by using biases to prune away sections of incompatible goals.

We applaud Lesh's efforts to scale up plan recognition systems. However, there is still much work to be done in this area. Lesh's algorithm only works in such fast time because

his set of goals are represented in a subsumption hierarchy. Many of the 500,000 goals in his system are actually conjunctions of more primitive goals. This subsumption allows his algorithm to prune away huge sections of the search space at a time. It is not clear to what extent dialogue-type goals could be represented in this representation. It is also unclear if this new representation and recognition routine could use any the extensions to prototypical plan recognition described in Section 5 (recognition of erroneous plans from the user, for example). We believe that it would be worthwhile to try to 'port' Lesh's techniques into plan recognition for dialogue systems.

6.2 Portability to New Domains

Another big problem with plan recognition in dialogue systems is that of portability. This actually encompasses two issues. First, most plan recognition systems have domain specific heuristics, and it is not clear if these are applicable across domains even within dialogue systems. Second is the problem of creation of new plan libraries for new domains. We discuss the second issue here.

Creating plan libraries by hand is a slow and painstaking process. This process may be acceptable for domain independent plans (such as discourse plans, meta plans, etc. as discussed above), which are most likely a manageable number and need only be specified once for any dialogue system. However, the need to hand craft new plan libraries for the domain plans of new domains is a serious impediment to dialogue system portability[5]. The slow process of hand construction of plan libraries is also a major problem to scaling up plan recognition systems to domains of more than several hundred plans (see the discussion above).

Again, we consider this an open issue. Lesh [LE96, Les97, Les98] suggests using concept learning to automatically build plan libraries (at least libraries of goals). Concept learning uses a set of primitives to hypothesize sets of goals in the domain.

A possible similar approach to construct a plan library (as opposed to a goal library) would be to use a set of primitive actions as well as a planner. The planner could generate possible plans in the domain for accomplishing certain goals. This is an area of our future work (see Section 8 below).

6.3 Differentiation between Immediate Execution and Future Planning

One problem with plan recognition in dialogue systems is that of distinguishing if the intention of a user's utterance is to *execute* a plan (or action) or to simply add it to an overall plan which is intended to be executed later. Most plan recognition systems have either been in domains where the user (or system, or both) is actually executing the steps of the plan as the discussion takes place (such as expert/apprentice domains), or where the user and system are simply constructing a plan to be executed at some later time.

Of course, there are domains in which this activity is mixed. Consider the TRIPS 911 domain, where a user is working with the system to formulate *and execute* plans to rescue

[5]Portability of a *core* dialogue system is one of the goals of the TRIPS project [ABD+00].

various victims around the county. Here, the user and system will formulate a certain plan and execute it, and then formulate other plans. In certain situations which do not allow for careful planning, the user may request that actions be executed without first formulating a specific plan (such as sending an ambulance to a heart attack victim). Future plan recognition systems will need to be able to recognize and deal with these kinds of domains.

Although Ramshaw and Ardissono's work [Ram89b, Ram89a, Ram91, ABL96] provide hints at how one might begin to structure such a solution, this problem has not yet been addressed by the literature.

6.4 Incremental Understanding

Nearly all current dialogue systems operate at an utterance level of granularity. Human dialogue is very different [Cla97]. In human-human dialogue, participants are clearly understanding and reacting at a much finer granularity. Back-channeling, interruptions, and turn-taking are several examples that humans are understanding at a word-level at least.

Although continuous understanding is a problem area for all levels of natural language understanding [Sto01], it is particularly problematic for plan recognition. Most plan recognition systems only work on a complete speech act, which means that the entire user utterance must be said and processed before the plan recognition process ever begins. In order to build dialogue systems which are capable of continuous interaction, this is a problem which must be overcome.

The only dialogue system we know of that attempts to do incremental understanding is DUG–1 [NMJ$^+$99]. This system, however does not attempt to do plan recognition. It works in a simplified domain of booking rooms for presentations, which only requires basic slot filling for understanding. Continuous understanding is provided by keeping and filling, in parallel, all ambiguous versions of the slots, and then ranking the best by a set of heuristics. It does not seem that this approach would generalize to plan recognition.

There is hope of a possible solution, which may be able to be built on the belief network plan recognition system of Goldman and Charniak [Gol90, CG91, CG93] (described in Section 5.3 above). Although the focus of their research does not appear to be incremental understanding, we believe that there are concepts in this work that could be used for incremental plan recognition. Firstly, their system seems to already support a limited form of incremental plan recognition. Their system treats each filler of a plan role as a separate evidence node in the belief network. An event (i.e. action type) without any fillers can be added to the network and cause net plan hypotheses to be formed. Event rolls can be added separately as well, although Goldman and Charniak did not allow them to lead to new hypotheses, only to affect the likelihood of preexisting hypotheses. They state the ambiguity problem as the reason for this disallowance.

With their system, upon hearing and understanding what type of event has occurred, the event node can be added to the network and begin to cause new hypotheses to be proposed. There is still much work to be done in this area, however we believe that it may be possible to start such work on the base of Goldman and Charniak's belief network plan recognition.

6.5 Interleaved Plans

Another outstanding issue for plan recognition systems is recognition of interleaved plans, especially when one action shared among two or more plans. Lesh [LE96] suggests a method for goal recognition which simply treats the two goals as a single conjoined goal, although it is unclear how such a representation could be used in a dialogue system which needs to explicitly keep track of each goal, as users may want to perform problem solving activities on each goal separately, regardless if they were explained in an interleaved way. Below we present an algorithm which can account for interleaved goals and still keep each goal (and associated information) separate.

6.6 Repairing Mistaken Inferences

One topic that has typically been absent in the discussion of plan recognition systems for dialogues is this. What happens when the system makes the wrong hypothesis? Many of the natural language systems discussed above to not seem to have a repair mechanism. The system is based on the assumption that all previous hypotheses were correct. When a mistake has been made, probably the only thing these systems could do would be to re-recognize all of the previous input, keeping in mind the new evidence we have received.

Logical systems like Kautz' treat plan recognition as non-monotonic reasoning, so incorrect inferences can be changed by new evidence. However, most natural language systems, ([Car90b, LC91, Ram91] for example), use heuristics and seem to have no mechanism for correcting incorrect inferences.

Our system below follows more closely along the human model discussed in [SSG78], and is able to recover from incorrect hypotheses in light of new evidence.

6.7 Integration with Linguistic Representation

The last problem with current plan recognition in dialogue which we discuss is perhaps the most amorphous, since it deals with integrating plan recognition into a larger system. Most plan recognition work, even for dialogue systems, has either been a standalone proof-of-concept system, for which input was 'magically' converted in the absence of a computational front end, or has been part of a system with limited processing capability of natural language input. Either way, the knowledge representation has been more geared towards plan recognition and, as pointed out by Di Eugenio [DE95], has not been very linguistically motivated. If plan recognition systems in dialogue are going to be useful at a general level, they must be able to function with more linguistically motivated input which can be processed by state-of-the-art natural language processing systems.

Below we propose the a system preprocessor, which converts linguistic actions into their planning atomic actions. We also tie linguistics and planning together by having a linguistic description field in our plan library, which allows understanding of more general linguistic signals such as "use instrument".

7 Our Plan Recognition System

We have created a plan recognition system for use in the TRIPS dialogue system [AFS01, ABD+01, ABD+00, FA98] that overcomes several of the shortcomings mentioned above. In this section we describe this system, which we plan to integrate into the TRIPS system this summer. We first talk about motivations for the system and how it needs to interact with the TRIPS system. We then describe knowledge representation in the system and the plan recognition algorithm. Finally we discuss how this system overcomes several of the above-mentioned shortcomings of current plan recognition systems, namely: dealing with interleaved plans, repair of mistaken inferences, and a linguistic decomposition preprocessor, which allows the system to be more fully integrated with natural language input.

