Plan-Based Dialogue Management in a Physics Tutor

Reva Freedman
Learning Research and Development Center
University of Pittsburgh
Pittsburgh, PA 15260
freedrk+@pitt.edu
http://www.pitt.edu/~freedrk

Abstract

This paper describes an application of APE (the Atlas Planning Engine), an integrated planning and execution system at the heart of the Atlas dialogue management system. APE controls a mixed-initiative dialogue between a human user and a host system, where turns in the 'conversation' may include graphical actions and/or written text. APE has full unification and can handle arbitrarily nested discourse constructs, making it more powerful than dialogue managers based on finite-state machines. We illustrate this work by describing Atlas-Andes, an intelligent tutoring system built using APE with the Andes physics tutor as the host.

1 Introduction

The purpose of the Atlas project is to enlarge the scope of student interaction in an intelligent tutoring system (ITS) to include coherent conversational sequences, including both written text and GUI actions. A key component of Atlas is APE, the Atlas Planning Engine, a "just-in-time" planner specialized for easy construction and quick generation of hierarchically organized dialogues. APE is a domain- and task-independent system. Although to date we have used APE as a dialogue manager for intelligent tutoring systems, APE could also be used to manage other types of human-computer conversation, such as an advice-giving system or an interactive help system.

Planning is an essential component of a dialogue-based ITS. Although there are many reasons for using natural language in an ITS, as soon as the student gives an unexpected response to a tutor question, the tutor needs to be able to plan in order to achieve its goals as well as respond appropriately to the student's statement. Yet classical planning is inappropriate for dialogue generation precisely because it assumes an unchanging world. A more appropriate approach is the "practical reason" approach pioneered by Bratman (1987, 1990). According to Bratman, human beings maintain plans and prefer to follow them, but they are also capable of changing the plans on the fly when needed. Bratman's approach has been introduced into computer science under the name of *reactive planning* (Georgeff and Ingrand 1989, Wilkins et al. 1995).

In this paper we discuss the rationale for the use of reactive planning as well as the use of the hierarchical task network (HTN) style of plan operators. Then we describe APE (the Atlas Planning Engine), a dialogue planner we have implemented to embody the above concepts. We demonstrate the use of APE by showing how we have used it to add a dialogue capability to an existing ITS, the Andes physics tutor. By showing dialogues that Atlas-Andes can generate, we demonstrate the advantages of this architecture over the finite-state machine approach to dialogue management.

2 Integrated planning and execution for dialogue generation

2.1 'Practical reason' and the BDI model

For an ITS, planning is required in order to ensure a coherent conversation as well as to accomplish tutorial goals. But it is impossible to plan a whole conversation in advance when the student can respond freely at every turn, just as human beings cannot plan their daily lives in advance because of possible changes in conditions. Classical planning algorithms are inappropriate because the tutor must be able to change plans based on the

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.

student's responses.

For this reason we have adopted the ideas of the philosopher Michael Bratman (1987, 1990). Bratman uses the term "practical reason" to describe his analysis since he is concerned with how to reason about practical matters. For human beings, planning is required in order to accomplish one's goals. Bratman's key insight is that human beings tend to follow a plan once they have one, although they are capable of dropping an intention or changing a partial plan when necessary. In other words, human beings do not decide what to do from scratch at each turn.

Bratman and others who have adopted his approach use a tripartite mental model that includes *beliefs*, *desires* and *intentions* (Bratman, Israel and Pollack 1988, Pollack 1992, Georgeff et al. 1998), hence the name "BDI model." Beliefs, which are uninstantiated plans in the speaker's head, are reified by the plan library. Desires are expressed as the agent's goals. Intentions, or plan steps that the agent has committed to but not yet acted on, are stored in an agenda. Thus the agent's partial plan for achieving a goal is a network of intentions. A plan can be left in a partially expanded state until it is necessary to refine it further.

