Cooperative Plan Reasoning for Dialogue Systems*

George Ferguson and James F. Allen
University of Rochester
Rochester, NY, 14627-0226
{ferguson,james}@cs.rochester.edu

1 Introduction

For several years, we have been studying collaborative, dialogue-based interaction between humans and between humans and a computer system as part of the TRAINS project [Allen and Schubert, 1991]. The goal of the project is an intelligent planning assistant that is conversationally proficient in natural language and that will assist a human manager (the user) formulate and execute plans. The domain of discourse is a transportation world that includes trains, cities, connections between them, warehouses and factories, and different types of commodities, all of which are part of a simulated world in which the mutually-decided-upon plans are executed. Work in the TRAINS project consists of developing both formal models of the various tasks involved in collaborative dialogue and programs that implement these theories. Both will be described in this report.

This work is of direct relevance to all three stated areas of interest for the symposium:

1. Sharing responsibility between person and computer:

   Since the interaction between the user and the system is in natural language, issues of coordination are extremely complicated. The system is considered to have good knowledge of how to achieve things in the world, but only partial knowledge of the actual state of the world. The user has goals that must be communicated to the system, which must recognize these and recognize how other utterances contribute to the user's intended plan.

   To determine what aspects of communication the system needs to understand to cooperate effectively, we have gathered a large number of dialogues with another person playing the role of the system. The only shared knowledge is a map of the initial scenario. These dialogues are transcribed and annotated to support research into prosody, discourse, and plan reasoning.

2. Managing the person-computer interaction:

   Clearly, the goal of the TRAINS project is "natural communication," using natural language. We are developing a broad coverage parsing system for a wide range of linguistic phenomena arising in dialogue, discourse management techniques for realistic cooperative dialogue including corrections and repairs, and a knowledge representation and reasoning system for both plans and world knowledge that is capable of handling the complexities of natural language.

   The TRAINS world and the task-oriented dialogues the system participates in are clearly amenable to certain forms of iconic display, such as maps. We are developing graphical tools for displaying such data, and in future work we intend to use iconic interaction in addition to natural language input.

3. Clarifying assumptions concerning the cognitive capabilities and knowledge requirements required for effective communication:

   Since the TRAINS system is intended to be an intelligent planning assistant, it needs to have knowledge about the world and about how actions that it can have TRAINS agents perform can change the world. All of this is expressed using a natural and powerful interval-based temporal logic. Such a logic is not only necessary for capturing the content of natural language utterances, it is also necessary for reasoning about action in complex, realistic domains.

   Furthermore, the system must explicitly represent the plan(s) under consideration, since the user's utterances must be recognized in terms of their contribution to the current plan. In our framework, plans include explicit statements of what the goals of the plan are and, importantly, the assumptions underlying the plan. Assumptions can arise both during plan recognition (to fill in details left unspecified by the user but necessary to connect the current utterance to the plan) and during planning (for example, persistence assumptions). Since these assumptions often drive the dialogue (for example, there can be a choice of assumptions, or an assumption can be inconsistent), we are developing an explicit representation of plans as arguments based on assumptions.

   The TRAINS plan reasoner supports interleaved planning and recognition (as necessary in processing dialogue), and exports a variety of functions to other modules of the system, in particular the dialogue manager.

This position paper will elaborate on the last of these points, in particular on the forms of knowledge and reasoning about plans, actions, and the world required to support dialogue in the TRAINS system. We will begin with a brief overview of the TRAINS system architecture. We will then focus on the theoretical and practical issues surrounding the development of the TRAINS plan reasoner, the module responsible for reasoning about plans and actions once the user's utterances have been interpreted by

*This work is supported by ONR/ARPA research grant no. N00014-92J-15121 and Air Force – Rome Air Development Center research contract no. F30602-91-C-0010.

