

Workshop W8

**Dialog-based Intelligent Tutoring
Systems**

**State of the Art
and New Research Directions**

**Held in conjunction with ITS 2004, the Seventh International
Conference on Intelligent Tutoring Systems**

Maceió, Brazil, August 31, 2004

Preface

Intelligent tutors should be like a human you can sit down and talk to and learn something from, or so goes some of the intuition behind wanting to build dialog-based tutors. While none of the systems mentioned in these papers quite meet that definition, the enclosed papers represent some of the most recent attempts to incorporate dialog into intelligent tutoring systems. Human dialog is complex and has many aspects, and this range is reflected in the different aspects of dialog that these papers attempt to model. For example, speech, natural language understanding, and discourse (dealing with what to ask next) are some of the areas explored in these papers.

Organizing Committee

Neil Heffernan (Co-Chair), Worcester Polytechnic University

Peter Wiemer-Hastings (Co-Chair), DePaul University

Vincent Aleven, Carnegie Mellon University

Ivon Arroyo, University of Massachusetts

Paul Brna, University of Northumbria at Newcastle

Mark Core, University of Edinburgh

Martha Evens, Illinois Institute of Technology

Reva Freedman, Northern Illinois University,

Michael Glass, Valparaiso University

Art Graesser, University of Memphis

Pamela Jordan, University of Pittsburgh

Ken Koedinger, Carnegie Mellon University

Diane Litman, University of Pittsburgh

Evelyn Lulis, DePaul University

Helen Pain, University of Edinburgh

Carolyn Rose, Carnegie Mellon University

Beverly Woolf, University of Massachusetts at Amherst

Claus Zinn, University of Edinburgh

Table of contents

<i>iii</i>	Preface
<i>v</i>	Organizing Committee
<i>1</i>	Towards Easier Creation of Tutorial Dialogue Systems: Integration of Authoring Environments for Tutoring and Dialogue Systems <i>Vincent Alevan & Carolyn Penstein Rosé</i>
<i>9</i>	Adaptive Tutorial Dialogue in AutoTutor <i>G. Tanner Jackson, Natalie K. Person, and Arthur C. Graesser</i>
<i>15</i>	Looking at the Student Input to a Natural-Language Based ITS <i>Chung Hee Lee, Martha W. Evens, and Michael S. Glass</i>
<i>23</i>	Evaluating the Effectiveness of SCoT-a Spoken Conversational Tutor <i>Heather Pon-Barry, Brady Clark, Elizabeth Owen Bratt, Karl Schultz and Stanley Peters</i>
<i>33</i>	Tutorial Dialog in an Equation Solving Intelligent Tutoring System <i>Leena M. Razzaq and Neil Heffernan</i>
<i>43</i>	Guided Exploratory Learning versus Directed Learning in a Simulation Environment for Thermodynamics: A Pilot Study <i>Carolyn Penstein Rosé, Cristen Torrey, and Vincent Alevan</i>
<i>49</i>	Conceptual Architecture for Generating Examples in a Socratic Tutor for Qualitative Reasoning <i>Leo C. Ureel II</i>
<i>55</i>	The design and architecture of Research Methods Tutor, a second-generation dialog-based tutor <i>Peter Wiemer-Hastings</i>

Towards Easier Creation of Tutorial Dialogue Systems: Integration of Authoring Environments for Tutoring and Dialogue Systems

Vincent Alevén & Carolyn Penstein Rosé

Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh PA 15213
aleven,cprose@cs.cmu.edu

In order for tutorial dialogue systems to become widespread, it is imperative that we develop ways to reduce both the development time needed and the level of computational linguistics and AI expertise required. We describe early work towards achieving that goal, focused on the integration of the Cognitive Tutor Authoring Tools (CTAT), Carmel Tools for authoring semantic knowledge sources for language understanding, and the KCD Authoring Tool Suite for authoring interactive directed lines of reasoning. This tool combination has been and is being tried out in two contexts: During the 4th Annual Summer School on Intelligent Tutoring Systems, participants with no prior authoring experience with the tools developed small tutorial dialogue systems in one week's time using these tools. We are also using the tool combination during the development of a novel tutorial dialogue system, CycleTalk. Lessons learned in those contexts will doubtlessly be very helpful in other similar efforts.

Keywords: tutorial dialogue, cognitive modeling, authoring tools

Introduction

Given the convenience of conversational interfaces, whether speech-based or keyboard-based, tutorial dialogue systems will no doubt become popular alternatives to intelligent tutoring systems based on more traditional WIMP interfaces. However, in order for these systems to become widespread and have a real impact on education, it is imperative that they become easier to build. Most tutorial dialogue systems that to date have undergone successful evaluations (CIRCSIM, AutoTutor, Atlas-Andes, WHY-Atlas, the Geometry Explanation Tutor) represent development efforts of many man-years. These systems were instrumental in pushing the technology forward and in proving that tutorial dialogue systems are feasible and useful in realistic educational contexts, although not always provably better than the more challenging alternatives to which they have been compared. We are now entering a new phase in which we as a research community must not only continue to improve the effectiveness of tutorial dialogue but also must find ways of making system development more efficient.

The current paper presents our early work on creating an authoring environment that will make one general type of tutorial dialogue system dramatically easier to build. These systems involve offering knowledge construction dialogues in the context of problem-solving exercises and as such share some global traits with Atlas-Andes (Rosé et al., 2001), in particular in their dialogue component. (The underlying model-tracing approach is different, as will become clear later.)

Key to the approach is the integration of existing authoring tools. Specifically, the Cognitive Tutoring Authoring Tools (Koedinger, Alevén, Heffernan, McLaren, & Hockenberry, in press) will be used to implement the model tracing portion of the system and the Carmel Tools (Rosé et al., 2003; Rosé & Hall, 2004) and KCD Authoring Tools (Jordan, Rose, & VanLehn, 2001) with recent enhancements for more sophisticated dialogue management techniques (Rosé & Torrey, 2004) will be used to implement the natural language dialogue portions. This approach has been and continues to be attempted in two different contexts: First, during the 4th Summer School on Intelligent Tutoring Systems, participant teams with little or no prior experience in the development of tutorial dialogue systems did in a single week create small prototypes of tutoring systems with a

dialogue component. Second, the approach, with some modification described below, is being used in the context of a novel tutorial dialogue system, CycleTalk (Rosé et al., 2004) that will engage students in discussion about design solutions as they work with a thermodynamics simulator (Forbus et al., 1999). We anticipate that the tools will greatly facilitate the development of this system. In these contexts, we expect to learn a great deal about various tool integration issues. We also expect that the integration effort will serve as a forcing function towards developing the tools further.

In the current paper, we briefly describe each of the different toolkits and describe how they have been integrated.

Cognitive Tutor Authoring Tools (CTAT) for non-AI programmers

The primary goal of the Cognitive Tutor Authoring Tools (CTAT) project is to make development of intelligent tutors (without dialogue) easier for experts (i.e., AI programmers) and feasible for non-experts. Cognitive Tutors are very effective (Anderson, Corbett, Koedinger, & Pelletier, 1995; Koedinger, Anderson, Hadley, & Mark, 1997) but their development is costly. A recent estimate puts development time at 200 hours per hour of instruction. To dramatically reduce the development time and the level of required expertise, CTAT will (and to a considerable degree already does) support all phases of tutor development, drawing on key HCI and AI techniques such as programming by demonstration. The tools will support different methods for authoring intelligent tutoring behavior. At the time of this writing, CTAT supports the authoring intelligent tutoring behavior in two different ways, first, by creating full-blown Cognitive Tutors and second, by creating a novel type of tutors that we have dubbed “Pseudo Tutors.” These two types of systems represent different trade-offs between the generality of the tutoring system and the time and expertise required to develop the system. The cognitive model that is a key component of Cognitive Tutors provides the generality needed to deal with large and homogeneous problem sets. Developing such a model is however not an easy task, requiring considerable expertise in AI programming. Pseudo Tutors on the other hand can be created in hours, but lack the scalability and generality of Cognitive Tutors. The term “Pseudo” in “Pseudo-Tutors” indicates not the absence of real tutoring (our initial experience with Pseudo Tutor development indicates that within their limited scope, they support the most important behaviors of Cognitive Tutors, see Koedinger et al., in press) but rather the absence of a model that supports intelligent tutoring across a broad range of problems. The approach to tool integration described in this paper applies equally to Cognitive Tutors and Pseudo Tutors, but in this paper we focus primarily on Pseudo Tutor development.

Using CTAT, a non-programmer author can a Pseudo Tutor that does interactive tutoring in a few hours, in the following steps (Koedinger et al., in press):

Create a graphical user interface (GUI) where student problem solving is performed, without programming, using a drag-and-drop GUI Builder tool (which we implemented by extending existing GUI Builders). Demonstrate alternative correct and incorrect solutions to the given problem in the interface – the steps are recorded by a tool called the Behavior Recorder in a “Behavior Graph” – all demonstrated actions are assumed to represent correct behavior unless the author indicates that they represent errors.

Add context-sensitive instruction by annotating solution steps in the Behavior Graph with hint messages and feedback messages (the latter related to incorrect actions).

The Behavior Recorder can then use the annotated Behavior Graph to tutor students as they attempt to solve the given problem. It uses the Behavior Graph to monitor student behavior and provide hints and feedback in a process called “example tracing,” analogous to the model-tracing algorithm used by Cognitive Tutors (Anderson et al., 1995), except that student behavior is being traced with respect to a specific set of example behaviors rather than a more general model.

Carmel-Tools for authoring natural language understanding systems

Carmel-Tools (Rosé & Hall, 2004) is a first attempt at a behavior oriented approach to facilitate rapid development of advanced language understanding interfaces. What we mean by behavior oriented is that Carmel-Tools provides a layer of abstraction between the author and the necessary linguistic knowledge sources (Rosé, 2000) freeing up the author to focus on the desired behavior rather than the linguistic details of the knowledge sources that would make this behavior possible. By insulating authors from the underlying linguistic issues, we aim to eventually bring the authoring process within the capabilities of typical interface programmers.

What we mean by advanced conversational interfaces is that they go beyond the capabilities that are possible using state-of-the-art authoring tools. Current authoring tools for building semantic knowledge sources, such as GATE (Cunningham et al., 2003), Alembic (Jay et al., 1997) and SGStudio (Wang & Acero, 2003) are tailored for information extraction and similar tasks that emphasize the identification of named entities such as people, locations, organizations, dates, and addresses. While regular expression based recognizers, such as JAPE (Cunningham, Maynard, & Tablan, 2000) used in such systems, are not strictly limited to these standard entity types, it is doubtful that they would be able to handle concepts that express complex relationships between entities, where the complexity in the conceptual representation can be encoded in natural language with a much greater degree of variation.

Carmel-Tools aims to overcome this limitation by inducing patterns that match against a more sophisticated underlying linguistic analysis rather than a stream of words, in order to normalize as much of the variation as possible, and thus reduce the number of patterns that the authored rules must account for. Furthermore, Carmel-Tools offers greater flexibility in output representation than the context-free rewrite rules produced by previous semantic authoring tools, allowing authors to design their own predicate language representations that are not constrained to follow the surface structure of the input text. An evaluation of the effectiveness of Carmel-Tools in the hands of an experienced user is presented in (Rosé & Hall, 2004). User testing of Carmel-Tools with inexperienced users shows that it has some usability issues that must be addressed before it can have the impact that it was designed to achieve. Overcoming these difficulties, in particular the problem that authors have trouble predicting the full extent of the ramifications of their authoring actions, is an important direction of our current research.

The KCD Authoring Toolkit for creating knowledge construction dialogues

The KCD Authoring Toolkit (Jordan, Rosé, & VanLehn, 2001) facilitates the development of Knowledge Construction Dialogues (KCDs), building upon the foundation of the Atlas Planning Engine (APE) (Freedman, 2000). KCDs were originally motivated by the idea of Socratic tutoring. KCDs are interactive directed lines of reasoning that are each designed to lead students to learn as independently as possible one or a small number of concepts, thus implementing a preference for an “Ask, don’t tell” strategy. When a question is presented to a student, the student types a response in a text box in natural language. The student may also simply click on Continue, and thus neglect to answer the question. If the student enters a wrong or empty response, the system will engage the student in a remediation sub-dialogue designed to lead the student to the right answer to the corresponding question. The system selects a subdialogue based on the content of the student’s response, so that incorrect responses that provide evidence of an underlying misconception can be handled differently than responses that simply show ignorance of correct concepts. Once the remediation is complete, the KCD returns to the next question in the directed line of reasoning.

KCDs have a very simple underlying dialogue management mechanism, specifically a finite state push down automaton. Thus, they do not make full use of the reactive capabilities of the APE tutorial dialogue manager. They make use of very simple shallow semantic parsing grammars to analyze student input, classifying it into one of a small number of pre-defined answer classes, although the integration with Carmel makes it possible to do a deeper analysis instead or in addition to the shallow, KCD approach. A set of accompanying authoring tools with a GUI interface (Jordan, Rosé, & VanLehn, 2001) makes it possible for domain experts to author the lines of reasoning underlying the KCDs. These authoring tools have been used successfully by domain experts with no technical or linguistic background whatsoever. KCDs invite students to enter freeform natural language responses to tutor questions. These tools make KCD development fast and easy. The most time consuming aspect of developing a knowledge construction dialogue is taking the time to thoughtfully design a line of reasoning that will be compelling enough to facilitate student understanding and student learning. Thus, the simplicity of the KCD technology allows developers to invest the majority of their time and energy on pedagogical concerns.

KCDs are a means for directly encoding the pedagogical content knowledge that is required to teach a concept effectively. Nevertheless, while KCDs have proved themselves robust enough to stand up to evaluations with real students, they fall short of the ideal of human tutorial dialogue. For example, KCDs are designed to lead students through a predetermined directed line of reasoning. While they have the ability to engage students in subdialogues when they answer questions incorrectly, they are designed to keep the student from straying too far away from the main line of reasoning. In order to do this, they respond to a wide range of responses that do not express the correct answer to questions in the same way. As a recent extension to this framework, the DReSDeN tutorial dialogue manager (Rosé & Torrey, 2004) provides a level of dialogue management above the level of individual KCDs. The goal is to build on what was valuable in the KCD approach while enabling a more flexible dialogue management approach that makes it practical to support mixed initiative and multi-threaded negotiation dialogues using general thread management strategies rather than making use of operator specific reactive mechanisms already provided in APE (Freedman, 2000).

A generic architecture for tutorial dialogue systems

The three authoring environments have now been combined to enable an author to create tutorial dialogue systems according to a generic architecture which involves the loose integration of three main components (see Figure 1): a Cognitive Tutor or Pseudo Tutor (to emphasize ease of authoring Figure 1 shows a Pseudo Tutor), a Dialogue Subsystem, and a Student Interface. The Pseudo Tutor provides all functionality that a tutorial dialogue system must provide besides dialogue. As the student solves the problem in the Student Interface, the student's actions are passed to the Pseudo Tutor. The Pseudo Tutor keeps track of the student's progress through the problem, by means of its example-tracing method, and provides hints and feedback at points in the interaction where dialogue is deemed not to be necessary or is expected not to have greater impact than standard non-interactive hints. At carefully selected points in the interaction, the Pseudo Tutor will trigger a dialogue, as is described further below. At that point, the dialogue manager takes control and engages the student in a dialogue. Once the dialogue is completed, the student resumes work on the problem and the Pseudo Tutor resumes its example-tracing approach to providing hints and feedback.

When creating a tutorial dialogue system of this type, the author must indicate when the system should initiate a dialogue with the student. In our first attempt at integrating, dialogues can be initiated in response to specific student errors or in response to hint requests. Typically, the Pseudo Tutor responds to errors and hint request by presenting messages that the author has attached to the links in the Behavior Graph (see step 3 of the Pseudo Tutor authoring process outlined above). However, when used to create a tutorial dialogue system rather than a Pseudo Tutor, CTAT enables the author to attach a dialogue name to any link in the Behavior Graph, rather than a sequence of hints or a "bug message." Then, when a student hint request or error matches one of those links, the attached dialogue name is sent over to the dialogue system, which then starts the dialogue with the given name. Thus, the coupling between the Pseudo Tutor and the dialogue system is very loose. A key assumption here is that the dialogue system does not need to have access to the details of the interaction that the student has had so far with the Pseudo Tutor, only that apparently, the student is struggling at a particular junction in the problem-solving process and an indication of what topic the student needs help on.

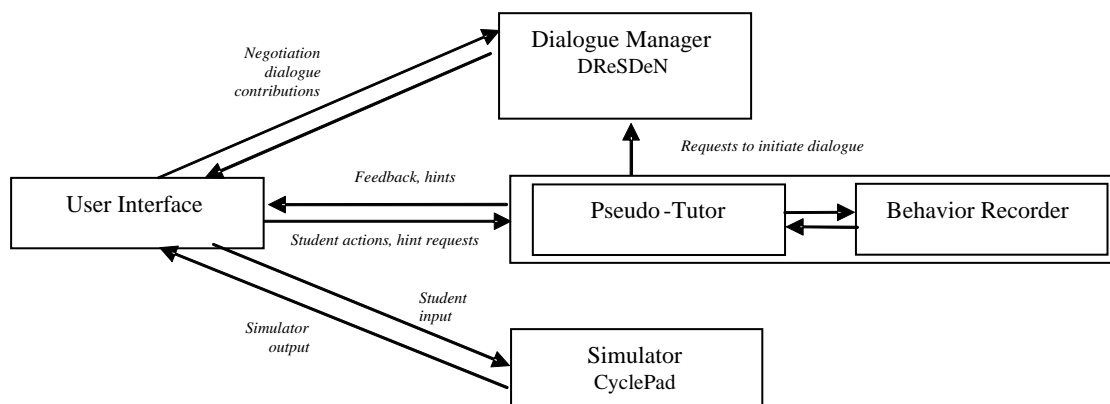


Figure 2: CycleTalk Architecture

Figure 2 presents a detailed picture of the CycleTalk architecture. As the student works with the simulator, the student's actions in the interface (the simulator interface, augmented with windows for hints and dialogue) are passed on from the interface to both the simulator and to the relevant Pseudo Tutor. The simulator updates the state of the simulated thermodynamic cycle and displays the new state in the interface. At the same time, the Pseudo Tutor keeps track of the student's progress through the exploration scenario. It provides hints and feedback. At points in the interaction where dialogue is deemed to be especially valuable (these judgments will be based as much as possible on empirical work, such as that reported in Rosé et al., 2004.) the Pseudo Tutor will trigger a dialogue. Once the dialogue is completed, the student resumes work in the simulator and the Pseudo Tutor resumes its monitoring.

Acknowledgements

The CycleTalk project is sponsored by ONR Award No.: N000140410107 "CycleTalk: A Tutorial Dialogue System that Supports Negotiation in a Design Context." The Cognitive Tutor Authoring Tools is sponsored by ONR Award No.: N000140310220 "Affordable Cognitive Model Authoring Tools Using HCI Methods." We gratefully acknowledge their contributions.

References

- Anderson, J., Corbett, A., Koedinger, K., & Pelletier, R., (1995). Cognitive Tutors: Lessons Learned. *The Journal of the Learning Sciences*, 4, 167-207.
- Cunningham, H., Maynard, D., and Tablan, V. (2000). Jape: a java annotations patterns engine. Institute for Language, Speech, and Hearing, University of Sheffield, Tech Report CS-00-10.
- Cunningham, H., Maynard, D., Bontcheva, K., and Tablan, V. (2003). Gate: an architecture for development of robust hlt applications. In *Recent Advances in Language Processing*.
- Forbus, K. D., Whalley, P. B., Evrett, J. O., Ureel, L., Brokowski, M., Baher, J., Kuehne, S. E. (1999). CyclePad: An Articulate Virtual Laboratory for Engineering Thermodynamics. *Artificial Intelligence* 114(1-2): 297-347.
- Freedman, R. (2000). Using a Reactive Planner as the Basis for a Dialogue Agent, *Proceedings of FLAIRS 2000*, Orlando.
- Jay, D., Aberdeen, J., Hirschman, L., Kozierok, R., Robinson, P., and Vilain, M. (1997). Mixed-initiative development of language processing systems. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*.

