Knowledge Representation in the TRAINS System

M. Poesio,[*] G. Ferguson,[†] P. Heeman,[†] C. H. Hwang,[†] D. R. Traum,[†]
J. F. Allen,[†] N. Martin,[†] L. K. Schubert[†]

Abstract

The long term goal of the TRAINS project is to develop an intelligent planning assistant that is conversationally proficient in natural language. The TRAINS system helps a user construct and monitor plans about a railroad freight system; their interaction takes place in natural language. The representational needs of a system like TRAINS include representing lexical meaning, dealing with the problem of ambiguity, make use of information about context, and finding a connection between the content of the current utterance and the plan being jointly developed by system and user. The goal of the paper is to describe how TRAINS-93, the latest prototype of TRAINS, deals with these issues.

1 The TRAINS System

The long term goal of the TRAINS project is to develop an intelligent planning assistant that is conversationally proficient in natural language [Traum *et al.*, 1994; Allen *et al.*, 1994]. The TRAINS system helps a user construct and monitor plans about a railroad freight system; their interaction takes place in natural language.

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. The system aids the user in all aspects of task. 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;

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

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

UU#	Speaker: Utterance
0	S: Hello, this is the TRAINS system. I can help you plan and execute actions in the TRAINS World.
1.1	U: we better ship a boxcar of oranges to Bath by eight a.m.
2.1	S: okay.
3.1	U: so we need to get a boxcar to Corning, where there are oranges.
3.2	: there are oranges at Corning.
3.3	: right?
4.1	S: right.
5.1	U: so we need an engine to move the boxcar.
5.2	: right?
6.1	S: right.
7.1	U: so there's an engine at Avon.
7.2	: right?
8.1	S: right.
9.1	U: so we should move the engine at Avon, engine E1,
	: to Dansville to pick up the boxcar there.
10.1	S: okay.
11.1	U: and move it from Dansville to Corning,
11.2	: load up some oranges into the boxcar,
11.3	: and then move it on to Bath.
12.1	S: okay.
13.1	U: how does that sound?
14.1	S: that's no problem.
15.1	U: good.

Figure 1: An example of TRAINS dialog

- 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.

Figure 2 shows a typical initial scenario. One of the conversations on which the system has been tested is shown in Figure 1; the system replicates the behavior of the participant marked as 'S'.

The representational needs of a system like TRAINS include representing lexical meaning, dealing with the problem of ambiguity, make use of information about context, and finding a connection between the content of the current utterance and the plan being jointly developed by system and user. The goal of the paper is to describe how TRAINS-93, the latest prototype of TRAINS, deals with these issues.