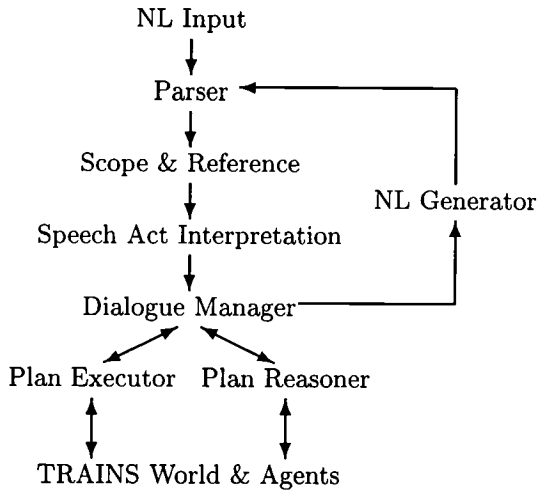

Figure 1: TRAINS System Architecture

the natural language modules. We will briefly describe the interval temporal logic and our motivation for using it, and compare other representations of time and action. We will then describe our representation of plans as graph-like arguments, and describe the types of reasoning that are necessary for cooperative dialogue systems. Examples from the dialogues we are working with will be presented throughout.

2 The TRAINS System

The TRAINS system architecture, shown in Figure 1, is that of a semi-hierarchical arrangement of modules, each of which corresponds to a "step" in the processing of the natural language input from the user. At the top, the input (currently typed, although we are working on prosodic features and speech input) is parsed by a GPSG-style parser to produce an indexical logical form in Episodic Logic [Hwang and Schubert, 1992]. This is processed by the scope and reference module to assign scope to tense and other operators and to resolve definite references [Poesio, 1993b; Poesio, 1993a].

The resulting deindexed logical form is then given to the dialogue manager [Traum, 1993] which produces a set of possible interpretations of the utterance as speech acts. The system determines turntaking, grounding, and argumentation acts as well as core speech acts using Conversation Act Analysis [Traum and Hinkelman, 1992]. These speech acts, such as `INFORM`, `SUGGEST`, `ASK-IF`, *etc.*, are represented using a language based on interval temporal logic [Heeman, 1993]. The dialogue manager prunes some speech act interpretations on the basis of its current knowledge.

The dialogue manager maintains a hierarchy of belief contexts that represent the system's notion of the state of the discourse. These include, for example, `HPROP`, for items proposed by the user but not yet accepted by the system, `SPROP` for the system's proposals, `SPLAN` for the private (unproposed) beliefs of the system, and `SHARED` for the mutally agreed upon beliefs. Different speech acts have different effects on these belief contexts, as do actions that the dialogue manager can instigate (*e.g.*, generating an utterance or invoking the plan reasoner). The dialogue manager maintains the obligations of the system, for example to answer questions or achieve goals, and updates them in light of the current speech acts.

At the "bottom" of the system, the plan reasoner (described more fully in what follows) provides planning and plan recognition services and performs reasoning about the state of the world. The dialogue manager uses the results of the plan reasoner to disambiguate speech act interpretations, update beliefs, and generate new conversational elements (for example, an ambiguity detected by the plan reasoner could cause a clarification sub-dialogue to be started). Once a mutually agreed upon plan is constructed, it is dispatched for execution and monitoring by the TRAINS agents in a simulated TRAINS world [Martin, 1993].

The following simple dialogue is an example of a conversation that the implemented system handles. It was distilled from a transcript of a conversation between two humans.

M: We have to make OJ.
M: There are oranges at Avon and an OJ factory at Bath.
M: Engine E3 is scheduled to arrive at Avon at 3pm.
M: Shall we ship the oranges?
S: Ok.
S: Shall I start loading the oranges into the empty car at Avon?
M: Ok.

A more elaborate dialogue of the sort motivating our current research is given in Appendix A, and others we have collected are available in annotated form in [Gross *et al.*, 1993].

3 Event-based Interval Temporal Logic

The system's knowledge (world knowledge, planning knowledge, and knowledge of the discourse state) is represented using an expressive, temporally-explicit logic. This section describes the language and its use in representing action. Details are necessarily brief but references to more in-depth presentations are given.