7.1 Motivations

As stated above, the motivation for this project was the TRIPS dialogue system. (For a general overview of TRIPS see Section 2.3 above). We desired to extend the plan recognition principles used in the TRAINS project [All97, FAMR96] (also described in Section 4.6 above), as well as improving and expanding both the range and accuracy of plan recognition. The plan recognition system herein described will be integrated into TRIPS over the next few months.

In analyzing the domain of TRIPS-911 (a dialogue system for handling 911 emergencies), we found examples of phenomena that current plan recognition systems could not process. We will discuss these phenomena later as specific motivations for some of the features of our plan recognition system. First, however, we describe the overall TRIPS architecture and then how plan recognition (within the Task Manager component) interacts with the rest of the system.

In AI applications, plan recognition never occurs in a vacuum. Rather it is just one piece of a complex architecture of components with which it must interwork in order to provide desired result. These next few sections show plan recognition within TRIPS. We will then describe some of the plan phenomena one can encounter in the TRIPS-911 domain.

TRIPS Architecture

TRIPS[6] has a modularized, asynchronous, agent-based, message passing architecture. Figure 1 shows a generalized view of the architecture.

As shown in Figure 1, multi-modal input comes in to the system either through speech recognition, the keyboard, or other GUI interaction. This input is then passed to the Parser, which outputs a list of (direct[7]) speech acts which describe the input.

[6] This section only provides very high level details of the system (those needed to understand the interaction of plan recognition). For more details about architecture see [AFS01, ABD+00], or our webpage at *http://www.cs.rochester.edu/research/cisd/*.

[7] As opposed to indirect speech acts [Sea75], the processing of which is described below.

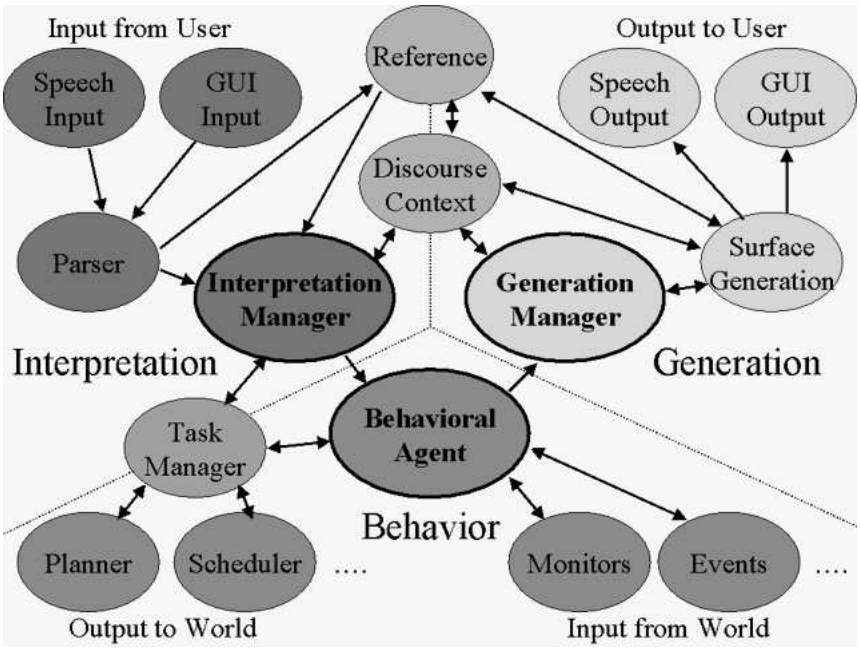

Figure 1: TRIPS Architecture

These speech acts are then sent to the Interpretation Manager, which is the main interpretation agent of the system. The Interpretation Manager, attempts to resolve all references in the input (via the Reference Manager), and the, using the method of Hinkelman and Allen [HA89, FAMR96] (also Section 4.6 above), computes a set of preliminary indirect speech acts that the input could account for. It then asks the Task Manager to evaluate each of these possibilities. The Task Manager (more details below), uses plan recognition to attempt to fit the new evidence into its existing plan structure and returns a score that describes how well the proposed indirect speech act integrates with the current state of the plan recognizer.

The Interpretation Manager then passes its best interpretation on to the Behavioral Agent. This autonomous agent then deals with the input as it sees fit (i.e. by accepting and performing the request, tabling the request in order to perform some more urgent action, etc.) The Behavioral Agent then communicates its intentions and/or results to the Generation Manager, whose job it is to communicate this back to the user in a socially and linguistically coherent manner.

Again, this has been just a quick overview of the system architecture. But, it gives us enough detail to be able to discuss in more detail how plan recognition fits into the system by describing the Task Manager.

The Task Manager

The Task Manager, as one of its many jobs, performs plan recognition in the system. The purpose of this section is not to describe the actual plan recognition (which is described

later), but to discuss the *interaction* of the plan recognition with the rest of the system.

As stated above, the input to the plan recognizer is an (indirect) speech act hypothesis (generated by the Interpretation Manager), and the output is the problem solving (planning) act the user intended with the speech act, and a score of goodness, which the Interpretation Manager uses to decide which (problem solving) interpretation of the input is the best. It then sends it off to the Behavioral Agent.

Interesting Plans from the TRIPS-911 Domain

Many of the plan recognition systems mentioned above were designed to be used in a specific application. Carberry's focusing heuristics [Car90b], for example, were developed from intuition from student advising dialogues. Grosz' [Gro78, Gro81] observation that dialogues tend to follow the structure of the task was obtained by looking at expert/apprentice dialogues.

However, what is true for one domain, may not necessarily (or to the same degree) be true in another. In student advising dialogues, people do probably tend to flush out a plan in a certain order. Also, in expert/apprentice dialogues where actions are being executed during the dialogue, it is certainly true that the dialogue will tend to follow the (temporal) order of the task at hand.

We believe that, by looking at examples from more and more domains, we are slowly beginning to see the wide range of *plan navigation* strategies that humans use in dialogue. We believe that within the TRIPS-911 domain we have found a new (at least to plan recognition in dialogue) such strategy, that of interleaved plans.

Before we look at this phenomenon, though, let us first consider a 'normal' bottom up exchange in this domain, which will serve as a comparison.

Dialogue 1
U1: Send Ambulance 1 to Irondeqoit Mall
U2: Pick up the person there
U3: Go to Strong Hospital
U4: And drop the person off there

Dialogue 1 is a prototypical example of a bottom-up, single plan navigation sequence. The user's goal is to rescue (i.e. get to a hospital) the person at Irondeqoit Mall (to Strong Hospital). The user's plan is to send Ambulance 1 to the mall, pick up the person, take him to Strong Hospital, and drop him off. This sequence follows both Carberry's and Grosz' observations of focusing. The user focuses on one goal, and even navigates it in the same temporal order that its actions will take.

Now, however, let us consider an interleaved example that seems plausible enough in this domain where it is sometimes more efficient (in terms of execution time) to interleave the actions of plans.

Dialogue 2
U1: Send Ambulance 1 to Irondeqoit Mall
U2: Pick up the person there
U3: Go to Midtown Plaza
U4: Pick up the person there
U5: Now go on to Highland Hospital
U6: Drop off the person from Midtown there
U7: Go on to Strong Hospital
U8: And drop off the Irondeqoit person there

In Dialogue 2, the user has two goals: to rescue the person at Irondeqoit Mall (to Strong Hospital) and to rescue the person at Midtown Plaza (to Highland Hospital). Note that the first of these goals is identical to that of Dialogue 1.

Instead of navigating the details of one goal and then the other, the user switches between the two. Why is this? It turns out in this domain that combining (i.e. sharing and/or temporally interleaving) the steps of multiple plans can actually result in a more efficient executing (in this case in terms of time taken). Midtown Plaza and Highland Hospital are 'on the way' from Irondeqoit Mall to Strong Hospital, and thus combining steps results in a more efficient plan throughput. We will call this phenomenon *plan synergy*. We do not believe that plan synergy is not limited to 911 domains. There are many domains in which one could imagine plan synergy to be naturally occurring, and thus potentially utilized by humans. For this reason, a general plan recognition system for natural language dialogue must be able to handle interleaved plans, and, as a result, plans which share actions.