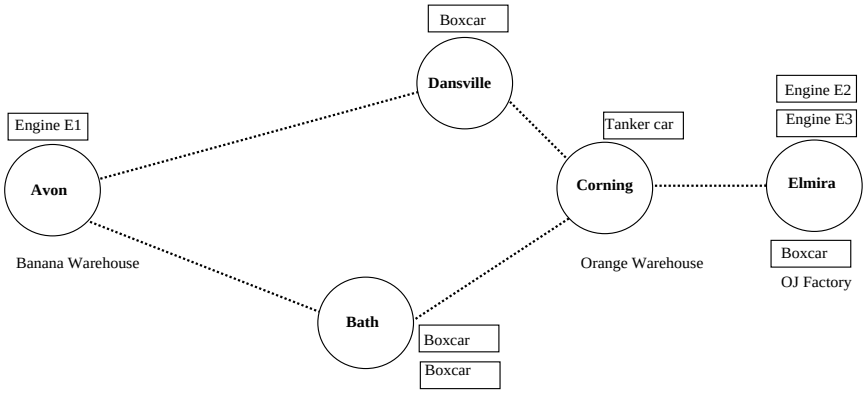

Figure 2: Trains World Set-up for Dialogue Collection

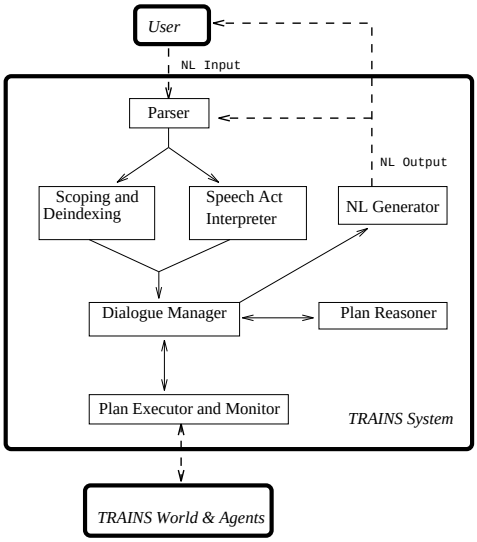

Figure 3: TRAINS System Architecture

2 The Representation Formalisms Used in TRAINS-93

Roughly put, there are four kinds of 'knowledge' that the modules of TRAINS-93—the 1993 implementation of the TRAINS system, whose architecture is shown in Figure 3—need to represent: information about the meaning of an utterance, information about the current context of interpretation, information about the 'world', and information about the plan. At the beginning of the TRAINS project, the decision was made not to adopt a single representation formalism to be used by all modules; the developers of each module were left free to adopt whichever representation was best suited for their task. As a result, several different representations and their associated reasoning methods are currently used in the TRAINS system. Each of these will be presented briefly, and its use illustrated by means of an example from the conversation in Figure 1, sentence `5.1`, repeated here for convenience. In section 3 we return to the issue of having multiple formalisms in an NLP system.

(1) `5.1` U: So we need an engine to move the boxcar.

2.1 Episodic Logic

The task of the modules concerned with the interpretation of utterances, including the Parser, the Scope and Deindexing (SAD) module,[1] the Speech Act Interpreter, and the Dialogue Manager (see Figure 3) is to arrive at one or more hypotheses about the intended meaning of an utterance. The language used to describe the meaning of the utterance must be rich enough to allow distinctions in meaning to be captured.

For example, it won't do to represent the meaning of `5.1` in Figure 1 as "There is an engine x and we need x to move the boxcar," because it is possible to use that utterance even in circumstances where there is no engine, but this paraphrase would be false under these circumstances. Ideally, the language used to represent lexical meanings should also make it simple to combine the meanings of lexical items in a single meaning.

The representation used by the Parser to represent lexical semantics is called Episodic Logic (EL). EL is a situational logic developed as a knowledge and semantic representation suitable for general NLU [Hwang and Schubert, 1993a; Hwang and Schubert, 1993b]. The most distinctive feature of EL is its natural-language like expressiveness. It allows for generalized quantifiers, lambda abstraction, sentence and predicate nominalization, sentence and predicate modifiers, intensional predicates (corresponding to wanting, needing, believing, etc.), tense and aspect, surface speech act operators, and explicit situational variables together with operators to specify what's true at a situation. Each of the phrase structure rules in the TRAINS grammar is paired with a semantic rule, and the Parser computes the initial, underspecified EL representation of an utterance as it builds a parse tree for the utterance.

The modules of TRAINS more directly involved with the interpretation of utterances also have to worry about *ambiguity*. As the parser does not do reference resolution and scope disambiguation, the representation it produces must be *underspecified*, in the sense that it must indicate which aspects of the utterance's meaning have to be resolved in context; as well, it must not represent a commitment to a specific scope of operators. In other words, the ability to represent semantic ambiguity *implicitly* is required.

[1] This module performs scope assignment and reference resolution.

The initial representation produced by the Parser is underspecified in that it may contain unscoped operators (quantifiers, in particular), indexical expressions (e.g., tense operators) and ambiguous predicates. For example, adverbial infinitives are initially translated using the ambiguous predicate, IN-DISCOURSE-RELATION, which will be later particularized into FOR-PURPOSE, WITH-RESULT, etc. Situational variables are introduced into the representation at a later stage when tense-aspect operators are processed. Both the Scope and Deindexing module and the Speech Act Analysis module operate on the basis of such underspecified representations, and generate hypotheses about which one among the available interpretations was intended.

