Mental State in the TRAINS-92 Dialogue Manager[*]

David R. Traum
Computer Science Department
University of Rochester
Rochester, New York 14627-0226 USA
traum@cs.rochester.edu

Abstract

This paper describes the dialogue manager of the TRAINS-92 system. The general dialogue model is described with emphasis on the mentalistic attitudes represented.

1 Introduction: The TRAINS Conversation System

The TRAINS project involves research in a wide range of areas of natural language and knowledge representation, including natural (spoken and written) language understanding, dialogue and discourse modelling, and real-time plan-guided execution. A centerpiece of this research is the design and implementation of a system which serves as a planning assistant to a human user, communicating with the user in an English conversation, with the duty of sending execution directives to agents in the simulated TRAINS world to achieve the goal of the user. More detailed information on the design and scope of the system can be found in [Allen and Schubert, 1991].

1.1 The Dialogue Manager

The dialogue manager is responsible for maintaining the flow of conversation, and making sure that the conversational goals are met. For this system, the main goals are that an executable plan which meets the user's goals is constructed and agreed upon by both the system and the user and then that the plan is executed.

The dialogue manager must keep track of the user's current understanding of the state of the dialogue, determine the intentions behind utterances of the user, generate utterances back, and send commands to the domain plan reasoner and domain plan executor when appropriate.

1.2 Other Related Modules

The **Natural Language Subsystem** is responsible for producing a semantic interpretation of an input utterance. This interpretation is a formula in Episodic Logic [Schubert and Hwang, 1989]. Parsing, semantic interpretation, and reference resolution and inference are steps in the interpretation process. The user's processed utterances are translated into a knowledge representation language built in RHET [Allen and Miller, 1989] and passed over to the dialogue manager for speech act analysis and further action, if necessary. The **NL Generator** takes speech acts produced by the dialogue manager and converts them to natural language utterances which are then output to the user (and fed back to the NL subsystem). The **Domain Plan Reasoner** does plan recognition and plan elaboration. Plan recognition is a crucial part of understanding utterances of the user, and plan elaboration produces the new information that the Dialogue Manager will then have to communicate to the user. The **Plan Executor** takes a plan and sends the necessary commands to the individual agents (engineers, factory and warehouse attendants) to have that plan carried out in the simulated world. It also monitors the progress of the plan (by remaining in communication with these agents as they attempt to perform their tasks) to make sure the plan execution is successful.

2 Elements of Conversational State

2.1 Beliefs

Several modalities of domain belief must be tracked in order to fulfill the conversational purposes. Private beliefs must be maintained in

[*]Presented at AAAI Spring Symposium on Reasoning about Mental States: Formal Theories & Applications, March 1993

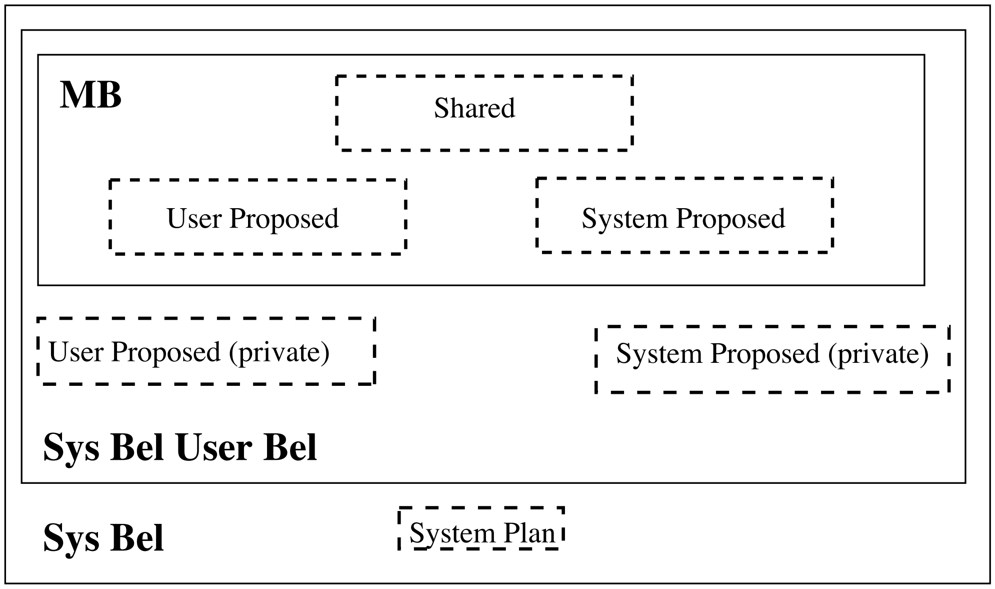

Figure 1: TRAINS-92 Domain Plan Contexts

order to do accurate plan reasoning. In addition, the system must maintain beliefs about the user's beliefs and about mutual beliefs both to interpret user utterances and formulate its own expressions coherently. These modalities are represented in the TRAINS system as belief spaces, maintained by the RHET KR system [Allen and Miller, 1989].