2.2 Implementation via reactive planning

Bratman's approach has been elaborated in a computer science context by subsequent researchers (Bratman, Israel and Pollack 1988, Pollack 1992, Georgeff et al. 1998). Reactive planning (Georgeff and Ingrand 1989, Wilkins et al. 1995), originally known as "integrated planning and execution," is one way of implementing Bratman's model. Originally developed for real-time control of the space shuttle, reactive planning has since been used in a variety of other domains. For the Atlas project we have developed a reactive planner called APE (Atlas Planning Engine) which uses these ideas to conduct a conversation. After each student response, the planner can choose to continue with its previous intention or change something in the plan to respond better to the student's utterance.

Like most reactive planners, APE is a hierarchical task network (HTN) style planner (Yang 1990, Erol, Hendler and Nau 1994). Hierarchical decomposition asserts that each goal can be achieved via a series of subgoals instead of relying on means-end reasoning. Hierarchical decomposition is more appropriate to dialogue generation for a number of reasons. First, decomposition is better suited to the type of large-scale dialogue planning required in a real-world tutoring system, as it is easier to establish what a human speaker will say in a given situation than to be able to understand why in sufficient detail and generality to do means-end planning. Second, Hierarchical decomposition minimizes search time. Third, our dialogues are task-oriented and have a hierarchical structure (Grosz and Sidner 1986). In such a case, matching the structure of the domain simplifies operator development because they can often be derived from transcripts of human tutoring sessions. The hierarchy information is also useful in determining appropriate referring expressions. Fourth, interleaved planning and execution is important for dialogue generation because we cannot predict the human user's future utterances. In an HTN-based system, it is straightforward to implement interleaved planning and execution because one only needs to expand the portion of the plan that is about to be executed. Finally, the conversation is in a certain sense the trace of the plan. In other words, we care much more about the actions generated by the planner than the states involved, whether implicitly or explicitly specified. Hierarchical decomposition provides this trace naturally.

3 Background: the Andes physics tutor

Andes (Gertner, Conati and VanLehn 1998) is an intelligent tutoring system in the domain of first-year college physics. Andes teaches via *coached problem solving* (VanLehn 1996). In coached problem solving, the tutoring system 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, the tutoring system provides hints and other assistance.

A sample Andes problem is shown in mid-solution in Figure 1. A physics problem is given in the upper-left corner with a picture below it. Next to the picture the student has begun to sketch the vectors involved using the GUI buttons along the left-hand edge of the screen. As the student draws vectors, Andes and the student cooperatively fill in the variable definitions in the upper-right corner. Later the student will use the space below to write equations connecting the variables.

