Papers from the Workshop on Adaptation in Dialogue Systems, NAACL 2001, Pittsburgh.

Adaptive Processing in a Medical Intelligent Tutoring System

Reva Freedman
Dept. of Computer Science
Northern Illinois University
DeKalb, IL
freedman@cs.niu.edu

Byung-In Cho
Korean Institute for
Defense Analysis
Seoul, Korea
chobyun@charlie.cns.iit.edu

Michael Glass
Department of EECS
U. of Illinois at Chicago
Chicago, IL
mglass@eecs.uic.edu

Yujian Zhou
WebEx Corp.
San José, CA
zhouyuj@hotmail.com

Jung Hee Kim
Dept. of Computer Science
North Carolina A & T University
Greensboro, NC
jkim@ncat.edu

Bruce Mills, Feng-Jen Yang and Martha W. Evens
Department of Computer Science
Illinois Institute of Technology
Chicago, IL
bruce_mills@wlc.edu, yangfen@charlie.cns.iit.edu, evens@iit.edu

The CIRCSIM-Tutor Project
http://www.cs.iit.edu/~circsim

Abstract

CIRCSIM-Tutor v. 3 is a dialogue-based intelligent tutoring system (ITS) that uses a uniform plan operator representation to make decisions at all levels of dialogue generation. It was designed to allow dynamic adaptation to the student at five levels of detail as the (typed) conversation progresses. In this paper we describe the studies that we used to make design decisions, including analysis of transcripts of human tutoring sessions, use of C4.5/C5.0, and user studies using an earlier version of the system. We are particularly interested in distinguishing decisions that are best made using user model information from those where recent dialogue history is the most relevant input.

1 Introduction

CIRCSIM-Tutor v. 3 is a dialogue-based intelligent tutoring system (ITS) that uses a uniform plan operator representation to make decisions at all levels of dialogue generation. It was designed to allow dynamic adaptation to the student at five levels of detail as the (typed) conversation progresses. These levels include the following:

- Topic of conversation
- High-level dialogue structure
- Transaction schemata
- On-the-fly updates to the dialogue plan
- Acknowledgments and discourse markers

To decide which areas of the system would benefit from adaptation as well as to decide what rules to use, we relied on three processes:

- Analysis of human-to-human transcripts
- Use of C4.5/C5.0 (Quinlan, 1993)
- Studies using previous versions of the system

In this paper we describe each of these areas of decision-making. For each area we outline the range of choices available, some rules we have developed, and the studies we did to choose those rules. We are especially interested in the degree to which variation in dialogue is determined by global data structures such as the user model versus local information such as the recent dialogue history.

Although the annotated portion of our corpus is not yet large enough for statistically significant conclusions, we have been able to derive some interesting rules, validate some intuitions, and provide a methodology for further research.

2 Generating Tutorial Dialogues

2.1 Structure of Tutoring Dialogues

Human-computer tutorial dialogues in problem-solving domains appear to have a consistent structure regardless of the domain. In problem-solving tutors, the student advances by making correct problem-solving steps via the system's user interface. If tutor and student are conducting a conversation in parallel with this activity, the resulting dialogue can be modeled at an abstract level as a hierarchically structured task-oriented dialogue (Grosz and Sidner 1986). Using this model, each leaf node represents the dialogue required to assist the student in achieving one correct problem-solving step, usually visible as one correct entry on the GUI. The higher-level nodes represent stages of the problem solution. The relative breadth and depth of the resulting tree depends on the domain and the task.

We call each leaf node a *tutoring episode*. Our data has shown that the conversation inside tutoring episodes is largely independent.

For each type of error a student might make, tutors maintain a repertoire of script-like methods for correcting them, labeled *tutoring methods* by Kim, Freedman and Evens (1998b). Each method consists of a sequence of one or more topics. Topics can be nested methods or speech act primitives such as *elicit* and *inform*. When the student needs help, the tutor chooses a tutoring method and goes through it one step at a time. When the student has difficulty with a question inside a tutoring method, the tutor can choose to respond to the student before going on with the tutoring method. As a result, the conversation within a tutoring episode does not necessarily map directly onto the original tutoring method. In addition, the tutor can choose to drop a tutoring method and replace it by another one if the conversation is not going well.