2.2 Domain Plans

From the point of view of the Dialogue Manager, Domain Plans are abstract entities which contain a number of parts. These include: the goals of the plan, the actions which are to be performed in executing the plan, objects used in the plan, and constraints on the execution of the plan. The composition of plans is negotiated by the conversational participants in order to come up with an agreement on an executable plan, which can then be carried out. Seen this way, the conversational participants can have different ideas about the composition of a particular plan, even though they are both talking about the "same" plan. The structure of TRAINS-92 domain plans and view of plans as *arguments* are described in detail in [Ferguson, 1992].

In order to keep track of the negotiation of the composition of a plan during a conversation, a number of plan contexts are used. These are shown in the dashed boxes in Figure 1, along with their related belief modalities. The solid boxes represent the belief modalities of (the system's belief about) mutual belief, the system's belief about the user's beliefs and the system's private beliefs. Each of these contains the former. Some of these beliefs will be about domain plans. The system's private beliefs about a plan is kept in the **System Plan** context. Items (including the goal of the plan, constituent action and event types, and other constraints on the plan) which have been suggested by the system but not yet acknowledged or accepted by the user are in the **System Proposed** (private) context. Similarly, items which have been suggested by the user but not accepted by the system are in the **User Proposed** (private) context. Items which have been proposed by one party and acknowledged but not accepted by another are in the appropriate Proposed-MB context. Items which have been proposed by one party and accepted by the other are in the **Shared** context.

There is no **User Plan** context, because the system has no direct knowledge of and does not represent the private non-communicated plan reasoning of the user. Spaces inherit from the spaces shown above them in the diagram. That is, everything in **Shared** will be in both **System Proposed-mb**, **User Proposed-mb**, and **System Plan**. In addition, everything in **User Proposed-mb** will be in **User Proposed**, and everything in **System Proposed-mb** will be in **System Proposed**. **System Plan** does not inherit from **System Proposed**, because the system may have erroneously communicated some portion of it's private plan (which would then lead the system to try to perform a *repair*).

In Pollack's terms, the constituents of the plan spaces will be (partial) *recipes for action* [Pollack, 1990]. The mental state of *having a plan* is not achieved until the shared plan is fully formed and the system adopts the intention to execute it.

2.3 Discourse Goals

The system maintains a set of **Discourse Goals** in order to use the conversation to satisfy its own goals. The over-riding goal for the TRAINS domain is to work out and perform an executable plan that is shared between the two participants. This leads to other goals such as accepting suggestions that the other agent has suggested, performing domain plan synthesis, or proposing plans to the other agent that the domain plan reasoner has constructed. Another top level goal is to adhere to conventions of conversational structure.

2.4 Action

Actions are represented in the reasoning system as semantic primitives. Of particular relevance to the dialogue manager are speech actions. These are the primary means by which the dialogue manager changes the conversational state and recognizes such changes. We are using the multi-level **Conversation Acts** model [Traum and Allen, 1991; Traum and Hinkelman, 1992] of action in conversation. In addition to traditional, propositional level **core speech acts** (e.g. suggest, inform, request), we also have levels of **turn-taking acts**, **grounding acts** which coordinate the aquisition of mutual belief, and **argumentation acts**, which coordinate the higher-level coherency in conversation. Associated with each action type are conditions describing their occurrence and effects on the conversational state.

2.5 Obligations

Obligations represent what an agent *should* do, according to external norms. They are different in kind from goals or intentions, though a well-behaved agent will choose to meet its obligations, and in fact there is generally a social cost for not meeting obligations which will encourage a purely strategic agent to endeavor to meet them. In a conversational setting, an accepted offer or a promise will incur an obligation. Also a request or command by the other party will bring an obligation to perform or address the requested action. If these requests are that the system say something (as in an acknowledgement request) or to inform (as in a question), then a **discourse obligation** is incurred.

