ITS Tools for Natural Language Dialogue: A Domain-Independent Parser and Planner

Reva Freedman, Carolyn Penstein Rosé, Michael A. Ringenberg, and Kurt VanLehn*

Learning Research and Development Center
University of Pittsburgh
Pittsburgh, PA 15260
{freedrk, rosecp, mringenb, vanlehn}@pitt.edu
http://www.pitt.edu/~circle

Abstract. The goal of the Atlas project is to increase the opportunities for students to construct their own knowledge by conversing (in typed form) with a natural language-based ITS. In this paper we describe two components of Atlas—APE, the integrated planning and execution system at the heart of Atlas, and CARMEL, the natural language understanding component. These components have been designed as domain-independent rule-based software, with the goal of making them both extensible and reusable. We illustrate the use of CARMEL and APE by describing Atlas-Andes, a prototype ITS built with Atlas using the Andes physics tutor as the host.

1 Motivation

The goal of the Atlas project is to enable the involvement of students in a more active style of learning by engaging them in a typed dialogue with an ITS. This dialogue can include both natural language and GUI actions. In this paper we motivate the use of dialogue in intelligent tutoring. We also describe resources developed on the Atlas project that are available for use on tutoring projects interested in including dialogue capabilities in their applications. The two key domain-independent components described here are APE, the Atlas Planning Engine, and CARMEL, the natural language understanding component. APE is a "just-in-time" planner specialized for easy construction and rapid generation of hierarchically organized dialogues. CARMEL is a general purpose engine for language understanding composed of robust and efficient algorithms for parsing, semantic interpretation, and repair. We explain how we used these components to build a prototype for a new tutor, Atlas-Andes, that adds a dialogue capability to the existing Andes physics tutor.

* This research was supported by NSF grant number 9720359 to CIRCLE, the Center for Interdisciplinary Research on Constructive Learning Environments at the University of Pittsburgh and Carnegie-Mellon University.

Collaborative dialogue between student and tutor is a well-documented prominent component of effective human tutoring [1-3]. A recent corpus study of reflective follow-up dialogues [4] demonstrates the potential for natural language dialogue to enhance the ability of tutoring systems to effectively diagnose student misconceptions. Furthermore, recent research on student self-explanations supports the view that when students express their thinking in words it enhances their learning [5,6]. Students learn more effectively when they are forced to construct knowledge for themselves.

Without natural language dialogue, the best remediation tools available to tutoring systems are hint sequences. Hints are a unidirectional form of natural language: the student can't take the initiative or ask a question. In addition, there is no way for the system to lead the student through a multi-step directed line of reasoning or to ask the student a question, except via a list of pre-coded answers. As a result, there is no way to use some of the effective rhetorical methods used by skilled human tutors, such as analogy and *reductio ad absurdum*. Thus, the use of natural language dialogue allows us to extend the tutor's repertoire to include the types of remediation subdialogues seen in corpus studies.

An elevator slows to a stop from an initial downward velocity of 10 ms in 2 sec. A passenger in the elevator is holding a 3 kg package by a string. What is the tension in the string?

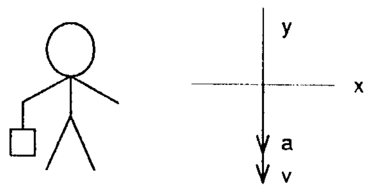

Fig. 1. Sample problem from the Andes physics tutor

2 Natural Language Dialogue in Atlas-Andes

Atlas-Andes is a dialogue-extended version of the Andes [7] tutoring system using the capabilities offered by the Atlas tool set. Andes is an ITS whose domain is first-year college physics. It tracks the student as the latter attempts to solve a problem. If the student gets stuck or deviates too far from a correct solution path, it provides hints and other assistance. The left-hand side of Fig. 1 shows a typical Andes problem with its accompanying drawing. On the right is seen the student's partially completed free-body diagram, or sketch of the vectors involved.