Interval temporal logic was initially developed for representing the content of natural language [Allen, 1983]. The twelve interval relations shown in Figure 2 correspond to several common locutions for temporally-qualified statements, and together with equality they can represent all possible relationships between two intervals. When there is a correspondence between point-based and interval temporal logics, certain common relations (in particular *Disjoint*, a disjunction of primitive relations) are more simply expressed in the interval representation. On the other hand, the addition of metric temporal information (*i.e.*, durations) is more difficult in the interval logic. In any case, it turns out that the interval relations are also appropriate for representing the knowledge required for planning and plan recognition [Allen, 1991; Allen and Ferguson, 1993a; Allen and Ferguson, 1993b]. For example, the main feature of non-linear planning is to allow two actions to be non-overlapping but unordered, that is, *Disjoint*.

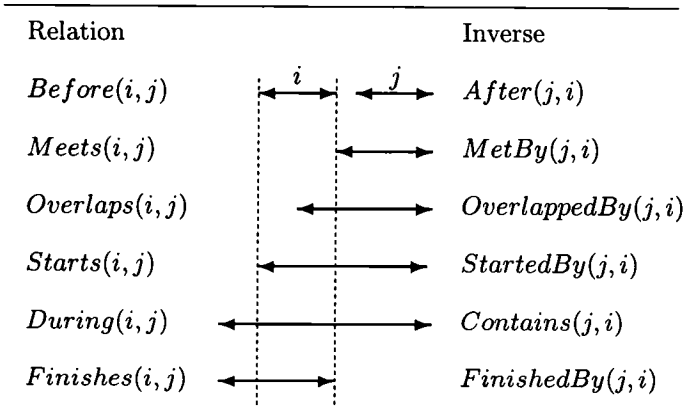

Figure 2: Interval temporal logic relations

In order to reason about change in the world and how changes can be brought about, we introduce events and actions into the ontology. Semantically, events are neutral objects except that for any event there is a function *time* which denotes the interval over which the event occurs. Events are organized into a hierarchy of event types, where different types define different role functions—functions that, given an event, denote objects playing certain roles in the events. There can be both temporal (*i.e.*, interval-denoting) and non-temporal (*i.e.*, object-denoting) roles. Each event type has a predicate which is true of an event if the event is of that type. For example, $Load(e)$ is true if the event e is a loading event. These predicates can be extended to accept additional parameters representing roles of the event.

We represent knowledge about events in several ways. The first is by defining necessary conditions on the event occurring, which we call **EDEF** axioms. These axioms state what must be (or have been) the case if an event of a particular type occurred. They do not capture any sense of causality. In particular, they do not state what properties are caused by an action and what properties simply must be true whenever the action succeeds. This is the distinction that state-based planners like STRIPS makes between preconditions and effects. We view *actions* as sensory-motor programs that can be run by agents. There is no *a priori* connection between attempting to execute a program (perform an action) and it having certain effects. Rather, these effects are specified axiomatically, using what we call **ETRY** axioms. Finally, for completeness, there are also event generation axioms (**EGEN**) that state that, under certain conditions, the occurrence of an event of one type entails the occurrence of an event of another type.

An important aspect of our approach is the separation of knowledge about what must be true given that an event of some type occurs (EDEF) from the knowledge about how to bring about the occurrence of events (ETRY, EGEN), a separation that is essential if we are to be able to reason about action attempts and failures. This has sometimes been referred to as the difference between "knowing how it works" and "knowing how to work it," two very different sorts of knowledge. In many situations, an agent may know the former but not the latter. For example, we can all recognize that the mechanic fixed our car, even if we have no idea what enabled the mechanic to do the job. This might be expressed by an EDEF axiom stating that fixing something means that the thing was once broken and is now working. With this information, the agent knows what it means for the car to be fixed although it does not know how it was done (or how to do it). Furthermore, such axioms could be used by a system to recognize situations in which the car has been fixed without having any knowledge of how it was done. Knowledge of this sort is also essential for much of natural language semantics, where many verbs are defined and used without the agent's knowing the necessary causal knowledge. This is discussed at length in [Allen, 1984].