We use these discourse obligations to efficiently and naturally represent the connection between a question and its answer. This can be contrasted to previous systems (e.g. [Allen and Perrault, 1980]), which go through an elaborate plan reasoning procedure starting from the fact that the question being asked means that the speaker wants to know something, which should then cause the system to adopt a goal to answer. In the current system, meeting the request is registered directly as an obligation, regardless of the intent of the questioner or the other goal structure of the system. If the system didn't address the obligation, then it would have to deal with the usual social problems of obligations which have not been met. This helps distinguish responses expected by convention (e.g. that a question be answered) from simple cooperative behavior (e.g. doing what one believes another agent wants). Other parts of the system might also bring about discourse obligations. For example, in some circumstances if the execution of the plan goes wrong, this would bring an obligation to inform the user. Discourse Obligations are tracked so that the obligations can be discharged appropriately. The current system does not do any complex deontic reasoning, merely choosing to meet its obligations as quickly as feasible.

2.6 Intentions

As the system discharges it's obligations and meets it's discourse goals, it constructs a set of **Intended Speech Acts**, actions it will perform given an opportunity. Because the system attempts to adhere to conventional interactional patterns, it does not always perform these right away, and may not get a chance to perform some of them. For example a suggestion may get preempted by a similar or conflicting suggestion by the user, an answer to a question may become irrelevant if the user retracts it).

3 Components of the Dialogue Manager

The TRAINS dialogue model includes the following major components: the speech act analyzer, the discourse context which contains elements of the mental and conversational state, and the discourse actor which decides what the system should do next, based on the discourse context.

3.1 The Speech Act Analyzer

The **Speech Act Analyzer** takes semantic interpretations of utterances and recognizes which acts have been performed by the speaker in making the utterance. Currently only the Core Speech act level has been fully implemented, with partial implementations of the other three levels. The method of analysis is based on the one proposed by [Hinkelman, 1990], using linguistic information as a filter to decide which acts are possible interpretations, and using plan based and contextual information to choose the most likely among the possible interpretations. One extension is that a one-to-one correspondence between utterances and speech acts is not assumed: some utterances will contain multiple act interpretations (e.g. an **inform** and a **suggest**) and some acts will take more than one utterance to be completed.

The analyzer is divided into two parts, the **Speech Act Interpreter** which provides a structured list of act possibilities based on the linguistic form of the utterance, and the **Speech Act Pruner**, which filters these possibilities based on contextual information and binds indexical information to the appropriate values, producing a list of acts which have actually been determined to occur. These acts are then used to update the discourse context.

The following core speech act types are handled in the current implementation:

inform Speaker presents Hearer new knowledge in an attempt to add a new mutual belief.

ynq Speaker asks Hearer to provide information Speaker is missing but suspects Hearer may know; imposes a discourse obligation to respond.

check Like a ynq, but Speaker already suspects the answer; Speaker wants to move the proposition in question from individual to mutual belief.

suggest Speaker proposes a new item as part of a plan. Plan recognition is also performed to incorporate presuppositions and implicatures of the suggestion.

request Like a suggest, but imposes a discourse obligation to respond.

accept Speaker agrees to a proposal by Hearer; proposal is now shared.

reject Speaker rejects a proposal by Hearer.

Multiple act possibilities springing from a single utterance can be connected with any of the following operators:

- `or` : any or all of the constituents are possible
- `and` : all of the constituents must succeed or this interpretation fails
- `ex-or` : the first successful act is the interpretation
- `sequence` : each constituent is to be verified in the context resulting from interpreting the prior constituent

These constituents may also recursively contain act lists, e.g. the surface interpretation of sentence (8) in the sample dialogue in figure 2 below, "Yes, and we'll have E3 pick it up." is

```
(:SEQUENCE
    (:OR [ACCEPT-25] [INFORM-26])
    (:OR [INFORM-23] [SUGGEST-24]))
```

Since the "yes" can either be an answer to a question or an acceptance of a suggestion, and the second part can be either an inform about the event in question or a suggestion to add it to the current plan.