In this example, the elevator is decelerating, so the acceleration vector should face the opposite direction from the velocity vector.[1] Andes responds to this frequent student error by turning the incorrect item red. If requested, it will

[1] If the acceleration vector went in the same direction as the velocity vector, the elevator would speed up and smash into the ground.

provide the first hint of a sequence, in this case "Think about the direction of the acceleration vector." Follow-up hints include further information about the direction of acceleration and, eventually, the answer. The prototype Atlas-Andes system replaces these hints by a choice of generated subdialogues, such as the one shown in Fig. 2.

```
S: <draws acceleration vector in same direction as velocity>
T: What is the definition of acceleration?
S: I don't have any clue.
T: OK, let's try this. If a car was driving along east, which way would
   you have to push on it to make it stop?
S: West.
T: Exactly, the opposite direction. What does that tell you about the
   direction of acceleration?
S: It goes west too.
T: Right. The net force goes the opposite direction, and so does the
   acceleration. Try to draw the acceleration vector again now.
S: <draws acceleration vector correctly>
```

Fig. 2. Example of generated dialogue

This example shows some prominent features of both APE and CARMEL. From the planning point of view, after the student's initial response, Atlas-Andes was able to drop an unpromising line of attack and try a different one. Later, it was able to give specific responses to statements by the student. From the language understanding point of view, we can see that Andes-Atlas can effectively interpret student responses, even idioms like "I don't have a clue."

3 Implementation of a Prototype Tutor

In this section we describe how a pre-existing tutoring system can be extended to offer dialogue capabilities using Atlas. Figure 3 illustrates the architecture of the resulting extended system. While we focus our discussion on the prototype Atlas-Andes system, the same approach could be used to add dialogue capabilities to a wide range of tutoring systems. In Fig. 3 we see that students interact with the dialogue-extended system through the User Interface Manager, which interprets mouse clicks and key presses. GUI actions are then channeled through the GUI Interpreter which interprets them and stores a representation of the interpreted input for the Tutorial Planner (APE), described in Section 4. Natural language input is channeled through the Input Understander (CARMEL), which interprets the student's input. Just as the GUI Interpreter does, it stores a representation of the interpreted natural language input for the Tutorial Planner. The Tutorial Planner then uses the input representation, as well as data from the host system (in this case Andes) and other sources, to formulate a response which it sends back to the student via the User Interface Manager.