To allow for this type of elaboration as the tutoring methods are played out, we have found it more useful to use a model based on Discourse Analysis (Sinclair and Coulthard 1975, 1992) within tutoring episodes rather than a purely hierarchical model. Discourse Analysis is a variant of Conversation Analysis (Schegloff and Sacks 1973). While both models focus on describing how turns accrete to form a conversation rather than the hierarchical structure of the conversation, Discourse Analysis is specifically oriented toward teacher-student interactions.

The highest level of analysis, the *transaction*, is defined as the discussion of one topic. In tutoring systems, a transaction usually corresponds to a tutoring episode. Transactions are built from exchanges. An *exchange* consists of a dialogue move by one party, a reply by the other, and possible followup by the originator. A turn often includes the followup from one exchange and the initiating move of the next. Each move can be expressed as a series of discourse acts, the smallest unit of discourse structure.

Additionally, although in human conversation non-verbal cues can be used to indicate a change of speaker, the student interacting with an ITS needs an explicit marker for the end of a turn. Thus the ITS must end each turn with a question or action request.

Using the Discourse Analysis approach allows us to derive the following structure for a turn in tutorial dialogue (Freedman and Evens 1996). Each of the three sections is optional, although one of the last two must be provided.

- Acknowledgment of student's statement
- Content-based reply
    If last section, must end in question/request
- New material
    If last section, must end in question/request

The *acknowledgment* is a domain-independent response such as "correct" or "wrong." The *content-based reply* allows the tutor to reply directly to the student's previous statement. In general this is used when the student's answer is wrong or unexpected, and might include a denial or counterexample. If the student's answer is basically correct, this slot can be used for a restatement of the student's answer or for further supporting evidence. The *new material* slot is used for the tutor to continue a tutoring method when the student's previous reply was correct or the tutor does not feel the need to belabor the point.

```
    <T-does-neural-DLR>
      <T-tutors-mechanism>
        <T-elicits-mechanism>
(29.4) Can you tell me how TPR is
controlled?
<S-answer catg=correct>
(30.1) Autonomic nervous system.
</S-answer>
<T-ack type=positive>
(31.1) Yes.
</T-ack>
        </T-elicits-mechanism>
      </T-tutors-mechanism>
      <T-tutors-DR-info>
        <T-informs-DR-info>
(31.2) And the predictions that you
are making are for the period before
any neural changes take place.
        </T-informs-DR-info>
      </T-tutors-DR-info>
      <T-tutors-value>
        <T-elicits-value>
(31.3) So what about TPR?
              ...
<S-ans catg=correct>
(38.4) I would like to change my
response re TPR to zero change.
</S-ans>
<T-ack type=positive>
(39.1) Good.
</T-ack>
        </T-elicits-value>
      </T-tutors-value>
    </T-does-neural-DLR>
```

Figure 1: Example of annotated text

2.2 Reactive planning for dialogue generation

Reactive planning (Georgeff and Ingrand 1989, Wilkins et al. 1995), also known as interleaved planning and execution or "just-in-time" planning, is an approach to generating tutorial dialogues that fits the model described in the previous section. As opposed to classical planning, where a complete plan is generated before execution, a reactive planner plans only as far as necessary to generate the next action, which in the case of dialogue planning is usually a turn. An agenda of unsatisfied and partially satisfied goals is maintained for continuity between turns. Reactive planning is well suited for dialogue generation because it models an essential characteristic of dialogue, that each party cannot know in advance how the other party is going to respond. Thus, after receiving an unexpected student reply, the tutor can change the agenda as part of planning its next turn.

Although syntax and semantics differ from system to system, plan operators in a reactive planning system generally contain a goal, preconditions, and an action recipe. Since reactive planners often include hierarchical decomposition as a method of planning in addition to means-end analysis, the syntax frequently allows for multi-step recipes. An agent plans by choosing an operator from those whose goal matches its current goal and whose preconditions are satisfied. The recipe from the chosen operator is then executed. This cycle is repeated, executing primitives and updating the agenda as necessary, until an operator needs input from the external world.