- Jordan, P., Rosé, C. P., & VanLehn, K. (2001). Tools for Authoring Tutorial Dialogue Knowledge. In J. D. Moore, C. L. Redfield, & W. L. Johnson (Eds.), *Artificial Intelligence in Education: AI-ED in the Wired and Wireless Future*, Proceedings of AI-ED 2001 (pp. 222-233). Amsterdam, IOS Press.
- Koedinger, K., Alevan, V., Heffernan, N., McLaren, B., & Hockenberry, M. (in press). Opening the Door to Non-Programmers: Authoring Intelligent Tutor Behavior by Demonstration, in Proceedings of ITS 2004.
- Koedinger, K., Anderson, J., Hadley, W., & Mark, M. (1997). Intelligent Tutoring Goes to School in the Big City. *International Journal of AI in Education*, 8.
- Rosé, C. P. (2000). A framework for robust semantic interpretation. In Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics, pages 311–318.
- Rosé, C. P., Jordan, P., Ringenberg, M., Siler, S., VanLehn, K., & Weinstein, A. (2001). Interactive Conceptual Tutoring in Atlas-Andes, In J. D. Moore, C. L. Redfield, & W. L. Johnson (Eds.), *Artificial Intelligence in Education: AI-ED in the Wired and Wireless Future*, Proceedings of AI-ED 2001 (pp. 256-266). Amsterdam, IOS Press.
- Rosé, C. P., Gaydos, A., Hall, B., Roque, A., & VanLehn, K. (2003). Overcoming the Knowledge Engineering Bottleneck for Understanding Student Language Input, Proceedings of AI in Education, 2003
- Rosé, C. P. and Hall, B. (2004) A Little Goes a Long Way: Quick Authoring of Semantic Knowledge Sources for Interpretation, In Proceedings of the 2nd International Workshop on Scalable Language Understanding (ScaNaLu'04).
- Rosé, C. P., Torrey, C., Alevan, V., Robinson, A., Wu, C. & Forbus, K. (2004). CycleTalk: Towards a Dialogue Agent that Guides Design with an Articulate Simulator, Proceedings of the Intelligent Tutoring Systems Conference.
- Rosé, C. P., & Torrey, C. (2004). DReSDeN: Towards a Trainable Tutorial Dialogue Manager to Support Negotiation Dialogues for Learning and Reflection, Proceedings of the Intelligent Tutoring Systems Conference.
- Wang, Y. and Acero, A. (2003). Concept acquisition in example-based grammar authoring. In Proceedings of the International Conference on Acoustics, Speech, and Signal Processing.

Adaptive Tutorial Dialogue in AutoTutor

G. Tanner Jackson, Natalie K. Person, and Arthur C. Graesser

Institute for Intelligent Systems, University of Memphis, Memphis, TN 38152, USA
{gtjacksn, nperson, a-graesser} @memphis.edu

Abstract. AutoTutor is a computer program that is designed to tutor students in computer literacy or conceptual physics, by simulating the tutoring strategies used by human tutors. AutoTutor interacts with students and attempts to cover the topic material through the course of a natural conversation. During this interaction, AutoTutor uses specific dialog moves to: elicit knowledge from the student, correct student errors, and elaborate/summarize the covered material. Previous research on human tutoring has demonstrated the complex relationship between dialog moves and their effects on student learning gains. The current research with AutoTutor is aimed at examining the relationship between dialog moves and student learning as well as the program's ability to use dialog to model the student users.

Keywords: AutoTutor, tutorial dialogue, student modeling

1 Introduction

One-on-one tutoring is the most effective method of teaching. Although most tutors in schools are untrained, they are still very effective in producing learning gains [1, 3]. Some explanations for these positive effects are that the human tutoring involves: (a) the occurrence of interactive dialogue throughout the learning process, (b) collaborative construction of knowledge, (c) explanations, (d) concrete examples, and (e) questions that promote deep reasoning [4]. In depth analyses of tutorial interaction indicate that effective tutors do their best to elicit knowledge from students and force them to take a more active/constructive role in the learning process. This active role is achieved by organizing tasks in such a way that cause the student to ask questions, generate their own explanations, recognize their misconceptions, and synthesize information [2, 4]. Recently, ITSs have begun to incorporate these processes as part of their tutoring repertoire. This paper describes how one such system, AutoTutor, has been designed with adaptive dialogue that helps students actively contribute to their learning.

2 AutoTutor

AutoTutor is an intelligent tutoring designed by the Tutoring Research Group (TRG) at the University of Memphis [8]. AutoTutor interacts with users by having a natural language conversation with them. There are currently two versions of AutoTutor, one for computer literacy and one for conceptual physics. In AutoTutor, the dialog strategies are modeled to simulate those of untrained human tutors along with a few sophisticated teaching tactics [7]. The architecture of AutoTutor is supported by six major modules: 1) animated agent, 2) curriculum script, 3) latent semantic analysis (LSA) [5], 4) language analyzers, 5) dialog advancer network [6], and 6) dialog move generator. The agent uses synthesized speech, facial expressions, and hand gestures to communicate with the student. The curriculum script is basically a pre-made lesson plan set out with loosely ordered concepts and examples that are used to cover the topic material. LSA is used to assess the quality of student contributions, as well as to keep a track of topic coverage and student ability. The language analyzers are used to classify types of student contributions so that AutoTutor can respond adaptively to student answers, ques-

tions, and assertions. The dialog advancer network is a mechanism that is designed to increase the smoothness and clarity of the conversation by implementing discourse markers. This dialog network, in conjunction with the LSA assessment, helps determine the next AutoTutor dialog move: main question, short feedback, pump, prompt, hint, assertion, correction, or summary.

3 Dialog Moves in AutoTutor

AutoTutor considers three parameters to determine which dialog move to deliver: student assertion quality, student ability level, and topic coverage. These parameters are updated by LSA after every student contribution. The dialog moves included in AutoTutor (plus examples) are provided below.

- (1) Positive immediate feedback. "That's right" "Yeah"
- (2) Neutral immediate feedback. "Okay" "Uh-huh"
- (3) Negative immediate feedback. "Not quite" "No"
- (4) Pumping for more information. "What else?"
- (5) Hinting. "How does tossing the pumpkin affect horizontal velocity?"
- (6) Prompting for specific information. "Vertical acceleration does not affect horizontal _____."
- (7) Asserting. "Vertical acceleration does not affect horizontal velocity."
- (8) Corrections in the correct content after a student error. "Air resistance is negligible."
- (9) Summarizing. "So to recap, <succinct summary of answer to question>"

During the tutoring session AutoTutor scaffolds student learning by using a hint-prompt-assertion cycle. This cycle is designed to shift gradually the cognitive burden from student to the tutor. Specifically, AutoTutor first presents the student with a hint that requires the student to provide the majority of the information. If the student does not supply an adequate answer, AutoTutor prompts the student for more specific information (generally one or two words). If the student is unable to provide the information required by the prompt, AutoTutor generates an assertion. If the student provides a sufficient answer at any point during the dialog move cycle, AutoTutor exits the cycle and proceeds to the next tutoring topic. The relationship between dialog moves and student learning measures is the focus of these studies.

4 Evaluations of AutoTutor

We performed two studies to examine whether types of dialog moves are related to student learning outcomes. The dialog moves considered in these analyses were pumps, hints, prompts, and assertions. As mentioned earlier, some of these dialog moves require active student involvement (i.e., pumps and hints), whereas others require very little effort from the student (i.e., prompts and assertions). We expected that higher knowledge students would be more likely to provide correct answers to the initial pumps and hints; and therefore, less likely to need prompts and assertions from the tutor. Conversely, we expected lower knowledge students to rely more heavily on AutoTutor to supply the information via prompts and assertions. If these patterns are reflected in the correlations between dialog move proportions and student outcomes measures, it can be concluded that AutoTutor's dialog is adaptive to students' knowledge levels.

4.1 Studies

We conducted two studies, one for each version of AutoTutor (computer literacy and conceptual physics). College students were recruited from their respective introductory courses (computer literacy, $n = 70$; or physics, $n = 24$) and received extra credit for their participation. In each study, participants completed a pretest, interacted with AutoTutor, and completed a posttest. The pre- and posttests included multiple-choice questions that tapped deep levels of knowledge. AutoTutor did not utilize any information from the pretests during the tutoring sessions. The proportions of correct responses from the multiple-choice pre- and posttest questions were used as measures of student knowledge. A normalized learning gain score was computed as one measure of

student outcomes $((\text{Posttest} - \text{Pretest})/SD_{\text{pre}})$. A second learning gains measure was computed as an Estes score $((\text{Posttest} - \text{Pretest})/(1 - \text{Pretest}))$.

4.2 Predictions

The predictions for these studies are based on two main questions. First, does AutoTutor adapt to the prior knowledge levels of students? If AutoTutor adapts to students' prior knowledge, then higher knowledge students (higher pretest scores) should be able to answer most of the pumps and hints correctly. Hence, AutoTutor should not force them to progress through the remaining prompts and assertions for a particular topic. Students with lower prior knowledge (lower pretest scores), however, should have a higher proportion of prompts and assertions. If AutoTutor adequately adapts to student knowledge, then there should be a positive correlation between prior knowledge measures (i.e., pretest scores) and the proportion of pumps and hints that students receive and a negative correlation between prior knowledge measures and the proportion of prompts and assertions.

Second, does active knowledge construction by the student *during* tutoring lead to greater learning outcomes? Many educational researchers have reported that this is indeed the case [2, 4]. We attempted to address this question by examining the relation between different dialog move proportions and student outcome measures (posttest scores, normalized learning gains, and Estes scores). Pumps and hints are the two types of dialog moves that place most of the cognitive burden on the students, while prompts and assertions require AutoTutor to assume more of the burden. Therefore, students who received a higher proportion of pumps and hints should have constructed more knowledge on their own, and should therefore, have higher learning gains, whereas students who received a higher proportion of prompts and assertions constructed less information on their own and should have lower learning gains. Thus, there should be a positive correlation between learning gain measures (posttest scores, normalized learning gains, and Estes scores) and the proportion of pumps and hints, and a negative correlation between learning gains and the proportion of prompts and assertions.

5 Results and Discussion

Correlations were computed to address the predictions outlined in the previous section. The predictions for each question are addressed separately in the sections below.

5.1 Question 1: Does AutoTutor Adapt to the Prior Knowledge of Students?

Recall that prior domain knowledge was measured by pretests in both AutoTutor studies. Correlations between pretests scores and dialog move proportions were computed. The results are presented in Table 1. The directions of the correlations support the predictions made about prior knowledge and AutoTutor's dialog moves. Physics students with greater prior knowledge received more pumps and hints, and fewer prompts and assertions. Computer literacy students with higher prior knowledge received more pumps and fewer assertions. Of course, the exact opposite is true for students with less prior knowledge. Specifically, low knowledge students did not answer as many pumps and hints correctly, and therefore, received a greater proportion of prompts and assertions from AutoTutor.

Table 1. Correlations on proportions for prior knowledge and dialog move type

	Pump	Hint	Prompt	Assertion
COMPUTER LITERACY				
Pretest Score	-.035	.216*	.033	-.186
PHYSICS				
Pretest Score	.398*	.134	-.081	-.287

* $p < .10$, ** $p < .05$

5.2 Question 2: Does Active Knowledge Construction Lead to Greater Learning Outcomes?

Learning outcome measures are reported in three ways: posttest scores, normalized gains and Estes Scores. Correlations between these learning outcome measures and the dialog move proportions were computed. The results are reported in Table 2. The direction of these correlations is also consistent with previous research; lower learning gains occurred for students that constructed less knowledge. Specifically, students who received more pumps and hints (student controls knowledge construction) during the tutoring session learned more than those who received more prompts and assertions (tutor controls knowledge construction).

Table 2. Correlations on proportions for student learning and dialog move type

	Pump	Hint	Prompt	Assertion
COMPUTER LITERACY				
Posttest	.188	.466**	-.165	-.433**
Normalized Gains	.151	.200	-.120	-.198
Estes Scores	.131	.141	-.117	-.136
PHYSICS				
Posttest	.527*	.235	-.166	-.418*
Normalized Gains	.043	.103	-.096	-.089
Estes Scores	.026	.373*	-.322	-.290

* $p < .10$, ** $p < .05$

Adaptive tutorial dialog and effective user-modeling techniques are trendy research topics for ITS designers. It seems quite likely that the future of ITSs and their widespread adoption in school settings may depend on the success of their adaptive facilities and general user-modeling capabilities. In this work, we simply focused on the occurrence of individual dialog moves and how they are related to student prior knowledge and learning. Future research should also address how sequences of dialog moves promote learning, and perhaps, whether the tutorial dialog should adapt to multiple aspects of students. For instance, an ITS that could adapt dialog to students' affective states as well as their cognitive states could very well lead to greater learning outcomes than ITSs currently deliver.

6 Acknowledgements

The research on AutoTutor (www.autotutor.org) was supported by the National Science Foundation (SBR 9720314, REC 0106965, REC 0126265, ITR 0325428) and the DoD Multidisciplinary University Research Initiative (MURI) administered by ONR under grant N00014-00-1-0600. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DoD, ONR, or NSF. Kurt VanLehn, Carolyn Rose, Pam Jordan, Stephanie Siler, Dumisizwe Bhembe, and others at the University of Pittsburgh collaborated with us in preparing AutoTutor materials on conceptual physics.

References

1. Bloom, B. S.: The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-one Tutoring. *Educational Researcher* 1 (1984) 4-16
2. Chi, M. T. H.: Constructing Self Explanations and Scaffolded Explanations in Tutoring. *Applied Cognitive Psychology* 10 (1996) S33-S49
3. Cohen, P. A., Kulik, J. A., Kulik, C. C.: Educational Outcomes of Tutoring: A Meta-analysis of Findings. *American Educational Research Journal* 19 (1982) 237-248
4. Graesser, A. C., Person, N. K., & Magliano, J. P.: Collaborative Dialogue Patterns in Naturalistic One-to-one Tutoring Sessions. *Applied Cognitive Psychology* 9 (1995) 1-28

5. Graesser, A. C., Wiemer-Hastings, P., Wiemer-Hastings, K., Harter, D., Person, N. K., the TRG: Using Latent Semantic Analysis to Evaluate the Contributions of Students in AutoTutor. *Interactive Learning Environments* 8 (2000) 129-148
6. Person, N. K., Bautista, L., Kreuz, R. J., Graesser, A. C., the TRG: The Dialog Advancer Network: A Conversation Manager for AutoTutor. *Proceedings of the Workshop on Modeling Human Teaching Tactics and Strategies at the Intelligent Tutoring Systems 2000 Conference, Montreal* (2000) 86-92
7. Person, N. K., Graesser, A. C., Kreuz, R. J., Pomeroy, V., the TRG: Simulating human tutor dialog moves in AutoTutor. *International Journal of Artificial Intelligence in Education* 12 (2001) 23-39
8. Wiemer-Hastings, P., Graesser, A. C., Harter, D., the TRG: The foundations and architecture of AutoTutor. *Proceedings of the 4th International Conference on Intelligent Tutoring Systems, Springer-Verlag, Berlin* (1998) 334-343

Looking at the Student Input to a Natural-Language Based ITS

Chung Hee Lee¹, Martha W. Evens¹, and Michael S. Glass²

¹ Department of Computer Science, Illinois Institute of Technology
10 West 31st Street, Chicago, Illinois 60616, leechun@iit.edu, evens@iit.edu

² Department of Mathematics and Computer Science, Valparaiso University
Valparaiso, IN 46383, Michael.Glass@valpo.edu

Abstract. First year medical students at Rush Medical College used CIRCSIM-Tutor, a natural-language based ITS, in a physiology laboratory in November 2002. Analysis of the 66 hour-long machine sessions revealed many of the syntactic and spelling errors that we expected, along with a number of occasions when students did not understand the question that the tutor asked. In an attempt to encourage students to explain their causal reasoning, we added some open questions – the only student input that the system does not try to parse. This effort was at least partially effective. Almost all of the students answered some open questions seriously, and they produced much longer answers than to the ordinary tutor questions, though many stopped trying when they realized that the system was not trying to understand these answers. We also discuss some issues regarding student hedges and initiatives.

Key words: tutoring dialogue, student input, hedges, student initiatives, student affect

1. Introduction

Our original goal in building the CIRCSIM-Tutor system was to demonstrate that it is feasible to build a system that uses natural language interaction in solving problems as its main approach to tutoring and to demonstrate that this approach is effective. In fact, students show significant learning gains in pre- and post-tests; they report that they feel they learn from the system; they come back to the laboratory to use it on their own time; and they ask for copies to take home (Michael et al., 2003, Evens and Michael, to appear). In November 2002 we carried out an experiment comparing the learning gains made by students using the CIRCSIM-Tutor system with those made by students reading a carefully chosen and edited text. It showed, as we hoped, that the 40 students who used CIRCSIM-Tutor for an hour made significantly greater learning gains than the 33 who read the text. Actually 26 of the students in the control group chose to also come and use CIRCSIM-Tutor in the laboratory the next day or the day after. We wound up with 66 hour-long machine sessions. What is more, over 80% of the students completed all eight problems as compared to about 60% in the experiment in November, 1999. In this paper we describe the language input by the students in these 66 sessions and describe some of the problems in understanding and responding to this student language. Much of this language was very terse. Students tend to respond to focused questions with single words or phrases. There are abbreviations everywhere. When the students run out of official medical abbreviations, they invent their own. In Section 2 we present a fragment of a session and describe some of the recent changes made to the system. We provide some overall statistics about the number of student inputs and characterize the range of inputs to ordinary system questions in Section 3. Then in Section 4 we discuss spelling errors and spelling correction. In Section 5 we describe some of the open questions added to the system for this experiment and the student answers. In Section 6 we raise some issues involving hedges and initiatives.

2. A Snapshot of a Dialogue between Student and Machine

The system begins by asking the student to make predictions about the qualitative changes in seven important parameters of the cardiovascular system. The system marks any errors in red and launches a tutorial dialogue designed to help the student figure out what changes to make. The following is a short example:

T: What are the determinants of Stroke Volume?
S: CO, HR
T: It is true that $CO=SV*HR$, but what I am asking is what determines how much blood is ejected from the heart each time it beats (the sv)?
S: VENTRICULAR FILLING
T: Do you know what parameter in the prediction table determines filling?
S: CVP
T: Right, Central Venous Pressure determines filling, therefore determines Stroke Volume.
T: The other determinant is neurally controlled. What is the other determinant of Stroke Volume?
S: TPR
T: No, Total Peripheral Resistance is not a determinant. Stroke Volume is determined by Central Venous Pressure and Inotropic State. Which determinant is dominant in this case?

Before we ran the experiment we created a new version of CIRCSIM-Tutor, Version 2.9, that corrected many of the problems that appeared in the last major experiment in November, 1999. This version also gave more and better hints and asked a number of open questions, described below. The spelling correction routine, adapted from (Elmi, 1998), worked well on single words but not on phrases. We modified it to look for phrasal possibilities first.

When the system cannot understand the input, most commonly the student is lost and does not understand what the system was attempting to elicit. The system thus issues category messages. The five category messages listed below actually appeared in these sessions; the number in parentheses records the number of actual occurrences in the 66 sessions:

Please indicate increased, decreased, or unchanged. (18)
Is the mechanism of control neural or physical? (24)
Please respond with prediction table parameters. (61)
Please indicate a stage: DR, RR, or SS. (17)
Please indicate directly or inversely related. (9)

Reading the transcripts of the machine sessions from Fall, 1999, revealed that the system really short-changed the stronger students. When the student made no prediction errors the system provided no tutoring. It just proceeded to the next stage or next problem. Expert tutors, when faced with no errors to tutor, often ask open questions about the functioning of the baroreceptor reflex or ask the student to make generalizations about the problem-solving process. We had always avoided making the system ask such open questions for fear that it would not be able to parse the answers.

We decided to kill two birds with one stone: we could both provide a greater challenge to the student and collect linguistic data for extending the parser in the future by asking such open questions in the dialogue. The system, without parsing the answer, rolls out a “canned” expert answer in response. Although a canned response is a poor substitute for giving the tailored critique of the student answer, this would ensure that they see a correct answer. We thus obtained longer and richer dialogues with a large number of attested examples of lengthy student answers.

This ruse, asking questions without parsing the answers, was only partly successful. A number of students realized that the system was not parsing their answers and the result was some interesting testing behavior and some expressions of affect.

3. Overview of Student Inputs in November 2002

We obtained 66 transcripts from machine sessions on November 11 and 12, 2002, in a regularly scheduled laboratory. There were 40 students who had not been part of the control group doing the reading over the weekend. There were 33 in the control group. Most, but not all, of the students in the control group chose to come to the laboratory as well, so we wound up with 66 transcripts. Table 1 displays the number of student inputs, the number of open questions seen by this student, the number of error messages, the number of spelling errors and the number of problems completed by the student. During the laboratory session each student took a pretest, worked with CIRCSIM-Tutor, and then took a post-test. We measured learning gains from the differences between the pre-test and the post-test. These learning gains are not reported here.

We were pleased to see that there was a definite gain in terms of the number of problems completed. As shown in Table 2, 54/66 (or 81.2%) transcripts include all 8 problems. Five students did more than eight problems; that is they repeated problems, a phenomenon that we had never seen before. By comparison only 60% of the transcripts from Fall 1999 included all eight problems (21/35). The two students who did only one problem left the laboratory after about fifteen minutes. Our impression from observing students during the laboratory sessions is that the students who did fewer than eight problems were not forced to stop by our time limit but chose to stop because they felt they had learned what they could from the system.

Table 1. Statistics of Student Inputs in November, 2002

	Student Inputs	Open Qs Asked	Open Qs Answered	Category Messages	Spelling Errors	Problems Completed
Per Session	45.2	8.1	5.9	2.0	1.6	7.4
Total	2980	535	390	130	106	487

The total number of student inputs was 2980, so the average was 45.2 inputs per session. Because the focus of the dialogue is the baroreceptor reflex, the negative reflex system that controls blood pressure in the human body, many questions involve qualitative changes in the important parameters. Thus, appropriate student answers tend to involve parameter names, verbs of change and directional adverbs, adjectives that answer questions about whether a relationship is direct or inverse, and though, these are less frequent, answers to yes/no questions. The most common ways of indicating change are shown in Table 3.

