Integrating Natural Language Understanding and Plan Reasoning in the TRAINS-93 Conversation System *

D. R. Traum[†], J. F. Allen[†], G. Ferguson[†], P. A. Heeman[†], C. H. Hwang[†], T. Kato[‡], N. Martin[†], M. Poesio[§] and L. K. Schubert[†]

Abstract

This paper describes the TRAINS-93 Conversation System, an implemented system that acts as an intelligent planning assistant and converses with the user in natural language. The architecture of the system is described and particular attention is paid to the interactions between the language understanding and plan reasoning components. We examine how these two tasks constrain and inform each other in an integrated nl-based system.

1 The TRAINS Project

The TRAINS project is a long-term research project to develop an intelligent planning assistant that is conversationally proficient in natural language[Allen and Schubert, 1991]. The TRAINS system helps a user construct and monitor plans about a railroad freight system. The user is responsible for assigning cargo to trains and scheduling shipments, scheduling various simple manufacturing tasks, and for revising the original plans when unexpected situations arise during plan execution. Figure 1 shows a typical initial scenario. The system aids the user in all aspects of this task by interacting in natural language. In particular, the system typically will perform the following tasks:

- Evaluating courses of action, calculating expected completion times, detecting conflicts, and so on;
- Filling in details of the proposed plan that do not require the user's attention;
- Suggesting ways to solve particular subproblems as they arise;
- Presenting and describing the current state of the world and how the proposed plan may affect it;
- Dispatching the plan to the different agents in the world, including the train engineers and factory users;
- Interpreting reports back from the engineers and factory users in order to monitor the progress of the plan and to anticipate problems before they arise; and
- Coordinating the correction and modification of plans with the user.

While we aim to produce a functioning system, the system itself is not really the goal of the effort. Rather, the domain is a tool for forcing our research to address the problems that arise in building a complete dialogue system. For instance, the dialogue module must be able to handle a wide range of everyday discourse phenomena rather than handling a few selected problems of theoretical interest. While this approach focuses our research directions, the solutions that we seek are general solutions to the phenomena rather than specific solutions that happen to work in the TRAINS domain. The TRAINS project currently has several main foci of research:

- Parsing and semantically interpreting utterances as they arise in spoken language, including sentence fragments, repairs and corrections;
- Accounting for the discourse behavior present in natural dialogue;
- Representing the reasoning and control of the discourse agent, including reasoning about plans in the TRAINS domain as well as the plans that drive the system's behavior in the dialogue itself; and
- Providing the knowledge representation and reasoning tools that are needed for planning and scheduling in realistic size domains, including reasoning about time, events, actions and plans.

Four prototypes of the TRAINS system have been developed, one per year starting in 1990. We feel that most of the issues to be discussed at this symposium were raised at one point or the other during our work of developing and refining the TRAINS system. In the final analysis, we can say the system "understands" NL input from the user if it executes the plan as the user expects.[1] More locally, the system can

*This material is based upon work supported by ONR/DARPA under grant number N00014-92-J-1512, the US Air Force under Rome Laboratory research contract number F30602-91-C-0010, and ONR under research grant number N00014-90-J-1811

[†]{traum, james, ferguson, heeman, hwang, martin, schubert}@cs.rochester.edu University of Rochester; Rochester, NY, 14627-0226.

[‡]kato@nttnly.ntt.jp NTT Network Information Systems Labs; 1-2356 Take, yokosuka-shi, Kanagawa 238-03 JAPAN.

[§]poesio@cogsci.ed.ac.uk University of Edinburgh, Centre for Cognitive Science, Edinburgh, EH8 9LW, Scotland, UK.

[1] Actually, this condition is too strong, as a user might fail to convey expectations properly, so a more realistic objective might be to execute a plan according to expectations of an *observer* of the conversation.