7.2 Knowledge Representation

This section discusses the knowledge representation of our system. We first describe the representation of speech acts (which are input to the system). We then discuss the representation of objective frames and finally we discuss how actions are represented in our system. This system uses a simplified version of the knowledge representation used in TRIPS.

Speech Act Input

The input to the plan recognition system (from the Interpretation Manager), is a reference resolved speech act. We describe here the knowledge representation of speech acts only insofar as is necessary to understand the plan recognition mechanism. For more information about knowledge representation in TRIPS, see [FAMR96].

The content of a speech act is essentially a logical form from the parser. Each event has a number of argument slots which are filled with other events or entities or with variables when they are not indicated by the speaker. The domain dependent parser [FAMR96, All94] maps ambiguous surface verbs onto relevant domain events. Figures 2 and 3 show examples of surface strings and their corresponding speech acts.

```
(REQUEST
  :contents
     (move BUS1 GREECE-MALL))
```

Figure 2: Speech Act for "Send bus one to the mall in Greece"

```
(REQUEST
  :contents
     (use
        AMB1
        (achieve
           (at-loc P-PITTS STRONG))))
```

Figure 3: Speech Act for "Use an ambulance to get the person from the Pittsford fire station to Strong Hospital"

Figure 2 shows a simple speech act. The speech act type is `REQUEST`. In the contents, the surface verb "send" has been mapped to the predicate `move` in the logical form. The all-caps entries `BUS1` and `GREECE-MALL` are world entities which have been resolved from the referring expressions "bus one" and "the mall in Greece" respectively. Implicit in this compact representation is that the predicate `move` has two arguments. The first (occupied in this example by `BUS1`) is a vehicle and the second, a destination. If the input had not (explicitly) specified one of the arguments, say, for example, in the sentence "move to the mall in Greece", the corresponding speech act contents would use an (implicitly) typed variable, as shown here.

```
:contents
  (move ?v GREECE-MALL)
```

The speech act in Figure 3 shows how propositions can be nested. The outer level has the predicate `use`, the second argument of which is a proposition, namely, `achieve` (mapped from the verb "get"). `Achieve` also takes a proposition, which is `(at-loc P-PITTS STRONG)`. The handling of nested propositions adds a layer of complexity to systems which match observed actions bottom-up (as for the speech act in Figure 2).

Another complication is that, although the parser is domain specific and maps verbs onto predicates which represent relevant senses in the domain, it has no knowledge of what the system planner considers to be 'atomic actions' in the domain. This is an example of the mismatch between linguistic and planning representation discussed in Section 6.7 above. Consider the following utterance and corresponding speech act (where "here" refers to the mall in Greece).

U1: Take the person here to Strong Hospital with Bus 1

```
(REQUEST
  :contents
     (transport P-GREECE BUS1 GREECE-MALL))
```

The `transport` action takes three arguments: a person (`P-GREECE`), a vehicle (`BUS1`), and a location (`GREECE-MALL`). `Transport` is not an atomic action in the planner, which knows nothing about it. However, we still use this linguistically motivated representation. We describe our strategy for dealing with this incongruity of representations in Section 7.4 below.

Plan Representation

Our representation of plans is hierarchical and similar to that of Carberry [Car90b] and Allen [FAMR96, All97]. As Allen notes [All97], a dialogue system must not only keep track of the goal and 'plan' (as defined by a set of temporally ordered atomic actions), but also *how* the (hierarchical) steps of the plan relate to each other. This is especially true in domains where the user and system are constructing a plan to be executed at some later time. Users will often want to revisit parts of the plan and do revisions, comparisons, etc. We will call (along the lines of [AFS01, ABD$^+$01]) this augmented representation an *objective* and the set of (possibly) hierarchical actions that accomplish an objective, a *solution*.

An objective is comprised of several elements (besides just the goal statement), which help in plan recognition as well as in providing the rich information needed to support user dialogue as discussed above. For the remainder of this section we discuss each of these elements. Further sections describe in detail the representation of solutions and of atomic actions. We discuss the `RescuePerson` objective as a detailed example. The following are elements of an objective.

Goal The goal describes the desired end state of the objective. This is what is typically thought of as a goal in planning literature and will not be discussed further here.

Resources This is a list of typed variables that are considered resources for the objective. A resource is typically something that is considered 'scarce' in a planning domain and that must be allocated in order to find a solution. See [AFS01, ABD$^+$01] for a more detailed discussion of resources.

Input Variables We start with frames for abstract objectives. Input variables *instantiate* the objective. Typically these are parameters which distinguish this objective from other similar objectives. This is a typed list of variables which are instantiated as the dialogue progresses.

Linguistic Description As stated above, we are not doing plan recognition in a vacuum, but are using it to support linguistic dialogue. The linguistic description of an objective describes (in slot-argument form) the goal of the objective, its resources, and so on. This description is used to interpret domain independent linguistic constructs which are used to specify parts of a plan. Further details of this process are given below.

Solutions The set of solutions (plans) for this objective which have been discussed. A common activity in problem solving is actually discussing different solutions for an objective, comparing them, and so forth. This representation allows us to model this behavior in the dialogue. The representation of a single solution is discussed below in more detail.

Focused Solution This is a pointer to which solution is currently in focus (i.e. being developed, evaluated, etc.)

Constraints The set of constraints on the objective.

Solutions

As stated above, a solution is what is typically called a 'plan' in the plan synthesis literature. It is an ordered set of (hierarchical) actions whose execution is intended[8] to bring about the goal(s) of the objective.

A solution is a set of *expected actions* and *agreed actions*. Before a dialogue begins, an solution has a set of expected actions that constitute a common *recipe* for that plan. As actions are grounded with a user, they are moved from the set of expected actions to the agreed action list. This builds expectations into the recognizer, which behavior was also found in humans [SSG78].

Atomic Actions

Atomic actions are represented as an action predicate with arguments (much like the `move` example above). In addition, each action is followed by a list of preconditions that must be true at the time the action is executed.

An Example: The `RescuePerson` Objective Frame

To illustrate the above sections, and to serve as a base for further discussion of the plan recognition system, we consider the `RescuePerson` objective frame, which is shown in Figure 4. This frame represents the objective of rescuing a person (i.e. getting them to a hospital) in a 911 domain. We will briefly discuss each of its features.

Goal The goal is to get a person (`?p`) to a hospital (`?h`).

[8] For an in-depth discussion of what exactly constitutes a plan, see [Pol90]

```
(RescuePersonFrame
     :goal (at-loc ?p ?h)
     :ling-desc (:id D1 :predicate Rescue :theme ?p
                 :goal ?h :instrument ?v)
     :resources ((Vehicle ?v) (Hospital ?h))
     :input-vars ((Person ?p) (Ailment ?a) (Location ?l))
     :constraints ((Hospital ?h) (Person ?p) (Vehicle ?v)
                   (Location ?l) (Ailment ?a) (has ?p ?a)
                   (treats ?h ?a) (at-loc ?p ?l))
     :focused-solution S1
     :solutions ((:id S1
         :agreed-actions
           NIL
         :expected-actions
           (((move ?v ?l) NIL)
            ((load ?p ?v) ((at-loc ?p ?l) (at-loc ?v ?l)))
            ((move ?v ?h) ((in ?v ?p)))
            ((unload ?p ?v) ((in ?v ?p) (at-loc ?v ?h)))))))
```

Figure 4: The `RescuePerson` Objective Frame

Resources Resources are a vehicle (to carry the person) and a hospital.

Input Variables Input variables are the person, their ailment, and their current location.

Linguistic Description The linguistic description contains the predicate `Rescue` (i.e. "rescue a person"), with the theme being the person, the goal (in a linguistic sense) being the hospital, and the instrument being the vehicle. These provide, as the name denotes, a linguistic description of this objective.

Solutions The solutions contain just one solution (since we haven't discussed any solutions yet), which has a typical recipe in its expected actions. The recipe is to move the vehicle to where the person is, load the person into the vehicle, move the vehicle to a hospital, and unload the person. Individual preconditions constrain this to be a solution which actually accomplishes the goal (discussed above). The `load`, for example, has the preconditions that both the vehicle and the person be at the location `?l`. The second `move` has a precondition that the person still be in the vehicle when it is moved, and so forth.