3 Application Background

3.1 The CIRCSIM-Tutor Task

CIRCSIM-Tutor is a dialogue-based intelligent tutoring system that helps medical students master the reasoning they need to learn in Introduction to Physiology. Students typically have difficulty in this course because the processing in many body organs involves negative feedback loops. Since the heart is covered first, mastering the underlying principles here improves the student's chance of succeeding in the course.

Students are given a simplified qualitative model of the heart, followed by a series of problems which utilize the model. The student interface screen is shown in Figure 2.

In each problem, an incident such as the administration of a drug affects the processing of the heart. The problem is shown in the upper-right corner of the screen. The student is then asked to predict the direction of change of seven core variables at three points in time. Below the problem there is a table where the student can fill in predictions for the three stages. The left-hand side contains a scrolling dialogue window. Either during or after the prediction phase, depending on the interaction protocol, the tutor engages the student in a dialogue to help the student learn the correct answers and the concepts underlying them.

There is a logical order for solving each problem, depending on the causal relations between the variables. In particular, the first variable the student should examine in the first stage, the *primary variable*, drives the solution to the problem. Thus tutors often insist that the student get the primary variable correct even when they don't require that the student follow a logical order.

Three of the seven core variables are controlled by the nervous system; these are called *neural variables* and need different pedagogical content.

CIRCSIM-Tutor can currently handle 83 problems at various levels of difficulty.

3.2 Collecting Human Dialogue Data

The CIRCSIM-Tutor project has accumulated over 5000 typed turns of expert human tutors helping students solve similar problems. The data were collected in approximately 60 one- and two-hour sessions.

We have hand-coded about 10% of the corpus using an SGML-based markup scheme that highlights the hierarchical nature of the tutor's goals while also showing the student's responses. The sections annotated were chosen to include about 50% of the instances of the features discussed in this paper. (The first author is currently working on converting the markup to XML and providing a platform-independent application to aid in annotation.)

Figure 1 shows an example of our markup. The numbers in parentheses are the turn number and sentence number from the original corpus. Each argument only occurs on the uppermost level to which it applies; we use a mechanical process to copy the arguments to lower levels. The *t-ack* form is part of the exchange structure but is not part of the goal hierarchy of the tutorial planner. The *s-answer* forms are produced by the student, not by the tutor.

This example shows a successful attempt to teach the student the correct value of the variable TPR in the DR stage. The following schema is used:

- Teach about the mechanism of control (neural)
- Teach about the current stage (DR)
- Check whether the student now knows the answer

For reasons of space, no content-based replies are shown. If the tutor had replied *yes, but …* at (31.1), we would have annotated it with a form such as *t-rebuts* and either *t-informs* or *t-elicits* below that.

The student's input during the initial prediction phase has also been captured in the transcripts so that the non-verbal part of the conversation is available as well. This information can be used to calculate performance scores for simulating the student model.

3.3 CIRCSIM-Tutor Architecture

CIRCSIM-Tutor has separate components for choosing discourse goals and for realizing them as text. The text realizer (or "turn planner") buffers all of the discourse goals comprising a turn so that it can generate more natural language. The turn planner, which is still being developed, is based on the semantic grammar developed by Kim, Freedman and Evens (1998a).

The tutorial planner is implemented using APE, a domain-independent dialogue plan interpreter (Freedman, 2000). APE plans are augmented schemata permitting both goal-oriented planning and hierarchical decomposition. APE allows CIRCSIM-Tutor to do update its agenda whenever required, something that most other ITSs, based on augmented finite-state machines, cannot do. Although APE can implement any programming construct, because we were interested in a declarative representation, we tried to limit our use of this feature to three cases: to skip the remainder of a tutoring strategy when circumstances change, to replace a strategy with another strategy that has the same goal, and to add or replace a sequence of goals at the top of the agenda. The last category is especially useful for adding a response to a student utterance without disturbing the global plan.

All of the rules described in the paper except for the choice of lexical items are implemented as APE plan operators. Since the turn planner is still being developed, no final decision has been made on the location of the lexical choice component.