3.2 The Discourse Context

The **Discourse Context** contains the following kinds of information which must be maintained by the dialogue manager during the conversation:

- **Turn-taking**: the notion of who has the turn is important in deciding whether to wait for the other agent to speak, or whether to formulate an utterance. It will also shape the type of utterance that will be made, e.g. whether to use some kind of interrupting form or not. The TRAINS-92 system represents the *turn* with a variable which indicates the current holder. The turn is changed by means of **turn-taking acts** which are realized in particular utterances. **turn-taking acts** are described in more detail in [Traum and Hinkelman, 1992].
- **Discourse Segmentation** information [Grosz and Sidner, 1986] is kept for a variety of purposes in linguistic interpretation and generation, including the ability to determine the possible referents for a referring expression, and the intentional relations between utterances. The currently open segment structure will guide how certain utterances will be interpreted. Utterances like "yes" can be seen as an acceptance or confirmation of the last question asked but unanswered, if one exists in an open segment. Certain utterances such as "by the way", or "anyway", or "let's go

back to ..” or “let’s talk about ..” will signal a shift in segments, while other phenomena such as clarifications will signal their changes in structure just by the information content. In addition to general segmentation information, a structure of conversationally accessible domain plans is maintained. Since the purpose of the trains system is to negotiate a shared plan for achieving domain goals, suggestions of domain actions are taken to be suggestions that the action be part of a particular plan.

- **Discourse Obligation Stacks** are kept for each participant to monitor current discourse obligations. In the current system two types of obligations are possible: to answer a question or address a request.
- **Discourse Goal Structure:** two types of goals are present, local goals, which spring up in the course of the conversation, and more global goals governing the purpose of the conversation. Local goals are not represented explicitly, but are embedded in the control structure of the discourse actor (see below). Local goals include satisfying obligations, and moving from private to shared plans.

  Global goals are represented by discourse scripts, prebuilt recipes for general interaction. The top level script for the TRAINS situation will have the following components, in addition to the general opening and closing phases:

  - Specify goal
  - Construct and verify plan
  - Execute plan

  The current goal will be important in verifying particular speech act interpretations and will guide the discourse actor when there are no local goals.

3.3 The Discourse Actor

The **Discourse Actor** is the central *agent* of the Dialogue Manageer. It decides what to do next, given the current state of the conversation and plan. It can perform speech acts, by sending directives to the **NL Generator**, make calls to the domain plan reasoner to do plan recognition or plan construction in one of the domain plan contexts, or call the plan executor to execute a domain plan.

The implemented version of the discourse actor is a simple reactive agent, its behavior determined by relationships among elements of the discourse context. It will first try to address all obligations commencing with the most recent, then generate whichever intended speech acts it can, finally addressing discourse goals. It will first try to maintain a *balance* among the domain plan spaces by performing suggestions, repairs, acknowledgements, requests for acknowledgement, acceptances, or requests for acceptance as needed, and when none are necessary it will work on the global goals. There is no discourse planning done by the current system. For each of the possible goals in the discourse scripts, there is a verification procedure to check if it has been completed, and a simple set of actions to execute to contribute to meeting the goal. For example, when the top goal is to construct a plan, the domain reasoner is called to elaborate the current plan. This will result in new items added to the **System Plan** context, which will then introduce local goals to get this information shared by proposals and solicitations of acceptance.

The dialogue model, while very simple, is thus very flexible. It can follow its own initiative toward meeting the plan construction goal, or it can follow the initiative of the user. It is also able to shift control dynamically, taking the initiative given an opportunity, but reacting to user input following conversational conventions.

4 Capabilities of The Model

The TRAINS-92 dialogue model can handle a fairly complex range of task oriented conversations along the lines of the one in Figure 2. It can process indirect speech acts, and infer plans which are never explicitly stated. It can carry on a fairly sophisticated negotiation of the content of plans, until an executable plan is shared. It has a rudimentary way of dealing with turn taking, acknowledgements and first person repair, and handles obligations incurred in conversation more straightforwardly than previous systems.

As an example of the model, consider how it handles the conversation in Figure 2, with the relevant portion of the trains world shown in Figure 3. (1) introduces the current plan and outlines its goal, to make OJ. The rest of this fragment is devoted to working out an implementable plan to fulfill this goal. Utterances 2-4, while they have the surface form of inform acts, are be interpreted in the context of building the plan as suggestions. Thus the user is not informing the system of the locations of various objects in the TRAINS world (this interpretation is ruled out because it is mutually assumed that the system already knows these facts), but is suggesting that they are somehow relevant to

USER:	(1)	We have to make OJ.
	(2)	There are oranges at I
	(3)	and an OJ Factory at B.
	(4)	Engine E3 is scheduled to arrive at I at 3PM
	(5)	Shall we ship the oranges?
SYSTEM:	(6)	Yes,
	(7)	shall I start loading the oranges in the empty car at I?
USER:	(8)	Yes, and we'll have E3 pick it up.
	(9)	OK?
SYSTEM:	(10)	OK