Event-based interval temporal logic is a powerful logic for representing and reasoning about action and change. In comparison to state-based approaches like the situation calculus, it can directly express the content of natural language utterances. Furthermore, it can express statements about the future without referring to the sequence of actions that leads to the future state. To the extent that state-based approaches can be made to do this (for example, by quantifying over situations), they become more and more like temporal logic. Event-based temporal logic also allows natural reasoning about external events, either due to the actions of other agents or due to natural forces in the world. It can reason about such events even in the face of uncertainty, unlike certain approaches based on minimization of "miracles". Finally, a crucial issue in reasoning about actions is the ability to deal with simultaneous and interfering events and actions. The event-based interval logic allows us to deal with several cases of simultaneous action directly: when two actions cannot occur simultaneously, when they occur simultaneously and are independent, and when they have additional effects that neither would have individually. In the difficult case when two actions partially or conditionally interfere, there are several possibilities offered by the expressiveness of the logic.

4 Representing Plans

The way people talk about plans in cooperative dialogue offers several challenges for representing plans and reasoning about them that are not encountered in more restrictive settings. First, the representation needs to be able to represent incorrect or sub-optimal plans in order to correctly express the content of natural language utterances about plans. The system should then be able to reason about why the plans are incorrect or sub-optimal and then use the results of such reasoning to drive the dialogue towards repairing or refining the plan. Second, the plans under discussion are rarely completely specified, nor are they necessarily described in a systematic way. Our experience in collecting TRAINS dialogues shows that people jump around to various aspects of plan that they consider salient. For example, a goal of moving some oranges some might be introduced, followed immediately by a discussion of which engine to use. The problem for the system attempting to connect the second statement to the plan introduced by the first statement is that, as far as its knowledge about events and actions is concerned, there are several steps or levels of abstraction between moving the

oranges and moving the engine. Third, in a dialogue setting, plan recognition and planning are interleaved. Plan recognition is performed to determine the content of the user's utterances, and planning is used to fill out details of the plan or to criticize previously-specified details. Both of these operations need to be performed in the context of different sets of beliefs and assumptions. These requirements mean that the plan reasoner must be able to reason from an arbitrary plan description in an an arbitrary knowledge context. Finally, as the previous example illustrates, the system must make assumptions both during plan recognition (*i.e.*, assume parts of the plan necessary to connect the utterance to the plan) and during planning (*e.g.*, traditional assumptions like persistences). These must be made explicitly rather than implicitly since they can figure in the conversation.

Based on these considerations, we are developing a representation of plans based on the intuition that a plan is an *argument* that a certain course of action under certain explicit conditions will achieve certain explicit goals. Plans include both the goals that they purport to achieve and the assumptions that underlie them. Plans can be analyzed and compared in terms of both their internal structure and the relative merit of their underlying assumptions. A plan for some goal can be good or bad, definite or impossible, depending on our knowledge about the world.

To formalize plans-as-arguments, we treat them like proofs, where the goals of the plan are the conclusion of the proof, and the proof branches out in a graph-like manner below the conclusion, terminating with the premises at the sources of the graph. A *plan graph* then is a connected, directed, acyclic graph consisting of three types of nodes labelled by formulas:

- *Event* nodes are labelled with $E(e)$ for some event type predicate E and event term e.
- *Action* nodes are labelled with $Try(\pi, t, e)$ for some action term π, temporal term t, and event term e.
- *Fact* nodes are labelled with any other formula.

The premises of a plan graph are its sources, the goals are its sinks. Since plan graphs are seen as arguments and not necessarily causal chains, we can have not just goals of achievement, but also of prevention and maintenance. The requirement that goals be just the sinks and all the sinks makes goals be "ends in themselves," which is intuitively plausible, and separates goals from "incidental effects" by effectively disallowing the latter. Of course we can still reason about such effects, they just aren't part of the plan proper. (By analogy with proofs, they would be corollaries.)

The internal structure of a plan graph can be evaluated relative to a knowledge base that includes knowledge about the preconditions and effects of actions such as is provided in the event-based temporal logic. The assumptions underlying a plan can be evaluated relative to our knowledge about the world, for example to verify that they are consistent with what we know. The user's plan might in fact rest on an assumption that we know to be false, in which case it ought to the subject of further conversation. The assumptions can also be evaluated relative to each other, for example preferring the plan based on the weakest set of assumptions or preferring the plan that assumed the tightest bounds on certain resources of interest.