Potential criteria for these decisions, i.e. prerequisites for the plan operators, include information from any of the following sources:

- *Dialogue history*. The dialogue history includes a trace of the plan as it is executed, including information about which schemata were chosen, what changes have been made to the agenda, and current open goals. The dialogue history also includes the student's response and our categorization of it.
- *Student model*. The student model tells us whether we are conversing with a strong or a weak student. We use the student's initial solution of the problem as well as later responses during the conversation to maintain a static and a dynamic evaluation of the student.
- *Domain knowledge base*. Domain knowledge is used to choose correct pedagogical content when required.

3.4 Studies with Users

During the extended gestation of CIRCSIM-Tutor v. 3,

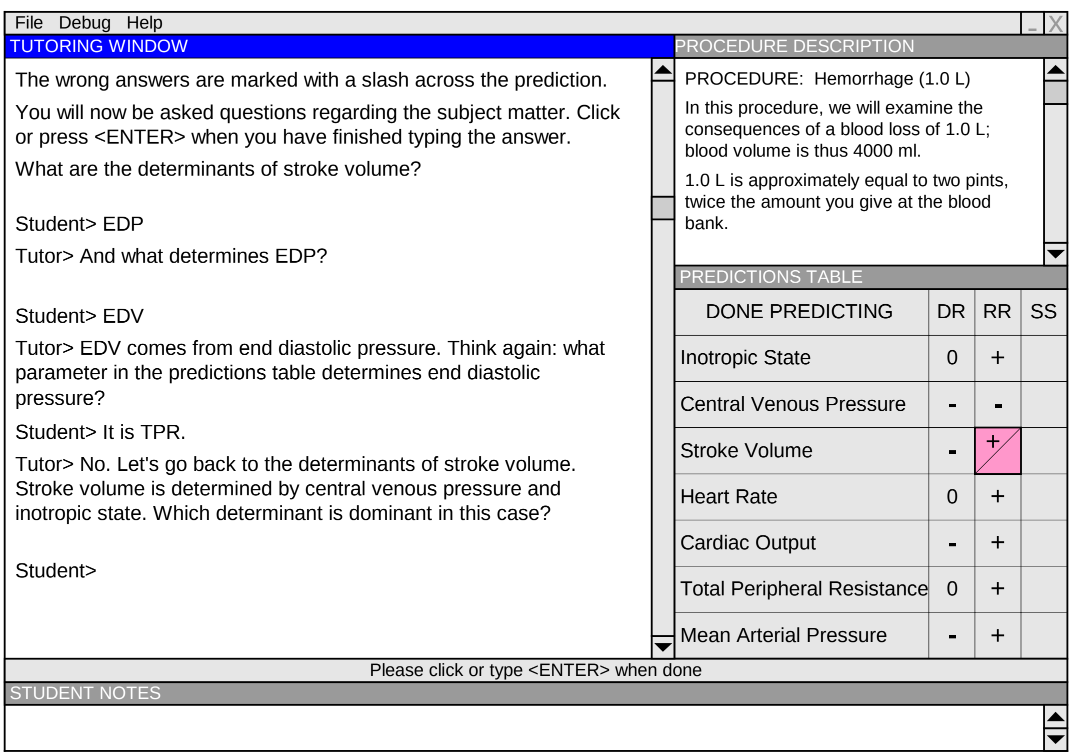

Figure 2: CIRCSIM-Tutor user interface

we have been able to obtain feedback on some possible algorithms by implementing them in earlier versions of CIRCSIM-Tutor. Although many ideas that can be expressed in the plan operator language in v. 3 must be hard-coded in earlier versions, this approach has allowed us to prototype a number of ideas before the completion of the full system.

This paper includes data from two studies with paid subjects. The subjects were volunteer medical students enrolled in Introduction to Physiology. The studies were scheduled so that the students had heard the material in lectures but had not yet been tested on it. This schedule was selected because CIRCSIM-Tutor is not intended to teach the material from scratch. The April 1998 study involved 24 students using v. 2.6 and the November 1998 study involved 48 students using v. 2.7.