Constraints These are general constraints on the solution. Some of these constrain types on variables (i.e. `?p` has to be a person, `?v` a vehicle, etc.) Others provide additional constrains such that the hospital of our solution must be able to handle cases of the ailment the person has and so forth.

7.3 Algorithm

In this section we describe the basic plan recognition algorithm used in our system. In later sections we discuss additional details of the algorithm which allow us to handle the phenomena as we discussed in Section 7.1 above.

In our restricted domain, we do not utilize hierarchical plans, although we believe the concepts here can be easily extended to handle such a model[9]. Instead, a user's overall objective simply consists of a set of `RescuePerson` objectives, each of which corresponds to the rescuing of a particular person.

The system keeps track of all objectives, present and past. This allows the user to always go back and query[10] about previous objectives. Objectives which are fully instantiated, and are out of focus are determined to be *grounded*, while recently discussed objectives, or objectives which have not fully been instantiated are *pending*. This dichotomy allows us to see where to look to 'fit' new input in to the existing plan structure.

Whereas many other plan recognition systems generate a large number of possible interpretations, ours does not. We attempt to follow the observation that humans only generate one hypothesis, which is then revised when necessary [SSG78]. Details of this hypothesis revision mechanism are described later.

Linguistically, humans use a number of (sometimes mixed) plan navigation strategies to convey their intentions. In a top-down strategy, the human first explicitly states their goal, and then explores the various details. In a bottom-up strategy, the human describes individual actions, from which the system must recognize the implicit goal. Humans also use domain independent linguistic constructs to specify certain planning level roles. We describe below how our system recognizes plans with input from different plan navigation strategies. Our system is also capable of handling compound statements with linguistic roles. We describe this lastly below.

Top-down Recognition

In a top-down strategy, the user states their explicit goal. Consider the following input from the user.

U1: Let's get the person from the Pittsford fire station to Strong Hospital

The speech act content of this statement is

```
(achieve
 (at-loc P-PITTS STRONG))
```

[9]Indeed, that is one of the subjects of our future work (see below)

[10]Queries, comparisons, and other problem solving actions will be supported in a future version of the system.

Notice that the verb "get" from above has been recognized by the parser as the `achieve` sense of the word. The algorithm recognizes that `achieve` is used to describe goal statements. A goal statement such as this is a sign that the user is moving focus to a new objective, so the system looks through its plan library to see if the goal statement matches (unifies with) the goal of any of the objective frames in the objective library.

In this case, `(at-loc P-PITTS STRONG)` unifies with the goal `(at-loc ?p ?h)` in the `RescuePerson` objective. We then instantiate the unified variables (`?p` and `?h`) throughout the entire frame and check the solution preconditions and general constraints. If these are ok, then we have a match.

This (partially) instantiated objective is then placed on the list of recognized objectives. It is now salient and activated for possible further input relating to this objective[11].

Bottom-up Recognition

Another strategy used by humans is to describe the solution action by action, by which the hearer is expected to recognize their plan. Consider this set of bottom-up commands from the user[12].

U1: Send Bus 1 to the mall in Greece
U2: Pick up the person there
U3: Now go to Rochester General Hospital
U4: And unload the person there

Let us consider this example in the context of several previous objectives that the user and system have been discussing. Here, with statement U1, the user intends to move focus to a new objective. The speech act content for U1 is `(move BUS1 GREECE)`.

We know from our ontology that `move` is an action. We first try to match this to an expected action of any pending objectives. Objectives are searched in order of how recently they have been mentioned. We discuss below several cases where this does match (falsely) a pending objective, but for now, we will assume that this did not match any expected actions.

If no match is found with pending objectives, we hypothesize that the user is moving on to a new objective. The system searches through its objective library for an expected action in one of the typical recipes which matches this `move`. In this case, a match is found with the first `move` action in the `RescuePerson` frame. Variables are instantiated and constraints are checked as described above. This checks out, so we move this action to the agreed actions list, and add this instantiated frame to our list of pending objectives. The state of the frame is shown in Figure 5.

[11] Within the real TRIPS system, the plan recognizer only returns this to the Interpretation Manager as its hypothesis for the intention. This is then passed to the Behavioral Agent, which, if it decides to so update the shared plan, instructs the plan recognizer (i.e. Task Manager) to update its structures. In our discussion, however, we will assume this happens seamlessly in order to simplify the explanations.

[12] For simplification, we have omitted system responses in these and other examples.

```
(RescuePersonFrame
     :goal (at-loc ?p ?h)
     :ling-desc (:id D1 :predicate Rescue :theme ?p
                 :goal ?h :instrument BUS1)
     :resources ((Vehicle BUS1) (Hospital ?h))
     :input-vars ((Person ?p) (Ailment ?a) (Location GREECE))
     :constraints ((Hospital ?h) (Person ?p) (Vehicle BUS1)
                   (Location GREECE) (Ailment ?a) (has ?p ?a)
                   (treats ?h ?a) (at-loc ?p GREECE))
     :focused-solution S1
     :solutions ((:id S1
         :agreed-actions
          (((move BUS1 GREECE) NIL))
         :expected-actions
           (((load ?p BUS1) ((at-loc ?p GREECE) (at-loc BUS1 GREECE)))
            ((move BUS1 ?h) ((in BUS1 ?p)))
            ((unload ?p BUS1) ((in BUS1 ?p) (at-loc BUS1 ?h)))))))
```

Figure 5: RescuePerson Frame after U1

```
(RescuePersonFrame
     :goal (at-loc P-GREECE ?h)
     :ling-desc (:id D1 :predicate Rescue :theme P-GREECE
                 :goal ?h :instrument BUS1)
     :resources ((Vehicle BUS1) (Hospital ?h))
     :input-vars ((Person P-GREECE) (Ailment HA) (Location GREECE))
     :constraints ((Hospital ?h) (Person P-GREECE) (Vehicle BUS1)
                   (Location GREECE) (Ailment HA) (has P-GREECE HA)
                   (treats ?h HA) (at-loc P-GREECE GREECE))
     :focused-solution S1
     :solutions ((:id S1
         :agreed-actions
          (((move BUS1 GREECE) NIL)
           ((load P-GREECE BUS1) ((at-loc P-GREECE GREECE) (at-loc BUS1 GREECE))))
         :expected-actions
           (((move BUS1 ?h) ((in BUS1 P-GREECE)))
            ((unload P-GREECE BUS1) ((in BUS1 P-GREECE) (at-loc BUS1 ?h)))))))
```

Figure 6: RescuePerson Frame after U2

```
(RescuePersonFrame
     :goal (at-loc P-GREECE ROC-GEN)
     :ling-desc (:id D1 :predicate Rescue :theme P-GREECE
                 :goal ROC-GEN :instrument BUS1)
     :resources ((Vehicle BUS1) (Hospital ROC-GEN))
     :input-vars ((Person P-GREECE) (Ailment HA) (Location GREECE))
     :constraints ((Hospital ROC-GEN) (Person P-GREECE) (Vehicle BUS1)
                   (Location GREECE) (Ailment HA) (has P-GREECE HA)
                   (treats ROC-GEN HA) (at-loc P-GREECE GREECE))
     :focused-solution S1
     :solutions ((:id S1
         :agreed-actions
          (((move BUS1 GREECE) NIL)
           ((load P-GREECE BUS1) ((at-loc P-GREECE GREECE) (at-loc BUS1 GREECE)))
           ((move BUS1 ROC-GEN) ((in BUS1 P-GREECE))))
         :expected-actions
           (((unload P-GREECE BUS1) ((in BUS1 P-GREECE) (at-loc BUS1 ROC-GEN)))))))
```