Relationships between physiological parameters are typically typed by the student as “direct, dir, di, d, +, inverse, indirect, ind, inv, in, I,” and “- .” This means that we have some genuine ambiguity, since “I, in, +, d,” and “-” may sometimes tell us about a direction of change and sometimes about a relationship.

Table 2. Numbers of Problems Completed in 2002

No. of Problems Completed	No. of Students
1	2
2	0
3	0
4	1
5	2
6	3
7	4
8	49
more than 8	5
Total	66

Table 3: How Students Indicate Change

increased, inc, in, i, I, +, up
decrease, decreased, dec, de, d, D, -, down, less
unchanged, unch, unc, 0, o, no change, same

Our ability to interpret these answers is totally dependent on remembering what question the system asked. We implemented a small ontology (Glass, 1999, 2000, 2001) for answering mechanism questions, since students seem to use almost any component or aspect of the nervous system to indicate a neural mechanism. We are now trying to expand this ontology for use in reasoning about answers to open questions (Lee et al., 2002a, 2002b). Students sometimes spell out parameter names, as in “Central Venous Pressure,” more often they are abbreviated, e.g. “CVP”, or replaced by a synonym such as “preload.” When the system asks a question that involves several variables at once, the student sometimes types a list with “and” or commas or spaces or even an arithmetic operator as separator, so “SV HR” or “SV and HR” or “SV, HR” or “SVxHR” are all attested ways for specifying the two variables SV and HR.

4. Spelling Correction in CIRCSIM-Tutor

The students made 106 spelling errors – fewer than 2 per session. Over the last ten years their ability to type on a computer keyboard has improved markedly, making spelling correction a less overwhelming problem than it was when this project began. It is still important to any project like this, however. Students do not want to worry about making spelling corrections in the middle of solving a complex problem. The system corrected 104 out of these 106 spelling errors without making any miscorrections that we could identify, but it did miss two corrections that it should have made. It failed to correct “soconstriction” to “vasoconstriction” and “lood volume” to “blood volume.”

The tutor also rejected two answers that should have been recognized as correct, because of a lack of vocabulary. The system turned down “calcium” as a mechanism, when in fact calcium ions cause the Inotropic State to increase. We have now added “calcium” to the ontology of neural mechanisms, as well as “less” and “more” as meaning decrease and increase.

These 66 sessions contain 130 category messages or almost two per session, with the prediction table message by far the most frequent. Of these 130 messages 95 were correctly interpreted by the student, 35 were not. These 35 misinterpretations were all made by a set of 12 students. Some of these students seem to be trying to correct their answer to a previous question and to be so focused on this task that they did not read the category message. An expert human tutor would almost certainly recognize this situation and accept the correction with enthusiasm; our system should do this too.

5. Open Questions and Student Answers

The list of open questions added to CIRCSIM-Tutor for this experiment (Version 2.9) is shown in Table 4. During most of the dialogue the system parsed the student input and would not let a student get away without answering a question, but the system did not even attempt to parse the student responses to open questions. The system did not ask the student one of these questions until that student entered a complete column of correct predictions, so most students did not see one of these questions until fairly late in the session. The largest number of open questions that a student could see without repeating problems was 11. The number of open questions that the system asked is shown in Table 1; as are the number of serious answers provided by the students. As the table shows, 73% of open questions elicited serious responses; most of the students took these questions seriously at first and tried to answer them. Only two students did not answer any open questions at all. There were also 125 blank responses and 20 other responses. Some students gave up answering these questions when they realized that the system was not parsing this part of their input. Others continued to answer these questions to the end. We will describe the serious answers here and defer discussion of the others until the next section.

Table 4. List of Open Questions Asked by CIRCSIM-Tutor in November, 2002.

(DR, RR, and SS identify the phases in the problem-solving process where this question is appropriate.)

DR:	Why did you predict that IS would not change?
	Can you explain why HR did not change?
	Why did you enter 'no change' for TPR?
RR:	Why did you predict that IS would increase?
	Why did you predict that HR would increase?
	Why did you predict that TPR would increase?

	Why did you predict that CVP would decrease?
	Why did MAP change in the manner that you predicted?
SS:	SV increased in DR and decreased in RR. Why did you predict that it would increase in SS?
	CO decreased in DR and increased in RR. Why did you predict that it would decrease in SS?
	Why is MAP still decreased/increased? <whichever applies>
	What does the baroreceptor reflex do?
	CVP increased in DR and decreased in RR. Why did you predict that it would increase in SS?

The behavior of Student M48 was typical. He gave a serious try at answering the open questions, as illustrated below. The first, second, and fourth answers are pretty good; the third is not so good.

T: What does the baroreceptor reflex do?

S: try and maintain MAP

T: Why did you enter 'no change' for TPR?

S: it is under autonomic control

T: Why did MAP change in the manner that you predicted?

S: cardiac output and hr increase3d

T: Why did you predict that CVP would decrease?

S: co increased

These answers are far from verbose; they are definitely shorter than the ideal answers written by the professor and reeled out by the system, but they certainly display more syntactic sophistication (three sentences and a verb phrase) than this student's answers to ordinary questions. What is more, they potentially reveal much more about the student's thought processes.

There were 47 answers to the first question listed in Table 4. There were 11 complete sentences ("There has not been a baroreceptor reflex yet"), 7 complete because clauses ("Because only the baroreceptor firing rate directly affects IS."), 7 noun phrases ("part of the baroreceptor reflex"), 12 responses consisting of a negative plus a noun phrase ("no reflex", "no reflex response yet"), 3 prepositional phrases ("under neural control"), and 7 other fragments ("baroreceptor not yet activated"). It is clear that we need to try to parse these inputs and determine how to respond. One option is to add more cascaded finite-state machines to the existing parser, since although these inputs are syntactically and semantically more complex than those that CIRCSIM-Tutor is parsing now, they are relatively short. We could also look at an LSA approach like that used in Auto-Tutor (Graesser et al., 1998; Person et al., 2000, 2001). Lee (2003, 2004) is working on an HPSG parser to possibly handle this problem.

6. Hedges and Student Initiatives

Students hedge to human tutors all the time. They stick in "perhaps" or "maybe" or "I think" or "I guess"; they add question marks to declarative sentences. Bhatt et al. (2004) identified 218 hedges (151 hedged answers and 67 hedged initiatives) in 25 human tutoring sessions from November 1999.

All the students hedged but the number of hedges in a session varied widely from 2 in one session to 22 in another. During an ITS session at a workshop at ACL 2001 there was a discussion of the possibility that hedges might provide useful clues to models of student knowledge or student affect. This seems unlikely since Bhatt et al. have found that, although hedged answers are more likely to be in error than answers that are not hedged, more than half of hedged answers are, in fact, correct.

These results seem to confirm the perceptions of our expert tutors. After the first eight tutoring sessions in 1989, Michael and Rovick decided to stop responding to hedges on the grounds that hedges seemed to say more about the student's preferred style of communication than about the state of the student's knowledge of the subject

matter. They have continued to respond in those cases where the student indicated some serious distress or confusion, however.

But as much as students hedge during dialogue with human tutors, we have not before observed them to hedge when conversing with the computer tutor. Although we have analyzed a number of issues involved in parsing hedges (Glass, 1999), and they are theoretically ignored during parsing of normal answers, we have no experience with them. Thus we were startled by this exchange:

```
T: SV increased in DR and decreased in RR. Why
    did you predict that it would increase in SS?
S: 9/10 times the dr will dominate because the
    rr can't bring all the way back
```

This answer is semantically correct, if syntactically incomplete, except that the quantifier/qualifier “9/10 times” is not justified by anything that the student has seen in the course. The student’s answer is hedged.

We are left with some interesting questions. Is hedging really an expression of uncertainty or is an interpersonal expression of politeness or deference – politeness and deference that are due to human tutors but not to machines. Have students avoided hedging to CIRCSIM-Tutor because its language performance does not match human standards? Can we expect to see more hedges as the system’s conversational performance improves? Is hedging really an unconscious expression of uncertainty or is it a conscious conversational move that expresses an interpersonal relationship?

Student initiatives are defined by Shah et al. (2002) as inputs by the student that are not intended to answer the question asked by the tutor. In the human tutoring sessions such inputs are typically explanations by the student or questions about the physiology or even challenges to whatever the tutor last said. These inputs are intended to take over the course of the dialogue, so they are truly conversational initiatives as well. In the machine sessions, in addition to many blank answers, there were twenty inputs from students in answers to open questions that were not apparently intended as serious answers to the question; all are listed in Table 5. They seem to be expressions of affect or tests of the system by the student intended to verify that the system is not paying attention to these inputs. The open question “Why did you enter no change for TPR?” received the answers: “You know why” and “Nimesh said so.” The question “Why is MAP still decreased?” was answered with “I don't want to tell you.” and “Blalal.” Another student answered an open question with the canned answer for the previous question and then answered the next question by telling the system that she knows all. Recent work on ITS emphasizes the importance of understanding and responding to student affect (Aist et al., 2002; Vicente and Pain, 2000).

Should we call these inputs initiatives or not? They are initiatives in the sense that they are not intended to carry the conversation forward in the direction that the tutor is going, but they are not intended to take over the direction of the dialogue, either – they are produced only because the student believes that they will not be understood. These inputs raise another difficult question. How can the system recognize them and how should the system respond to them when it does recognize them?

7. Conclusion

In many ways the results of the 2002 experiment were very encouraging. The system did not get caught in any of the confusions that turned up in earlier experiments. The students made impressive learning gains and expressed enthusiasm in the survey. None of the students felt impelled to curse the system. We would like to believe that this was because the system is definitely less frustrating to use, but it may just have been due to the number of observers present.

This experiment left us with a number of problems in system implementation, however. How can we improve the language produced by the system so students will venture on more initiatives? How can we parse answers to open questions and any initiatives that students produce. How can we recognize expressions of student affect and how should we respond to them? Beyond the level of our own system problems, we wonder about the role of hedges in tutoring sessions.

Acknowledgments

This work was partially supported by the Cognitive Science Program, Office of Naval Research under Grant 00014-00-1-0660 to Stanford University as well as Grants No. N00014-94-1-0338 and N00014-02-1-0442 to Illinois Institute of Technology. The content does not reflect the position or policy of the government and no official endorsement should be inferred.

Table 5. List of Nonserious Answers to Open Questions.

(Each entry contains a session number, the tutor question and the entire student response. In session T56 the student typed the response given by the system to the previous open question.)

M52 T: Why did you predict that IS would not change?	S: 0
M52 T: Why did you predict that IS would increase?	S: 1
M52 T: What does the baroreceptor reflex do?	S: 1
M52 T: Can you explain why HR did not change?	S: no
M59 T: What does the baroreceptor reflex do?	S: nothing
M65 T: Can you explain why HR did not change?	S: +
M67 T: Why did you enter 'no change' for TPR?	S: BC
T48 T: Why did you enter 'no change' for TPR?	S: you know why.
T48 T: Can you explain why HR did not change?	S: yes, i can.
T48 T: Why is MAP still decreased?	S: I don't want to tell you.
T56 T: Why did MAP change in the manner that you predicted?	S: If we think about what we mean by DR -- a period of time BEFORE any change in neural activity has occurs, then we can see that heartrate will not change because HR is determined by sympathetic and parasympathetic activity to the hear.
T60 T: Why did MAP change in the manner that you predicted?	S: In other words, <student's own name> knows all...
T60 T: Why did you predict that IS would not change?	S: it
T65 T: CO decreased in DR and increased in RR. Why did you predict hat it would decrease in SS?	S: dr
T74 T: Why is MAP still decreased?	S: blalaal
T76 T: Why did you enter 'no change' for TPR?	S: the TPR can
T79 T: Why did you predict that IS would not change?	S: whatever
T81 T: Why is MAP still decreased?	S: asdf
T81 T: What does the baroreceptor reflex do?	S: t
T81 T: Why did you enter 'no change' for TPR?	S: Nimesh said so

References

- Aist, G., Kort, B., Reilly, R., Mostow, J., and Picard, R. (2002). Experimentally augmenting an intelligent tutoring system with human-supplied capabilities: Adding human-provided emotional scaffolding to an automated reading tutor that listens. *ITS 2002 Workshop on Empirical Methods for Tutorial Dialogue Systems*, San Sebastian, Spain.
- Bhatt, K., Agamon, S., and Evens, M. (2004). Hedged responses and expressions of affect in human/human and human/computer tutorial interactions. *COGSCI 2004*, Chicago, IL.
- Elmi, M., and Evens, M. (1998). Spelling correction using context. *Proceedings of COLING 98*, Montreal, Canada. 360-364.

- Evens, M. and Michael, J. (to appear). *One-on-One Tutoring by Man and Machine*. Erlbaum.
- Glass, M.S. (1999). *Broadening input understanding in a language-based intelligent tutoring system*. Unpublished Ph.D. Dissertation, Computer Science Department, Illinois Institute of Technology, Chicago, IL, May.
- Glass, M.S. (2000). Processing language input in the CIRCSIM-Tutor intelligent tutoring system. In C.P. Rosé & R. Freedman (Eds.) *Proceedings of the AAI Fall Symposium on Building Dialogue Systems for Tutorial Applications*, Menlo Park, CA: AAAI Press. 74-79.
- Glass, M.S. (2001). Processing language input for an intelligent tutoring system. In J.D. Moore, C.L. Redfield, & W.L. Johnson (Eds.), *Proceedings of Artificial Intelligence in Education*. Amsterdam:IOS Press. 210-221.
- Graesser, A.C., Franklin, S., Wiemer-Hastings, P. (1998). Simulating smooth tutorial dialogue with pedagogical value. *Proc. FLAIRS 98*. Sanibel, Island, FL. 163-167.
- Lee, C.H., Seu, J.H., and Evens, M. (2002a). Building an ontology for CIRCSIM-Tutor. *Proc. MAICS 2002*. 161-168.
- Lee, C.H., Seu, J.H., and Evens, M. (2002b). Automating the construction of case frames for CIRCSIM-Tutor. *Proc. ICAST 2002*. 59-65.
- Lee, C.H., and Evens, M.W. (2003). Interleaved syntactic and semantic processing for CIRCSIM-Tutor dialogues. *Proceedings of the Midwest Artificial Intelligence and Cognitive Science Conference MAICS'03*, pp. 69-73. Cincinnati, OH.
- Lee, C.H., and Evens, M. (2004). Using selectional restrictions to parse and interpret student answers in a cardiovascular tutoring system. *Proceedings of MAICS 2004*, Schaumburg, IL. 63-67.
- Michael, J., Rovick, A., Glass, M., Zhou, Y., and Evens, M. (2003). Learning from a computer tutor with natural language capabilities. *Interactive Learning Environments*. 11(3) 233-262.
- Person, N.K., Graesser, A.C., Harter, D.C., Mathews, E.C. & the Tutoring Research Group, (2000). Dialog move generation and conversation management in Auto-Tutor. In C.P. Rosé & R. Freedman (Eds.) *Proceedings of the AAI Fall Symposium on Building Dialogue Systems for Tutorial Applications*, Menlo Park, CA: AAAI Press. 87-94.
- Person, N.K., Bautista, L., Graesser, A.C., Mathews, E.C., & The Tutoring Research Group. (2001). Evaluating student learning gains in two versions of Auto-Tutor. In J.D. Moore, C.L.Redfield & W.L. Johnson (Eds.), *Proceedings of Artificial Intelligence in Education*. Amsterdam: IOS Press. 246-255.
- Shah, F., Evens, M.W., Michael, J.A., & Rovick, A.A. (2002). Classifying student initiatives and tutor responses in human tutoring keyboard to keyboard tutoring sessions. *DiscourseProcesses* 33(1) 23-52.
- Vicente, A., and Pain, H. (2000). A computational model of affective educational dialogues. In C.P.Rosé & R. Freedman (Eds.) *Proceedings of the AAI Fall Symposium on Building Dialogue Systems for Tutorial Applications*. Menlo Park, CA: AAAI Press. 113-121.

Evaluating the Effectiveness of SCoT: A Spoken Conversational Tutor

Heather Pon-Barry, Brady Clark, Elizabeth Owen Bratt, Karl Schultz and Stanley Peters

Center for the Study of Language and Information
Stanford University
Stanford, CA 94305 USA
{ponbarry, bzack, ebratt, schultzk, peters}@csli.stanford.edu

Abstract. SCoT is a tutorial dialogue system that engages students in natural language discussions through a speech interface. The current instantiation, SCoT-DC, is applied to the domain of shipboard damage control—the task of containing the effects of crises (e.g. fires) that occur aboard Navy vessels. This paper describes a recent evaluation of SCoT-DC and presents preliminary results showing: (1) the effectiveness of SCoT-DC as a learning tool, and (2) that speech recognition technology is mature enough to support use in tutorial dialogue systems.

Keywords: Intelligent tutoring systems, spoken dialogue, evaluation, empirical results, speech technology, shipboard damage control

1 Introduction

There seems to be an assumption within the ITS community that the current state of speech technology is too primitive to provide effective tutorial interaction. Although some tutoring systems have been making use of spoken input for years (Aist & Mostow, 1997), only recently have researchers begun incorporating spoken input into dialogue-based tutoring systems (Clark et al., 2001; Litman & Silliman, 2004). The majority of tutorial dialogue systems currently rely on typed input from the student (e.g., Graesser et al., 2000; Evens et al., 2001; Heffernan & Koedinger, 2002).

SCoT, a Spoken Conversational Tutor, was developed in order to investigate the advantages of spoken language interaction in intelligent tutoring systems. While it has yet to be shown whether spoken tutorial dialogue systems can be more effective than typed tutorial dialogue systems, arguments involving access to prosodic and temporal information as well as multi-modality have been made in favor of spoken dialogue (Litman & Forbes, 2003; Pon-Barry et al., 2004). Furthermore, a recent study comparing spoken versus typed tutoring found significantly greater learning gains in the spoken condition when the tutor was a human, but little difference between the two conditions when the tutor was a computer (Litman et al., 2004)—suggesting that there is an advantage to spoken interaction, but that current tutorial dialogue systems are still far from human-like levels of sophistication.

In the winter and spring of 2004, we ran an experiment at Stanford University to evaluate the effectiveness of SCoT. In particular, the following two hypotheses were tested and confirmed:

- (1) Tutorial interactions with SCoT-DC (the current instantiation of SCoT) will help students learn shipboard damage control
- (2) Speech recognition, when combined with current technology for natural language understanding and dialogue management, is accurate enough to support effective spoken tutoring interactions

This paper is organized as follows. Section 2 gives an overview of how SCoT-DC works, Section 3 describes the experimental procedure, Section 4 presents empirical results, and Section 5 offers some conclusions.

2 System Overview

SCoT-DC, the current instantiation of the SCoT tutoring system, is applied to the domain of shipboard damage control. Shipboard damage control refers to the task of containing the effects of fires, floods, explosions, and other critical events that can occur aboard Navy vessels. Students carry out a reflective discussion with SCoT-DC after completing a problem-solving session with DC-Train (Bulitko & Wilkins, 1999), a fast paced, real time, speech-enabled training environment for damage control. Figure 1 shows a screenshot of DC-Train.

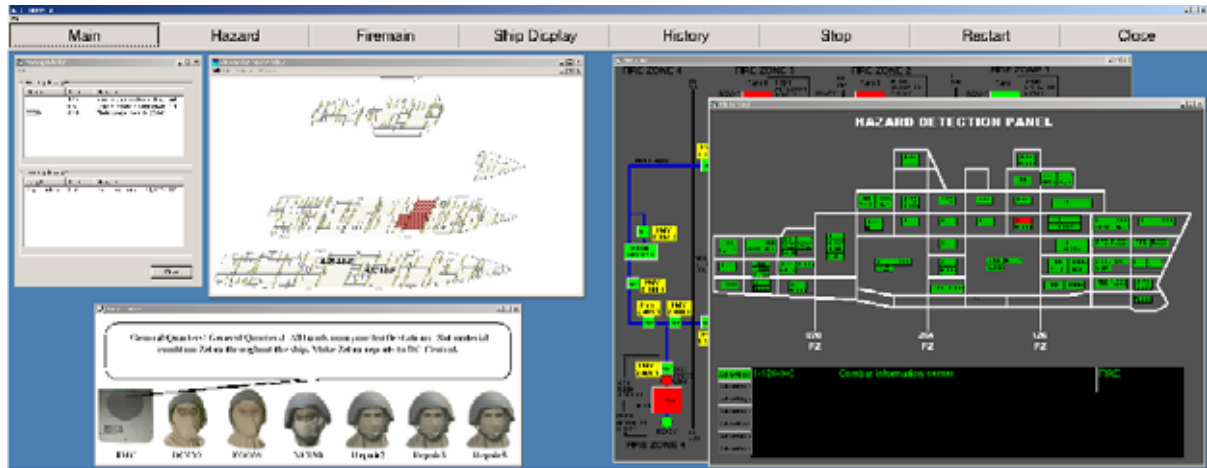


Figure 1. DC-Train Simulator

Figure 2 shows a screenshot of SCoT-DC. The window on the right depicts the multiple decks of the ship; the window in the bottom left corner contains a history of the tutorial dialogue as well as buttons for starting the tutor; and the window in the upper left corner is the *common workspace*—a space where both the student and the tutor can zoom in or out, and select (i.e. point to) compartments, regions, or bulkheads (lateral walls in the ship).