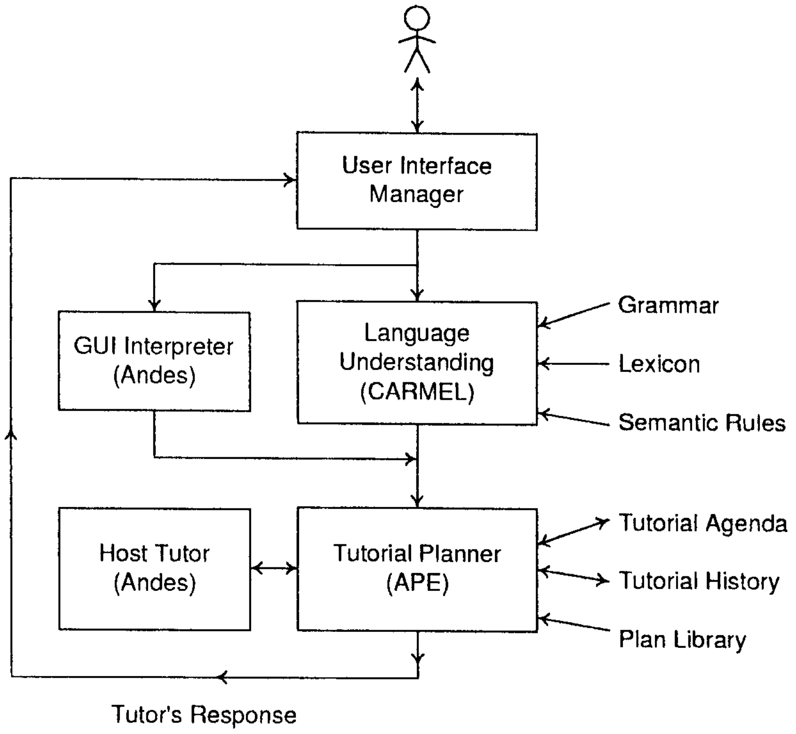

Fig. 3. Architecture of Atlas-Andes

Two domain-specific knowledge sources are required to apply the Atlas tools (APE and CARMEL) to a new domain, namely a plan library to guide the Tutorial Planner and semantic mapping rules to guide the Input Understander. A corpus of transcribed, spoken human-human dialogues using two experienced tutors and 20 students attempting to solve physics problems informed the development of the prototype Atlas-Andes system. The prototype system contains 21 semantic mapping rules and a plan library of approximately 100 plan operators, including roughly equal numbers of operators for dialogue creation, responding to specific student misconceptions, and handling domain-independent dialogue issues. In addition to API and GUI handling, the latter category includes general tutoring policies such as whether the student should be allowed to take the initiative and return to the GUI without finishing a subdialogue in process. With this knowledge the system can generate a large number of variations of the dialogue in Fig. 2 as well as selected examples of other ways of teaching about the direction of acceleration, such as the mini-*reductio* in Fig. 4. Thus the resulting system has the ability to tailor its approach to a wide variety of student inputs. Operators are selected based on historical information gathered by the tutor (discourse/interaction history), information about the current situation (the tutor's current goal and the student's latest response), and domain

knowledge. As an example of the latter, if a student draws an acceleration vector which is incorrect but not opposite to the velocity vector, a different response will be generated.

In the remainder of the paper, we will discuss the APE tutorial planner and the CARMEL input understander in greater depth.

4 APE: The Atlas Tutorial Planner

Planning is required in dialogue-based ITSs in order to ensure that a coherent conversation ensues as the tutor's pedagogical goals are accomplished. If the system just responds to student actions, the resulting conversation will not necessarily be coherent, and the tutor has no way to ensure that its own teaching goals are met. Although Wenger [8] wrote in 1987 that using a global planner to control an ITS would be too inefficient, developments in reactive planning have made this goal a realistic possibility.

One cannot plan a conversation in advance unless the student's responses are classified into a small number of categories, and even then it would be wasteful. Furthermore, depending on the quality of the student's answers, one might need to change the plan during the conversation. For these reasons we work with partial plans that are expanded and refined only as needed. This style of planning is often called ***reactive planning*** [9, 10].

For adding dialogue to ITSs, we have developed a reactive planner called APE (Atlas Planning Engine) that is specialized for dialogue. In a previous study [11], we showed how modeling human-human tutorial dialogues according to the hierarchical structure of task-oriented dialogues [12] can make them tractable for plan-based generation. In the tutoring dialogues we have studied, a main building block of the discourse hierarchy, corresponding to the transaction level in Conversation Analysis [13], matches the *tutoring episode* defined by VanLehn [14]. A tutoring episode consists of the turns necessary to help the student accomplish one correct problem-solving step, e. g. to make one correct entry on a graphical interface. Our planner makes it convenient to satisfy local goals without disturbing the basic hierarchical structure.

Figure 4 shows a sample plan operator from Atlas-Andes. For legibility, we have shown the key elements in English instead of in Lisp.

To initiate a planning session, the user invokes the planner with an initial goal. The system searches its operator library to find all operators whose goal field matches the next goal on the agenda and whose filter conditions and preconditions are satisfied. Goals are represented by first-order logic without quantifiers and are matched using full unification. Since APE is intended especially for the generation of hierarchically organized task-oriented discourse, operators have multi-step recipes. When a match is found, the matching goal is removed from the agenda and replaced by the steps in the recipe. This operation is repeated until a primitive (non-decomposable) step is reached. If the primitive step corresponds to a question, the tutor asks the question and ends its turn. If the

```
(def-operator handle-same-direction
  :goal (...)
  :filter  (...)
  :precond (...)
     ; We have asked a question about acceleration
     ; ... and the student has given an answer
     ; ... from which we can deduce that he/she thinks acceleration and
         velocity go in the same direction
     ; and we have not given the explanation below yet
  :recipe (...)
     ; Tell the student: "But if the acceleration went the same direction
         as the velocity, then the elevator would be speeding up."
     ; Mark that we are giving this explanation
     ; Tell the student that the tutor is requesting another answer
     ;   ("Try again.")
     ; Edit the agenda so that tutor is ready to receive another answer
  :hiercx  ())
```

Fig. 4. Sample plan operator

primitive step corresponds to a statement, the tutor utters the statement but continues to plan, allowing the generation of multi-sentence turns.