Figure 7: `RescuePerson` Frame after U3

```
(RescuePersonFrame
     :goal (at-loc P-GREECE ROC-GEN)
     :ling-desc (:id D1 :predicate Rescue :theme P-GREECE
                 :goal ROC-GEN :instrument BUS1)
     :resources ((Vehicle BUS1) (Hospital ROC-GEN))
     :input-vars ((Person P-GREECE) (Ailment HA) (Location GREECE))
     :constraints ((Hospital ROC-GEN) (Person P-GREECE) (Vehicle BUS1)
                   (Location GREECE) (Ailment HA) (has P-GREECE HA)
                   (treats ROC-GEN HA) (at-loc P-GREECE GREECE))
     :focused-solution S1
     :solutions ((:id S1
         :agreed-actions
          (((move BUS1 GREECE) NIL))
           ((load P-GREECE BUS1) ((at-loc P-GREECE GREECE) (at-loc BUS1 GREECE)))
           ((move BUS1 ROC-GEN) ((in BUS1 P-GREECE)))
           ((unload P-GREECE BUS1) ((in BUS1 P-GREECE) (at-loc BUS1 ROC-GEN))))))
         :expected-actions
           NIL
```

Figure 8: `RescuePerson` Frame after U4

Notice that we now have several expectations about further actions in the solution of this objective. Since we have instantiated the vehicle as `BUS1`, we now expect a `load` of a person into `BUS1`. Other expectations can also be seen in the same figure.

When we receive the input for utterance U2, `(load P-GREECE ?v)`, we follow the same procedure outlined above. This time, we find a match right away in the pending objectives. The most recently mentioned objective is the one we have just created, and its `load` action matches. The search stops here. We do not search any further, either through other pending objectives or through possible new objectives once we have found a match. If an older pending objective is still waiting for a `load`, it will not confuse the system. Most likely, only one `load` will match and pass the constraint check. If not, we believe that humans (following focusing principles) will only consider the most recent, as it is the most salient objective. Recall that dialogue is intended recognition. The user is trying to convey his intentions in the most straightforward manner possible.

This utterance also provides us with the person variable. We also assume that we knew previously that `P-GREECE` has the ailment of a heart attack (`HA`), which fills in our Ailment variable. The result of utterance U2 is shown in Figure 6.

The next two utterances, `(move ?v ROC-GEN)` and `(unload P-GREECE ?v)`, are likewise matched, as shown in Figures 7 and 8. Notice that after the second `move` is mentioned, the system predicts the final `unload` perfectly. Input of the expected `unload` confirms our prediction, and it too is moved to the agreed actions list.

Linguistic Role Filling

Besides just mentioning actions or goals, users also sometimes use domain independent linguistic descriptions to signal objective rolls. Consider the following dialogue.

U1: Let's get the person from the Pittsford fire station to Strong Hospital
U2: Use Ambulance 1

Notice that U1 is the same utterance from our top-down navigation example above, and processing of this utterance precedes exactly as described in Section 7.3. After the user has stated his goal, `(at-loc P-PITTS STRONG)`, the system has instantiated the person (`?p`) variable, as well as the hospital (`?h`). However, it is still not clear which vehicle the user intends to use. Utterance U2 has a local form of `(use AMB1)`. Here, the predicate `use` is recognized as a domain independent linguistic strategy which specifies an instrument. The system then tries to unify the specified instrument (`AMB1`) with the `:instrument` slot in the objective's linguistic description. Constraints and preconditions check out, so we now know what vehicle we should use.

Compound Statements

The previous example may be stated more concisely by a user. Consider the following utterance.

U1: Use Ambulance 1 to get the person from the Pittsford fire station to Strong Hospital

The logical form of this statement is

```
(use
   AMB1
   (achieve
      (at-loc P-PITTS STRONG)))
```

Here we have a goal statement (with `achieve`) embedded in a linguistic role description (with `use`). The system handles embedded cases by recursion. It first considers the `achieve`, which is the same as the example above. The `use` is then considered in the environment of the *instantiated* objective from the `achieve`. If constraints and preconditions check out, for both of these, we have found our answer. Otherwise, the search is continued until a match which satisfies both of these is found.

7.4 Linguistic Decomposition

One problem with the approach of bottom-up plan recognition described above is that the original solution recipe (i.e. the expected actions) are a set of 'atomic' actions. Linguistically, however, there are many ways to describe an action, or even set of actions. Consider the following dialogue.

U1: Send Ambulance 2 to Marketplace Mall
U2: Take the person there to Highland Hospital

Utterance U1 is similar to the `move` predicates we have already seen, except, instead of using the verb "move", the user has chosen the verb "send". The TRIPS parser already handles this by recognizing that this "send" is semantically the same as our `move` predicate and mapping it onto that predicate. In fact, its logical form is just `(move AMB2 MARKETPLACE)`. So, different ways of expressing an action is already handled in the parser.

Utterance U2, however, does not quite correspond to any of the actions we have seen before. The verb "take" here seems to, in fact, include the last three actions of our solution recipe, namely, loading a person, moving a vehicle to a location, and unloading the person. The parser does not notice this and give us three different actions. The logical form given to us by the Interpretation Manager is `(transport P-UR STRONG ?v)`. `Transport` does not match any one atomic actions in our solution recipe. We do not want, however, to preclude the user from using this sense of "take".

The fact that actions and groups of actions can be expressed in many different ways linguistically is a very big problem in dialogue systems. Our approach to the group of actions problem is to use a preprocessing step which decomposes predicates into a compound statement of atomic actions, which can then be processed by the mechanisms described above (and below).

Compound linguistic predicates are decomposed by a library of simple decomposition rules. For example, the decomposition rule for "transport" is

```
(transport ?p ?h ?v) --> ((load ?p ?v)
                          (move ?v ?h)
                          (unload ?p ?v))
```

Plan recognition can now run normally on this generated compound statement. This simple decomposition step allows our plan recognizer to handle a much richer set of natural language input.

7.5 Interleaved Plans

In these last two sections, we examine how the plan recognizer handles the plan navigation phenomena described in Section 7.1. In this section we describe how simple interleaved plans are recognized. In the next section we describe how the system revises hypotheses that are determined to be incorrect by further evidence.

Let us consider Dialogue 2 (reprinted here for convenience), which was an example above of an interleaved plan in the 911 domain. We show that the plan recognition algorithm can handle this interleaved plan navigation without need for revision.

Dialogue 2
U1: Send Ambulance 1 to Irondeqoit Mall
U2: Pick up the person there
U3: Go to Midtown Plaza
U4: Pick up the person there
U5: Now go on to Highland Hospital
U6: Drop off the person from Midtown there
U7: Go on to Strong Hospital
U8: And drop off the Irondeqoit person there

The first two utterances proceed just as a normal bottom-up dialogue, filling in the `move` and `load` of an objective. However, when utterance U3 is received, the attempt to match it to the third move of the objective fails because Midtown Plaza is not a hospital.

The system then hypothesizes that the user is discussing a new objective and finds a match with the first `move` of another `RescuePerson` objective. Utterance U4 then matches with the `load` of this new objective, which is the most recently mentioned, and therefore the first searched.

Utterance U5 could match the second `move` of either objective, but since the second objective is the most salient, we search this one first, and upon finding a match, we do not search any further. Utterance U6 confirms this hypothesis when we drop the people from Midtown Plaza off there.

Utterance U7 matches the second move of the original objective (since all actions of the second objective are already agreed), and finally we match the final `unload`. Planning experts may notice that the 'plan' in our solutions is not canonical in the sense that the second `move` does not leave from the place we originally picked up the people. However, since the precondition for the `move` that the people stay in the vehicle has not been violated, this solution actually does accomplish the goal.

It is also important to mention here again, that there are many points in this dialogue that would be ambiguous to one of the plan recognition systems above. However, our algorithm does not even generate or explore any of these ambiguities. Our heuristic of stopping at the first fitting hypothesis helps us avoid some of the runtime expansion problems that most plan recognition systems have[13]. Our heuristics also define an implicit method of choosing between these (otherwise logically equivalent) hypotheses.