4 Results of Empirical Studies

4.1 Choice of Conversation Topic

The highest-level decision that CIRCSIM-Tutor makes is the choice of the next problem. In a problem-based ITS like CIRCSIM-Tutor, this decision is considered as part of the *curriculum planning* phase. The curriculum planning rules in CIRCSIM-Tutor were developed based on the intuition of expert teachers. They were then implemented by Cho et al. (1999) and tested on students. These rules consider only two factors:

- Student's desired difficulty level
- Global assessment of student performance

The student's desire is obtained by asking whether the student wants the next problem to be harder, easier or the same difficulty as the previous problem.

Results from the April 1998 study reinforced the importance of early testing with users. After initial results appeared to contradict the authors' beliefs about the relative difficulty level of the problems, it was discovered that many of the problem names used on the problem selection menu contained the name of the primary variable, thus providing an inadvertent extra hint. ("Hemorrhage" in Figure 2 doesn't, but many of the others did.) As a result the menu was changed to show the problem description instead.

4.2 Choice of High-Level Dialogue Organization

The next decision that must be made is about the high-level conversation structure, or *interaction protocol* in ITS terminology. Khuwaja et al. (1995) identified three protocols used by expert human tutors through analysis of the human-to-human transcripts.

The protocols differ along two axes. On the first axis, the tutor can insist that the student solve the problem in a logical order, not require this, or only require that the student get started in the right direction. When the student makes errors, the second axis determines when the tutor interrupts the student's problem solving for a correction dialogue. Although many educators believe that it is important to interrupt the student as soon as possible to give immediate feedback, other experienced teachers (Michael et al., 1992) find it worthwhile to let the student solve a larger part of the problem before interrupting in order to get a better idea of what the student's underlying confusion is.

There are three logical places to interrupt a student in CIRCSIM-Tutor: after a single variable (one cell in the matrix) has been calculated, after the completion of each temporal stage (a column), or after the completion of the entire problem. (Although the latter might seem extreme, CIRCSIM, an earlier non-adaptive CAI system using that protocol, has been successfully used in the same physiology course for several years.)

In the most commonly used protocol studied by Khuwaja, the tutor ensures that the student has the correct answer for the primary variable, then lets the student progress a column at a time. Cho et al. (2000) extended this research by looking at the nine longest human-to-human transcripts to see if the tutors changed their protocol as the session progressed. They identified a new protocol where the tutor started with Khuwaja's protocol but switched to requiring the correct order and giving immediate feedback when the student's performance was poor. The tutor switched back when student performance improved.

C5.0 was used to identify possible rules for switching between protocols. Eleven attributes were identified for each temporal stage in the transcripts:

- *Discussion type*: whether the tutor started the conversation with a discussion of a basic concept or the student started it with a request for an explanation. The initial discussion ends when the primary variable is identified.
- *Discussion success*: whether the tutor was satisfied with the student's responses in the initial discussion at least 50% of the time.
- *Discussion length*: number of turns in the initial discussion.
- Seven numerical values measuring the student's progress.
- Temporal stage of the data point.

Three factors were most useful in generating a parsimonious set of rules:

- Discussion success
- A selection of global performance measures
- Temporal stage

When one considers that discussion success is another performance measure, it is clear that most of the factors involved are performance measures. We do not know why the temporal stage was important, whether because human tutors handle the content of the various stages differently or because their intervention preference changes as the session continues.

It is interesting to note that in a questionnaire completed as part of the November 1998 study, students who did poorly were much more likely to want immediate feedback than students who were doing well. Of course, the fact that students preferred one of the protocols does not mean that they did better when using it.

4.3 Choice of Tutoring Strategy

Freedman et al. (1998) did some initial studies using C4.5 to identify the conditions under which various tutoring strategies used in the transcripts are chosen by the tutor. In the first study, they annotated the first temporal stage of every transcript dealing with a broken pacemaker. There were 23 data points, where each data point represented one attempt to tutor one variable. The following features were annotated:

- Whether the variable was neural or non-neural
- Sequence of the variable within the neural or non-neural group. The tutors usually tutor all the neural variables first, followed by the others.
- How many previous attempts had been made to tutor this variable
- Total number of variables predicted incorrectly
- Number of neural variables predicted incorrectly (three of the seven core variables are controlled by the nervous system)

The outcome was one of five possible strategies. Only two factors were relevant, whether or not the variable was neural and its sequence within the group of similar variables. It is interesting to note that none of the performance assessment parameters were chosen, only dialogue history phenomena.