To tailor the tutor's responses to the student as much as possible, one needs the ability to change plans during a conversation. This ability is provided in APE through the use of three types of recipe steps that can update the agenda. APE can skip the remainder of a strategy if circumstances have changed; it can replace a strategy with another strategy that has the same goal; and it can replace a sequence of goals at the top of the agenda. The last type is especially useful for responding to a student utterance without disturbing the global plan. In this way our approach differs from that of Vassileva [15]. Her work, based on AND-OR graphs, uses a separate set of rules for reacting to unexpected events.

A second way to tailor the tutor's response to the student is to take context into account before choosing a response. APE provides this ability in two ways. The "hierarchical context" or *hiercx* slot of an operator, shown in the last line of Fig. 4, provides a way for the planner to be aware of the goal hierarchy in which a decomposition is proposed. Additionally, operators can update and test predicates in a dynamic knowledge base.

APE communicates with the host system via an API. It obtains information from the world—the GUI interface, the natural language understanding component (CARMEL), and the host tutoring system—through preconditions on its plan operators. It returns information and action requests through recipe steps that update its knowledge base and execute external actions. Further details about the APE planner can be found in [16], and a deeper treatment of the role of reactive planning in dialogue generation can be found in [17].

Many previous dialogue-based ITSs have been implemented with finite-state machines, either simple or augmented. In the most common finite-state model,

each time the human user issues an utterance, the processor reduces it to one of a small number of categories. These categories represent the possible transitions between states. There are several problems with this approach. First, it limits the richness of the student's input that can be appreciated. With APE, on the other hand, the author can write arbitrarily complex predicates, evaluable at run time, to define a class of input. Second, one can only take history and context into account by expanding the number of states, putting an arbitrary restriction on the amount of context or depth of conversational nesting that can be considered. Third, the finite-state approach misses the significant generalization that tutorial dialogues are hierarchical: larger units contain repeated instances of the same smaller units in different sequences and instantiated with different values. Finally, the finite-state machine approach does not allow the author to drop one line of attack and replace it by another without hard-coding every possible transition, thus limiting the tutor's ability to tailor its responses.

The prototype Atlas-Andes system described above shows that APE permits one not only to build more sophisticated ITSs but to build them faster. Since the domain-specific tutorial strategies are built from a small vocabulary of lower-level operators, there is a considerable economy of scale when expanding such a prototype to a full-scale tutoring system. Additionally, many of the operators that express general tutoring policies and conversational strategies are domain-independent and do not need to be repeated when expanding domain coverage.

5 CARMEL: The Atlas Input Understander

The task of the Atlas input understander is to extract relevant information from student explanations and other natural language input to pass back to the planner. This information can take the form of single atomic values or collections of flat propositional clauses, depending upon what the planner requires in specific contexts. In either case, CARMEL, the Core component for Assessing the Meaning of Explanatory Language, is used to parse the student input onto a feature structure representation that contains both syntactic and semantic information. Domain specific pattern matchers called semantic mapping rules are then used to match against particular patterns of features in order to identify and extract the needed information.

The overarching goal behind the design of the Atlas input understander is to facilitate the rapid development of robust natural language understanding interfaces for multiple domains. While interest in language understanding interfaces for tutoring systems has grown in recent years, progress towards making such interfaces commonplace has been greatly hindered by the tremendous time, effort, and expertise that is normally required for such an endeavor. Our long term goal is to build a tool set to semi-automate the process by applying machine learning techniques that require system developers only to annotate corpora with information pertinent to tutoring, thus insulating them from the underlying linguistic aspects of the development. At the heart of our design is the CARMEL core language understanding component, which is available for use on other tutoring projects.[2] Its underlying robust understanding technology [18–20] has already proven successful in the context of a large scale multi-lingual speech-to-speech translation system [21, 22].

CARMEL provides a broad foundation for language understanding. It is composed of a broad coverage English syntactic parsing grammar and lexicon; robust and efficient algorithms for parsing, semantic interpretation, and repair; and a formalism for entering idiomatic and domain-specific semantic knowledge. Current dialogue-based tutoring systems, such as CIRCSIM-Tutor [23] and AutoTutor [24], rely on shallow processing strategies to handle student input. This technology has so far proven effective for efficiently processing short student answers and for evaluating content based on inclusion of relevant vocabulary. In contrast, the goal of CARMEL is to support a deeper level of analysis in order to identify arbitrarily complex relationships between concepts within longer student answers.