5 Plan Reasoning for Dialogue Systems

We characterize plan reasoning for dialogue systems as search through a space of plan graphs. The termination criterion for the search depends on the type of reasoning being done, as will be described presently. Since the plan graph formalism sanctions arbitrarily complex graphs labelled with arbitrarily complex formulas, searching all possible plan graphs is impossible. We therefore rely on additional properties of the underlying representation to restrict the search.

First, we assume the ability to test whether two objects (including events and facts) unify and, optionally, to determine assumptions under which they would unify. Simple objects use simple equality. In the temporal logic, two events are equal if their roles are equal. Two facts unify if there are assumptions that make them logically equivalent. This use of equality and inequality corresponds to the posting of *codesignation* constraints in traditional planners.

Second, we assume that events are defined using relations corresponding to *enablers*, *effects*, and *generators*. This should not be controversial. In the temporal logic, these descriptions can be obtained from the event definition axioms (ETRY, EDEF, EGEN). For a STRIPS system, they correspond to the add- and delete-lists. Existing plan recognition systems use an event taxonomy, which corresponds to the generators slot. There can be multiple definitions of an event type, thereby allowing alternative decompositions or conditional effects.

The search then only considers plan graphs that reflect the structure of the event definitions, we call such plan graphs *acceptable*. In this respect, the search will only find plan graphs that agree with the assumed-shared "event library." However, information returned from failed searches can be used to guide the repair of apparent incompatibilities at the discourse level.

The TRAINS plan reasoner searches a plan graph described declaratively in a belief context. It performs both plan recognition ("incorporation") and planning ("elaboration") using a variety of interfaces corresponding to certain common locutions that we have observed in our dialogs. The search is breadth-first, thereby imposing a "shortest-path" heuristic—we prefer the simplest connection to the existing plan. Rather than dwell on the details, we will simply present each of the procedures together with a motivating example. More details are given in [Ferguson and Allen, 1993].

The procedure `incorp-event` takes as parameters a plan graph and an event (a term or a lambda expression representing an event type). For example, the sentence:

(1) Send engine E3 to Dansville.

results in the following call:

```
(incorp-event
  (lambda ?e*Move-Engine
          (And (Eq (eng ?e) ENG3)
               (Eq (dst ?e) DANSVILLE)))
  THE-PLAN)
```

where `?e*Move-Engine` is an event variable of type `Move-Engine`. That is, the plan should contain an instance of a `Move-Engine` event where the engine being moved is `ENG3` and the destination of the moving is `DANSVILLE`.

The procedure `incorp-role-filler` is used for statements that mention objects to be used in the plan, such as:

(2) There's an OJ factory at Dansville.

Once the dialogue manager determines that this represents a request to use the OJ factory in the current plan, it makes the following call:

```
(incorp-role-filler
  (lambda ?x*OJ-Factory (At ?x DANSVILLE NOW))
  THE-PLAN)
```

In this example, the lambda-expression arises from the natural language "an OJ factory."

Finally, there is the procedure `incorp-fact` that searches for a fact node that would unify with the given one. This is used for utterances like:

(3) We must get the oranges there by 3 PM.

(4) The car will be there because is it attached to engine E1.

since the plan graph representation supports inferential (fact-fact) links. Again however, the search space of potential unifying formulas is infinite. We therefore only consider certain candidates, based on syntactic considerations. These include facts that the underlying reasoning system is particularly good at, such as temporal constraints or location reasoning.

There are also ways to specify the incorporation of a goal, and to guide the search using information contained in such natural language locutions as purpose clauses (*e.g.*, "Use the engine to pick up the boxcar") and relative clauses (*e.g.*, "Send it to Avon, where there are oranges"). To summarize, the state of the plan after utterance (5) of the example dialog is shown in Figure 3. The `Move-Oranges` event `e7` (an abstraction of `e5`) has just been added as a result of interpreting the question as also suggesting that a `Move-Oranges` event be in the plan.