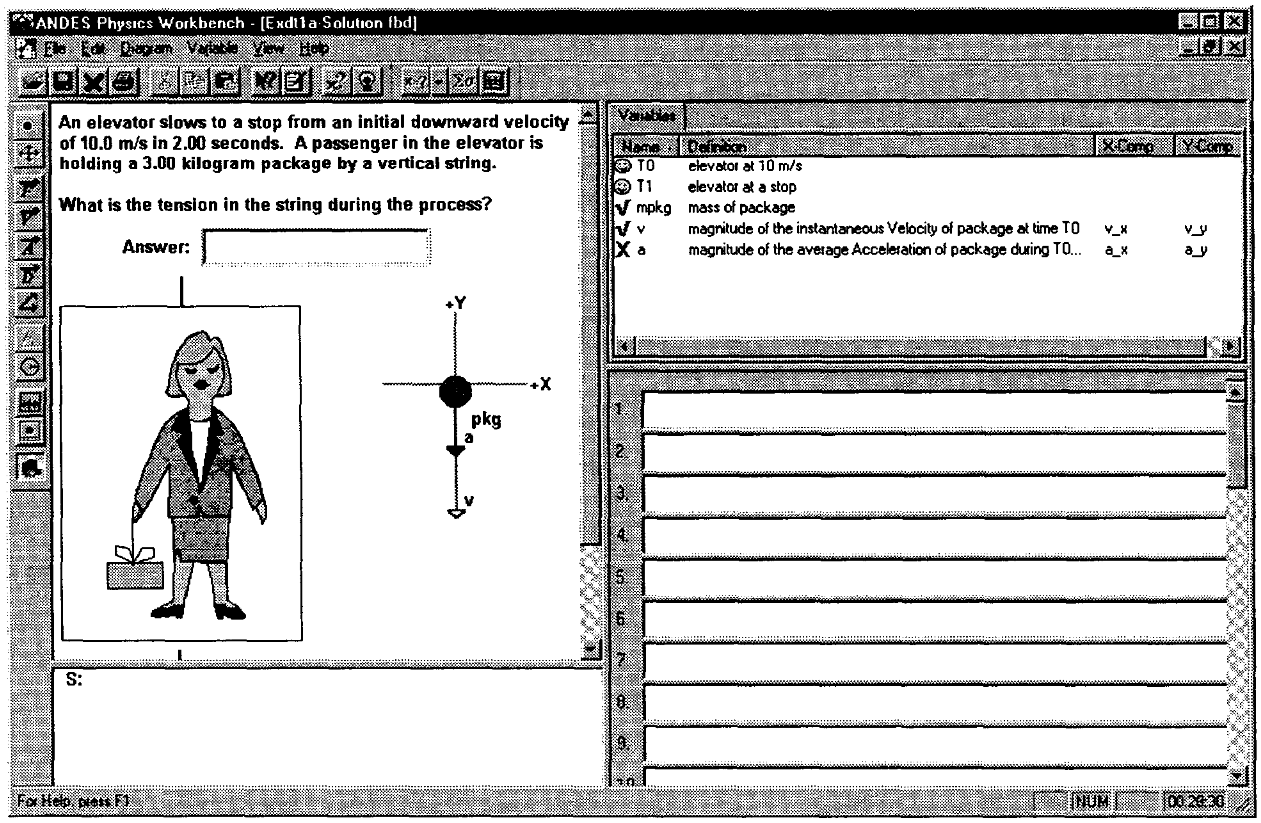
In this example, the elevator is decelerating, so the acceleration vector should face the opposite direction from the velocity vector. (If the acceleration vector went the same direction as the velocity vector, the speed of the elevator would increase and it would crash into the ground.) This is an important issue in beginning physics; it occurs in five Andes problems.

When such errors occur, Andes turns the incorrect item red and provides hints to students in the lower-left corner of the screen. A sample of these hints, shown in the order a student would encounter them, is shown in Fig. 2. But hints are an output-only 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 ask the student a question or lead the student through a multi-step directed line of reasoning. Thus there is no way to use some of the effective rhetorical methods used by skilled human tutors, such as analogy and *reductio ad absurdum*. Current psychological research suggests that active methods, where students have to answer questions, will improve the performance of tutoring systems.

4 Structure of the Atlas Planning Engine

Figure 3 shows a sample plan operator. For legibility, the key elements have been rendered in English instead of in Lisp. The *hiercx* slot provides a way for the planner to be aware of the context in which a decomposition is proposed. Items in the *hiercx* slot are instantiated and added to the transient database only so long as the operator which spawned them is in the agenda.

To initiate a planning session, the user invokes the planner with an initial goal. The system searches the operator library to find all operators whose goal field matches the next goal on the agenda and whose filter conditions and precon-


ANDES Physics Workbench - [Exdt1a-Solution fbd]
File Edit Diagram Variable View Help
Variables
+Y
+X
pkg
a
v
S:
For Help, press F1
NUM
00:28:30

An elevator slows to a stop from an initial downward velocity of 10.0 m/s in 2.00 seconds. A passenger in the elevator is holding a 3.00 kilogram package by a vertical string.

What is the tension in the string during the process?