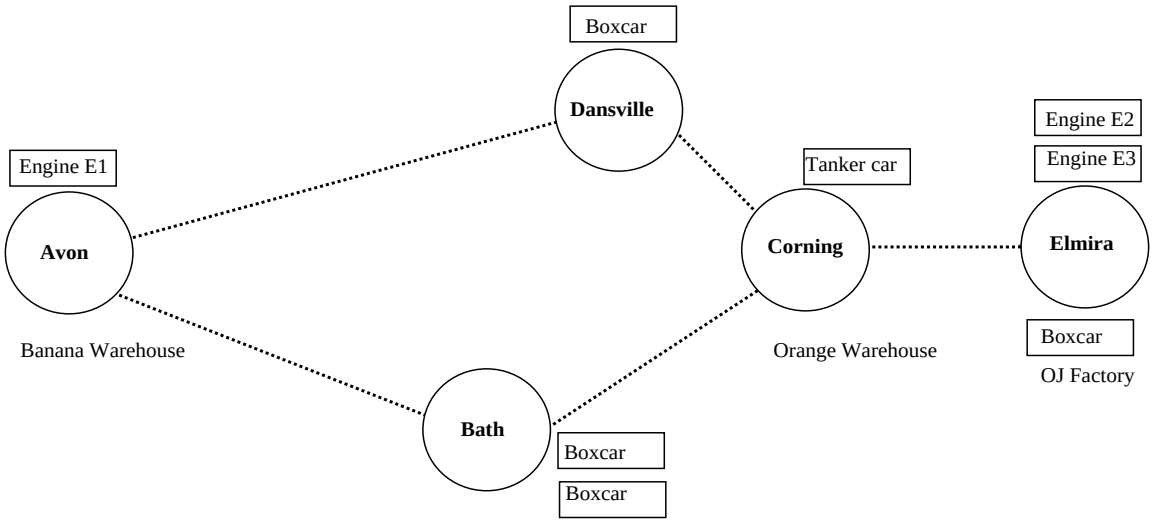

Figure 1: Trains World Set-up for Dialogue Collection

assume it understands a particular utterance if it can produce a coherent interpretation of the utterance, both in terms of the additions to the plan, and in terms of discourse structure expectations and conventions.

2 Modules of the TRAINS-93 System

Figure 2 shows the modules of the 1993 implementation of the TRAINS System, as well as the main flow of communication between modules.

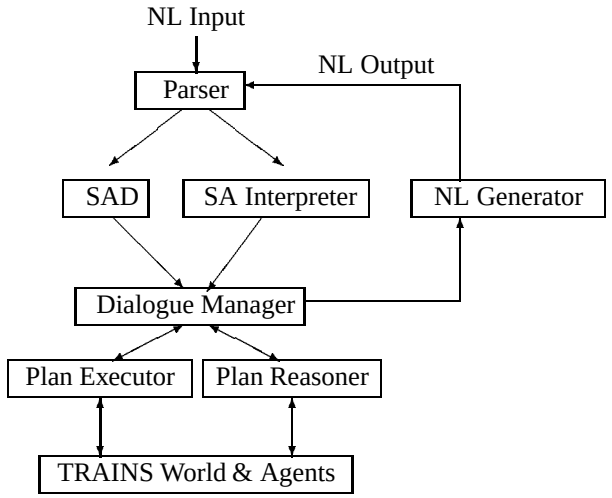

Figure 2: TRAINS System Architecture

As the figure indicates, the language interpretation modules are tightly integrated with the planning and plan execution task modules. Granularity of interleaving in the implementation is at the sentential utterance level. Each utterance (from either the user or the system) is processed up to the dialogue manager (which calls the domain reasoner to disambiguate hypotheses about meaning and to update the representation of the current plan). Information from the planner can be used as a basis for forming NL responses, e.g., acceptances, rejections, or repairs of plan-based suggestions.

2.1 Parsing and LF-Computation

The first module in the interpretation process is a parser which takes an utterance and produces a representation that combines the result of syntactic analysis and of lexical interpretation. We use A GPSG style grammar that makes extensive use of a feature system including subcategorization features and several feature principles governing feature percolation. Each rule in the grammar consists of a syntactic rule coupled with a corresponding semantic rule. This allows us to build the interpretation compositionally in the sense of Montague's semantics.