The `elaborate` procedure is typically invoked by the dialogue manager when the system gets the turn, in order to detect problems with the current plan and add missing items that might then result in system utterances. This procedure performs fairly traditional means-ends planning to attempt to flesh out the plan. In so doing, it attempts to satisfy or assume preconditions and bind roles to objects in order to generate a supported plan. It freely makes consistent persistence assumptions by assuming inclusion of one unconstrained temporal interval within another known one. It can ignore some details of the plan, for example the exact route an engine should take. These can be reasoned about if necessary (*i.e.*, if the user mentions them) but can be left up to the executor otherwise. Again, ambiguities are reported to the dialogue manager together with whatever could be unambiguously inferred. The ambiguity is typically used to invoke a clarification sub-dialog or make a suggestion. In the sample dialogue given previosuly, the plan as it stands when the system gets the turn is ambiguous as to exactly which boxcar is to be used to transport the oranges. The system generates the utterance

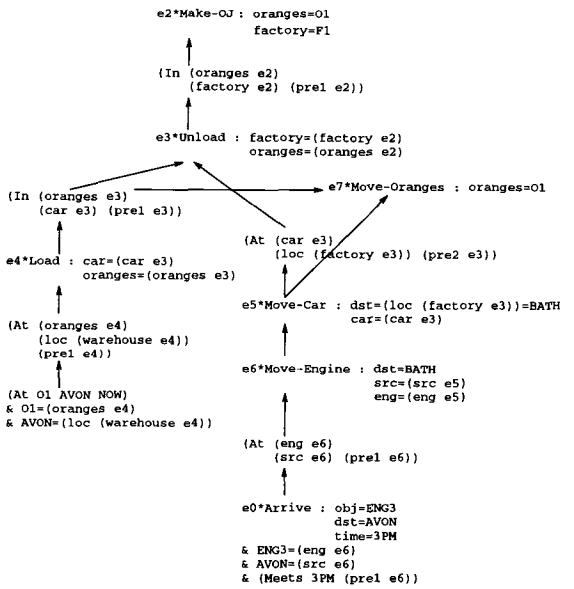

Figure 3: Example dialog plan graph

(5) Shall I start loading the oranges into the empty car at Avon?

after choosing one of the alternatives.

6 Conclusions

The TRAINS project is an ideal testbed for both theoretical and practical aspects of human-computer collaboration. The complexity of unrestricted natural language discourse about plans and actions requires expressive representations both for reasoning about the discourse and for reasoning about the world under discussion. Sample dialogs between humans illustrate the wide range of phenomena that an intelligent, cooperative system must handle.

We have developed and are continuing to develop representations of time, action, and plans that are adequate for such interaction. We acknowledge the fundamental role of assumptions in such behaviour and are developing formalisms that allow reasoning about such assumptions to be made explicit. These formal theories are not ends in themselves. Rather, they serve to define the proper behaviour of the programs that we develop as part of the TRAINS project. Use of the programs, in turn, uncovers issues that force refinement of the formalisms. Together, we hope to converge towards the specification and construction of a comprehensive, intelligent, cooperative natural language-based assistant in a complex, realistic world.

References

[Allen and Ferguson, 1993a] James F. Allen and George Ferguson. Action in interval temporal logic. In *Working notes of the Second Symposium on Logical Formalizations of Commonsense Reasoning*, pages 12–22, Austin, TX, 11–13 January 1993.

[Allen and Ferguson, 1993b] James F. Allen and George Ferguson. Actions and events in interval temporal logic. Submitted for publication, 1993.

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

[Allen, 1983] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[Allen, 1984] James F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.

[Allen, 1991] James F. Allen. Temporal reasoning and planning. In *Reasoning about Plans*, pages 1–68. Morgan Kaufmann, San Mateo, CA, 1991.

[Ferguson and Allen, 1993] George Ferguson and James F. Allen. Generic plan recognition for dialogue systems. In *Proc. ARPA Workshop on Human Language Technology*, Princeton, NJ, 21–23 March 1993.

[Gross *et al.*, 1993] Derek Gross, James F. Allen, and David R. Traum. The TRAINS-91 dialogues. TRAINS Technical Note 92-1, Dept. of Computer Science, University of Rochester, Rochester, NY, June 1993.