Our approach is to achieve the most complete deep analysis possible within practical limits by relaxing constraints only as needed. CARMEL first attempts to construct analyses that satisfy both syntactic and semantic well-formedness conditions. A spelling corrector [25] is integrated with the lexical look-up mechanism in order to robustly recognize the student's intended input in the face of typos and spelling errors. The robust parser [19] has the ability to efficiently relax syntactic constraints as needed and as allowed by parameterized flexibility settings. For sentences remaining beyond the coverage of its syntactic knowledge, a repair stage [18], relying solely on semantic constraints compiled from a meaning representation specification, is used to assemble the pieces of a fragmentary parse. Thus, robustness techniques are applied at each stage in processing student input in order to address the wide variety of phenomena that make language understanding challenging.

In a recent evaluation of CARMEL's syntactic coverage, we measured the parser's ability to robustly analyze student input by testing it on a subset of our corpus of tutoring dialogues that had not been used for development of the prototype. The test corpus contained 50 student sentences and 50 multi-sentence student turns randomly extracted from the full corpus. The utterances ranged in length from 1 to 20 words, with an average length of 8 words per utterance. The parser was able to construct analyses covering 87% of the corpus when a high flexibility setting was used, taking on average .1 seconds per sentence.

When the parser is unable to construct an analysis of a sentence that deviates too far from the grammar's coverage, a fragmentary analysis is passed on to the repair module that quickly assembles the fragments [18]. Our approach to repair is unique in that no hand-coded repair rules are required as in other approaches to recovery from parser failure [26, 27]. A recent evaluation demonstrates that CARMEL's repair stage can increase the number of acceptable interpretations produced by between 3% (when using a high flexibility setting) and 9% (when a restricted flexibility setting is used), taking on average only .3 seconds per sentence.

[2] Interested parties should contact Carolyn Rosé at rosecp@pitt.edu.

6 Conclusions

One goal of the Atlas project is to develop reusable software for implementing natural-language based ITSs. In this paper we described CARMEL and APE, the parser and planner, respectively, for Atlas. We illustrated this work with an example from Atlas-Andes, a prototype physics tutor built using the Atlas framework. We showed how using these components could enable not only better tutoring but reduced authoring time as well.

7 Acknowledgments

We are grateful to Abigail Gertner for her generous assistance with the Andes system. Mohammed Elmi and Michael Glass of Illinois Institute of Technology provided the spelling correction code. Pamela Jordan provided constructive commentary on the manuscript.

References

1. Merrill, D.C., Reiser, B.J., Landes, S.: Human tutoring: Pedagogical strategies and learning outcomes (1992) Paper presented at the annual meeting of the American Educational Research Association.
2. Fox, B.A.: The Human Tutorial Dialogue Project: Issues in the design of instructional systems. Hillsdale, NJ: Erlbaum (1993)
3. Graesser, A.C., Person, N.K., Magliano, J.P.: Collaborative dialogue patterns in naturalistic one-to-one tutoring. Applied Cognitive Psychology **9** (1995) 495–522
4. Rosé, C.P.: The role of natural language interaction in electronics troubleshooting. In: Proceedings of the Energy Week Conference and Exhibition, Houston (1997)
5. Chi, M.T.H., Bassok, M., Lewis, M.W., Reimann, P., Glaser, R.: Self-explanations: How students study and use examples in learning to solve problems. Cognitive Science **13** (1989) 145–182
6. Chi, M.T.H., de Leeuw, N., Chiu, M.H., LaVancher, C.: Eliciting self-explanations improves understanding. Cognitive Science **18** (1994) 439–477
7. Gertner, A., VanLehn, K.: Andes: A coached problem solving environment for physics. In: Proceedings of the Fifth International Conference on Intelligent Tutoring Systems (ITS '00), Montreal (2000)
8. Wenger, E.: Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge. San Mateo, CA: Morgan Kaufmann (1987)
9. Georgeff, M.P., Ingrand, F.F.: Decision-making in an embedded reasoning system. In: Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI '89), Detroit (1989) 972–978
10. Wilkins, D., Myers, K., Lowrance, J., Wesley, L.: Planning and reacting in uncertain and dynamic environments. Journal of Experimental and Theoretical Artificial Intelligence **7** (1995) 121–152
11. Kim, J., Freedman, R., Evens, M.: Responding to unexpected student utterances in CIRCSIM-Tutor v. 3: Analysis of transcripts. In: Proceedings of the Eleventh Florida Artificial Intelligence Research Symposium (FLAIRS '98), Sanibel Island, Menlo Park: AAAI Press (1998) 153–157