Answer:

Name	Definition	X-Comp	Y-Comp
T0	elevator at 10 m/s		
T1	elevator at a stop		
✓ mpkg	mass of package		
✓ v	magnitude of the instantaneous Velocity of package at time T0	v_x	v_y
X a	magnitude of the average Acceleration of package during T0...	a_x	a_y

Figure 1: Screen shot of the Andes physics tutor

S: ⟨draws acceleration vector in same direction as velocity⟩
T: Wrong.
S: What's wrong with that?
T: Think about the direction of the acceleration vector.
S: Please explain further.
T: Remember that the direction of acceleration is the direction of the change in velocity.
S: Please explain further.
T: The direction of the acceleration vector is straight up.
S: ⟨draws acceleration vector correctly⟩

Figure 2: Andes hint sequence formatted as dialogue

ditions are satisfied. Goals are represented in first-order logic without quantifiers and matched via unification. Since APE is intended especially for generation of hierarchically organized task-oriented discourse, each operator has a multi-step recipe in the style of Wilkins (1988). When a match is found, the matching goal is removed from the agenda and is replaced by the steps in the recipe. APE has two kinds of primitive actions; one ends a turn and the other doesn't.

From the point of view of discourse generation, the most important APE recipe items are those allowing the planner to change the agenda when necessary. These three types of recipe items make APE more powerful than a classical planner.

• *Fact:* Evaluate a condition. If false, skip the rest of the recipe. *Fact* is used to allow run-time decision making by bypassing the rest of an operator when circumstances change during its execution. *Fact* can be used with *retry-at* to implement a loop just as in Prolog.

• *Retry-at:* The purpose of *retry-at* is to allow the planner to back up to a choice point and make a new decision. It removes goals sequentially from the top of the agenda, a full operator at a time, until the supplied argument is false. Then it restores the parent goal of the last operator removed, so that further planning can choose a new way to achieve it. *Retry-at* implements a Prolog-like choice of alternatives, but it differs from backtracking in that the new operator is chosen based on conditions that apply when the retry operation is executed, rather than on a list of possible operators formed when the original operator was chosen. For *retry-at* to be useful, the author must provide multiple operators for the same goal. Each operator must have a set of preconditions enabling it to be chosen at the appropriate time.

• *Prune-replace:* The intent of *prune-replace* is

```
(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 s/he thinks accel. 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 tutor is requesting another answer ("Try again.")
     ; Edit the agenda (using prune-replace) so that responding to another
         answer is at the top of the agenda
  :hiercx  ())
```

Figure 3: Sample plan operator

to allow the planner to remove goals from the agenda based on a change in circumstances. It removes goals sequentially from the top of the agenda, one at a time, until the supplied argument becomes false. Then it replaces the removed goals with an optional list of new goals. *Prune-replace* allows a type of decision-making frequently used in dialogue generation. When a conversation partner does not give the expected response, one would often like to remove the next goal from the agenda and replace it with one or more replacement goals. *Prune-replace* implements a generalized version of this concept.

APE is domain-independent and communicates with a host system via an API. As a partner in a dialogue, it needs to obtain information from the world as well as produce output turns. Preconditions on plan operators can be used to access information from external knowledge sources. APE contains a recipe item type that can be used to execute an external program such as a call to a GUI interface. APE also has recipe items allowing the user to assert and retract facts in a knowledge base. Further details about the APE planner can be found in (Freedman, 2000).

5 Implementation of Atlas-Andes

5.1 Architecture of Atlas-Andes

The first system we have implemented with APE is a prototype Atlas-Andes system that replaces the hints usually given for an incorrect acceleration vector by a choice of generated subdialogues. Figure 4 shows the architecture of Atlas-Andes; any other system built with APE would look similar. Robust natural language understanding in Atlas-Andes is provided by Rosé's CARMEL system (Rosé 2000); it uses the spelling correction algorithm devised by Elmi and Evens (1998).

5.2 Structure of human tutorial dialogues

In an earlier analysis (Kim, Freedman and Evens 1998) we showed that a significant portion of human-human tutorial dialogues can be modeled with the hierarchical structure of task-oriented dialogues (Grosz and Sidner 1986). Furthermore, a main building block of the discourse hierarchy, corresponding to the transaction level in Conversation Analysis (Sinclair and Coulthard 1975), matches the *tutoring episode* defined by VanLehn et al. (1998). A tutoring episode consists of the turns necessary to help the student make one correct entry on the interface.