Figure 2: Sample TRAINS Conversation

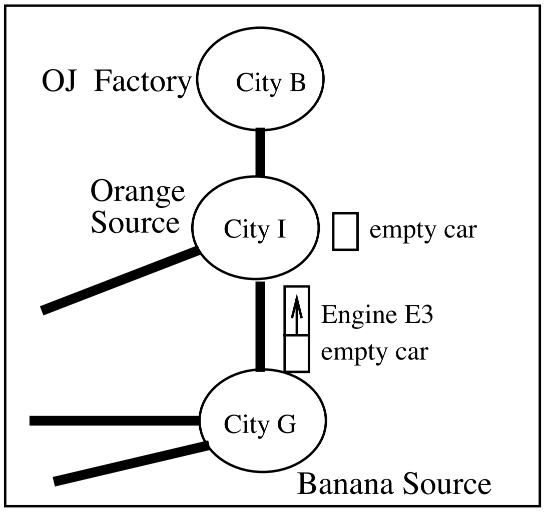

Figure 3: Trains World Scenario for Figure 2

the plan. In performing plan recognition, the system discovers that the user is suggesting using the OJ Factory at City B to make OJ from the oranges at City I, using Engine E3 to transport them. This also fills in the missing context for utterance 5: we want to ship the oranges at I to B using engine E3, as part of out plan to make OJ. Utterance 5 is also seen as a *release-turn* action, mainly in virtue of its question form.

The first thing the system does after taking up the turn is to accept the previous suggestions. While the previous plan recognition and inference had all been going on within the **User Proposed** context, this acceptance moves the entire plan, as so far constructed, to the **Shared** context. Now the dialogue manager calls the domain plan reasoner to do further *plan construction* on this plan to fill in any missing pieces. It comes back with the information (in the **System Plan** context) that in order to transport the oranges, a car is needed to put them in. There are two likely candidates, as shown in Figure 3, one being C1, the empty car already at City I, the other C2, the car already attached to e3. The system arbitrarily decides to pick C1, and suggests this to the user in utterance (7), copying the new plan item to **System Proposed**. This utterance also releases the turn back to the user. The user accepts this suggestion with utterance (8) (moving this part to **Shared**), and also adds the the item that E3 will couple to C1 and take it to B. Utterance (9) requests acceptance, and releases the turn to the system. Everything now seems complete (the unexpressed actions of unloading the oranges and starting the factory having been assumed recognized previously by plan recognition), so the system accepts the plan (utterance (10)), and sends commands off to the executer to put the plan into effect.

Acknowledgements

I would like to thank the TRAINS group at the University of Rochester for providing a stimulating research environment and a context for implementing these ideas. Special thanks to James Allen who supervised this part of the system, Peter Heeman who wrote the current implementation of the speech act interpreter, Brad Miller who maintained and extended the Rhet system when needed, and George Ferguson, Massimo Poesio, and Len Schubert for providing valuable feedback. This work was supported in part by the NSF under research grant no. IRI-9003841, by ONR under research grant no. N00014-90-J-1811, and by DARPA/ONR under contract N00014-92-J-1512.

References

[Allen and Miller, 1989] James F. Allen and Bradford W. Miller. The rhetorical knowledge representation system: A user's manual. Technical Report TR238R, Computer Science Dept. University of Rochester, March 1989.

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

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

[Ferguson, 1992] George Ferguson. Explicit representation of events, actions, and plans for assumption-based plan reasoning. Technical Report 428, Computer Science Dept. University of Rochester, June 1992.

[Grosz and Sidner, 1986] Barbara Grosz and Candice Sidner. Attention, intention, and the structure of discourse. *CL*, 12(3):175–204, 1986.

[Hinkelman, 1990] Elizabeth Hinkelman. *Linguisic and Pragmatic Constraints on Utterance Interpretation.* PhD thesis, University of Rochester, 1990.

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

[Schubert and Hwang, 1989] L. K. Schubert and C. H. Hwang. An episodic knowledge representation for narrative texts. In *1st Inter. Conf. on Principles of Knowledge Representation and Reasoning (KR89)*, pages 444–458, Toronto, Canada, May 15-18, 1989.

[Traum and Allen, 1991] David R. Traum and James F. Allen. Conversation actions. In *Working Notes AAAI Fall Symposium on Discourse Structure in Natural Langauge Understanding and Generation*, November 1991.

[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. Special Issue on Non-literal language.