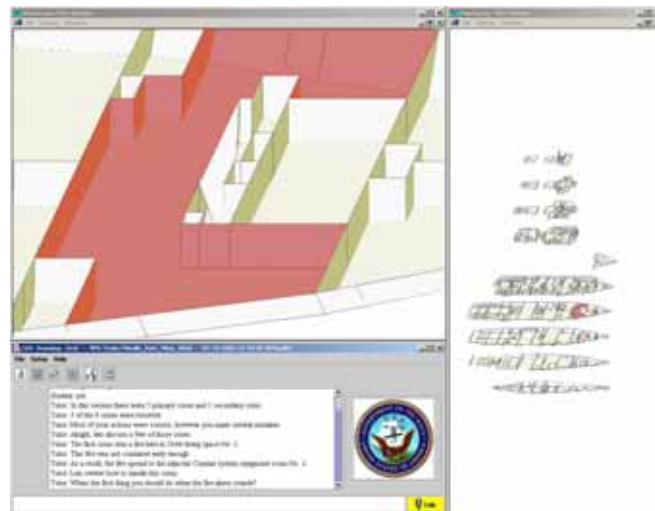


Figure 2. SCoT-DC Tutor

SCoT is developed within the Architecture for Conversational Intelligence (Lemon et al., 2002), a general purpose architecture which supports multi-modal, mixed-initiative dialogue. The version of SCoT used in these experiments consists of three separate components: a dialogue manager, a tutor, and a set of natural language tools. These three components are described briefly in sections 2.1 through 2.3. A more detailed description can be found in Schultz et al. (2003).¹ Figure 3 is the overall architecture of our system.

¹ Also, a downloadable video is available at www-csli.stanford.edu/semlab/muri/November2002Demo.html

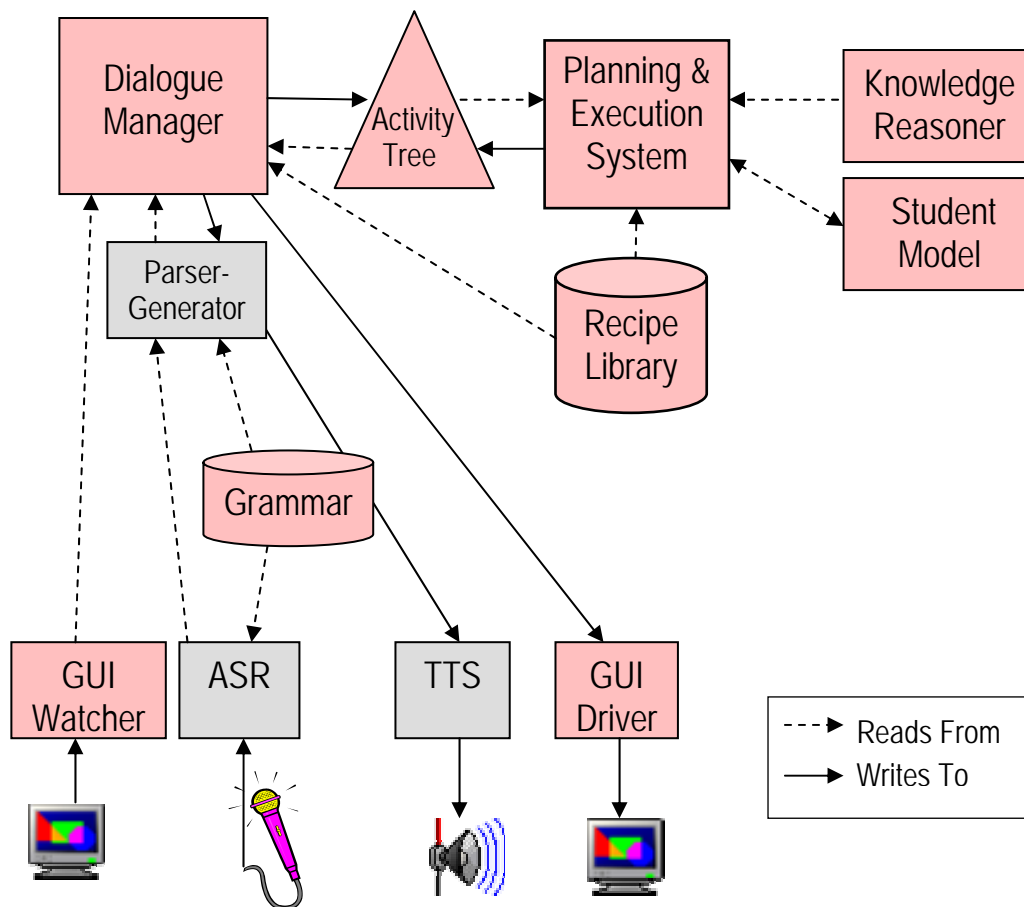


Figure 3 SCoT-DC Tutor architecture.

2.1 Dialogue Manager

The dialogue manager mediates communication between the system and the user by handling aspects of conversational intelligence such as turn management and coordination of multi-modal input and output. It contains multiple dynamically updated components—the two main ones are (1) the *dialogue move tree*, a structured history of dialogue moves, and (2) the *activity tree*, a hierarchical representation of the past, current, and planned activities initiated by either the tutor or the student. In SCoT, each activity initiated by the tutor corresponds to a tutorial goal; the decompositions of these goals are specified by activity recipes contained in the *recipe library* (see section 2.2).

2.2 Tutor

The tutor component contains the tutorial knowledge necessary to plan and carry out a flexible and coherent tutorial dialogue. It is implemented as a set of domain-general tutorial activities (the *recipe library*) and a system for planning and executing these activities. The *recipe library* contains activity recipes that specify how to decompose a tutorial activity into other activities and low-level actions. The *planning and execution system* uses information from external knowledge sources to (1) generate an initial plan at the start of the dialogue, (2) decide when and how to revise these plans during the dialogue, (3) classify student utterances, and (4) decide how to respond to student answers, i.e., which recipe to put on the activity tree. A high-level activity recipe for discussing a partially-correct answer is shown in Figure 4, followed by the resulting dialogue in Figure 5 (dialogue turns resulting from recipe are shown in boldface). The recipe in Figure 4 decomposes the activity of discussing a partially-correct answer into four sub-activities (acknowledge, hint, appraise, and generalize). The tutor places these activities on the activity tree, and the dialogue manager begins to execute their respective recipes.

<p>Goal</p> <ul style="list-style-type: none"> • Discuss partially-correct answer/error of omission <p>Preconditions</p> <ul style="list-style-type: none"> • Student Answer (from dialogue) = Partially Correct • Student Action (from DC-Train) = Error of Omission <p>Recipe Body</p> <ul style="list-style-type: none"> • Acknowledge: <i>Muted Praise</i> • Hint: <i>Partially Correct Hinting Sequence</i> • Appraise: <i>State Error of Omission</i> • Generalize: <i>Comment on Repeated Errors</i>
--

Figure 4. Activity recipe for discussing a problem solving sequence

Tutor: All right, suppose the fire has been overhauled.
Tutor: Let's now discuss the resulting smoke. [displays grey color in compartment].
Tutor: What should you do first?
Student: Send repair two to desmoke the compartment.
Tutor: Almost.
Tutor: You should send a different repair team.
Tutor: Which team should you send instead?
Student: Repair five.
Tutor: Yes, exactly. You should order repair five to desmoke the compartment.
Tutor: Repair 2 was overtasked, so repair 5 should have been sent.
Tutor: But you sent the wrong repair team during the session.
Tutor: This is a mistake you made more than once.

Figure 5. Tutorial dialogue resulting from activity recipe in Figure 4

2.3 Natural Language Components

The natural language components which make the spoken dialogue possible include a bi-directional unification grammar and off-the-shelf tools for automatic speech recognition and text-to-speech synthesis. Incoming student utterances are handled by SCoT in the following way. First, the utterance is recognized using Nuance² speech recognition, which uses a grammar compiled from a Gemini natural language understanding grammar. Gemini (Dowding et al., 1993) translates word strings from Nuance into logical forms, which the dialogue manager interprets in context and routes to the tutor. The system responds to the student via a FestVox³ limited domain synthesized voice.

3 Experiment

This experiment was designed to test the hypothesis that SCoT-DC will help students learn damage control. In order to test this hypothesis, we divided the tutorial content into three knowledge areas: sequencing, boundaries, and jurisdiction. Sequencing refers to giving orders for actions in response to crises (e.g. fires, floods) at the correct times. Setting boundaries refers to the task of correctly specifying six parameters that determine the location of the bulkheads (upright partitions that separate ship compartments) that need to be cooled or sealed to prevent a crisis from spreading. Jurisdiction refers to the task of giving orders to the appropriate personnel on the ship—personnel are assigned to different regions such as forward, aft, and midship.

² <http://www.nuance.com>

³ <http://festvox.org>

3.1 Participants

Thirty native English speakers were recruited to participate in this experiment (16 male, 14 female). All subjects were novices in the domain of damage control, twenty-nine had no prior experience in dialogue system studies.

3.2 Experimental design

Subjects were randomly assigned to three groups. All groups ran through the same four DC-Train scenarios (which increased in difficulty). Between each DC-Train session, all groups received tutoring in one of the three knowledge areas (sequencing, boundaries, and jurisdiction), but at different times. For example, group I received tutoring on sequencing between scenario 1 and scenario 2, group II received tutoring on sequencing between scenario 3 and scenario 4, and group III received tutoring on sequencing between scenario 2 and scenario 3. This allowed us to separate learning gains due to the tutorial interaction from learning gains due to practice alone. In this way, each subject served as their own control, and all groups served as a comparison for each other. Table 1 shows the layout for each group.

Subject Group	DC-Train Session	SCoT-DC Tutoring	DC-Train Session	SCoT-DC Tutoring	DC-Train Session	SCoT-DC Tutoring	DC-Train Session
I	Scenario 1	<i>Sequencing</i>	Scenario 2	<i>Boundaries</i>	Scenario 3	<i>Jurisdiction</i>	Scenario 4
II	Scenario 1	<i>Boundaries</i>	Scenario 2	<i>Jurisdiction</i>	Scenario 3	<i>Sequencing</i>	Scenario 4
III	Scenario 1	<i>Jurisdiction</i>	Scenario 2	<i>Sequencing</i>	Scenario 3	<i>Boundaries</i>	Scenario 4

Table 1. Experiment design

Learning was measured in two ways. Firstly, general knowledge was tested in the form of a multiple-choice pre-test and a post-test. Secondly, and crucially, quantitative performance measures were drawn from each of the four DC-Train scenarios. Based on the logfiles of each scenario, students were scored for their performance in sequencing, boundaries, and jurisdiction.

3.3 Procedure

The experimental procedure is illustrated below in Table 2. Steps 4 through 10 (shown in boldface) constitute the main body of the experiment and correspond to the steps listed in Table 1. In addition to these main steps, all subjects went through an interactive multimedia introduction to (1) familiarize them with DC-Train and basic damage control knowledge, and (2) give them practice using the speech recognition interface. After the multimedia introduction, subjects took a 20 question multiple-choice pre-test, and had one practice DC-Train session. Following the main body of the experiment, subjects took a 20 question post-test and filled out a questionnaire. The total duration of the experiment was roughly three hours per subject.

Step 1	Multimedia Introduction	30-40 min
Step 2	Pre-test	5-10 min
Step 3	Practice DC-Train session	10 min
Step 4	DC-Train session 1	15 min
Step 5	Tutoring	< 15 min
Step 6	DC-Train session 2	15 min
Step 7	Tutoring	< 15 min
Step 8	DC-Train session 3	15 min
Step 9	Tutoring	< 15 min
Step 10	DC-Train session 4	15 min
Step 11	Post-test	5-10 min
Step 12	Questionnaire	< 5 min

Table 2. Experiment Procedure

4 Results

4.1 Hypothesis (1) – Effectiveness of Tutoring

Students showed improvement both on the written test and on their performance in the DC-Train scenarios. Every student earned a higher score on the post-test than on the pre-test, and the mean post-test score (84%) was significantly higher (Paired-Samples T Test: $p = 0.000$) than the mean pre-test score (67%). Looking at performance in the DC-Train simulator, students were performing better in their fourth session with the simulator than in the first for two of the three knowledge areas (sequencing and boundaries). These performance gains are shown in Figure 6.

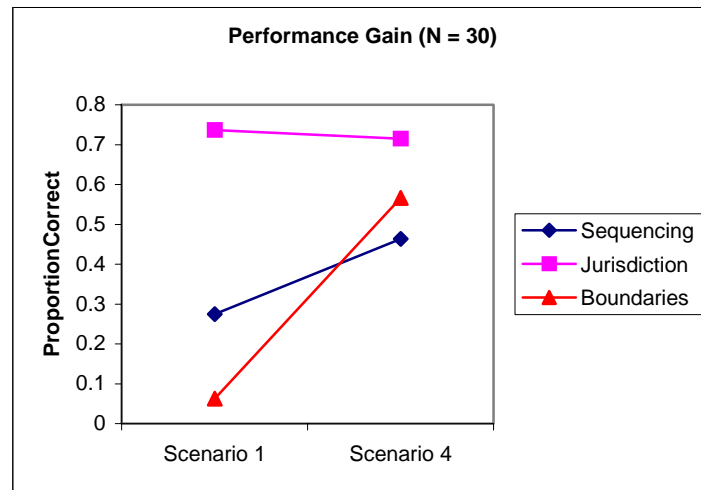


Figure 6. DC-Train performance gains

The lack of improvement in the area of jurisdiction may be due to a ceiling effect—because initial performance in jurisdiction was already very high. However, there are clear performance gains in the areas of sequencing and boundaries. This leads to the question of whether these gains should be attributed to the SCoT-DC tutoring or just to improvement over time with practice on the DC-Train simulator.

Because all three groups received tutoring in all three knowledge areas (at different times), we can separate gains due to SCoT tutoring from gains due to practice alone. We do this by comparing, for a particular knowledge area, performance gains across sessions with tutoring on that knowledge area to performance gains across sessions with tutoring on some other knowledge area. For example, consider the graphs in Figure 7. In Figure 7a, the left column depicts the average gain across two DC-Train scenarios between which the student received tutoring on sequencing (i.e. for subject group I: between scenario 1 and scenario 2, for subject group II: between scenario 3 and scenario 4, for subject group III: between scenario 2 and scenario 3). The right column depicts the average gain across sessions between which the student received tutoring on either boundaries or on jurisdiction. Figures 7b and 7c show the same data, but for boundaries and jurisdiction respectively.

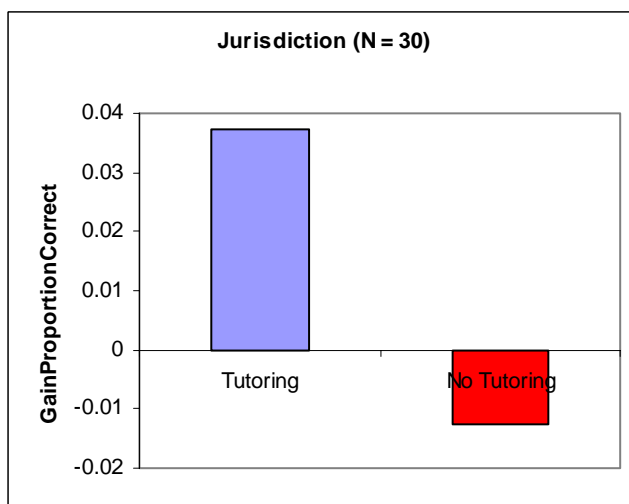
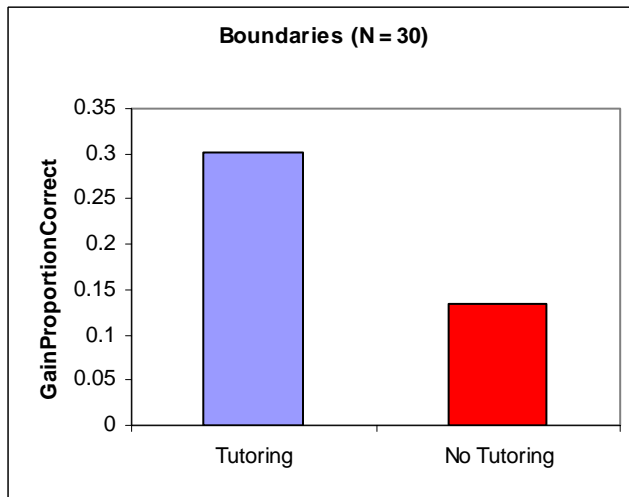
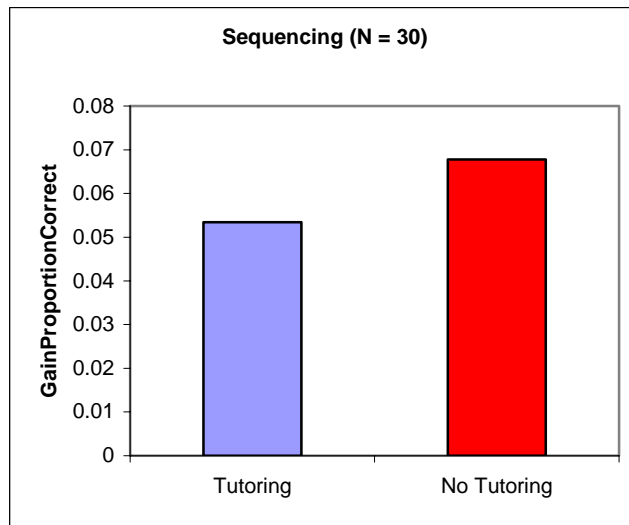


Figure 7. Gains due to tutoring vs. gains due to practice for (a) sequencing, (b) boundaries, and (c) jurisdiction

Figure 8 shows the average gains due to tutoring and due to practice (no tutoring) across all three knowledge areas. On average, gains in performance after being tutored in a particular knowledge area are over twice as high as gains in performance after being tutored in some other knowledge area.

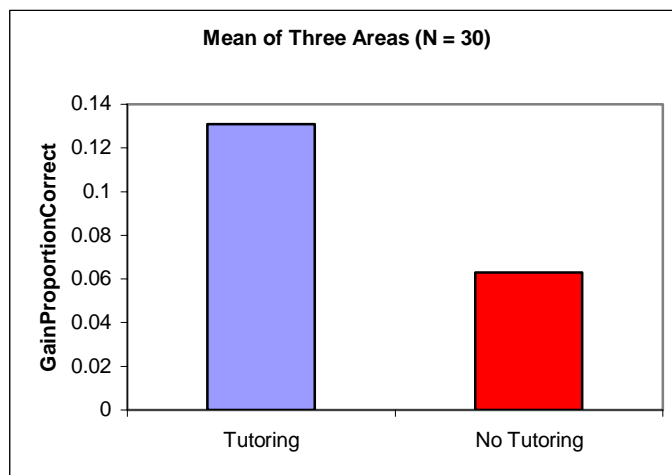


Figure 8. Gains due to tutoring vs. gains due to practice for all three knowledge areas

We have already seen that jurisdiction performance did not improve much with use of the system, perhaps because it started at such a high level, so the small difference seen in Figure 7c between the means of 0.04 (stdev=0.40) and -0.01 (stdev=0.21) in jurisdiction performance may not be as important to consider as the areas in which students did improve.

In the area of boundaries, the mean performance gain with tutoring (0.30) is close to one standard deviation above the performance gain with tutoring (0.13, stdev=0.21). This statistic gives the clearest evidence of the benefit of tutoring combined with simulator practice.

The fact that in the area of sequencing subjects had smaller average performance gains with tutoring (0.05, stdev=0.27) than those without tutoring (0.06, stdev=0.10) seems to argue against the effectiveness of the tutor. However, on examining the performance of each subject group separately, an interesting pattern emerges. Subjects who received sequencing tutoring first (group I) showed larger gains in sequencing in the following DC-Train scenario than they did in their subsequent scenarios (0.13 with tutoring, 0.03 with no tutoring). However, the other two subject groups, who received sequencing tutoring later, did not appear to benefit from the tutoring (group II: -0.01 with tutoring, 0.12 with no tutoring; group III: 0.04 with tutoring, 0.05 with no tutoring). A possible explanation is that the subject matter of sequencing is fundamental to performance on DC-Train and that it is critical to be tutored on it early on. Allowing students to practice their mistakes before reviewing the correct actions may lead them into habits that are hard to unlearn. We hope that future experiments may clarify whether this explanation holds.

4.2 Hypothesis (2) – Effectiveness of Speech Interface

We collected the following statistics on speech recognition performance and on speech quantity:

- percentage of words correctly recognized
- percentage of sentences recognized with no word errors
- percentage of sentences rejected by the speech recognizer
- mean length of utterance in words
- mean number of utterances per session

Twenty-six speakers have been analyzed so far.

Our main hypothesis was that speech recognition is accurate enough to support effective spoken tutoring interactions. We predicted that speech recognition performance would correlate with improvements on the written tests and with performance in simulator. We found no correlation with either one. For test score gains, the correlation with percent words correct is -0.266 (p=0.190), and the correlation with rejection rate is 0.251

($p=0.215$). For overall performance in the simulator, the correlation with percentage words correct is -0.250 ($p=0.218$), and the correlation with rejection rate is -0.159 ($p=0.437$). Figure 9 shows a scatter-plot of average percent words correct versus the gains from the pre-test to the post-test.