7.6 Hypothesis Revision

The previous section showed how interleaved plans are handled by the recognition system, and how (potential) ambiguities are solved. However, there are several places in Dialogue 2 where the system was essentially 'lucky' in choosing the hypotheses it did. For example, the user could have just as easily dropped off the person from Irondeqoit Mall (the person in the first objective) at Highland Hospital instead of the person from Midtown Plaza, in which case our previous hypothesis that the move to Highland was part of the second objective, would have been incorrect. In this section we discuss how the system recovers from such incorrect hypotheses.

We will take as an example, however, an even more complicated example. Dialogue 3 is the same as Dialogue 2 except we substitute Rochester General Hospital for Midtown Plaza (as indicated below).

Dialogue 3
U1: Send Ambulance 1 to Irondeqoit Mall
U2: Pick up the person there
U3: Go to **Rochester General Hospital**
U4: Pick up the person there
U5: Now go on to Highland Hospital
U6: Drop off the person from **Rochester General** there
U7: Go on to Strong Hospital
U8: And drop off the Irondeqoit person there

Note that the *only* change we have made is from Midtown Plaza to Rochester General Hospital. Recall, however, that in the original example, U3 was the point at which we knew that the user was constructing a new objective, since Midtown Plaza wasn't a hospital. In the case of Dialogue 3, however, our matching step will match this as the second `move` of the

[13] Although we still have to search through the entire plan library when finding a new objective. We also have to deal with the ambiguity that arises when more than one objective is matches on a new objective search.

first objective. This seems like a natural hypothesis to make at this point. All things being equal, after hearing U3, we would expect the user to be taking the person from Irondeqoit Mall there. We trace though the recognition process from this point (U4).

U4 is a `load` of the person at Rochester General. This does not match anything in the current objective (we have already loaded the person from Irondeqoit Mall), so we hypothesize that this is part of a new objective and search our objective library. We find a match with the `load` in a new `RescuePerson` objective. Because of the implicit temporal constraints on the order of actions, we would have expected a `move` before the `load`. Whenever a match is not the first item in the list of expected actions, we search (in order of most recent first) all previous actions we have encountered to see if there is a match. In this case we find the previous `move` (U3) matches the first `move` of the new objective. This means that this `move` is now playing a role in two solutions. We also do not alter the previous hypothesis that the `move` was part of the previous objective.

Utterance U5 is a `move` to Highland Hospital. Before we begin searching this, we notice that this action causes us to violate the precondition on the `unload` of the first solution that the vehicle be at the hospital when the unload occurs. This lets us know that our current hypothesis about that objective is incorrect in some way. We begin to 'undo' updates we have done to this objective, beginning with the most recent, which was the `move` to Highland Hospital (U3). After undoing U3, the above-mentioned precondition is no longer violated, since the variable `?h` is no longer bound to Highland Hospital.

We now try to reassign actions we have undone. We see, however, that the `move` from U3 was already playing a role in the second objective, so there is no need to try to reassign it.

Now that all constraints have been handled, we process the new `move` from U5, which now fits in to the second `move` of the second objective. The rest of the example now proceeds as described above, without need for further hypothesis revision.

There are still issues to be worked out with this method (see below). However, we believe that this solution is a step in the right direction. It allows us to account for several phenomena that most other plan recognizers cannot. It also takes into account the fact that we are doing this plan recognition in a dialogue system. It makes use of the intended nature of communication to reduce the search space considerably, as well as to reduce ambiguity. It also contains processing and information which make it easier to integrate linguistic and planning knowledge representation.

8 Conclusions and Future Work

We have described the current state of plan recognition, especially as it relates to natural language dialogue. We have also discussed several problems with the state of the art, and have discussed a plan recognition system which overcomes a few of these, namely interleaved plans, hypothesis revision and integration with linguistic processing. Our system takes advantage of the intended nature of plan recognition in dialogue in trying to process in a more human-like manner.

Immediate future work, as discussed above, is to integrate this plan recognition system into the TRIPS dialogue system. This will necessitate coverage of a broader range of problem solving activities, including queries about solutions, comparisons, and so forth.

We would also like to further investigate the issues of tying together planning and linguistics (the *linguistics-planning interface*). The division of labor between these two areas has not been explored much. Both camps seem to assume certain things from the other, although not much coordination has taken place. We believe that plan recognition systems have not yet taken into account the rich linguistic/discourse information available from natural language input, which could help to improve plan recognition. We also believe that any plan recognition system for natural language dialogue must account for the intentional nature of communication, which has not been explored as far as it should be.

Another direction of future work we would like to pursue is along the lines of the goals of the TRIPS project discussed in [ABD+00]. If dialogue systems are to be commercially viable, we need to be able to build a domain independent core which can be easily specialized to a specific domain. We are trying to build a dialogue system *core* which can be easily portable to other domains. As we discussed above, porting plan recognition systems and the associated plan/objective hierarchies has not been explored in much detail. We would like to explore how to separate the *core* of a plan recognition system from the domain specific knowledge and heuristics and to find a way to be able to easily port this core to new domains.

References

[ABD+00] J. Allen, D. Byron, M. Dzikovska, G. Ferguson, L. Galescu, and A. Stent. An architecture for a generic dialogue shell. *Journal of Natural Language Engineering, special issue on Best Practices in Spoken Language Dialogue Systems Engineering*, 6(3):1–16, December 2000.

[ABD+01] James F. Allen, Donna K. Byron, Myroslava Dzikovska, George Ferguson, Lucian Galescu, and Amanda Stent. Towards conversational human-computer interaction. *AI Magazine*, 2001. To appear.

[ABL96] L. Ardissono, G. Boella, and L. Lesmo. Recognition of problem-solving plans in dialogue interpretation. In *Proc. 5th Int. Conf. on User Modeling*, pages 195–197, Kailua-Kona, Hawaii, 1996.

[ABWF+98] Jan Alexandersson, Bianka Buschbeck-Wolf, Tsutomu Fujinami, Michael Kipp, Stephan Kock, Elisabeth Maier, Norbert Reithinger, Birte Schmitz, and Melanie Siegel. Dialogue acts in VERBMOBIL–2: Second edition. Verbmobil Report 226, DFKI Saarbrücken, Universität Stuttgart, TU Berlin, Universität des Saarlandes, July 1998.

[AFFH86] Jerome Azarewicz, Glenn Fala, Ralph Fink, and Christof Heithecker. Plan recognition for airborne tactical decision making. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 805–811, Philadelphia, 1986.

[AFH89] Jerome Azarewicz, Glenn Fala, and Christof Heithecker. Template-based multi-agent plan recognition for tactical situation assessment. In *Proceedings of the Fifth IEEE Conference on Artificial Intelligence Applications*, Miami, 1989.

[AFS01] James Allen, George Ferguson, and Amanda Stent. An architecture for more realistic conversational systems. In *Proceedings of Intelligent User Interfaces 2001 (IUI-01)*, pages 1–8, Santa Fe, NM, January 2001.

[Ale95] J. Alexandersson. Plan Recognition in VERBMOBIL. In M. Bauer, editor, *IJCAI 95 Workshop on The Next Generation of Plan Recognition Systems: Challenges for and Insight from Related Areas of AI (Working Notes)*, pages 2–7, Montreal, Canada, 1995.

[All79] James F. Allen. *A Plan-based Approach to Speech Act Recognition.* PhD thesis, University of Toronto, 1979. also Technical Report 131/79.

[All83] J. F. Allen. Recognizing intentions from natural language utterances. In M. Brady and R. C. Berwick, editors, *Computational Models of Discourse*, pages 107–166. MIT Press, 1983.

[All94] James F. Allen. *Natural Language Understanding.* Benjamin/Cummings, second edition, 1994.

[All97] James Allen. A problem solving model for mixed-initiative planning. Unpublished TRAINS Technical Note, April 1997.

[AP80] J. F. Allen and C. R. Perrault. Analyzing intention in utterances. *Artificial Intelligence*, 15(3):143–178, 1980.

[AS91] James F. Allen and Lenhart K. Schubert. The TRAINS project. TRAINS Technical Note 91-1, Dept. of Computer Science, University of Rochester, Rochester, NY, 1991.