12. Grosz, B.J., Sidner, C.L.: Attention, intentions, and the structure of discourse. Computational Linguistics **12** (1986) 175–204
13. Sinclair, J.M., Coulthard, R.M.: Towards an Analysis of Discourse: The English Used by Teachers and Pupils. London: Oxford University Press (1975)
14. VanLehn, K., Siler, S., Murray, C., Baggett, W.: What makes a tutorial event effective? In: Proceedings of the Twenty-first Annual Conference of the Cognitive Science Society, Madison (1998) 1084–1089
15. Vassileva, J.: Reactive instructional planning to support interacting teaching strategies. In: Proceedings of the Seventh World Conference on AI and Education (AI–ED '95), Washington, D. C., Charlottesville, VA: AACE (1995)
16. Freedman, R.: Using a reactive planner as the basis for a dialogue agent. In: Proceedings of the Thirteenth Florida Artificial Intelligence Research Symposium (FLAIRS '00), Orlando (2000)
17. Freedman, R.: Plan-based dialogue management in a physics tutor. In: Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP '00), Seattle (2000)
18. Rosé, C.P.: A framework for robust semantic interpretation. In: Proceedings of the First Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL '00), Seattle (2000)
19. Rosé, C.P., Lavie, A.: Balancing robustness and efficiency in unification augmented context-free parsers for large practical applications. In Junqua, J.C., Noord, G.V., eds.: Robustness in Language and Speech Technologies. Dordrecht: Kluwer (1999)
20. Rosé, C.P.: Robust Interactive Dialogue Interpretation. PhD thesis, School of Computer Science, Carnegie Mellon University (1997)
21. Woszcyna, M., Coccaro, N., Eisele, A., Lavie, A., McNair, A., Polzin, T., Rogina, I., Rosé, C.P., Sloboda, T., Tomita, M., Tsutsumi, J., Waibel, N., Waibel, A., Ward, W.: Recent advances in JANUS: a speech translation system. In: Proceedings of the ARPA Human Languages Technology Workshop, Princeton, NJ (1993)
22. Suhm, B., Levin, L., Coccaro, N., Carbonell, J., Horiguchi, K., Isotani, R., Lavie, A., Mayfield, L., Rosé, C.P., Dykema, C.V.E., Waibel, A.: Speech-language integration in a multi-lingual speech translation system. In: Proceedings of the AAAI Workshop on Integration of Natural Language and Speech Processing, Seattle (1994)
23. Glass, M.S.: Broadening Input Understanding in an Intelligent Tutoring System. PhD thesis, Illinois Institute of Technology (1999)
24. Wiemer-Hastings, P., Graesser, A., Harter, D., the Tutoring Research Group: The foundations and architecture of AutoTutor. In Goettl, B., Halff, H., Redfield, C., Shute, V., eds.: Intelligent Tutoring Systems: 4th International Conference (ITS '98). Berlin: Springer (1998) 334–343
25. Elmi, M., Evens, M.: Spelling correction using context. In: Proceedings of the 17th COLING/36th ACL (COLING-ACL '98), Montreal (1998)
26. Danieli, M., Gerbino, E.: Metrics for evaluating dialogue strategies in a spoken language system. In: Working Notes of the AAAI Spring Symposium on Empirical Methods in Discourse Interpretation and Generation, Stanford (1995)
27. Kasper, W., Kiefer, B., Krieger, H., Rupp, C., Worm, K.: Charting the depths of robust speech parsing. In: Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL '99), College Park (1999)