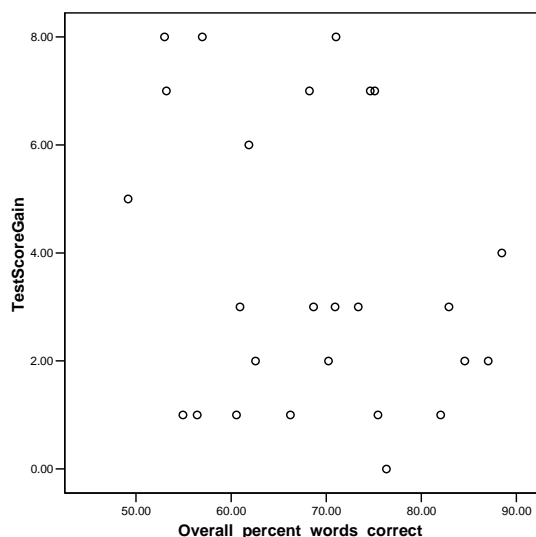


Figure 9. Scatter-plot of speech recognition success vs. gains in test scores.

Although our prediction was not validated, this result is highly relevant because it suggests that poor speech recognition does not diminish learning. Students who had only 50% of their words recognized correctly showed learning gains comparable to students who had 75% of their words recognized correctly. Even though speech recognition is far from perfect, students being tutored by SCoT-DC improved regardless of the number of misrecognitions they encountered. This result is in line with recent findings reported in (Litman et al., 2004).

While speech recognition accuracy did not affect learning, it did affect the student's desire to use the system regularly. As part of the questionnaire, subjects were asked to rate the following statement on a 1-7 Likert scale (1 = strongly disagree, 7 = strongly agree):

“Based on my experience using this tutoring system, I would like to use this kind of automated tutoring system regularly.”

Subjects said they would like to use this kind of system again when they had a higher percentage of sentences with no errors (Pearson correlation = 0.557, $p < 0.05$ level (2-tailed)). However, there was no correlation with percentage words correct, and paradoxically, high rejection rates also correlated with more desire to use the system (Pearson correlation = 0.629, $p < 0.01$ level (2-tailed)).

4.3 Future Analysis

In the future, we are interested in adding semantic error rate to the speech performance statistics, to see how many of the speech recognition errors made a difference in the interpretation the system assigned to the sentence. Semantic errors are the subset of word and sentence errors that result in the system misunderstanding the user (assigning an incorrect logical form to the utterance). For example, if the student says “send repair five to set fire boundaries” and the recognition hypothesis is “send repair five to set fire boundary”, it would be a word/sentence error but not a semantic error. However, if the recognition hypothesis was “send repair five to check the fire” it would be both a word/sentence error and a semantic error. The interplay between word error and semantic error is discussed in Wang et al. (2003).

We are also interested in examining speech characteristics that may reflect a student's level of certainty in their answer such as: speech rate, pauses, filled pauses (“um” and “uh”), disfluencies (“se- set boundaries”), hedges (“I guess”, “probably”, “maybe”), and latency (the delay between the end of the system's question and the start

of the student's response). We would like to see if any of these metrics can be used in updating the student model and selecting appropriate tutorial tactics.

5 Conclusions

These results demonstrate that SCoT-DC's tutoring on the three knowledge areas was effective. Subjects who started off knowing nothing about the domain learned a surprising amount about shipboard damage control. In just three hours, their performance with the damage control simulator (voice-enabled DC-Train) increased to a level of 51% fully correct actions and their scores on a written test about damage control rose to 84% correct.

Additionally, the initial results demonstrate that speech recognition technology is mature enough to support usable instructional technology. And, the remaining room for improvement in speech technology does not overwhelm the pedagogical virtues and shortcomings of instructional simulators and intelligent tutors.

Acknowledgements

This work is supported by the Office of Naval Research under research grant N000140010660, a multidisciplinary university research initiative on natural language interaction with intelligent tutoring systems. Further information is available at <http://www-csli.stanford.edu/semlab/muri>.

References

1. Aist, G., & Mostow, J. (1997). A time to be silent and a time to speak: Time-sensitive communicative actions in a reading tutor that listens. *AAAI Fall Symposium on Communicative Actions in Humans and Machines*, Boston, MA.
2. Anderson, J. R. (1993). *Rules of the mind*. Hillsdale, NJ: Erlbaum.
3. Bulitko, V., & Wilkins, D. C. (1999). Automated instructor assistant for ship damage control. In *Proceedings of AAAI-99*.
4. Clark, B., Fry, J., Ginzton, M., Peters, S., Pon-Barry, H., & Thomsen-Gray, Z. (2001). A Multimodal Intelligent Tutoring System for Shipboard Damage Control. In *Proceedings of 2001 International Workshop on Information Presentation and Multimodal Dialogue (IPNMD-2001)*. Verona, Italy. 121-125.
5. Dowding, J., Gawron, M., Appelt, D., Cherny, L., Moore, R., and Moran, D. (1993). Gemini: A natural language system for spoken language understanding. In *Proceedings of ACL 31*.
6. Evens, M., Brandle, S., Chang, R., Freedman, R., et al. (2001). CIRCSIM-Tutor: An Intelligent Tutoring System Using Natural Language Dialogue. In *Proceedings of the Twelfth Midwest AI and Cognitive Science Conference, MAICS 2001*.
7. Graesser, A., Wiemer-Hastings, K., Wiemer-Hastings, P., Kreuz, R., & the Tutoring Research Group. (2000). AutoTutor: a simulation of a human tutor. *Journal of Cognitive Systems Research*, 1, 35-51.
8. Heffernan, N., & Koedinger, K. (2002). An Intelligent Tutoring System Incorporating a Model of an Experienced Human Tutor. In *Proceedings of the 6th International Conference, ITS 2002*.
9. Lemon, O., Gruenstein, A., & Peters, S. (2002). Collaborative activities and multitasking in dialogue systems. In C. Gardent (Ed.), *Traitement Automatique des Langues (TAL, special issue on dialogue)*, 43(2), 131-154.
10. Litman, D., & Forbes, K. (2003). Recognizing Emotions from Student Speech in Tutoring Dialogues. In *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, St. Thomas, Virgin Islands.
11. Litman, D., & Silliman, S. (2004). ITSPOKE: An Intelligent Tutoring Spoken Dialogue System. In *Proceedings of the Human Language Technology Conference: 4th Meeting of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL) (Companion Proceedings)*, Boston, MA.
12. Litman, D., Rosé, C., Forbes-Riley, K., VanLehn, K., Bhembe, D., & Silliman, S. (2004). Spoken Versus Typed Human and Computer Dialogue Tutoring. In *Proceedings of the 7th International Conference on Intelligent Tutoring Systems (ITS)*, Maceió, Brazil.
13. Pon-Barry, H., Clark, B., Schultz, K., Bratt, E., & Peters, S. (2004). Advantages of Spoken Language Interaction in Tutorial Dialogue Systems. In *Proceedings of the 7th International Conference on Intelligent Tutoring Systems*, Maceió, Brazil.
14. Schultz, K., Bratt, E., Clark, B., Peters, S., Pon-Barry, H., & Treeratpituk, P. (2003). A Scalable, Reusable Spoken Conversational Tutor: SCoT. In *Proceedings of the AIED 2003 Workshop on Tutorial Dialogue Systems: With a View Towards the Classroom*.
15. Wang, Y., Acero, A., & Chelba, C. (2003). Is Word Error Rate a Good Indicator for Spoken Language Understanding Accuracy. In *Proc. of the IEEE Workshop on Automatic Speech Recognition and Understanding*. St. Thomas, Virgin Islands.

Tutorial Dialog in an Equation Solving Intelligent Tutoring System

Leena M. RAZZAQ
Neil T. HEFFERNAN

Computer Science Department
100 Institute Road
Worcester Polytechnic Institute
Worcester, MA, USA
Email: leenar@wpi.edu, nth@wpi.edu

Abstract. A new intelligent tutoring system is presented for the domain of solving equations. This system is novel, because it is an intelligent equation-solving tutor that combines a cognitive model of the domain with a model of dialog-based tutoring. The tutorial model is based on the observation of an experienced human tutor and captures tutorial strategies specific to the domain of equation solving. In this context, a tutorial dialog is the equivalent of breaking down problems into simpler steps and then asking new questions to the student before proceeding to the next step. The resulting system, named E-tutor, was compared, via a randomized controlled experiment, to an ITS similar to the “Cognitive Tutor” by Carnegie Learning, Inc[®]. The Cognitive Tutor provides traditional model-tracing feedback and buggy messages to students, but does not engage students in dialog. Preliminary results using a very small sample size showed that E-tutor capabilities performed better than the control. This result showed an effect size of 0.4 standard deviations for overall learning by condition. This set of preliminary results, though not statistically significant, shows promising opportunities to improve learning performance by adding tutorial dialog capabilities to ITSs. The system is available at www.wpi.edu/~leenar/E-tutor.

Keywords: Intelligent tutoring system, model-tracing, cognitive model, tutoring, mathematical education, artificial intelligence.

1. The Need for Better Intelligent Tutoring Systems

This research is focused on building a better tutor for the task of solving equations by replacing traditional model-tracing feedback in an ITS with a dialog-based feedback mechanism. This system, named “E-tutor”, for Equation Tutor, is novel, because it is the first intelligent equation-solving tutor that combines a cognitive model of the domain with a model of dialog-based tutoring. The tutorial model is novel because it is based on the observation of an experienced human tutor and captures tutorial strategies specific to the domain of equation-solving. In this context, a tutorial dialog is the equivalent of breaking down problems into simpler steps and then asking new questions before proceeding to the next step. This research does not deal with natural language processing (NLP), but rather with dialog planning.

Studies indicate that experienced human tutors provide the most effective form of instruction known (Bloom, 1984; Cohen, Kulik, & Kulik, 1982). They raise the mean performance about two standard deviations compared to students taught in classrooms. Intelligent tutoring systems can offer excellent instruction, but not as good as human tutors. The best ones raise performance about one standard deviation above classroom instruction (e.g., Anderson, Corbett, Koedinger, & Pelletier, 1995).

Although Ohlsson (1986) observed that teaching strategies and tactics should be one of the guiding principles in the development of ITSs, incorporating such principles in ITSs has remained largely unexplored

(McArthur, Stasz & Zmuidzinas, 1990). Hence, ITSs where a model of both the student and the tutor are created in an effort to improve performance was the natural extension to model-tracing tutors. Several researchers have developed ITSs that model the tutor as well as the student, such as Graesser, Person & Harter (2001) with AutoTutor, Heffernan (2000) with Ms. Lindquist and Rosé, Jordan, Ringenberg, Siler, VanLehn et al. (2001) with Atlas-Andes.

In studying what makes a tutoring session successful, VanLehn, Siler and Murray (1998) identified principles for effective teaching. One important principle was that tutors should not offer strong hints or apply rules to problems themselves when students make mistakes. Students miss the opportunity to learn how to solve a problem when they are given an answer and are not allowed to reason for themselves.

Merrill, Reiser, Ranney and Trafton (1992) compared the effectiveness of human tutors and intelligent tutoring systems. They concluded that a major reason that human tutors are more effective is that they let the students do most of the work in overcoming impasses, while at the same time provided as much assistance as necessary. Merrill, D., Reiser, Merrill, S., and Landes (1995) argue that the main thing human tutors do is to keep students on track and prevent them from following “garden paths” of reasoning that are unproductive and unlikely to lead to learning. Merrill et al. (1995) pointed to the large number of remarks made by tutors that helped keep students on track while learning Lisp programming.

Modeling, coaching, and scaffolding are described by Collins, Brown and Holum (1991) as the heart of cognitive apprenticeship, which they claim “help students acquire an integrated set of skills through processes of observation and guided practice.” An important part of scaffolding is fading, which entails progressively removing the support of scaffolding as the student demonstrates proficiency (Collins et al., 1991).

2. Our approach to making a better equation solving ITS

To incorporate human dialog in E-tutor, techniques used by a human math tutor were observed while teaching middle school students how to solve equations. The math tutor is also a math teacher at Doherty Memorial High School in Worcester, Massachusetts. Although the effectiveness of the math tutor was not determined and may not be representative, it was known that she also had one year of experience in one-on-one math tutoring for hire before becoming a math teacher. She was videotaped tutoring four students in solving linear equations. The students were three pre-Algebra students and one Algebra I student

A test of 20 algebra problems was given to the students to work on. The tutor was instructed to tutor the students on problems they got wrong on the test. This phase resulted in one and a half hours of one-on-one videotaped human tutoring. The human tutor helped the students on a total of 26 problems that the students had missed on the test. Similar problems were presented in the pre- and post-test of the experiment.

It was observed that there were several strategies that the tutor relied on. Often the tutor would introduce a subgoal in English to explain the next step when a student was stuck. Using subgoals can be helpful to students as described by Catrambone (1996):

“A particular subgoal will have a set of steps, or method, associated with it, and if the learner discovers that those steps cannot be used to achieve that subgoal on a particular problem, he or she will have a reduced search space to consider when trying to adapt the method. That is the learner knows on which steps to focus for changing the procedure. (p. 1020)”

This can be helpful for learning to solve equations and deciding what kind of transforms may be needed to solve a problem, or if one is needed at all. The human tutor often presented subgoals at the beginning of a problem based on how a variable could be isolated, such as in lines 291 – 294 of the tutor transcript (Razzaq (2003), Appendix A). The tutor also used a real example to illustrate a problem when a student seemed to have trouble with variables. This is consistent with Catrambone’s (1996) claim that “people typically prefer to study and use worked examples, or problems they have previously solved, when attempting to solve new problems (p. 1021).” Another common strategy that the math tutor used was to repeat subgoals that the student had just answered correctly.

Both immediate and delayed feedbacks have been shown to be helpful to students (Mathan and Koedinger, 2003). On most occasions, the tutor provided immediate feedback to student errors, keeping students on the correct solution path. There was one out of 26 problems where the tutor gave delayed feedback to a student error. In this instance, the tutor allowed the student to continue where she had made an error and then let her check her work and find the error (lines 351-377, Razzaq (2003), Appendix A), promoting evaluative skills. This happened with the student who was taking Algebra I and was a little more advanced than the others. However, for the other 25 problems, the tutor provided immediate feedback to promote the development of generative skills. This agrees with McArthur et al. (1990) in their examination of tutoring techniques in algebra where they collected one and a half hours of videotaped one-on-one tutoring sessions.

“In fact, for every student error we recorded, there was a remedial response. At least the tutors we observed were apparently not willing to let students explore on their own and perhaps discover their own errors...teachers may believe that such explorations too frequently lead to unprofitable confusion for the student (p. 209).”

3. Implementation

E-tutor was implemented using a production rule system, the Tutor Development Kit (TDK; Anderson and Pelletier, 1991). Production rules are if-then rules that operate on a set of working memory elements. A production rule system consists of components for working memory, rule memory, and a mechanism for repeating the “*match, conflict resolution, act*” cycle. The patterns contained in working memory are matched against the conditions of the production rules. This produces a subset of rules known as the conflict set, whose conditions match the contents of working memory. One or more of the rules in the conflict set is selected (conflict resolution) and fired, which means its action is performed. The process terminates when no rules match the contents of working memory. E-tutor was implemented with 89 production rules, including 12 buggy productions.

3.1 The control condition

The equation-solving section of the Cognitive Tutor by Carnegie Learning, Inc.[®] was used as a model for the control condition of this thesis. Researchers at Carnegie Mellon University (Koedinger et al., 1995) built the software to teach various skills in algebra, such as graphing, word problems and solving equations. This thesis was only concerned with the equation-solving section. Evaluation studies of the Cognitive Tutor Algebra I showed an average of 85% better performance on post-tests compared to students in traditional mathematics classes. (Koedinger, Corbett, Ritter and Shapiro, 2000) Using the Cognitive Tutor as a model provided a strong control condition for evaluating E-tutor.

The interface of the equation-solving section of the Cognitive Tutor consists of a problem window, a skillometer and a message window. The problem window presents the equation to be solved, shows the steps chosen by the student and changes to the problem as the student works. Negative feedback is indicated by coloring incorrect steps orange. The message window presents hint messages when the student presses a help button or bug messages when the system recognizes an error that the student made. Skills that the student is expected to complete are shown in the skillometer.

The Cognitive Tutor uses model-tracing to understand a student’s input. The Cognitive Tutor can indicate to students when they have made a mistake and provide hint messages, as well as provide canned explanations to some errors with bug messages. When a student continues asking for help, the hint messages get more explicit until the tutor is telling the student exactly what to do. The Cognitive Tutor does not engage the student in dialog by asking them new questions.

In the control condition, students can ask for canned hints and may receive bug messages to specific errors. The hint and bug messages are similar to and often identical to those in the Cognitive Tutor. While bug messages are presented to students when they make specific errors, the presentation of hints depends on student initiative, that is, only when a student asks for one. Hint sequences in the Cognitive Tutor typically contain three hints culminating with a very explicit hint that tells the student the correct action and input for the next step in the problem. These hint sequences are mimicked in the control condition of E-tutor.

3.2 The experimental condition

E-tutor is able to carry on a coherent dialog that consists of breaking down problems into smaller steps and asking new questions about those steps, rather than simply giving hints. Several tutorial dialogs were chosen from the transcripts of the human tutoring sessions to be incorporated in the ITS. The dialogs were designed to take the place of the hints that are available in the control condition. E-tutor does not have the hint button. When students make errors they are presented with a tutorial dialog if one is available. The student must respond to the dialog to exit it and return to solving the problem in the problem window. Students stay in the loop until they respond correctly or the tutor has run out of dialog. When the tutor has run out of dialog, the last tutorial response presents the student with the correct action and input similar to the last hint in a hint sequence. A close mapping between the human tutor dialog and the ITS was attempted.

Table 1 shows a comparison of a portion of the human tutor dialog collected and an E-tutor dialog. The dialogs in the table pertain to a problem that should be transformed by multiplication. In this scenario, the students are stuck and do not know what to do to solve the problem. Both dialogs start out by explaining that an opposite operation is needed and asking the student what the opposite of division is. In line 2, both students answer correctly that it is multiplication. In line 3, both tutors give some positive reinforcement by repeating that yes, multiplication is indeed the opposite of division. Now that the students know the operation, the tutors both ask the student what number is needed. When the students respond incorrectly in line 4, the tutors present some concrete examples. When the students get the correct number, the tutors, in line 9, give positive reinforcement by repeating the main concepts that the dialog dealt with.

Problem: $k/6 = 2$	
Experienced human tutor dialog	Real E-tutor dialog with simulated inputs
60. Tutor: Fractions are really division, right? What's the opposite of division?	1. Tutor says: You have to get rid of the fraction to get f by itself on the left. So we need to find a way to cancel the 1/6 on the left. Fractions are really division. What is the opposite of division?
61. Student: multiplication	2. Student chooses: multiplication
62. Tutor: yeah, multiplication's gonna cancel out division, right? Is there a number I can multiply this fraction by that's gonna get rid of that fraction?	3. Tutor says: Right, the opposite of division is multiplication What should you multiply both sides by?
63. Student: 12?	4. Student types in: 12
64. Tutor: 12 or even a smaller number ... Let me ask you a question. If I had 1/5 and I wanted to multiply by a number that's going to cancel the 5. What's the smallest number you can think of?	5. Tutor says: Well, here's how it works. To get rid of the denominator in a fraction you need to multiply the numerator by the same number, because 4/4 is 1, 7/7 is 1, 1394/1394 is also 1. 1/6 times what equals 1?
65. Student: 5	6. Student types in: 5
66. Tutor: if I had 1/6, what number?	7. Tutor says: you need to multiply both sides by a number that will cancel the 1/6 on the left side. What should you multiply k/6 by to cancel the 1/6?
67. Student: 6	8. Student types in: 6
68. Tutor: right. See if we multiply by 6 it cancels the 6 in the denominator.	9. Tutor says: Right, if you multiply both sides by 6 you get 1 k by itself on the left. You should go back to the problem and multiply both sides by 6.

Table 1: Comparison of human tutorial dialog and a simulated E-tutor dialog

The interface of the experimental condition of E-tutor was designed to be as similar to the control condition as possible (Figure 1). Both versions provide implicit feedback by using colors in the problem

window. The experimental version provided a skillometer such as provided in the control condition. The experimental E-tutor did not have a hint button, and dialogs appeared in the message window instead of hints. Dialogs often include a text input box, ok button or radio buttons for student input.

The dialogs typically have several “sub-dialogs” that are presented depending on the number of times a particular error is made and depending on factors such as whether the error is from a wrong student action or wrong student input. There are several different paths a dialog can take depending on student responses to the tutor’s questions. Table 2 shows an example of a dialog implemented in E-tutor.