The Parser produces for (1) the underspecified representation shown in (2). It can be thought of as a parse tree whose leaves have been replaced by the semantics of lexical items. The labels `S_N2+V2`, `V2_V+N2`, etc. represent the left-hand side and right-hand side categories occurring in the phrase structure rule used. The semantic aspect of the rule is formulated as Episodic Logic expressions, represented as Lisp expressions of the form `(<key> <item>+)`, where `<key>` indicates the semantic object represented by the expression. For example, the key `:F` indicates a functional application of the first argument to the rest, the key `:L` indicates a lambda-expression, and the key `:0` indicates an unscoped operator.

(2)
```
(1 UTT 1
 (1 S-TELL (:F :DECL 1)
  (1 S_CONJCOORD+S (:I :UTT-IMP 1 2)
   (1 SO1 :SO-COORD)
   (2 S_N2+V2 (:I 1 2) (1 WE1 :WE)
    (2 V2_V2+VINF
     (:F (:F :ADV-A
           (:F :IN-DISCOURSE-RELATION
               (:F :KA 2)))
      (:L :X (:I :X 1)))
     (1 V2_V+PREDNTYPE (:F 1 2)
      (1 NEED3.3 (:0 :PRES :NEED-REQUIRE))
      (2 PRED_DETAN+N1 2
          (1 AN1 :E)
          (2 ENGINE1.1 :ENGINE)))
     (2 V2_V+V2BASE (:F 1 2) (1 TO1 (:L :X :X))
      (2 V2_V+N2 (:P 1 2)
       (1 MOVE1.1 :MOVE)
       (2 N2_DET+N1SING (:Q 1 2)
        (1 THE1 :THE)
        (2 BOXCAR1.1 :BOXCAR)))))))
  (2 /PERIOD1 NIL)))
```

2.2 Conversation Representation Theory

Conversation Representation Theory (CRT) [Poesio, 1994] is an account of contextual interpretation and ambiguity resolution in conversations. The language of CRT is a superset of the language of Episodic Logic; the augmentations are designed to provide better tools to describe the context of interpretation, especially to indicate which referents are available and under which conditions, and which aspects of a sentence's meaning need to be resolved.

In CRT, what takes place after an utterance is produced is best seen as a process during which alternative hypotheses about the change in the *discourse situation* brought about by that utterance are obtained. A fundamental aspect of the theory is that hypothesis generation is a process of context update; this process is formalized in terms of operations on structures called DRSs[2] that represent the discourse situation. Semantically, the main characteristic of CRT is that its formulas denote sets of functions from situations to truth values, so that the language can be used to describe both semantically 'unambiguous' expressions (that denote singleton sets of such values) and ambiguous expressions, such as the underspecified representations obtained by the Parser (that denote sets of cardinality greater than one).

The underspecified representation produced by the Parser is converted into a CRT expression that represents the (surface) *conversational event*, or locutionary act, resulting from the production of that utterance, and this expression is added to the DRS representing the current discourse situation. The Scope and Deindexing (SAD) module uses the information contained in the resulting DRS to do inferences that resolve contextually-dependent expressions; this results in one or more hypotheses.

The hypothesis obtained by processing (2) is shown in (3). As the syntax of CRT is a superset of that of Episodic Logic, this conversion is mostly straightforward; we ignore the differences in this paper (but see [Poesio, 1994] for a discussion).