The computed representation is an unscoped logical form (ULF) in *Episodic Logic* [Hwang and Schubert, 1993]. Episodic logic is a very expressive knowledge and semantic representation which is capable of representing the nuances of a natural language, including tense and aspect, purposes, goals and plans, causes, facts, beliefs, surface speech acts, and various kinds of adverbials. A ULF is an *underspecified interpretation* in that it involves unscoped operators (e.g., quantifiers and coordinators) as well as indexical operators/terms (e.g., speech act operators, tense operators, and pronouns).

2.2 Scope and Deindexing

The task of the Scope and Deindexing module, SAD-93, is to 'deindex' (i.e., fix the value of) context-dependent aspects of an utterance's content such as referential expressions. The input to SAD-93 is the underspecified representation produced by the parser. The output is the set of alternative hypotheses about how to resolve the ambiguity that are suggested in the given context.

SAD-93 arrives at an hypothesis by applying discourse interpretation rules with varying strength, all of which operate directly off the underspecified interpretation. The decision to integrate scope interpretation with other aspects of pragmatic interpretation reflects the thesis put forth in [Poesio, 1994] that the scoping preferences of human subjects result from the interaction of several discourse interpretation processes.

2.3 Speech Act Analysis

The speech act interpreter is responsible for determining what the speaker means by her utterance. For instance, when a speaker utters "there are oranges at Corning", (as in utterance 3-7 in the conversation presented in Figure 3, below) she might be saying this to *inform* the hearer of this fact, to *check* whether the hearer agrees with this, to *question* the hearer about whether this is the case, or as a *suggestion* to use the oranges in the current plan. These alternatives are not all mutually exclusive. The speech act interpreter builds a list of hypotheses about the speech act interpretations of an utterance.

Rather than working from fully scoped and deindexed forms, the speech act analyzer, like the Scope and Deindexer, takes the unscoped Logical Form as its input. By working off of the unscoped Logical Form, the speech act analyzer can make use of clues that are present in the surface form, such as modals and left-clefts to aid in the analysis, clues which would be blurred by scope and deindexing. After both the speech act analysis and scope and deindexing are completed, the result of scope and deindexing is incorporated into the speech act hypotheses passed on to the dialogue manager for verification.

2.4 Dialogue Management

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 goal is 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, verify the hypotheses about intentions behind utterances of the user, send system intended speech acts to the NL generator to produce system utterances, and send commands to the domain plan reasoner and domain plan executor when appropriate. The dialogue manager is described in more detail in [Traum, 1993].

2.5 Domain Plan Reasoning

The plan reasoner provides planning and plan recognition services and performs reasoning about the state of the world. 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, we are developing an explicit representation of plans as arguments based on assumptions [Ferguson, 1992].

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. The dialogue manager uses the results of plan reasoning to disambiguate speech act interpretations, update beliefs, and generate new conversational elements (*e.g.*, an ambiguity detected by the plan reasoner could cause a clarification sub-dialogue to be started).

2.6 Plan Execution

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 (communicating with these agents as they perform their tasks) to make sure the plan execution is successful.

The plan executor also performs two additional services for other modules. It can aid the planner by making choices among a set of alternatives presented by the planner (*e.g.*, a choice of which among a set of alternative paths to take between destinations).

The executor also gathers the information that allows it to make better choices on subsequent queries. It bases its decisions on constraints on probability that are inferred from observations, from probabilities asserted by the programmer and from facts inferred from its knowledge base. The executor gathers information by observing events in the world and inferring the type of events that caused the observation. It then updates the statistics on the events observed and the events inferred. The principles behind the executor's operation are described in more detail in [Martin, 1993].

2.7 TRAINS World Simulation

The TRAINS World is a detailed simulation of action executions by agents on world objects. It is used for plan execution and monitoring (since, as far as the system is concerned, it is the real world). The intentions of the system have no direct bearing on the behavior of the World. Actions will take place regardless of the system's goals. The World implements a dynamic physical model unknown to the system. The only way the system can effect change in the world is through sending directives to the agents in the world: engineers, factory managers, etc. If they receive the messages (some may be lost), they execute the requested actions. These actions may or may not have the consequences anticipated.