[ASF+94] J. F. Allen, L. K. Schubert, G. M. Ferguson, P. A. Heeman, C. H. Hwang, T. Kato, M. N. Light, N. G. Martin, B. W. Miller, M. Poesio, and D. R. Traum. The TRAINS project: A case study in building a conversational planning agent. TRAINS Technical Note 94-3, University of Rochester, Department of Computer Science, September 1994. also TR 532.

[Aus62] J. L. Austin. *How to Do Things with Words*. Harvard University Press, Cambridge, Massachusetts, 1962.

[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.

[Bau94] Mathias Bauer. Integrating probabilistic reasoning into plan recognition. In *Proceedings of the 11th European Conference on Artificial Intelligence*, pages 620–624, Amsterdam, Netherlands, 1994.

[BBD+93] M. Bauer, S. Biundo, D. Dengler, J. Koehler, and G. Paul. PHI — a logic-based tool for intelligent help systems. In R. Bajcsy, editor, *Proceedings of the 13th IJCAI*, pages 460–466, Chambery, France, 1993. Morgan Kaufmann.

[BP93] M. Bauer and G. Paul. Logic-based plan recognition for intelligent help systems. Research Report 93-43, German Research Center for Artificial Intelligence (DFKI), 1993.

[Car87] Sandra Carberry. Pragmatic modeling: Toward a robust natural language interface. *Computational Intelligence*, 3:117–136, 1987.

[Car90a] Sandra Carberry. Incorporating default inferences into plan recognition. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 471–478, Boston, 1990. AAAI.

[Car90b] Sandra Carberry. *Plan Recognition in Natural Language Dialogue*. ACL-MIT Press Series on Natural Language Processing. MIT Press, 1990.

[CG91] Eugene Charniak and Robert Goldman. A probabilistic model for plan recognition. In *Proc. 9th Conf. AAAI*, pages 160–165, Anaheim, CA, USA, 1991.

[CG93] Eugene Charniak and Robert P. Goldman. A Bayesian model of plan recognition. *Artificial Intelligence*, 64(1):53–79, 1993.

[Cla97] Herbert H. Clark. Dogmas of understanding. *Discourse Processes*, 23:567–598, 1997.

[Coh78] P. R. Cohen. On knowing what to say: Planning speech acts. Technical Report TR 118, Department of Computer Science, University of Toronto, Ontario, 1978. PhD Thesis.

[CP79] Philip R. Cohen and C. Raymond Perrault. Elements of a plan-based theory of speech acts. *Cognitive Science*, 3:177–212, 1979.

[CPA82] Philip R. Cohen, C. Raymond Perrault, and James F. Allen. Beyond question answering. In Wendy G. Lehnert and Martin H. Ringle, editors, *Strategies for Natural Language Processing*, chapter 9, pages 245–274. Lawrence Erlbaum Associates, 1982.

[CSH94] Robin Cohen, Bruce Spencer, and Pat Hoyt. Designing a tool to allow updates during plan recognition - challenges and applications. In *1994 IEEE International Conference on Tools with Artificial Intelligence*, pages 63–70, 1994.

[CSSvB91] Robin Cohen, Fei Song, Bruce Spencer, and Peter van Beek. Exploiting temporal and novel information from the user in plan recognition. *User Modeling and User-Adapted Interaction*, 1(2):125–148, 1991.

[CY91] Randall J. Calistri-Yeh. Utilizing user models to handle ambiguity and misconceptions in robust plan recognition. *User Modeling and User-Adapted Interaction*, 1(4):289–322, 1991.

[DE95] B. Di Eugenio. Plan Recognition and Natural Language Understanding. In M. Bauer, editor, *IJCAI 95 Workshop on The Next Generation of Plan Recognition Systems: Challenges for and Insight from Related Areas of AI (Working Notes)*, pages 42–47, Montreal, Canada, 1995.

[FA98] George Ferguson and James F. Allen. TRIPS: An intelligent integrated problem-solving assistant. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 567–573, Madison, WI, July 1998.

[FAMR96] George M. Ferguson, James F. Allen, Brad W. Miller, and Erik K. Ringger. The design and implementation of the TRAINS-96 system: A prototype mixed-initiative planning assistant. TRAINS Technical Note 96-5, University of Rochester, Department of Computer Science, October 1996.

[Fin83] Timonthy Finin. Providing help and advice in task oriented systems. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 176–178, Karlsruhe, West Germany, 1983.

[GK95] J. Greer and G.M. Koehn. The Peculiarities of Plan Recognition for Intelligent Tutoring Systems. In M. Bauer, editor, *IJCAI 95 Workshop on The Next*

Generation of Plan Recognition Systems: Challenges for and Insight from Related Areas of AI (Working Notes), pages 54–59, Montreal, Canada, 1995.

[Gol90] Robert P. Goldman. A probabilistic approach to language understanding. Technical Report CS-90-34, Department of Computer Science, Brown University, December 1990. PhD Thesis.

[Gri57] H. Paul Grice. Meaning. *Philosophical Review*, 66(3):377–388, 1957.

[Gri69] H. Paul Grice. Utterer's meaning and intentions. *Philosophical Review*, 78(2):147–177, 1969.

[Gri75] H. Paul Grice. Logic and conversation. In P. Cole and J. L. Morgan, editors, *Speech Acts*, volume 3 of *Syntax and Semantics*, pages 41–58. Academic Press, New York, 1975.

[Gro78] Barbara J. Grosz. Focusing in dialog. In *Theoretical Issues in Natural Language Processing-2*, pages 96–103, University of Illinois at Urbana-Champaign, Champaign, Illinois, July 25-27 1978.

[Gro81] Barbara J. Grosz. Focusing and description in natural language dialogues. In A. Joshi, B. Webber, and I. Sag, editors, *Elements of Discourse Understanding*, pages 84–105. Cambridge University Press, New York, New York, 1981.

[HA89] Elizabeth A. Hinkelman and James F. Allen. Two constraints on speech act ambiguity. In *Proceedings of the Association for Computational Linguistics (ACL)*, Vancouver, Canada, 1989.

[HDW94] Marcus J. Huber, Edmund H. Durfee, and Michael P. Wellman. The automated mapping of plans for plan recognition. In *UAI94 — Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 344–351, Seattle, Washington, 1994.

[HL82] Karen Huff and Victor Lesser. Knowledge-based command understanding: An example for the software development environment. Technical Report TR 82-6, University of Massachusetts at Amherst, 1982.

[HP99] Eric Horvitz and Tim Paek. A computational architecture for conversation. In *Proceedings of the Seventh International Conference on User Modeling, Banff Canada*, pages 201–210. Springer Wien, June 1999.

[HP00] Eric Horvitz and Tim Paek. DeepListener: Harnessing expected utility to guide clarification dialog in spoken language systems. In *Proceedings of the 6th International Conference on Spoken Language Processing*, Beijing, China, 2000.

[JKM+95] Susanne Jekat, Alexandra Klein, Elisabeth Maier, Ilona Maleck, Marion Mast, and J. Joachim Quantz. Dialogue acts in VERBMOBIL. Verbmobil Report 65, Universität Hamburg, DFKI Saarbrücken, Universität Erlangen, TU Berlin, April 1995.

[Joh95] W.L. Johnson. Plan Recognition in a Situational Context. In M. Bauer, editor, *IJCAI 95 Workshop on The Next Generation of Plan Recognition Systems: Challenges for and Insight from Related Areas of AI (Working Notes)*, pages 66–71, Montreal, Canada, 1995.

[KA86] Henry Kautz and James Allen. Generalized plan recognition. In *Proceedings of AAAI-86*, pages 32–37, Philadelphia, 1986.

[Kau87] Henry A. Kautz. A formal theory of plan recognition. Technical Report 215, University of Rochester, Department of Computer Science, 1987. PhD Thesis.

[Kau90] H. Kautz. A circumscriptive theory of plan recognition. In P. R. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communication*, pages 105–134. MIT Press, Cambridge, MA, 1990.