(3)
```
(:SIT-DESCR :COA1
 (:DRS (:CE5950 :CE5951)
  (:EV-DESCR :CE5950
       (:DRS NIL (:I :HUM :SO-INIT)))
  (:I :CE5950 :AT-ABOUT :NOW8)
  (:I :CE5950 :SUBSIT :COA1)
  (:EV-DESCR :CE5951
   (:DRS NIL
    (:I :HUM :TELL :SYS
        (:SIT-DESCR (:PAR :*S5951* :SIT)
         (:DRS (:E5946 :E5945 :X5946)
          (:EV-DESCR :E5946
           (:DRS NIL
            (:I (:SET-OF :HUM :SYS)
                (:F :NEED-REQUIRE
                 (:LAMBDA :Y
                  (:Q
                   (:OP :A :X5945
                    (:I (:PAR :*RES-SIT5945* :SIT)
                        :* (:I :X5945 :ENGINE)))
                   (:I :Y := :X5945)))))))
          (:I :E5946 :AT-ABOUT :CE5951)
          (:I :E5946 :SUBSIT :S*)
          (:EV-DESCR :E5945
            (:DRS NIL
              (:I (:SET-OF :HUM :SYS)
                  :MOVE :X5946)))
          (:I :E5946 :FOR-PURPOSE :E5945)
          (:I :X5946 := :X2887)
          (:SIT-DESCR :PLAN1
            (:DRS NIL (:I :X5946 :BOXCAR)))))))))
  (:I :CE5951 :AT-ABOUT :NOW9)
  (:I :CE5951 :SUBSIT :COA1)
  (:I :CE5950 :BEFORE :CE5951)))
```

This expression states, roughly, that the discourse segment `:COA1` (a situation) has been augmented with two conversational events, `:CE950` and `:CE951`. An expression of the form `(:DRS (:A ... :B) F1 ... Fn+)` represents a DRS, that in CRT is interpreted as the representation of a situation type: `:A` ... `:B` are the constituents of a situation and each expression `Fj` represents a fact that has to hold at that situation.

Of the two conversational events that are part of `:COA1`, `:CE951` is most relevant; this is an event of the agent `:HUM` telling the agent `:SYS` that a situation to be contextually determined, `(:PAR :*S5951* :SIT)` is characterized by two eventualities: `:E5946` and `:E5945`. The expression `(:PAR :*S5951* :SIT)` is a *parameter*. Parameters stand for 'holes' in the meaning of an utterance that need to be filled by reasoning, and are used to represent pronouns, definite descriptions, and so forth. The representation includes the

[2] This construct is borrowed from DRT [Kamp and Reyle, 1993].

information that the referent :X5946 of the definite description "the boxcar" has been identified as being equal to the DRS constituent :X2887 introduced in utterance 3.1.

2.3 EBTL

The EBTL language (EBTL stands for *Event Based Temporal Logic*) is used as a communication language between the Speech Act Interpreter, Dialogue Manager and Plan Reasoner. It is also used by the Dialogue Manager to represent conversation level information. EBTL is a typed logic based on that presented in [Allen, 1991] and is implemented using the RHET knowledge representation system [Allen and Miller, 1991]. Rhet includes support for type reasoning, temporal reasoning, hierarchical belief contexts, equality and inequality reasoning. EBTL adds a much needed quotation facility (as some operators need to treat propositions as terms), lambda abstraction, existential and definite quantifiers, and discourse markers. The language is more restricted than those of EL or CRT: only events and some simple states can be represented directly rather than general episodes.

Both hypotheses such as (3) and the underspecified representation in (2) are fed to the Speech Act Interpreter, which produces a set of core speech act alternatives and a list of argumentation acts (which will not be discussed). In (4), the speech act definition, expressed in EBTL, is given for one of the possible core speech interpretations, [ST-CHECK-0024].

(4)
```
(:AND (:ROLE [ST-CHECK-0024] :R-SPEAKER [HUM])
      (:ROLE [ST-CHECK-0024] :R-HEARER [SYS])
      (:ROLE [ST-CHECK-0024]
            :R-TIME [F-TIME [CE5951]])
      (:CONTENT [ST-CHECK-0024]
       (:NEED-REQUIRE [SYSHUM]
        (:LAMBDA ?O*T-ENGINE
         (:DISC-MARKER [X5945] ?O*T-ENGINE ))))
      (:FOCUS [ST-CHECK-0024] [X5945]))
```

In general, there can be many interpretations for an utterance, and the manner in which they can provide an interpretation for the utterance is specified in an alternative list. This list relates the conversational event to its interpretations via the predicate SURF-INTERP. The manner in which the alternatives can combine is specified by the operators AND, OR, and EX-OR.