Although this result was not novel, as it matched the main tutorial schema used in an earlier version of CIRCSIM-Tutor, it showed that rule induction with a larger corpus could be a useful method of identifying preconditions for schema choice.

4.4 On-the-Fly Dialogue Plan Updating

The goal of the second Freedman et al. study was to obtain a rule for determining when the tutor would change strategies. This study had 57 data points. The possible outcomes were coded for implementation as APE operators:

- *Proceed*. Proceed with the next tutorial goal. Normal action when the student gave the desired answer.
- *Give info and proceed*. The tutor responded with some tutorial information before proceeding.
- *Give info and re-elicit*. The tutor responded with some tutorial information, then asked substantially the same question again.
- *Give answer and proceed*. The tutor gave the student the answer, then proceeded with the next tutorial goal.
- *Nested method*. The tutor introduced a nested tutoring method to address the current tutorial goal.
- *New method*. The tutor abandoned the current tutoring goal and all its descendants, and tried another method to tutor the same variable.

The most explanatory features were:

- Accuracy of the student's response
- Whether the variable was neural or non-neural
- Sequence of the variable within its group
- How many previous attempts had been made to tutor this variable

No global assessment features were useful. This indicates that the tutor's response to a student utterance appears to depend more on recent dialogue history than on a global assessment of the student's performance.

Finally, Freedman et al. looked for a low-level rule to determine when to make the student actively produce some information and when to provide it instead. Two simple rules were obtained:

If the topic is "what stage are we in"
    if the context is neural vs. non-neural control
        prefer *inform*
    else prefer *elicit*

If all three neural variables were incorrect
    prefer *elicit*
    else prefer *inform*

Note that neither rule depends on the student assessment. The second rule suggests that when the student makes the same error repeatedly, the tutor switches to a schema where the information involved is specifically probed for.

4.5 Acknowledgments and discourse markers

Two studies using C4.5 have contributed to our knowledge about lexical choice for these closed-class items.

Freedman et al. (1998) attempted to induce a rule for generating acknowledgments. Unlike human tutors, old-fashioned CAI systems give an acknowledgment after every student answer, one of the features that make them sound so unnatural. Although they had 62 cases, they were unable to find an acceptable rule. There is some evidence that tutorial planning features are not sufficient to explain the use of acknowledgments. We suspect that other features affecting dialogue coherence, such as the presence or absence of an initial discourse marker, are related to the decision to use an explicit acknowledgment.

Kim et al. (2000) attempted to find a rule to generate discourse markers by coding 60 instances from the same corpus for the following features:

- Accuracy of student's most recent answer
- Presence of acknowledgment in the sentence
- Sequence of tutorial topic within strategy
- Whether discourse goal is *elicit* or *inform*

The following rule was obtained.

If sentence precedes first topic, then use *now*
If first topic in strategy:
  if goal is *elicit*, use *so*
  else /* goal is *inform* */ use *and*
If medial topic in strategy, then use *and*
If last topic in strategy, then use *so*

The result is not novel; it had been predicted earlier by Kim, Freedman and Evens (1998a). But the study shows that the coding is sophisticated enough to predict useful rules.

5 Related Work

The idea of using machine learning, specifically the use of rule induction, to help identify reasonable rules from transcripts was first suggested to us by a series of papers connecting text generation goals with textual phenomena (Moser and Moore, 1995; Vander Linden and Di Eugenio 1996a, 1996b; Di Eugenio, Moore and Paolucci, 1997). Our approach differs from theirs in that it is based on a hierarchical markup; we use inputs from each level of the hierarchy in determining our rules.

6 Conclusions

CIRCSIM-Tutor v. 3 has opportunities for dialogue adaptation in five areas of processing: curriculum planning, choice of GUI protocol, choice of tutoring schemata, on-the-fly updates to the dialogue plan, and the choice of selected lexical items. This paper has described the use of three methods—the analysis of human tutorial transcripts, the use of C4.5/C5.0, and studies using earlier versions of the system—to help us decide what the alternatives should be and how the choice should be made. Although the annotated portion of our corpus is not yet large enough for statistically significant conclusions, we have been able to derive some interesting rules, validate some intuitions, and provide a methodology for further research.