<p>“Multiply-Transform” Dialog Example problem: $x/7=4$ Example of one of several paths for this tutor dialog:</p> <p>Tutor: You have to get rid of the fraction to get x by itself on the left. So we need to find a way to cancel the $1/7$ on the left. Fractions are really division. What is the opposite of division?(Student is offered choices of addition, subtraction, multiplication and division.) Student: (Student chooses multiplication) Tutor: Right the opposite of division is multiplication. What should you multiply both sides by? Student: (Student types in 4) Tutor: No, you need to multiply both sides by a number that will leave only $1 x$ on the left side. What should you multiply $x/7$ by to cancel the $1/7$? Student: (Student types in 1) Tutor: Well, here’s how it works. To get rid of the denominator in a fraction you need to multiply the numerator by the same number, because $4/4$ is 1, $9/9$ is 1, $1394/1394$ is also 1. $1/7$ times what equals 1? Student: (Student types in 7) Tutor: Yes, if you multiply both sides by 7 you get $1 x$ by itself on the left. You should go back to the problem and multiply both sides by 7.</p>

Table 2: Example of dialog in E-Tutor

Human tutors are unlikely to repeat the same questions and comments in exactly the same way when a student continues to make a certain error. To avoid giving the same dialog over and over, E-tutor keeps track of the number of times a student makes a particular error and presents different dialogs. Typically, a dialog is presented differently the first three times an error is made. If an error is made more than three times, the third dialog is presented again. Dialogs fade as students progress, rather than repeating concepts that students have mastered, in accordance with principles of cognitive apprenticeship as described by Collins et al. (1991).

E-tutor also recognizes when a student has chosen the correct action for the next step in the problem but has entered the wrong input. For instance, for the problem in Table 4, $3x + 4 = 6$, a student may choose the correct action of “subtract both sides” with the wrong input of 6. This kind of error may be a simple slip, based on the fact that the student got the action correct. In this case, the student will get the following dialog:

Tutor: You are right that you need to subtract something from both sides to isolate the $3x$. But if you subtract 6 from both sides then the 4 will not be canceled on the left. $4 - 6$ is not zero, so you need to subtract something that will cancel the 4.
Student: (Student types in 4)
Tutor: Right, so you see subtracting 6 from both sides doesn't leave $3x$ by itself on the left, but subtracting by 4 does You should go back to the problem and subtract 4 from both sides."

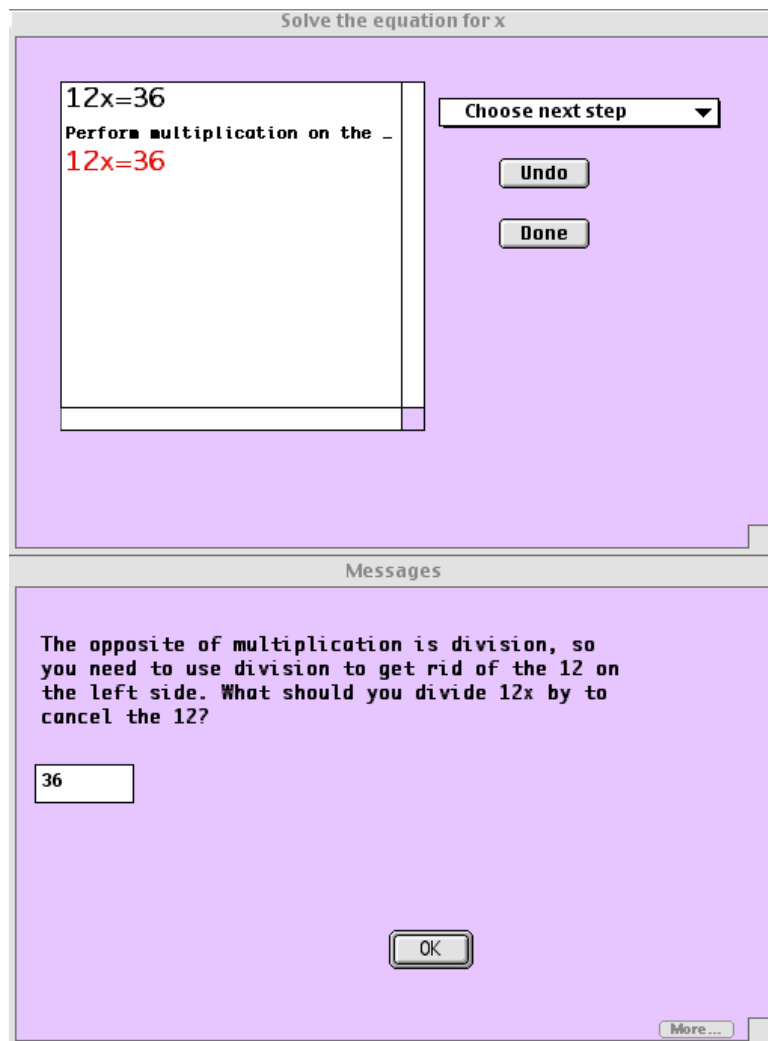
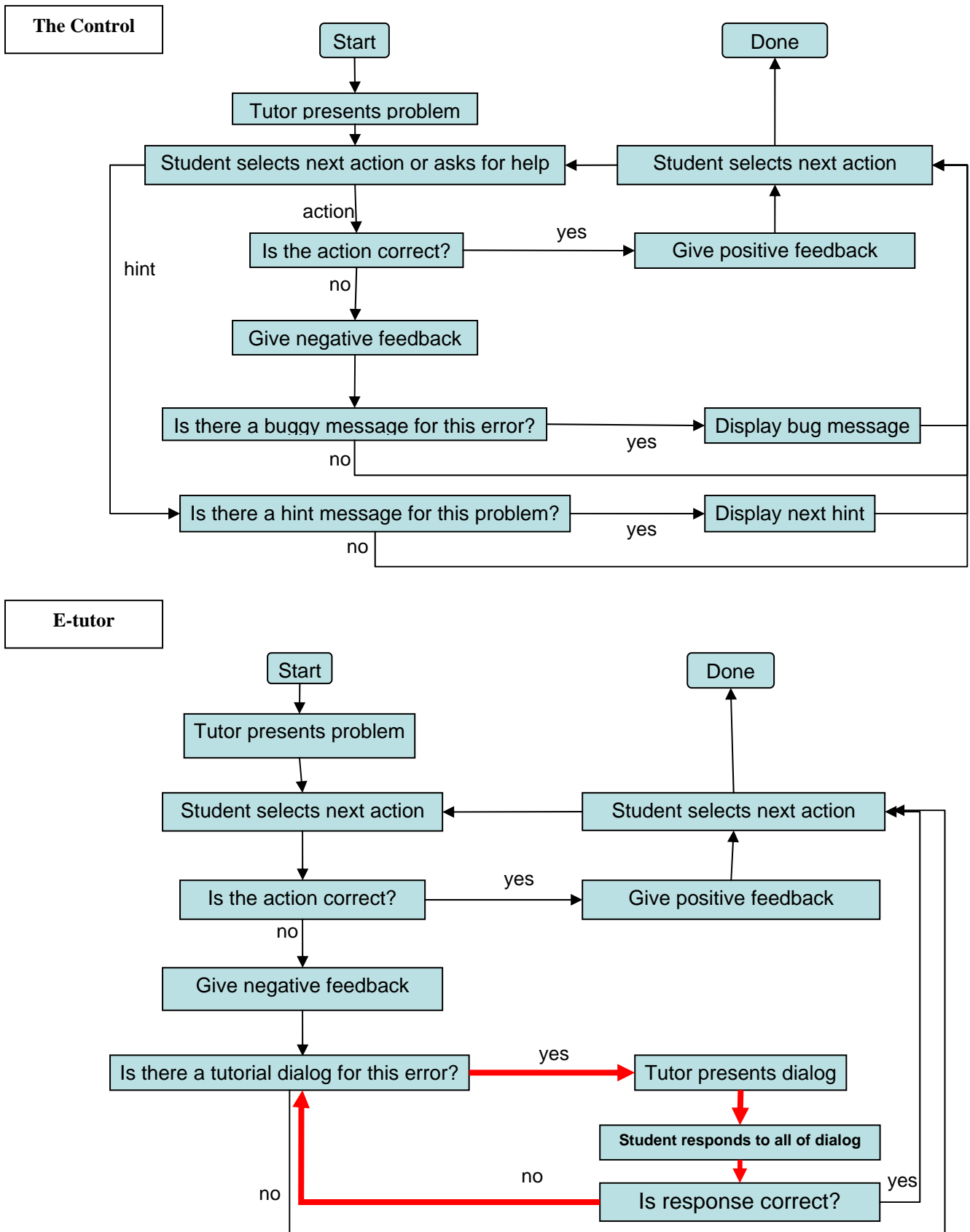


Figure 1: The interface of E-tutor showing a portion of the “Divide-Transform” dialog.

An important feature of the dialogs was keeping the student in the dialog until the tutor was satisfied. That meant that upon entering a dialog, the message window became modal. A student must respond correctly to the tutor’s new questions or the tutor must run out of comments and questions for the student to be allowed to work in the problem window. This forces the student to participate actively in the dialog. It is this loop that we hypothesize will do better at teaching equation-solving than hint sequences do. A look at Figure 2 shows the difference between the architectures of the control and experimental conditions.

Figure 2: Traditional Model-Tracing Tutor Architecture compared to E-tutor's Architecture



4. Evaluation

This section describes the design and setup of the experiment. Materials for the experiment are described, followed by a description of the subjects, the design and the procedure followed in the experiment.

Materials/Apparatus

The experiment was run at Clark University in Worcester, Massachusetts. The Jonas Clark building has a computer lab that contains 6 Macintosh computers running OSX. The software was installed so that no 2 adjacent computers were in the same condition. Students were placed in a condition based on where they sat to ensure random assignment.

Subjects

The experiment was ready to run during summer break; therefore summer school programs for middle and high school students were approached for subjects. A math teacher for “Upward Bound”, a summer enrichment program for high school students that focuses on college preparation, allowed the experiment with her Algebra I students. The students were participating in the Upward Bound program at Clark University in Worcester, Massachusetts.

The experiment was carried out with 15 Upward Bound students. The students would be entering their sophomore year in high school in the fall.

Design

Table 3 illustrates the design of the experiment.

Time	Control Condition (7 students)	Experimental Condition (8 students)
20 minutes	Paper and pencil pre-test	
5 minutes	Demonstration of both systems	
Control: average 47.8 minutes Experimental: average 55.6 minutes (over 2 sittings)	Used Control Tutor	Used E-tutor
20 minutes	Paper and pencil post-test	

Table 3: experimental design

Procedure

Eight students used the experimental E-tutor, and seven students used the control E-tutor. The experiment took place over two sessions, over a two-day period, of approximately one and a half hours each. The students were given 20 minutes to complete a pre-test before using the ITS, which appeared to be sufficient judging from students’ reactions. Upon completing the pre-test, the students were given a 5 minute demonstration showing all students both systems and the students were directed to solve as many problems as they could in the first session. In the second session, students were directed to complete the rest of the problems in the E-tutor or Control lesson if they had not done so in the first session. The experiment did not control for time but did control for the number of problems completed in the E-tutor or Control lesson. When the students had completed all of the problems in the lesson, they were given 20 minutes to complete the post-test.

Results

Four students were excluded from the analysis; three students who got perfect scores on the pre-test were excluded because we concluded that these students had already mastered the material. One student who skipped four problems on the pre-test, but attempted all four on the post-test was excluded because it was felt that she had neglected the problems on the pre-test by accident (presumably by not noticing the problems on the back of the paper). The analysis of the results was carried out with a very small sample size of 6 students in the experimental group and 5 students in the control group. Because of the small sample size, statistical significance was not obtainable in most of the analyses done in the following sections. It should be noted that with such small sample sizes, detecting statically significant effects is less likely. A large note of caution is also called for since using such small sample sizes does make our conclusions more sensitive to a single child, thus possibly skewing our results.

Overall learning

To confirm that there was learning in the experiment overall, a repeated measures ANOVA was carried out between the number of correct problems in the pre- and post-tests as a factor. The dependent variable being predicted was test score. This test showed a statistically significant overall learning ($F(1,10) = 5.213$, $P\text{-value} = .0455$). The analysis showed that overall there was over half a problem gained (pre-test = 4.9 correct, post-test = 5.6 correct).

Learning gains by condition

To check for learning by condition, a repeated measure ANOVA was performed using experimental or control condition as a factor. The repeated measure of pre-test and post-test was a factor with prediction of test score as the dependent variable. Due to the small sample size, we found that the experimental group did better on the pre-test by an average of about 1.5 points; the difference was bordering on statistical significance ($F(1,9) = 3.9$, $p = 0.07$). There was marginally statistically significant greater learning in the experimental condition than in the control condition ($F(1,9) = 2.3$, $p = 0.16$). The experimental condition had average pre-test score of 5.67 and post-test score of 6.67, showing a gain score of 1 problem. The control had average pre-test score of 4 problems correct and average post-test score of 4.2 problems correct. Taking the standard deviation of the control condition into account, the effect size was a reasonable 0.4 standard deviations between the experimental and control conditions, that is, an effect size for E-tutor over the Control. More experiments with larger numbers of students are definitely warranted to see if this effect holds up or is just a statistical fluke.

5. Conclusions

The experiment showed evidence that suggested incorporating dialog in an equation-solving tutor is helpful to students. Although the sample size was very small, there were some results in the analyses that suggest that, when controlling for number of problems, E-tutor performed better than the Control with an effect size of 0.4 standard deviations for overall learning by condition.

There were some limitations in this research that may have affected the results of the experiment. E-tutor presented tutorial dialogs to students when they made certain errors. However, the Control depended on student initiative for the appearance of hints. That is, the students had to press the Hint button if they wanted a hint. Although students in the control group were told that they could request hints whenever they wanted, the results may have been confounded by this dependence on student initiative in the control group. We may also be skeptical about the results because the sample size was very small. Additionally, the experimental group performed better on the pre-test than the control group, so they were already better at solving equations than the control group.

In the future, an experiment could be run with a larger and more balanced sample of students which would eliminate the differences between the groups on the pre-test. The confound with student initiative could be removed for a better evaluation of the two conditions. Another improvement would be to employ more tutorial strategies. A possible enhancement would be to allow the tutor to choose strategies dynamically according to students' learning styles or what types of errors they make. A question that could be answered with another experiment is whether tutorial dialogs are more motivating for students, or does dialog keep students' attention longer? Another experiment that controls for time rather than for the number of problems would examine whether E-tutor was worth the extra time.

References

- Anderson, J. R. (1993). *Rules of the Mind*. Hillsdale, NJ: Erlbaum.
- Anderson, J. R. & Pelletier, R. (1991). A development system for model-tracing tutors. In *Proceedings of the International Conference of the Learning Sciences*, 1-8. Evanston, IL.
- Bloom, B. S. (1984). The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-one Tutoring. *Educational Researcher*, 13, 4-16.

Burton, R. R., & Brown, J. S. (1982). An investigation of computer coaching for informal learning activities (pp. 79-98). In D. Sleeman and J. S. Brown (Ed.), *Intelligent Tutoring Systems*. New York: Academic Press. [20 pgs.]

Catrambone, R. (1996). Generalizing solution procedures learned from examples. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 22(4), 1020-1031.

Collins, A., Brown, J. S., Holum, A. (1991) Cognitive Apprenticeship: Making Thinking Visible, *American Educator*, 6(11), 38-46.

Corbett, A. & Trask, H. (2000). Instructional interventions in computer-based tutoring: differential impact on learning time and accuracy. *Proceedings of the SIGCHI conference on human factors in computing systems*. The Hague, The Netherlands.

Gertner, A. & VanLehn, K. (2000). Andes: A coached problem solving environment for physics. *Intelligent Tutoring Systems: 5th International Conference*, Montreal, Canada. Gauthier, Frasson, VanLehn (eds), Springer (Lecture Notes in Computer Science, Vol. 1839), pp. 133-142.

Graesser, A.C., Person, N., Harter, D., & TRG (2001). Teaching tactics and dialog in AutoTutor. *International Journal of Artificial Intelligence in Education*.

Heffernan, N. T., (2002-Accepted) Web-Based Evaluation Showing both Motivational and Cognitive Benefits of the Ms. Lindquist Tutor. [SIGdial](#) endorsed Workshop on "Empirical Methods for Tutorial Dialogue Systems" which was part of the International Conference on Intelligent Tutoring System 2002.

Heffernan, N. T (2001) *Intelligent Tutoring Systems have Forgotten the Tutor: Adding a Cognitive Model of Human Tutors*. Dissertation. Computer Science Department, School of Computer Science, Carnegie Mellon University. Technical Report CMU-CS-01-127 <<http://reports-archive.adm.cs.cmu.edu/anon/2001/abstracts/01-127.html>>

Koedinger, K. R., Anderson, J. R., Hadley, W. H. & Mark, M. A. (1995). Intelligent tutoring goes to school in the big city. In *Proceedings of the 7th World Conference on Artificial Intelligence in Education*, pp. 421-428. Charlottesville, VA: Association for the Advancement of Computing in Education.

Koedinger, K., Corbett, A., Ritter, S., Shapiro, L. (2000). Carnegie Learning's Cognitive TutorTM: Summary Research Results. http://www.carnegielearning.com/research/research_reports/CMU_research_results.pdf

Kulik, J. (1994). Meta-analytic studies of findings on computer-based instruction. In Baker, E. L. and O'Neil, H. F. Jr. (Eds.), *Technology assessment in education and training*. (pp. 9-33) Hillsdale, NJ: Lawrence Erlbaum.

Long, L. (1998). *Painless Algebra*. Hauppauge, NY: Barron's Educational Series, Inc.

McArthur, D., Stasz, C., & Zmuidzinis, M. (1990) Tutoring techniques in algebra. *Cognition and Instruction*. 7 (pp. 197-244.)

Merrill, D., Reiser, B. Ranney, M., and Trafton, J. (1992). Effective Tutoring Techniques: A Comparison of Human Tutors and Intelligent Tutoring Systems. *Journal of the Learning Sciences* 2(3): 277-305.

Razzaq, Leena M. (2003) *Tutorial Dialog in an Equation Solving Intelligent Tutoring System*. Master Thesis. Computer Science Department, Worcester Polytechnic Institute. <<http://www.wpi.edu/Pubs/ETD/Available/etd-0107104-155853>>

Rosé C, Jordan P, Ringenberg M, Siler S, VanLehn K, and Weinstein A. (2001). Interactive conceptual tutoring in Atlas-Andes. In J. D. Moore, C. L. Redfield & W. L. Johnson (Eds). *AI in Education: AI-ED in the Wired and Wireless Future*. Amsterdam: IOS Press, pp. 256-266.

Silver, Howard A. (1982). *Elementary Algebra*. Englewood Cliffs, NJ: Prentice-Hall.

Guided Exploratory Learning versus Directed Learning in a Simulation Environment for Thermodynamics: A Pilot Study

Carolyn Penstein Rosé, Cristen Torrey, and Vincent Alevén
Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh PA 15213
cprose,ctorrey,aleven@cs.cmu.edu

While the evidence in support of guided exploratory learning over pure exploratory learning seems clear, what is lacking is a set of guidelines that specify precisely how much guidance is ideal. In this paper we describe a small pilot study in which we contrast two forms of guidance. In particular, students in one condition use a simulation environment in a highly directed manner (lead by a script) whereas students in another condition use the same simulation environment in a more exploratory manner (supported by a human tutor). Although our long term goal is to learn precisely how most effectively to conduct the dialogue with the student in order to achieve the greatest instructional effectiveness, we chose to begin our exploration by contrasting human tutoring with a script condition among other reasons because previous evaluations of human tutoring in comparison to a targeted text control did not consistently show an advantage for human tutoring (Graesser et al., 2002; Rosé et al., 2003). We hypothesize that differences in the task domain, in particular the highly exploratory and creative nature of the task, will create an environment in which the advantages of tutorial dialogue will more clearly be seen than in previous evaluations. We present our preliminary results and observations from our collected human tutoring corpus. The results are encouraging but by no means conclusive due to the small population size. This pilot study is a precursor to a much larger study that we will run in Fall '04. We encourage workshop participants to raise questions and issues related to improving the experimental design in preparation for the Fall study.

Keywords: tutorial dialogue, guided exploratory learning

Motivation

We are conducting our research in the domain of thermodynamics, using as a foundation the CyclePad articulate simulator (Forbus et al., 1999; Wu, 2002). CyclePad offers students a rich, exploratory learning environment in which they apply their theoretical thermodynamics knowledge by constructing thermodynamic cycles and performing a wide range of efficiency analyses without expense or danger. These analyses offer students the opportunity to explore how their choice of properties for the devices in their designs affects the cycle's global properties, such as efficiency. CyclePad has been in active use in a range of thermodynamics courses at the Naval Academy and elsewhere since 1996 (Tuttle & Wu, 2001); a number of textbooks focus very strongly on activities with CyclePad (e.g., Wu, 2002). Active, hands-on learning with CyclePad stands in stark contrast to traditional engineering instruction in thermodynamics which emphasizes analysis rather than design. Design skills have been pinpointed as essential by the Accreditation Board for Engineering and Technology (ABET, 1998), yet are difficult for students to acquire. Qualitative evaluations of CyclePad have shown that students who use CyclePad have a deeper understanding of thermodynamics equations and a better handle on the meaning of technical terms (Baher, 1999; Baher & Ma, 1999).

While active learning and intense exploration have been demonstrated to be more effective for learning and transfer than more highly directed, procedural help (Dutke & Reimer, 2000; Dutke, 1994), pure exploratory learning has been hotly debated (Ausubel, 1968; de Jong & van Joolingen, 1998; Mayer, 2004). In particular, scientific exploratory learning requires students to be able to effectively form and test hypotheses. However, students have been observed to form ineffective hypotheses, design inconclusive experiments, demonstrate inefficient experimentation behaviour, follow a confirmation bias, and apply an engineering approach (i.e., to optimize a parameter) rather than a scientific one (i.e., to test a hypothesis) (de Jong & van Joolingen, 1998).