[Kau91] Henry Kautz. A formal theory of plan recognition and its implementation. In J. Allen, H. Kautz, R. Pelavin, and J. Tenenberg, editors, *Reasoning about Plans*, pages 69–125. Morgan Kaufman, San Mateo, CA, 1991.

[KGN94] Martin Kay, Jean Mark Gawron, and Peter Norvig. *Verbmobil – a Translation System for Face-to-Face Dialog*. Number 33 in CSLI Lecture Notes. Center for the Study of Language and Information, Stanford, CA, 1994.

[LA87] D. J. Litman and J. F. Allen. A plan recognition model for subdialogues in conversations. *Cognitive Science*, 11(2):163–200, 1987.

[LA90] Diane J. Litman and James F. Allen. Discourse processing and commonsense plans. In Philip R. Cohen, Jerry Morgan, and Martha E. Pollack, editors, *Intentions in Communication*, chapter 17, pages 365–388. MIT Press, 1990.

[LC91] Lynn Lambert and Sandra Carberry. A tripartite plan-based model of dialogue. In *Proceedings of the 29th ACL*, pages 47–54, Berkeley, CA, June 1991.

[LE95a] N. Lesh and O. Etzioni. Insights from Machine Learning for Plan Recognition. In M. Bauer, editor, *IJCAI 95 Workshop on The Next Generation of Plan Recognition Systems: Challenges for and Insight from Related Areas of AI (Working Notes)*, pages 78–83, Montreal, Canada, 1995.

[LE95b] Neal Lesh and Oren Etzioni. A sound and fast goal recognizer. In *IJCAI95 — Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1704–1710, Montreal, Canada, 1995.

[LE96] Neal Lesh and Oren Etzioni. Scaling up goal recognition. In *KR-96 — Principles of Knowledge Representation and Reasoning*, pages 178–189, 1996.

[Les97] Neal Lesh. Adaptive goal recognition. In *IJCAI '97*, 1997.

[Les98] Neal Lesh. *Scalable and Adaptive Goal Recognition*. PhD thesis, University of Washington, 1998.

[Lit85] Diane J. Litman. *Plan Recognition and Discourse Analysis: An Integrated Approach for Understanding Dialogues*. PhD thesis, University of Rochester, Department of Computer Science, 1985. also Technical Report TR170.

[Lit86] Diane J. Litman. Understanding plan ellipsis. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 619–624, Philadelphia, 1986.

[LRS99] Neal Lesh, Charles Rich, and Candace L. Sidner. Using plan recognition in human-computer collaboration. In *Proceedings of the Seventh International Conference on User Modeling*, Banff, Canada, July 1999.

[May89] James Mayfield. *Goal Analysis: Plan Recognition in Dialogue Systems*. PhD thesis, University of California at Berkeley, Computer Science Division (EECS), 1989. Technical Report UCB 89/521.

[May92] James Mayfield. Controlling inference in plan recognition. *User Modeling and User-Adapted Interaction*, 2(1-2):55–82, 1992.

[NMJ+99] Mikio Nakano, Noboru Miyazaki, Jun-ichi Hirasawa, Kohji Dohsaka, and Takeshi Kawabata. Understanding unsegmented user utterances in real-time spoken dialogue systems. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99)*, pages 200–207, 1999.

[PG95] F. Pachet and S. Giroux. Building plan recognition systems on arbitrary applications: the spying technique. In M. Bauer, editor, *IJCAI 95 Workshop on The Next Generation of Plan Recognition Systems: Challenges for and Insight from Related Areas of AI (Working Notes)*, pages 101–106, Montreal, Canada, 1995.

[PH00] Tim Paek and Eric Horvitz. Conversation as action under uncertainty. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI-2000)*, Stanford, CA, June 2000.

[PHR00] Tim Paek, Eric Horvitz, and Eric Ringger. Continuous listening for unconstrained spoken dialogue. In *Proceedings of the 6th International Conference on Spoken Language Processing*, Beijing, China, 2000.

[Pol86a] Martha E. Pollack. *Inferring Domain Plans in Question-Answering*. PhD thesis, University of Pennsylvania, 1986.

[Pol86b] Martha E. Pollack. A model of plan inference that distinguishes between the beliefs of actors and observers. In *Proceedings of the Twenty-Fourth Annual Meeting of the Association for Computational Linguistics*, pages 207–214, New York, June 1986.

[Pol87] Martha E. Pollack. Some requirements for a model of the plan-inference process in conversation. In Ronan Reilly, editor, *Communication Failure in Dialogue*, pages 245–256. North-Holland, Amsterdam, 1987.

[Pol90] Martha E. Pollack. Plans as complex mental attitudes. In P. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communication*. MIT Press, 1990.

[Qui89] Alexander Quilici. Detecting and responding to plan-oriented misconceptions. In Alfred Kobsa and Wolfgang Wahlster, editors, *User Models in Dialog Systems*, pages 108–132. Springer Verlag, Berlin, 1989.

[Ram89a] Lance A. Ramshaw. A metaplan model for problem-solving discourse. In *Proceedings of the Fourth Conference of the European Chapter of the Association for Computational Linguistics*, pages 35–42, Manchester, England, 1989.

[Ram89b] Lance A. Ramshaw. *Pragmatic Knowledge for Resolving Ill-Formedness*. PhD thesis, University of Delaware, Newark, Delaware, June 1989.

[Ram91] Lance A. Ramshaw. A three-level model for plan exploration. In *Proceedings of the 29th ACL*, pages 39–46, Berkeley, CA, June 1991.

[Rum75] David E. Rumelhart. Notes on schema for stories. In Daniel G. Bobrow and Allan Collins, editors, *Representation and Understanding: Studies on Cognitive Science*, Language, Thought, and Culture: Advances in the Study of Cognition, pages 211–236. Academic Press, New York, 1975.

[SA77] Roger Schank and Robert Abelson. *Scripts, Plans, Goals and Understanding*. Lawrence Erlbaum, Hillsdale, NJ, 1977.

[Sea70] John R. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, England, 1970.

[Sea75] John R. Searle. Indirect speech acts. In P. Cole and J. L. Morgan, editors, *Speech Acts*, volume 3 of *Syntax and Semantics*, pages 59–82. Academic Press, New York, 1975.

[SR81] Roger C. Schank and Christopher K. Riesbeck. *Inside Computer Understanding: Five Programs Plus Miniatures*. Lawrence Erlbaum, Hillsdale, NJ, 1981.

[SSG78] C. F. Schmidt, N. S. Sridharan, and J. L. Goodson. The plan recognition problem: An intersection of psychology and artificial intelligence. *Artificial Intelligence*, 11:45–83, 1978.

[Sto01] Scott C. Stoness. Continuous understanding: A first look at CAFE. University of Rochester, Department of Computer Science Area Paper: Draft, 2001.

[vBC91] Peter van Beek and Robin Cohen. Resolving plan ambiguity for cooperative response generation. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 938–944, 1991.

[VW01] Michael Van Wie. *Role Selection in Teams of Non-Communicating Agents*. PhD thesis, University of Rochester, Department of Computer Science, 2001.

[WAC84] Robert Wilensky, Yigal Arens, and David Chin. Talking to UNIX in English: An overview of UC. *Communications of the ACM*, 27(6):574–593, 1984.

[WCL+89] Robert Wilensky, David N. Chin, Marc Luria, James Martin, James Mayfield, and Dekai Wu. The Berkeley UNIX consultant project. Technical Report UCB/CSD 89/520, University of California at Berkeley, Computer Science Division (EECS), 1989.

[Wil78] Robert Wilensky. *Understanding Goal-Based Stories*. PhD thesis, Yale University, Computer Science Department, 1978. Research Report 140.

[Wil82] Robert Wilensky. Talking to UNIX in English: An overview of an on-line consultant. Technical Report UCB/CSD 82/104, University of California at Berkeley, Computer Science Division (EECS), Berkeley, California, September 1982.

[Wil83] Robert Wilensky. *Planning and Understanding: A Computational Approach to Human Reasoning*. Addison-Wesley, Reading, Massachusetts, 1983.