Acknowledgments

The studies discussed here were supported by the Cognitive Science Program, Office of Naval Research under Grant No. N00014–94–1–0338 to the Illinois Institute of Technology.

Professors Joel A. Michael and Allen A. Rovick of the Department of Physiology at Rush Medical College devised the data collection setup for the human tutoring sessions and served as expert tutors. We also thank the anonymous reviewers for their helpful suggestions.

References

Cho, B., Michael, J., Rovick, A. and Evens, M. 1999. A Curriculum Planning Model for an Intelligent Tutoring System. In Proceedings of the Twelfth Florida Artificial Intelligence Research Symposium (FLAIRS '99), Orlando.

Cho, B., Michael, J., Rovick, A. and Evens, M. 2000. An Analysis of Multiple Tutoring Protocols. In G. Gautier et al., eds., Intelligent Tutoring Systems: Fifth International Conference (ITS 2000), Montreal. Berlin: Springer. LNCS 1839.

Di Eugenio, B., Moore, J. D. and Paolucci, M. 1997. Learning Features that Predict Cue Usage. Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL '97), Madrid. `Cmp-lg/9710006`.

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

Freedman, R. and Evens, M. W. 1996. Generating and Revising Hierarchical Multi-turn Text Plans in an ITS. In C. Frasson, G. Gauthier and A. Lesgold, eds., Intelligent Tutoring Systems: Third International Conference (ITS '96), Montreal. Berlin: Springer. LNCS 1086.

Freedman, R., Zhou, Y., Glass, M., Kim, J. H. and Evens, M. W. 1998. Using Rule Induction to Assist in Rule Construction for a Natural-Language Based Intelligent Tutoring System. In Proceedings of the Twentieth Annual Conference of the Cognitive Science Society, 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.

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

Khuwaja, R., Rovick, A., Michael, A. and Evens, M. W. 1995. A Tale of Three Protocols: The Implications for Intelligent Tutoring Systems. In E. A. Yfantis, ed., Intelligent Systems: Third Golden West International Conference, Las Vegas, 1994. Dordrecht: Kluwer.

Kim, J., Freedman, R. and Evens, M. W. 1998a. Relationship between Tutorial Goals and Sentence Structure in a Corpus of Tutoring Transcripts. In Proceedings of Ninth Midwest Artificial Intelligence and Cognitive Science Society Conference (MAICS ’98), Dayton.

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

Kim, J., Glass, M., Freedman, R. and Evens, M. W. 2000. Learning the Use of Discourse Markers in Tutorial Dialogue for an Intelligent Tutoring System. In Proceedings of the Twenty-second Annual Conference of the Cognitive Science Society, Philadelphia.

Michael, J., Rovick, A., Evens, M., Shim, L., Woo, C. and Kim, N. 1992. The Uses of Multiple Student Inputs in Modeling and Lesson Planning in CAI and ICAI Programs. In I. Tomek, ed., Computer Assisted Learning: Proceedings of the Fourth International Conference on Computer Assisted Learning (ICCAL ’92), Wolfville, Nova Scotia. Berlin: Springer.

Moser, M. G. and Moore, J. D. 1995. Investigating Cue Selection and Placement in Tutorial Discourse. Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL ’95), Cambridge, MA.

Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning*. Los Altos, CA: Morgan Kaufmann.

Schegloff, E. and Sacks, H. 1973. Opening up Closings. *Semiotica* 7: 289–327.

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

Sinclair, J. M. and Coulthard, R. M. 1992. Towards an Analysis of Discourse. In M. Coulthard, ed., *Advances in Spoken Discourse Analysis*, pp. 1–34. London: Routledge.

Vander Linden, K. and Di Eugenio, B. 1996a. A corpus study of negative imperatives in Natural Language instructions. Proceedings of the 17th International Conference on Computational Linguistics (COLING ’96), Copenhagen. `Cmp-lg/9607014`.

Vander Linden, K. and Di Eugenio, B. 1996b. Learning Micro-Planning Rules for Preventative Expressions. Eighth International Workshop on Natural Language Generation (INLG ’96), Sussex, UK. `Cmp-lg/9607015`.

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.