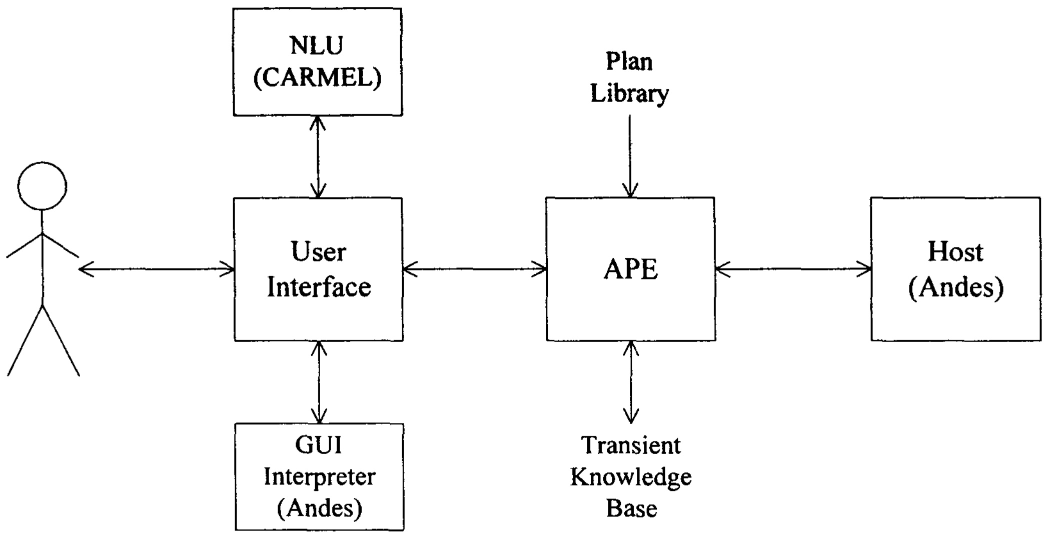


Figure 4: Interface between Atlas and host system

To obtain empirical data for the Atlas-Andes plan operators, we analyzed portions of a corpus of human tutors helping students solve similar physics problems. Two experienced tutors were used. Tutor A was a graduate student in computer science who had majored in physics; tutor B was a professional physics tutor.

The complete corpus contained solutions to five physics problems by 41 students each. We analyzed every tutoring episode dealing with the acceleration vector during deceleration, totaling 29 examples divided among 20 students and both tutors. The tutors had very different styles. Tutor A tended to provide encouragement rather than content, making those transcripts less useful for deriving an information-based approach. Tutor B used an information-based approach, but after one wrong answer tended to complete the solution as a monologue. Largely following tutor B's approach to sequence and content, we isolated six ways of teaching the student about direction of acceleration.

5.3 Sample output and evaluation

Figure 5 shows an example of text that can be generated by the Atlas-Andes system, showing an analogy-based approach to teaching this content. The operator library used to generate this text could generate a combinatorially large number of versions of this dialogue as well as selected examples of other ways of teaching about direction of acceleration.

This operator library used to generate this text contained 111 plan operators, divided as follows:

Tutoring schemata	4	4%
Switching between schemata	5	4%
API and GUI handling	33	30%
Answer handling	35	31%
Domain-dep. lex. insertion	24	22%
Domain-indep. lex. insertion	10	9%
TOTAL	111	100%

We are currently working on components that will allow us to increase the number of physics concepts covered without a corresponding increase in the number of operators. The schema switching operators prevent the tutor from repeating itself during a physics problem. They could be reduced or eliminated by a general discourse history component that tutoring schema operators could refer to. Domain-dependent lexical insertion refers to the choice of lexical items such as *car* and *east* in the sample dialogue, while domain-independent lexical insertion refers to items such as *OK* and *exactly*. Both categories could be eliminated, or at least severely reduced, through the use of a text realization package. Together that would provide a one-third reduction in the number of operators needed. As the set of API and GUI handling operators is fixed, that would reduce by half the number of application operators needed.