2.8 NL Generation

The NL Generator takes speech act representations produced by the dialogue manager and converts them to natural language text which is then "uttered" as output to the user (and fed back through the language analysis modules).

3 Example Dialogue

Figure 3 shows a (slightly edited) dialogue collected as part of the study in [Gross *et al.*, 1993], in which a person played the part of the system. The dialogue concerns the TRAINS world situation given in Figure 1. The current implementation can interpret the user utterances in this dialogue and provide the appropriate system responses, while building up an executable plan.

1	U: We better ship a boxcar of oranges to Bath by 8 AM.
2	S : Okay.
3-3=6	U: So we need to get a boxcar to Corning where there are oranges.
3-7	U: There are oranges at Corning.
3-8	U: Right?
4	S : Right.
5-1	U: So we need an engine to move the boxcar.
5-2	U: Right?
6	S : Right.
7-1=2	U: So there's an engine at Avon.
7-3	U: Right?
8	S : Right.
9=13	U: So we should move the engine at Avon, engine E1, to Dansville to pick up the boxcar there.
14	S : Okay.
15-2=4	U: And move it from Dansville to Corning.
15-5=7	U: Load up some oranges into the boxcar.
15-8=10	U: And then move it on to Bath.
16	S : Okay.
17	U: How does that sound?
18-3	S : That's no problem.
19	U: Good.

Figure 3: The dialog processed by TRAINS-93.

4 Domain Constraints on NL Interpretation Tasks

The domain itself and the tasks of plan reasoning and plan execution constrain NL interpretation in a number of ways. Consider for example the interpretation of NPs. In the TRAINS domain, there is a fixed number of engines and train cars, so that in the situation represented in Figure 1, any talk of "an engine" must eventually boil down to E1, E2, or E3. On the other hand, in this domain all train cars are equivalent for all purposes; therefore it is never the case that an indefinite is interpreted specifically (see, for example, utterances 3-3=6 or 5-1).

Another important characteristic of the task is that two 'domains of conversation' are actually involved: the situation described by the map, and the 'possible situation' described by the plan. This affects the interpretation of referential expressions, for example, that can refer either to some object in the plan (see, e.g., "the boxcar" in utterance 5-1) or to some object "in the world" ("the boxcar there" in 9=13). Tense interpretation is similarly affected. Ambiguities may result; information about the plan is often necessary to disambiguate between likely candidates for referential expressions.

Plan recognition will serve as the ultimate arbiter for the meanings of utterances. For example, an utterance which suggests an action to be performed will generally have more profound effects than merely adding the mentioned action to the plan — plan recognition is performed to *incorporate* the action into the known part of the plan, linking up the action itself with whatever else is needed to connect the action to the eventual goal, and adding this whole structure to the plan. This extra connective material (which might include other actions to be performed) can be thought of as the *implicatures* of the utterance. The utterance cannot really be said to have been understood without understanding how it fits in with the rest of the conversation (in this case the rest of the plan).

Information returned from plan recognition can be used to respond to the user in a variety of ways. If the plan recognizer cannot link up the current suggestion to the plan, some sort of clarification is in order. Similarly, if a suggestion could be linked up in any of several different ways, this is also a good time for a clarification. If quite a bit of material is needed to incorporate the action into the plan, then it might be a good idea for the system to try to check this material explicitly, otherwise an acknowledgement of some sort is in order. If the system recognizes the connection, but finds the suggested additions nonoptimal, it may reject the suggestion and offer a counter proposal.

As well as incorporating suggestions by the user into a plan using plan recognition, the domain plan reasoner can elaborate plans on its own. Often the level of detail mentioned by the user is insufficient to actually carry out the objective in the world. The plan reasoner can fill in missing pieces using standard planning techniques. If the dialogue manager judges these additions significant enough, they will need to be suggested back to the user in so that the system can maintain with the user a shared view of the plan.

5 NL Interaction Constraints on Domain Plan Reasoning

The embedding of the domain plan reasoner within the context of a dialogue system also poses many constraints on the plan reasoning process. Most importantly, the plan representation language must be expressive enough to represent the kinds of things people say about plans in cooperative dialogue. For one thing, 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. Also, 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.