(5)
```
(SEQUENCE
      (SURF-INTERP [CE5950] [ST-ACCEPT-0022])
      (SURF-INTERP [CE5951]
        (AND (OR (EX-OR [ST-INFORM-0023]
                        [ST-CHECK-0024]
                        [ST-YNQ-0025])
                 [ST-SUGGEST-0026])
             [ST-SUPP-SUG-0027])))
```

The Dialogue Manager, reasoning on the basis of the definition of the speech acts, consults the Plan Reasoner; in this particular case, the final decision is to incorporate a new component in the plan:

(6)
```
#[(INCORPORATE
    (:USE-OBJECT
      (:LAMBDA ?o*T-ENGINE
        (:DISC-MARKER [X5945] ?o*T-ENGINE )))
    [PLAN-2383]
    :INCORP-TYPE :ROLE-FILLER
    :FOCUS
       (:LAMBDA ?o*T-ENGINE
         (:DISC-MARKER [X5945] ?o*T-ENGINE ))
    :PURPOSE
     (:LF-EXISTS ?L*T-MOVE  [E5945]
       (:APPLY
         (:LAMBDA ?E*T-MOVE
           (:AND (:ROLE ?E*T-MOVE
                   :R-AGENT [SYSHUM])
                 (:ROLE ?E*T-MOVE
                   :R-OBJECT [X5946])))
         ?L*T-MOVE )
       (:OCCURS ?L*T-MOVE ))
    :CONTEXT     *PLANNER-RETURN3*)#]
```

2.4 Plans as Arguments

The TRAINS domain plan reasoner represents plans as connected, directed, acyclic graphs of *event* and *fact* nodes labelled by events and propositions, respectively. These plan graphs are given a formal semantics in terms of *arguments*: they represent an argument that a certain course of action under certain conditions will achieve certain goals. Since plans are objects in the ontology, plan description predicates can be defined that impose constraints on the plan-as-argument. There are *structural* predicates, such as ActionIn, Enables, Premise, Goal, etc., that are defined in terms of the structure of the plan graph. Then there are *evaluation* predicates. These include *absolute* predicates such as Plausible or Impossible and *relative* predicates that allow plans to be compared, for example, according to resource usage or time constraints. Further details are available in [Ferguson and Allen, 1994].

As a result of processing utterances 1.1-4.1, the plan graph corresponding to [PLAN-2545] contains two Move-Car events: one is moving a car to Corning (where there are oranges), the other is moving a car from Corning to Bath (to move the oranges). Although both Move-Car events in the plan could unify with the purpose formula in 5.1, the first one is preferred as it requires fewer equality assumptions. In fact though, either choice would be possible since in either case the Move-Car event has a role which can unify with the lambda expression describing the engine. Appropriate assumptions and new parts of the plan graph are returned to the dialog manager as plan description predicates, which can then be used to generate helpful suggestions or request confirmations.

3 Discussion

One of the issues mentioned in the call for papers—the relative advantages and disadvantages of having one or many formalisms—was repeatedly raised during our work on the project. Adopting different formalisms made it easier to develop general-purpose components that could also be used in isolation. In addition, it was not obvious that any existing formalism would have been appropriate to describe all the different kinds of information we needed to represent, and we felt it was best to postpone any unification attempts until we gained a better understanding of the problems we had to face.

On the other hand, not choosing a single representation formalism from the very beginning involved a risk, namely, that we would end up with a number of mutually incompatible representations. In fact, this did not happen. There are strong connections between the formalisms we use, and a number of ontological assumptions are shared, so that most translations are a straightforward matter.