We have observed mechanical engineering students struggling to use CyclePad effectively in an unsupported condition (Rosé et al., to appear). Guided exploratory learning, in which a teacher provides hints, direction, coaching, or feedback, has been demonstrated to be more effective than pure exploratory learning for the purpose of learning problem solving rules, discovering conservation strategies, and discovering programming concepts (Mayer, 2004). However, Veenman and Elshout (1995) found an aptitude treatment interaction in which poor students benefited from more structuring and guidance whereas students with strong working habits or high intelligence did not.

While the evidence in support of guided exploratory learning over pure exploratory learning seems clear, what is lacking is a set of guidelines that specify precisely how much guidance is ideal. We are conducting a series of experiments to pursue the answer to this question. Some of our specific research questions are as follows:

- ? What benefits of dialogue can we observe in an exploratory environment such as CyclePad?
- ? How can we maximize productive student activity and exploration while minimizing unproductive floundering?
 - ? How can we determine how capable students are to productively explore the space?
 - ? How can we unobtrusively manipulate how exploratory students are within the design space?

Experimental Design

Dialogue	Script
Active	Passive
Exploratory	Directed
Adaptive	One-size-fits-all
Slow	Fast
Downside: Floundering	Downside: Shallow learning

We designed an experiment to contrast the effectiveness of exploration supported by a human tutor with exploration supported by a script. The script contains a description of the entire design space, with rationale for each design configuration, with instruction for optimization of alternative design configurations, including instruction on interpreting analyses and basing design choices on data from these interpretations. Although our long term goal is to learn precisely how most effectively to conduct the dialogue with the student in order to achieve the greatest instructional effectiveness, we chose to contrast human tutoring with a script condition in this study for a couple of reasons. First, a script condition, because it is totally non-adaptive and non-interactive, creates the starkest contrast with human tutoring. Secondly, script based instruction is arguably the cheapest form of instruction, so comparing human tutoring, a very expensive form of instruction, with it as a baseline is an effective way to evaluate the value added that comes with the expense. Finally, targeted text instructional conditions have proven surprisingly effective in recent evaluations. Surprisingly, previous evaluations of human tutoring in comparison to a targeted text control did not consistently show an advantage for human tutoring (Graesser et al., 2002; Rosé et al., 2003). Thus, if our contrast between human tutoring and script based exploration support shows a significant advantage for human tutoring, it would be an outstanding result.

We designed isomorphic pretest and posttest, each with 19 items covering background knowledge, basic thermodynamics concepts, and prediction problems. We controlled for time. Participation in the experiment lasted for approximately 2 hours and 10 minutes altogether.

- Experimental procedure:
 - Pretest (30 min)
 - Introductory tutorial (30 min)
 - Dialogue with tutor (experimental condition) or script (control condition) (1 hour and 20 min)
 - Exploratory Design (20 min)
 - Post test (30 min)

Results

Due to problems getting students at the end of the semester, only 9 students participated in the study, five of whom were in the experimental condition and four in the control condition. All 9 students were mechanical engineering students from Carnegie Mellon University who had formal instruction in thermodynamics but who had not yet received instruction in cycle design. Students were evaluated both on their gain in conceptual understanding as measured by the pre/post test and on the quality of their resulting designs at the end of the 20 minutes of unsupported exploration with CyclePad.

Figure 1 gives an overview of the results from the 20 minutes of unsupported exploration each student had with CyclePad. Out of four script condition students, 50% were not able to produce an analyzable cycle. The other two produced cycles with efficiencies 37.39 and 41.04 respectively (average 39.2). Out of five tutoring students, only 20% were not able to produce an analyzable cycle. The others produced cycles with efficiency of 40.59, 41.80, 41.88, and 47.44 (average 42.9). Students in the script condition who produced an analyzable design all chose to implement one reheat cycle and one regeneration cycle while the students in the tutoring condition universally chose to implement one or more reheat cycles.

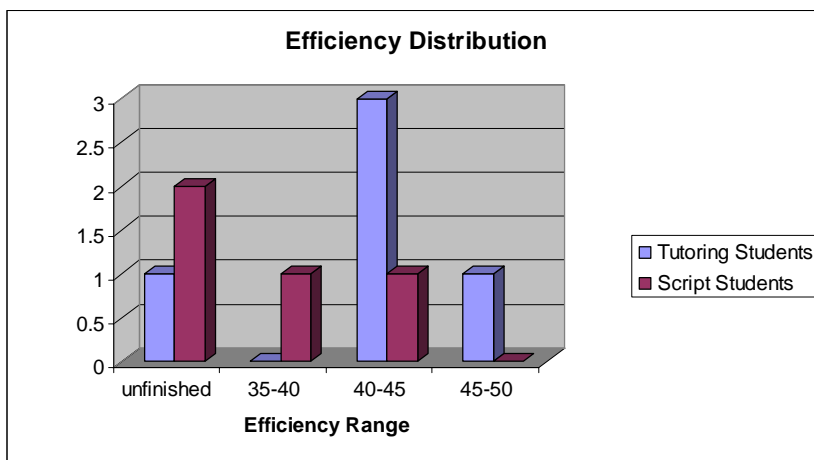


Figure 1 Distribution of efficiency of final designs produced by students during unsupported exploration

The pre/post test did not show any difference in conceptual learning between conditions. There was no significant difference in average pretest scores between conditions, nor was there a significant difference in average posttest scores between conditions. There was, however, a significant gain between pretest and posttest in both conditions. However, there was no significant difference between pretest and posttest on specifically the prediction problems. While these results are preliminary do to the very small population size, we are concerned about the lack of gain on prediction problems. Thus, we plan to include more activities requiring students to make explicit predictions on their own, without the help of CyclePad, in both conditions in subsequent experiments and in the CycleTalk system. Surprisingly, there was no relationship between pretest score and ability to produce an analysable design or final cycle efficiency, although this might be a function of lack of statistical power due to the small sample size.

In comparison to previous studies contrasting human tutoring with a targeted text control, we consider it very interesting that our preliminary results show a potential advantage for human tutoring in terms of teaching the

ability to do design where none is found on the conceptual pre/post test. One potential explanation is that the tests focused on much basic thermodynamic conceptual knowledge that is assumed and applied throughout the script and discussions with the human tutors but is not discussed explicitly with much frequency either in the script or in the human tutoring transcripts. However, another possible interpretation is that the value in human tutoring for instruction in contrast to a reading control lies in a different skill set than has been tapped in previous evaluations. For example, the contrasts presented in (Graesser et al., 2002; Rosé et al., 2003) focus entirely on conceptual physics knowledge.

We are currently analyzing the corpus of tutoring dialogues that we collected from the 5 students in the experimental condition as well as those from a small corpus of even more preliminary pilot data with students interacting with CyclePad with the aid of a human tutor. We are observing the students behaving at a high level of activity throughout their interaction with CyclePad and the tutor. In particular, students frequently ask questions, for one student the average was 23 questions per hour. In total, 12.2% of student contributions consisted entirely of student initiatives, including unsolicited questions, observations, and arguments. Students also frequently offered unsolicited explanations as elaborations on their answers to tutor questions. The tutors required the students to make choices and act according to their choices. Before asking the students to make a choice, the tutors tended to engage the students in a discussion of the options available, requesting the students to be the one to enumerate the set of options. Tutors let students try sub-optimal assumption setting but then explored why they were suboptimal. Thus, they allowed the students to explore, but did not allow them to go too far down an unproductive path. The tutors turned these sub-optimal choices into opportunities for learning.

Because exploration supported by a tutor takes more time than exploration guided by a script, there were differences in how students in the two conditions explored the design space. In general, students in the script condition covered entire design space, whereas tutoring students only covered simple rankine cycle. The following describes the observations we have made so far about the behavior of our tutors:

- Tutors encouraged students to keep it simple: actively combating what seems to be a very common misconception that more sophisticated designs will lead to higher efficiency
- Tutors pushed students to base designs on conclusions drawn from sensitivity analyses
- Tutors asked students to explain the implications of design choices – emphasizing understanding over exploration
- Tutors emphasized understanding the reasons for limitations on acceptable ranges for cycle parameters

Discussion

Currently, we are continuing our analysis of thermodynamics dialogue corpus we have collected. If possible, we will collect additional pilot data throughout the summer as we are building our prototype CycleTalk system. We plan to begin pilot testing the system in the Fall of '04.

In this paper we are presenting the results from a small pilot study contrasting exploration supported by a human tutor and exploration supported by a script. This small pilot study was conducted as a precursor to a larger study we plan to run in Fall of '04. While the results of this experiment are promising, they are by no means conclusive because of the small sample size, and they leave many questions unanswered. Most importantly, we question what is the most effective way to measure what students have learned through their interaction with CyclePad beyond the conceptual learning, which can be measured with an on-paper test. In our pilot study, students in the two conditions produced designs with very distinct characteristics. Where students in the human tutoring condition produced more efficient designs, students in the script condition produced more sophisticated designs. Both sophistication and efficiency demonstrate in some sense an understanding of the design space. We are inclined to believe that efficiency is a better measure of deep learning since more of an understanding of the internal model of thermodynamics is required in order to set model assumptions effectively than to mimic design aspects that had been demonstrated through instruction (i.e., using a reheat or regeneration cycle). Nevertheless, we are still grappling with this and other issues.

References

- Accreditation Board for Engineering and Technology (ABET) (1998). *Engineering Criteria 2000: How do you measure success*. ASEE Professional Books.
- Ausubel, D. (1978). *Educational Psychology: A Cognitive View*, Holt, Rinehart and Winston, Inc.
- Baher, J. (1999). Articulate Virtual Labs in Thermodynamics Education: A Multiple Case Study. *Journal of Engineering Education*, October 1999. 429-434.
- Baher, J., Ma, J., (1999). Using Case Studies to Evaluate Learning Technologies. *Proceedings of the 29th ASEE/IEEE Frontiers in Education Conference*, Puerto Rico.
- Dutke, S. (1994). Error handling: Visualizations in the human-computer interface and exploratory learning. *Applied Psychology: An International Review*, 43, 521-541.
- Dutke, S. & Reimer, T. (2000). Evaluation of two types of online help for application software, *Journal of Computer Assisted Learning*, 16, 307-315.
- Forbus, K. D., Whalley, P. B., Evrett, J. O., Ureel, L., Brokowski, M., Baher, J., Kuehne, S. E. (1999). CyclePad: An Articulate Virtual Laboratory for Engineering Thermodynamics. *Artificial Intelligence 114(1-2)*: 297-347.
- Graesser, A., VanLehn, K., the TRG, & the NLT (2002). Why2 Report: Evaluation of Why/Atlas, Why/AutoTutor, and Accomplished Human Tutors on Learning Gains for Qualitative Physics Problems and Explanations, LRDC Tech Report, University of Pittsburgh.
- de Jong, T. & van Joolingen, W. R. (1998). Scientific Discovery Learning With Computer Simulations of Conceptual Domains, *Review of Educational Research*, 68(2), pp 179-201.
- Mayer, R. E. (2004). Should there be a three-strikes rule against pure discovery learning? The Case for Guided Methods of Instruction, *American Psychologist* 59(1), pp 14-19.
- C. P. Rosé, C. Torrey, V. Aleven, A. Robinson, C. Wu, & K. Forbus (in press). CycleTalk: Towards a Dialogue Agent that Guides Design with an Articulate Simulator, *Proceedings of the Intelligent Tutoring Systems Conference*.
- Rosé, C. P., VanLehn, K., & the Natural Language Tutoring Group (2003b). Is Human Tutoring Always More Effective Than Reading: Implications for Tutorial Dialogue Systems, *Proceedings of the AIED 2003 Workshop on Tutorial Dialogue Systems: With a View Towards the Classroom*.
- Tuttle, K., Wu, Chih, (2001). Intelligent Computer Assisted Instruction in Thermodynamics at the U.S. Naval Academy, *Proceedings of the 15th Annual Workshop on Qualitative Reasoning*, San Antonio, Texas.
- Veenman, M. V. J. & Elshout, J. J. (1995). Differential effects of instructional support on learning in simulation environments. *Instructional Science*, 22. 363-383.
- Wu, C. (2002). *Intelligent Computer-Based Engineering Thermodynamics and Cycle Analysis*. New York: Nova Science Publishers.

A Conceptual Architecture for Generating Examples in a Socratic Tutor for Qualitative Modeling

Leo C. Ureel II

Qualitative Reasoning Group
Northwestern University
ureel@cs.northwestern.edu

Abstract: VModel is a qualitative modeling environment for middle school science students. Students using VModel learn to construct qualitative models in the language of Qualitative Process theory. A pervasive theme in the design of scaffolds and supports for VModel has been the importance of underlying structural relationships, in particular causal relationships, to a student's learning and understanding of system-level behavior. The Socratic Assistants project aims to further emphasize these structural relationships by guiding students through an expert analysis of their model and the problem space. Socratic Assistants are based on Collins and Stevens' cognitive theory of Socratic tutoring. However, Socratic Assistants incorporate new theory to explain the underlying cognitive mechanisms invoked by the tutor when executing the strategies prescribed by Collins & Stevens' theory. This paper describes the conceptual architecture for one class of these strategies; the generation of examples, (including hypothetical and counter examples).

Keywords: Socratic Tutoring, Structure Mapping Theory, Qualitative Process Theory, Qualitative Modeling

1 INTRODUCTION

The Socratic Assistants project at Northwestern University is developing a model of the underlying cognitive mechanisms used by tutors engaged in a Socratic dialogue. Our initial goal is to implement Socratic Assistants using VModel, a qualitative modeling environment incorporating a representation similar to concept-maps (Section 1.1), as the primary means by which students articulate their understanding of scientific phenomena. A pervasive theme in the design of scaffolds and supports for VModel has been the importance of underlying structural relationships, in particular causal relationships, to a student's learning and understanding of system-level behavior. Socratic Assistants will further emphasize these relationships by guiding students through a dialogue-like interaction concerning their model and the problem space. Through the course of a dialogue, the tutor will analyze and respond to the student's model and to student questions using three modes of interaction: 1. natural language templates (such as those already used by VModel's on-board analysis coach [4]), 2. a dynamically generated list of questions that can be asked by the student (similar to those used in evidence-driven ASK-systems [54]), 3. the highlighting and annotation of elements within a student's model or the proposal of new modeling elements/ models such as a counter-example to something the student has modeled.

Socratic Assistants are based on the theory of Socratic tutoring (Section 1.2) developed by Collins & Stevens [6] and extended through the curriculum planning work of Wong, et al. [14] and the natural language understanding work of Graesser, et al. [12]. The base theory developed by Collins & Stevens' describes a model-driven production system that guides a student through an analysis of the causes and effects in a problem domain. This theory of Socratic tutoring was instrumental in analyzing a model, tracing causal paths, and suggesting how to direct the dialog. Collins and Stevens [2] went on to describe the goals and strategies used by tutors when analyzing student knowledge and guiding tutorial dialogues. However, although the theory prescribes which strategies can be used in a given situation to meet a pedagogical goal, the underlying cognitive mechanisms invoked when executing those strategies are not described. This paper discusses a conceptual architecture for implementing some aspects of one class of these strategies: pick an example/counterexample/hypothetical example based on factors in the student's model.

We believe execution of these strategies can be explained by a combination of knowledge manipulation, qualitative simulation and similarity-based retrieval. In particular, we draw on the work of Forbus in Qualitative Process theory [9], a representation and method for qualitative reasoning (Section 1.3), and the work of Gentner

in Structure Mapping Theory [11], a cognitive theory of similarity (Section 1.4).

1.1 VModel

The VModel project at Northwestern University has developed a visual modeling tool (VModel) and supporting curricula to teach qualitative modeling in middle school science classrooms [1, 4, 8]. VModel provides a qualitative modeling environment that uses a concept map notation, which enables students to articulate their understandings of scientific phenomena and get feedback about them. VModel has been successfully fielded through several studies in the Chicago Public Schools, and will soon be available, along with sample curriculum materials, as open-source software. The representation utilized in VModel emphasizes causal structures over surface features (Figure 1). The combination of knowledge representation and structural emphasis is designed to encourage analogy between models and support learning in physical domains.

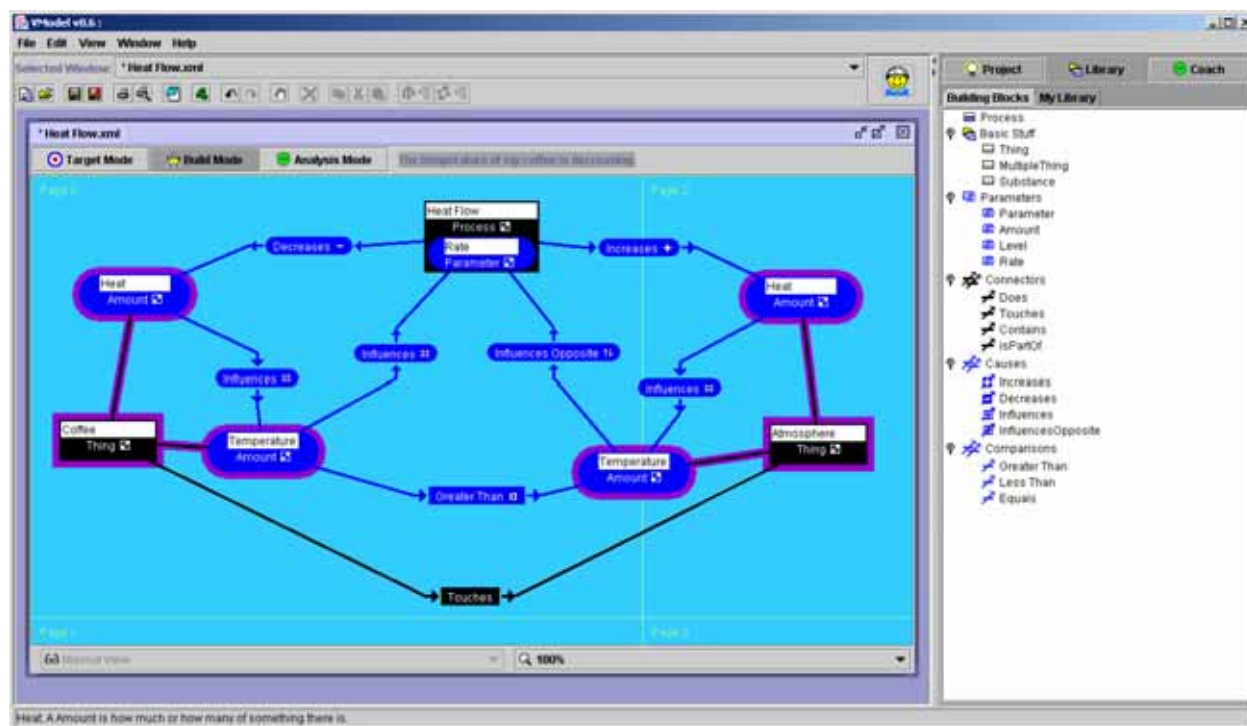


Figure 1: A model of Heat Transfer created using the VModel software.

VModel already contains some scaffolding and coaching feedback to guide the student through the creation and analysis of a model. While students are constructing a model, the coach uses a simple set of rules to identify when they are using modeling elements incorrectly, when a more appropriate modeling element is available, or when there are missing elements or inconsistencies in their model. For example, two of the coach’s rules, stated in English, are: “A direct influence relationship must be connected between the rate of a process and any type of parameter” and “No type of parameter can have both a direct influence and an indirect influence as input.”

The coach indicates when it finds a problem in two non-intrusive ways. First, an icon always depicts the emotional state of the coach. The default icon is smiling, but as the student’s model accumulates errors, it is replaced by a puzzled-looking icon. If a large number of errors are made, a frowning icon is used. Second, parts of the model that are implicated problems are highlighted in red. Students can query the coach as to the nature of the problem(s) by clicking on the coach icon or by right-clicking on the indicated modeling element. The coach responds by displaying its analysis of the current state of the model. It is important that the coach be non-intrusive because editing a model often involves temporary inconsistencies.

Once completed, a students’ model is analyzed via simple qualitative simulation. Resolving ambiguous influences via asking the student is straightforward and easy for them to understand. The results of qualitative simulation are provided to the student in terms of a textual explanation. For example, this explanation is generated from a model of heating water in a microwave oven:

- There is the process of Heating which
 - Increases the Heat of Water that is INCREASING and which
 - Influences the Temperature of Water that is INCREASING

The generated text serves several pedagogical purposes. First, it encourages students to name and decompose their entities properly, since the templates include conventions such as “<quantity> of <entity>”. This overcomes a student tendency to lump things together by relying on their language skills (e.g., “Trixie eating does eating.” grates, even though the fragment of concept map that gives rise to the statement at first did not.) Second, it illustrates the process that students should follow when reasoning with causal models on their own. This point is reinforced by animation used during qualitative simulation, where the quantity currently being considered is highlighted and intermediate results are displayed as graphical annotations on the student’s model.