The largest remaining category of operators is the answer handlers. These operators handle a variety of answers for each of the five questions that the system can ask. The answers we recognize include categories such as "don't know" as well as specific answers (e.g. a direction perpendicular to the correct answer) which we recognize because the tutor has specific replies for them. In order to reduce the number of

S: ⟨draws acceleration vector in same direction as velocity⟩
T: What is the definition of acceleration?
S: Don't know.
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. So 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⟩

Figure 5: Example of generated dialogue

operators further, we must investigate more general methods of handling student errors. In particular, we plan to investigate error-classifying predicates that apply to more than one question as well as the use of intention-based predicates. Since the system only covers one rule of physics, albeit in a variety of ways, we plan to make some of these efficiency improvements before adding new rules of physics and testing it with users.

Preconditions for the operators in the plan library utilize discourse or interaction history, the current goal hierarchy, recent information such as the tutor's current goal and the student's latest response, shared information such as a model of objects on the screen, and domain knowledge. As an example of the latter, if the student draws an acceleration vector which is incorrect but not opposite to the velocity vector, a different response will be generated.

5.4 Discussion

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. Thus history can be stored, and context considered, only by expanding the number of states. This approach puts an arbitrary restriction on the amount of context or depth of conversational nesting that can be considered. More importantly, it misses the significant generalization that these types of dialogues are hierarchical: larger units contain repeated instances of the same smaller units in different sequences and instantiated with different values. Furthermore, 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.

It is also clear that the dialogue-based approach has many benefits over the hint-sequence approach. In addition to providing a multi-step teaching methods with new content, it can respond flexibly to a variety of student answers at each step and take context into account when generating a reply.

6 Related work

Wenger (1987), still the chief textbook on ITSs, states that using a global planner to control an ITS is too inefficient to try. This is no longer true, if indeed it ever was. Vassileva (1995) proposes a system based on AND-OR graphs with a separate set of rules for reacting to unexpected events. Lehuen, Nicolle and Luzzati (1996) present a method of dialogue analysis that produces schemata very similar to ours. Earlier dialogue-based ITSs that use augmented finite-state machines or equivalent include CIRCSIM-Tutor (Woo et al. 1991, Zhou et al. 1999) and the system described by Woolf (1984). Cook (1998) uses levels of finite-state machines. None of these systems provides for predicates with variables or unification.

7 Conclusions

In this paper we described APE, an integrated planner and execution system that we have implemented as part of the Atlas dialogue manager. APE uses HTN-style operators and is based on reactive planning concepts. Although APE is intended largely for use in domains with hierarchical, multi-turn plans, it can be used to implement any conversation-based system, where turns in the 'conversation' may include graphical actions and/or text. We illustrated the use of APE with an example from the Atlas-Andes physics tutor. We showed that previous models based on finite-state machines are insufficient to handle the nested subdialogues and abandoned partial subdialogues that occur in practical applications. We showed how APE generated a sample dialogue that earlier systems could not handle.

Acknowledgments

We thank Abigail Gertner for her generous assistance with the Andes system, and Michael Ringenberg for indispensible programming support. Carolyn Rosé built the CARMEL natural language understanding component. Mohammed Elmi and Michael Glass of Illinois Institute of Technology provided the spelling correction code. We thank Pamela Jordan and the referees for their comments.

References

Bratman, M. E. 1987. *Intentions, Plans, and Practical Reason.* Cambridge, MA: Harvard.

Bratman, M. E. 1990. What is Intention? In P. R. Cohen, J. Morgan and M. E. Pollack, *Intentions in Communication.* Cambridge, MA: MIT Press.

Bratman, M. E., Israel, D. J. and Pollack, M. E. 1988. Plans and Resource-Bounded Practical Reasoning. *Computational Intelligence* 4(4): 349–355.