In a dialogue setting, plan recognition and planning must be 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 arbitrary knowledge context, making *incremental* changes in accord with user suggestions (e.g., utterances 1, 3-3=6, 5-1, 7-1=2, etc.).

The conversation setting also provides extra resources to a plan reasoner that an off-line system would not enjoy. For many computational problems (*e.g.*, NP-complete problems), proof construction is difficult, yet proof verification is relatively easy. If the plan reasoner had to come up with a plan on its own (even for a simple goal in a simple situation such as the example in Figure 1), it would run across many decision points, and many of the choices would lead to inefficient or unworkable plans. Allowing the user to make suggestions helps focus the search within a smaller search space.

Both when planning and when performing plan recognition, the domain plan reasoner is confronted with a number of choice points without any principled basis to choose one path over another. The executor can sometimes help, on the basis of its experience in similar situations, yet this may not always be sufficient (the executor may not have seen similar situations enough times). The only other alternative for a stand-alone domain plan reasoner is to choose randomly.

In an NL system, however, more options are available. An important task of the NL components of TRAINS is to exploit 'surface' cues and conversational implicatures to help the domain plan reasoner by focusing its search. Particular constructions such as purpose clauses and cue words (e.g., "so" and "and") provide important clues as to how the plan fragments described by utterances are related. Conversational implicatures based on Gricean maxims similarly play an important role: e.g., to suggest that the actions described by utterances 9=13 to 15-8=10 are related and temporally ordered.

In a NL system it is also possible for the planner to ask the user for a preference whenever alternative plans are available. Similarly, if the plan recognition process becomes too difficult (in the current implementation, this amounts to the search expanding beyond a preset depth without the system being able to incorporate the current event in the plan), the plan reasoner can signal the dialogue manager to request a clarification.

6 Discussion

Of particular interest is how the notion of *plan* is used differently by different modules. To the scope and deindexer, a plan represents a possible *situation*, which can be a resource for determining the referents of event and object descriptions. To the speech act interpreter and dialogue manager, a plan is an *object* which can contain various components (*e.g.*, events, goals). To the plan reasoner, a plan is an *argument* that an eventuality (the goal) will occur. To the plan executor, a plan is a list of instructions which must be translated into commands to send to the simulated agents in the world. Reconciling the ontological requirements of the separate modules is a problem that must be addressed by any system attempting to closely integrate natural language understanding, discourse, and planning in a principled way. The TRAINS project is an investigation into these issues; the TRAINS system distills the state of the art in formal work in several fields into a functional, interactive program.

The interest of the TRAINS project lies in the *generality* and theoretical defensibility of the multiple components that have been integrated into a single working system. This emphasis on well-foundedness ensures that the overall architecture and component integration are not simply a reflection of practical needs in a narrow task domain, but a possible general model of task-oriented dialogue agents. Our progress to date shows that our theoretical emphasis is not an impediment to practical implementation. Some small but realistic dialogues such as the one in Figure 3 are already handled by the system, without "shortcuts" in the requisite parsing, semantic interpretation, speech act analysis, dialogue management, plan inference, and world simulation. While the amount of knowledge possessed by the system remains small, the way that knowledge is represented, manipulated, and integrated into cooperating modules is in principle transferable to any other type of cooperative, task-oriented conversational domain.

References

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

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

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

[Hwang and Schubert, 1993] Hwang, C. H. and Schubert, L. K. 1993. Episodic Logic: A situational logic for natural language processing. In *Situation Theory and its Applications, V. 3*, CSLI, Stanford, CA.

[Martin, 1993] Martin, Nathaniel G. 1993. *Using Statistical Inference to Plan Under Uncertainty*. Ph.D. Dissertation, University of Rochester, Computer Science Department, Rochester, NY 14627.

[Poesio, 1994] Poesio, Massimo 1994. *Discourse Interpretation and the Scope of Operators*. Ph.D. Dissertation, University of Rochester. forthcoming.

[Traum, 1993] Traum, David R. 1993. Mental state in the TRAINS-92 dialogue manager. In *Working Notes AAAI Spring Symposium on Reasoning about Mental States: Formal Theories and Applications.* 143–149.