[Heeman, 1993] Peter A. Heeman. Speech actions and mental states in task-oriented dialogues. In *Working Notes of the AAAI Spring Symposium on Reasoning about Mental States*, Stanford, CA, March 1993.

[Hwang and Schubert, 1992] Chung Hee Hwang and Lenhart K. Schubert. Episodic logic: A situational logic for natural language processing. In *Proc. Third Conference on Situational Theory and its Applications*, Oiso, Japan, 18–21 November 1992. To appear in *Situation Theory and its Applications, vol. 3*, P. Aczel, D. Israel, Y. Katagiri, and S. Peters (eds.), CSLI, Stanford, CA, 1993.

[Martin, 1993] Nathaniel G. Martin. *Applying Statistical Inference to Planning under Uncertainty.* PhD thesis, Dept. of Computer Science, University of Rochester, Rochester, NY, 1993.

[Poesio, 1993a] Massimo Poesio. Assigning a semantic scope to operators. In *Proc. ACL-93*, Columbus, OH, June 1993.

[Poesio, 1993b] Massimo Poesio. A situation-theoretic formalization of definite description interpretation in plan elaboration dialogues. In P. Aczel, D. Israel, Y. Katagiri, and S. Peters, editors, *Situations Theory and its Applications, vol.3*, chapter 12, pages 343–378. CSLI, Stanford, 1993. To appear.

[Traum and Hinkelman, 1992] David R. Traum and Elizabeth A. Hinkelman. Conversation acts in task-oriented spoken dialogue. *Computational Intelligence*, 8(3):575–599, 1992. Also available as University of Rochester TR 425.

[Traum, 1993] David R. Traum. Mental state in the TRAINS-92 dialogue manager. In *Working Notes of the AAAI Spring Symposium on Reasoning about Mental States*, pages 143–149, Stanford, CA, March 1993.

A Sample Dialogue

The following dialogue is a verbatim transcript of a conversation between two humans, one playing the role of the manager, the other playing the system. We have edited out some mid-sentence corrections and several non-verbal "ums" and "ahs".

M: We have to ship a boxcar of oranges to Bath by 8 AM and it is now midnight.
S: Okay.
M: Okay. Um... all right. So, there are two boxcars at ... Bath and one at ... Dansville and ... [4sec]
S: And there's ...
M: [interrupts] Wait. I've forgotten where the oranges are. Where are the oranges?
S: The oranges are in the warehouse at Corning.
M: Okay. So we need to get a boxcar to ... Corning.
S: Right.
M: All right. So why don't we take one of the ones from Bath?
S: Okay.
M: So ...
S: [interrupts] We need ... Okay, which engine do you want to bring it?
M: Oh. How about ... Well, Let's see. What's shorter the distance between Avon and Bath or Elmira? It looks like it's shorter from Elmira to Corning, so why don't you send E2 to Corning?
S: Okay. [4sec]
M: In fact ... What did I say? Did I say ... Did I send a boxcar from Bath to Corning? It also looks like it's shorter from Dansville to Corning, so why don't you send that boxcar to ... Corning ...
S: [interrupts] Okay, with ...
M: [interrupts] instead ...
S: [interrupts] with engine E2.
M: Yes.
S: Okay, so you want ...
M: [interrupts] Oh wait. Okay, I ... All right, I misunderstood. So you have to have an engine in order to move a boxcar, right?
S: Right. [3sec]
M: What's ... Is it shorter to move an engine from Elmira? [3sec] Can you figure things out for me? Can you figure out what's the shortest? [3sec] What would be faster: to send an engine from Elmira to ... one of the boxcars or from Avon?
S: Well there's a boxcar already at Elmira [3sec] and you mean to go to Corning.
M: Yes. All right. I didn't see that boxcar. Okay. Great. Send E2 with the boxcar from Elmira to Corning.
S: Okay. [2sec] Okay. We can do that.
M: All right. And [3sec] once it ... gets there [2sec] What am I allowed to do? Okay. I can make a whole plan at this point. Right?
S: Yeah.
M: Okay. Once it gets there ... have it fill up with oranges and go straight to Bath. [2sec]
S: Okay. Then we get to Bath at ... 5 AM.
M: Great.
S: So we're done.
M: Okay.