Cook, J. 1998. Knowledge Mentoring as a Framework for Designing Computer-Based Agents for Supporting Musical Composition Learning. PhD. diss., Computing Department, The Open University.

Elmi, M. A. and Evens, M. W. 1998. Spelling Correction using Context. In Proceedings of the 17th COLING/36th ACL (COLING–ACL '98), Montreal.

Erol, K., Hendler, J. and Nau, D. S. 1994. HTN Planning: Complexity and Expressivity. In Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI '94), Seattle.

Freedman, R. 2000 (to appear). Using a Reactive Planner as the Basis for a Dialogue Agent. In Proceedings of the Thirteenth Florida Artificial Intelligence Research Symposium (FLAIRS '00), Orlando.

Gertner, A. S., Conati, C. and VanLehn, K. 1998. Procedural Help in Andes: Generating Hints Using a Bayesian Network Student Model. In Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI '98), Madison.

Georgeff, M. P. and Ingrand, F. F. 1989. Decision-Making in an Embedded Reasoning System. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI '89), Detroit.

Georgeff, M. P., Pell, B., Pollack, M. E., Tambe, M. and Wooldridge, M. 1998. The Belief-Desire-Intention Model of Agency. In N. Jenning, J. Muller, and M. Wooldridge (Eds.), *Intelligent Agents V.* Springer.

Grosz, B. J. and Sidner, C. L. 1986. Attention, Intentions, and the Structure of Discourse. *Computational Linguistics* 12(3): 175–204.

Kim, J., Freedman, R. and Evens, M. 1998. 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.

Lehuen, J., Nicolle, A. and Luzzati, D. 1996. Un modèle hypothético-expérimental dynamique pour la gestion des dialogues homme-machine. In Actes du dixième congrès de reconnaissance des formes et intelligence artificielle (RFIA '96), Rennes.

Pollack, M. E. 1992. The Uses of Plans. *Artificial Intelligence* 57(1): 43–69.

Rosé, C. P. 2000. 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).

Sinclair, J. M. and Coulthard, R. M. 1975. *Towards an Analysis of Discourse: The English Used by Teachers and Pupils.* London: Oxford University Press.

VanLehn, K. 1996. Conceptual and Meta Learning during Coached Problem Solving. In *Intelligent Tutoring Systems: Third International Conference (ITS '96)*, Montreal. Berlin: Springer. LNCS 1086.

VanLehn, K., Siler, S., Murray, C. and Baggett, W. 1998. What Makes a Tutorial Event Effective? In Proceedings of the Twenty-first Annual Conference of the Cognitive Science Society, Madison. Hillsdale, NJ: Erlbaum.

Vassileva, J. 1995. 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.

Wenger, E. 1987. *Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge.* San Mateo, CA: Morgan Kaufmann.

Wilkins, D. 1988. *Practical Planning: Extending the Classical AI Planning Paradigm.* San Mateo, CA: Morgan Kaufmann.

Wilkins, D., Myers, K., Lowrance, J. and Wesley, L. 1995. Planning and Reacting in Uncertain and Dynamic Environments. *Journal of Experimental and Theoretical Artificial Intelligence* 7: 121–152.

Woo, C., Evens, M. W., Michael, J. A. and Rovick, A. A. 1991. Dynamic Instructional Planning for an Intelligent Physiology Tutoring System. In Proceedings of the Fourth Annual IEEE Computer-Based Medical Systems Symposium, Baltimore.

Woolf, B. 1984. Context-Dependent Planning in a Machine Tutor. Ph.D. diss., Dept. of Computer and Information Science, University of Massachusetts at Amherst. COINS Technical Report 84–21.

Yang, Q. 1990. Formalizing planning knowledge for hierarchical planning. *Computational Intelligence* 6(1): 12–24.

Zhou, Y., Freedman, R., Glass, M., Michael, J. A., Rovick, A. A. and Evens, M. W. 1999. Delivering Hints in a Dialogue-Based Intelligent Tutoring System. In Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI '99), Orlando, FL.