A number of factors conspired in keeping the formalisms we use relatively close to each other. First of all, there was an unspoken agreement by all to adopt logic-based formalisms. Secondly, as work progressed, it became more and more clear that a tight interaction between the modules was needed; the

typical module of TRAINS has to interact with at least two other modules, and, in some crucial cases, this interaction has to be two-way. A particularly important case is the interaction between the Dialogue Manager and the Plan Reasoner: the Dialogue Manager has to be able, on the one hand, to make decisions on the basis of the current state of the dialog (e.g., what speech act is most plausible during a certain phase of the conversation); on the other hand, it also needs to interact with the Plan Reasoner, and use the information thus obtained to make decisions. This effectively forced the Dialogue Manager and the Plan Reasoner to adopt a unified ontology.

In fact, although at the moment most modules do not access the information stored by modules 'further down the pipeline' (e.g., the Parser need not access information maintained by the Plan Reasoner), it has become fairly clear that eventually this will have to change: e.g., the SAD module needs to know the results of Speech Act interpretation, as these affect discourse segmentation and therefore the availability of anaphoric references. It is likely that a tighter integration among formalisms will be reached in future versions of TRAINS.

This said, it is by no means clear that in the future all modules of the system will be able to access a unified database of information. In order for that to happen, it is not enough to adopt a single representation formalism; it is also necessary that all modules adopt the same ontological perspective, and it's by no means clear that that's possible, or even desirable. A case in point is the representation of the plan. From the perspective of the reference resolution module, the plan is a situation just like any other—an object that contains constituents that can be referred to. This is not, however, the most useful way for the Plan Reasoner to look at the plan; the 'argument' perspective is more fruitful.

References

[Allen and Miller, 1991] Allen, J. F. and Miller, B. W. 1991. The RHET system: A sequence of self-guided tutorials. Technical Report 325, University of Rochester - Computer Science.

[Allen *et al.*, 1994] Allen, J. F.; Schubert, L. K.; Ferguson, G.; Heeman, P.; Hwang, C. H.; Kato, T.; Light, M.; Martin, N.; Miller, B.; Poesio, M.; and Traum, D. R. 1994. The TRAINS project: A case study in building a conversational planning agent. In preparation.

[Allen, 1991] Allen, J. F. 1991. Time and planning. In J. Allen, R. Pelavin H. Kautz and Tenenberg, J., editors 1991, *Reasoning About Plans*. Morgan Kaufmann.

[Ferguson and Allen, 1994] Ferguson, G. and Allen, J. F. 1994. Arguing about plans: Plan representation and reasoning for mixed-initiative planning. In *Proceedings of the Second International Conference on AI Planning Systems (AIPS-94)*, Chicago, IL. To appear.

[Hwang and Schubert, 1993a] Hwang, C. H. and Schubert, L. K. 1993a. Episodic logic: A comprehensive, natural representation for language understanding. *Minds and Machines* 3:381–419.

[Hwang and Schubert, 1993b] Hwang, C. H. and Schubert, L. K. 1993b. Episodic logic: A situational logic for natural language processing. In Aczel, P.; Israel, D.; Katagiri, Y.; and Peters, S., editors 1993b, *Situation Theory and its Applications, v.3*. CSLI. 303–338.

[Kamp and Reyle, 1993] Kamp, H. and Reyle, U. 1993. *From Discourse to Logic*. D. Reidel, Dordrecht.

[Poesio, 1994] Poesio, M. 1994. *Discourse Interpretation and the Scope of Operators*. Ph.D. Dissertation, University of Rochester, Department of Computer Science, Rochester, NY.

[Traum *et al.*, 1994] Traum, D. R.; Allen, J. F.; Ferguson, G.; Heeman, P. A.; Hwang, C. H.; Kato, T.; Martin, N.; Poesio, M.; and Schubert, L. K. 1994. Integrating natural language understanding and plan reasoning in the TRAINS-93 conversation system. In *Working Notes AAAI Spring Symposium on Natural Language Understanding in Integrated Systems*, Stanford.