Here the coach also provides help in interpreting the results of their simulation. For example, if there is a mismatch between the student’s prediction and the results of the simulation, this is noted as part of the text generated along with the simulation, e.g.:

Level **Temperature**

- You predicted that **Temperature** would be DECREASING but instead it is INCREASING

with the offending parameter highlighted in red.

We aim to build a Socratic Tutor on top of VModel to help teach scientific reasoning and qualitative modeling skills. The modeling interface itself will comprise a major component in the communication of information between the tutor and the student.

1.2 Socratic Tutoring

Collins and Stevens’ application of the Socratic Method in the WHY Project was a model-based tutoring system for a global weather patterns micro domain [13]. The project led to the creation of a cognitive theory of inquiry teaching and went a long way toward the development of a computational model of Socratic tutoring strategies. Their theory constitutes a model of pedagogical discussion that is descriptive, prescriptive, and primarily Socratic in that it relies upon a dialectic process of discussion.

Implemented Socratic tutoring platforms, including Why-2K [12] and TAP-2 [14], implement and extend the theory developed by Collins and Stevens’. Their contributions include robust natural language processing and discourse/curriculum planning to guide the student in a coherent and organized manner. The current state of Socratic theory is that we now know *what* the tutor should do and *when*, but we do not yet fully understand the cognitive mechanisms involved in *how* tutorial actions are carried out. In order for a cognitive theory to be deeply satisfying, it must explain the underlying cognitive mechanisms invoked to carry out tutorial actions. We believe some recurring properties of these cognitive mechanisms include some form of qualitative knowledge representation (such as Qualitative Process Theory; Section 1.3), and a means of assessing similarity and generating analogies (Structure mapping Theory; Section 1.4).

1.3 Qualitative Process Theory

QP theory [9] provides a way of describing the world in terms of objects, their properties, and the processes and relationships that operate upon them. In QP-Theory, a physical scenario is modeled as a collection of entities and the relationships among them. The entities are described by attributes, such as temperature and pressure, represented by qualitative quantities. Qualitative quantities are said to be increasing, decreasing, or constant and can be compared using operations such as greater-than, less-than, or equal-to. Processes are an intangible type of entity that effects change in the qualitative properties of other entities. Processes are the sole mechanism of change in this representational system. Relationships between entities and/or qualitative quantities can be structural, constraining, or causal. Structural relationships describe how entities are configured. Constraining relationships tell when a process will be acting. Causal relationships describe what holds as a consequence of the process acting. The effects of a process can be traced by following causal relationships between qualitative quantities. Models described using QP theory can be used to make predictions and generate explanations concerning observable changes in the world. For example:

The entity Cup can contain the entity Coffee that has the qualitative quantities Heat and Temperature. The coffee is in contact with the Atmosphere, which also has associated qualitative quantities Heat and Temperature. When the Temperature of the Coffee is Greater-Than the Temperature of the Atmosphere, a process called Heat-Exchange is activated. The process of Heat-Exchange directly reduces the Heat of the Coffee (resulting in the Temperature of the Coffee going down) and likewise increases the Heat of the Atmosphere (resulting in an increase in the Temperature of the Atmosphere above the Coffee).

Although there are many different qualitative representations suitable for representing knowledge utilized by cognitive mechanisms, we had selected QP theory for our visual modeling system, VModel, because we believe it is uniquely accessible to students studying middle school physical science [8].

1.4 Structure Mapping Theory

The ability to compare student knowledge to known content knowledge or to knowledge from other domains (via analogy) is a key ability used by teachers in determining pedagogical goals and strategies for guiding a Socratic dialogue. Gentner's Structure Mapping Theory (SMT) describes a cognitive mechanism underlying people's ability to recognize similarities and differences as well as our ability to make analogies [11].

In structure mapping, knowledge domains are viewed as systems of entities and predicates. Predicates are knowledge representations that can be used to relate entities with other entities or attributes. In propositional notation an attribute is a predicate of one argument and relationships are predicated of two or more arguments. For example, a partial structure mapping might exist between a representation of the solar system and an atom. Using the solar system as a base knowledge domain, its objects (the sun and the planets) are mapped onto the objects of the atom (the nucleus and the electron). Given that the objects correspond, Structure Mapping Theory conveys that the relationships that hold between the nodes in the solar system also hold between the nodes in the atom (i.e. there is a central attracting force, peripheral objects revolve around the center, the center is more massive than the peripheral objects, and so on.)

Gentner and Forbus have collaborated on several computational simulations of cognitive processes based on SMT. The Structure-Mapping Engine (SME) provides a computational account of similarity comparisons in human reasoning [5]. MAC/FAC (Many Are Called / Few Are Chosen) is a computational simulation of similarity-based memory retrieval [6]. Using modules such as these, we believe we are in a position to construct and simulate a deep cognitive theory of Socratic interaction.

2 RETRIEVING AND GENERATING EXAMPLES

Collins and Stevens described the generation of tutor questions and feedback as being triggered by rules for Socratic Tutoring and guided by the goals and strategies of inquiry teachers. This combination of rules, goals, and strategies could result in simple questions that maintain a linear dialog course such as "Identify the factors influencing the POPULATION of GAZELLE." Simple questions are relatively easy to implement using model tracing techniques. More complex questions require the student to generate and explore an example model. For this type of question, the software must be able to spawn digressions, remember connections between dialog threads, and prompt returning to previous threads.

Even more sophisticated are questions wherein the tutor must generate a relevant example or counter-example, or a hypothetical example. Figure 2 shows our conceptual architecture for retrieving examples and counter-examples from memory. On the left we show two important case libraries within the knowledge-base: the student's model library, which includes all the previous models and modeling elements designed by the student (the model library comes to us free as a part of VModel), and a phenomena library containing other models of scientific phenomena that the coach knows about. This allows the tutor to retrieve examples that might be personally relevant to the student. MAC/FAC (Many Are Called/Few Are Chosen), a computational simulation of a cognitive model of similarity-based retrieval, is used to fetch related knowledge structures from these libraries within the knowledge base. The MAC stage does a fast featural match, using a salient portion of the student's model as a probe, resulting in many retrievals. In general, the FAC stage uses SME (Structure Mapping Engine) with the same probe to perform computationally expensive structural matches that filter the retrieval results to find the best match. The fitness of the match is determined by the amount of structural overlap between the probe and base knowledge structures. MAC/FAC has been validated as a psychologically plausible account

of similarity-based retrieval. However, here we hypothesize something new: the tutor can influence the FAC stage by modifying the probe's knowledge structures in between the MAC and FAC stages. The generation of counter-examples requires us to cache sign information for quantities post simulation in a way that can be matched by SME. This is because counter-examples and hypothetical cases can be searched for based on either a quantity sign-change (find an example where the volume is decreasing) or a structure change (find an example where this QPROP runs in the opposite direction).

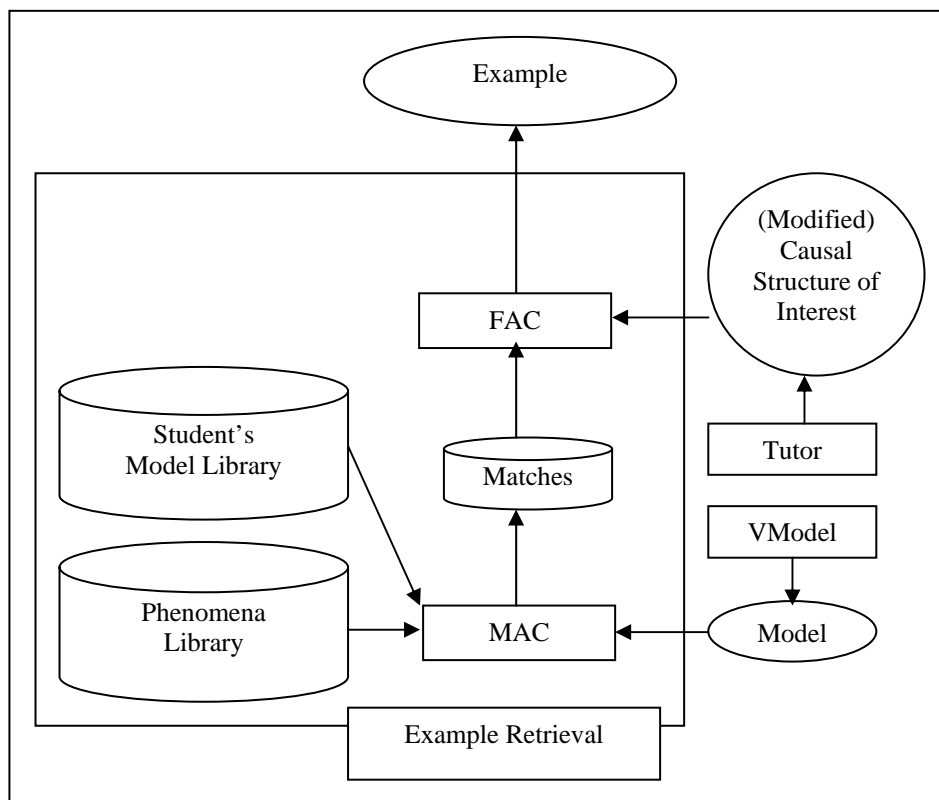


Figure 2: Architecture for example and counter-example generation and retrieval.

Once an example or counter-example has been retrieved from memory, the tutor can use it to generate a hypothetical example by reifying candidate inferences resulting from SME in the FAC stage. Candidate inferences are the structural information that can be inferred from the alignable differences between the probe and the base. The choice of candidate inferences to be reified is determined by the pedagogical goals currently in force and the strategies they trigger. For example, the tutor might decide to generate a hypothetical example surrounding the most relevant influencing factor and thus select only candidate influences involving that factor.

Lastly, a strategy may be triggered wherein the tutor confronts the student with an absurd situation (possibly for reasons of entrapment), “You agreed that the SOLAR SYSTEM is like your model of the ATOM. Therefore there should be a GRAVITATIONAL FORCE between the NUCLEUS and the ELECTRONS that constrains the ELECTRONS to their ORBIT. Do you agree with this example?” Such absurd examples can be formed using the techniques described above or by mutating the student’s own model.

3 DISCUSSION & FUTURE WORK

We describe a conceptual architecture for the cognitive mechanisms utilized by tutors when retrieving and generating examples, which is one class of strategies used by inquiry teachers. This conceptual architecture is psychologically interesting. In particular, allowing the hypothesized alteration of the probe between the MAC and FAC stages of similarity-based retrieval. This hypothesis must be verified through psychological studies and could perhaps be used to explain some consistent anomalous retrievals.

Using the VModel environment, students have opportunities to make explicit the structured relationships

between entities and processes using our visual qualitative vocabulary. This vocabulary is semantically well-defined making it extremely useful to the tutor as an explicit representation of the student's mental model of some phenomenon. Socratic Assistants are being developed to guide students through a deep causal analysis of their model while confronting them with analogical comparisons to other models. We hope that through the use of VModel and Socratic Assistants, students will become skilled in the *Art of Modeling*.

REFERENCES

1. Carney, K, Forbus K., Ureel, L., Sherin, B. (2002) "Using Modeling to Support Integration and Reuse of Knowledge in School Science: Vmodel, a New Educational Technology." American Educational Research Association annual conference, April, 2002.
2. Collins, A., & Stevens, A. L. (1982). Goals and strategies of inquiry teachers. In R. Glaser (Ed.), *Advances in instructional psychology* (Vol. 2, pp. 65-119). Hillsdale, NJ: Erlbaum.
3. Ferguson, W., Bareiss, R., Birnbaum, L., Osgood, R., (1992) *Ask Systems: An Approach to the Realization of Story-based Teachers.*, Technical Report #22, Institute for the Learning Sciences, Evanston, IL
4. Forbus, K., Carney, K., Sherin, B., Ureel II, L., (2004 in press) VModel: A visual qualitative modeling environment for middle-school students., *Proceedings of IAAI-04, The Sixteenth Innovative Applications of Artificial Intelligence Conference*, July 27-29, 2004, San Jose, California.
5. Forbus, K., Ferguson, R. and Gentner, D., (1994) Incremental structure-mapping. *Proceedings of the Cognitive Science Society*, August.
6. Forbus, Kenneth D., Gentner, Dedre., Law, Keith., (1994) *MAC/FAC: A Model Of Similarity-Based Retrieval.*, The Institute For The Learning Sciences, Technical Report # 59, Northwestern University, October 1994
7. Forbus, K, and Gentner, D. (1986) *Learning Physical Domains: Toward a Theoretical Framework.* In R. Michalski, J. Carbonell & T. Mitchell (eds.) *Machine learning: An artificial Intelligence Approach II.* USA, Morgan Kaufman.
8. Forbus, K., Ureel II, L. Carney, K., Sherin, B., (2004 in press) *Qualitative modeling for middle-school students.*, *Proceedings of QR04, 18th International Workshop on Qualitative Reasoning*, August 2-4, 2004, Northwestern University, Evanston, Illinois, USA.
9. Forbus, K., (1981). "Qualitative Reasoning about Physical Processes," *Proceedings of the Seventh IJCAI*, Vancouver, B.C., pp.326-30
10. Gentner, D., & Stevens, A. (Eds.) (1983) *Mental Models*, Lawrence Erlbaum Associates, Hillsdale, New Jersey.
11. Gentner, Dedre (1983) Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7, 155-170.
12. Graesser, Arthur, Kurt VanLehn, Carolyn Rosé, Pamela Jordan and Derek Harter "Intelligent Tutoring Systems with Conversational Dialogue", In *AI Magazine*.
13. Stevens, A. L., and Collins, A., (1977) *The goal structure of a Socratic Tutor.*, In *Proceedings of the National ACM Conference.*, Association for Computing Machinery, New York
14. Wong, L. H., Quek, C., Looi, C. K. (1998) TAP-2: A Framework for an inquiry Dialogue Based Tutoring System/ *International Journal of Artificial Intelligence in Education*, 9, 88-110

The design and architecture of Research Methods Tutor, a second generation dialog-based tutor

Peter Wiemer-Hastings

peterwh@cti.depaul.edu

School of Computer Science,
Telecommunications, and Information Systems
DePaul University
243 South Wabash Avenue
Chicago, Illinois

Abstract: RMT (Research Methods Tutor) is a dialog-based tutoring system that is designed to be used via the internet in order to augment traditional classroom experience. RMT was designed with a modular architecture to enable us to interchange and evaluate different tools and techniques for improving tutoring. The focus of this paper is to describe the architecture of the system and the current components that allow it to provide an effective complement to classroom teaching.

Keywords: Dialog-based tutoring, System architecture

Introduction

Research on human-to-human tutoring has identified one primary factor that influences learning: the cooperative solving of example problems [1]. Typically, a tutor poses a problem (selected from a relatively small set of problems that they frequently use), and gives it to the student. The student attempts to solve the problem, one piece at a time. The tutor gives feedback, but rarely gives direct negative feedback. The tutor uses pumps (e.g. “Go on.”), hints, and prompts (e.g. “The groups would be chosen ...”) to keep the interaction going. The student and tutor incrementally piece together a solution for the problem. Then the tutor often offers a summary of the final solution [1].

Over the last 10 years, the field of dialog-based intelligent tutoring systems (DBITS) has evolved and produced some quite successful systems that mimic this behavior. DBITS have been implemented and have been demonstrated to be help learning in such areas as medical diagnosis [2, 3, 4], geometry [5], electronics [6], computer literacy [7], and physics [8] to name just a few. This paper describes the architecture and the components of a DBITS called Research Methods Tutor (RMT). RMT was developed with two goals in mind: to provide a internet-based tutoring system that can augment classroom teaching, and to provide that in the form of a domain-independent, modular architecture that supports research in a variety of different tutoring techniques.

The paper begins with a high-level overview of the major components of the system. Then we focus on the individual components. As is the case for most DBITS, a critical aspect is its ability to understand student responses, that is, its natural language processing mechanism. RMT uses a technique called Latent Semantic Analysis (LSA) which learns a semantic representation from a corpus, and which provides a simple similarity metric between texts. RMT augments LSA with spelling correction and a keyword-matching mechanism for special circumstances. In addition to the natural language understanding techniques, RMT uses a modular dialog manager to control the interaction with the student, and a transition network to determine appropriate responses. These primary modules and the other support modules are described in the rest of this paper.

RMT architecture

RMT is a close descendant of the AutoTutor system, which was explicitly modeled after the behavior of human tutors. While AutoTutor incorporates a wide variety of artificial intelligence techniques, RMT was designed as a lightweight, modular system that would incorporate only those techniques required to provide educationally beneficial tutoring to the student. This section gives an overview of RMT's critical components, and these modules are described in detail in the following sections.

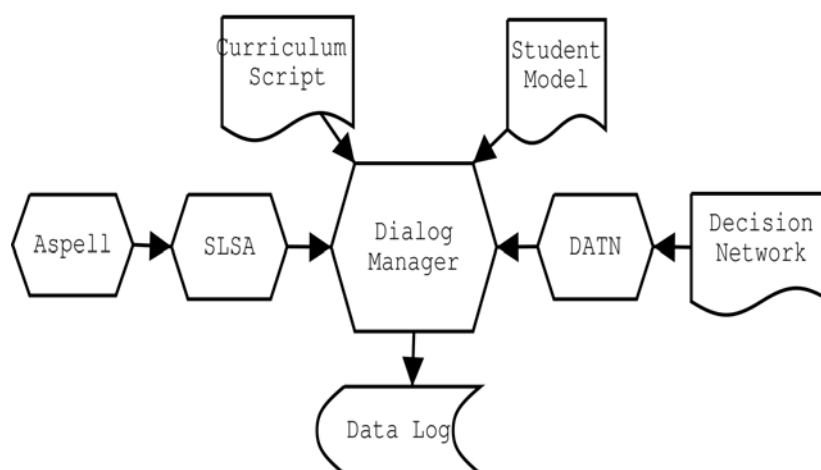


Figure 1: RMT Architecture

Figure 1 graphically depicts the relationships between RMT's major components. The dialog manager (described in the next section) is the hub of activity in the system. It controls the interactions with the user, presenting tutor responses, and requesting evaluation of the students' replies. On the left side of the graphic are the language understanding components, LSA with the help of a spellchecker. These are further described in section **Error! Reference source not found.** On the right are the domain-independent components which determine the *type* of the next tutor move (section **Error! Reference source not found.**). The Curriculum Script (section **Error! Reference source not found.**) provides all of the content material in the tutoring domain in a structure that combines questions, expected answers, and follow-up dialog moves. The data logger (section **Error! Reference source not found.**) is critical for research purposes and pedagogical purposes because it allows us to save information at a fine-grained level, and to flexibly retrieve it.

Dialog Manager

As shown in figure 1, the dialog manager (DM) is the central controller of the system. Because RMT is a web-based system, each tutoring session has its own dialog manager, and the DM maintains information about the parameters of the tutoring session and, crucially, the current state of the dialog. Traditional (non-internet-based) interactive programs can rely on the internal state of variables associated with some processing loop. But an internet server program must act as an agent which responds to user requests, and thus, it must maintain enough state information to allow it to recreate the context of the dialog whenever it is invoked.

The DM receives student responses as posts from a web page, and then asks the Dialog Act Transition Network (DATN) to compute an appropriate tutor response. RMT was designed to also work in a text-only console mode. RMT's design includes an object-oriented parallel representation to provide the same functionality in both operating environments.

Understanding student contributions

RMT uses Latent Semantic Analysis (LSA) to evaluate student contributions. LSA was first developed for information retrieval — selecting query-relevant texts from a database [9]. Landauer and Dumais [10] later showed that it accurately models human acquisition of word knowledge. LSA has also been shown to perform well at finding synonyms [11], suggesting appropriate texts for students to read [11], and even grading student essays [12]. AutoTutor was the first system to use LSA to “understand” student responses in an interactive

tutoring system [7], and it has subsequently been incorporated or evaluated for use by several other systems [5, 13, 14, 4].

LSA evaluates a student response by comparing it to a set of expected answers. This works well in the tutoring setting because the tutor asks most of the questions and knows what types of answers (good and bad) the student is likely to produce. LSA calculates the similarity between two sentences (or texts of any size) using a vector-based representation that is learned from a large corpus of domain-relevant texts [15]. The pattern of co-occurrence of words in the paragraphs of the corpus causes similar words to be represented with similar vectors. Word vectors are combined to represent entire texts, and the similarity between texts is determined by the cosine between the vectors [10]. In the tutoring setting, LSA has been shown to evaluate student responses nearly as well as human raters with intermediate domain knowledge [15]. For tutoring, this is good news, because even peer tutors have been shown to produce significant learning gains [16].

A strength of LSA is that it is robust to grammatical errors in the student inputs. RMT has additionally included an automatic spell-checking mechanism based on GNU Aspell, the most effective available [17]. For words that are not recognized by LSA, RMT compares Aspell's suggested respellings to find the first one that is defined in LSA. This allows RMT to avoid asking the student to either re-enter the response, or to confirm which word was intended.

Dialog Act Transition Network

Each tutor "turn" can perform three different functions: evaluate the student's previous utterance (e.g. "Good!"), confirm or add some additional information (e.g. "The dependent variable is test score."), and produce an utterance that keeps the dialog moving. Like AutoTutor, RMT uses pumps (e.g. "What else?"), prompts (fill-in-the-blank sentences), and hints (follow-up questions) to try to get the student to add information about the current topic. RMT also asks questions, summarizes topics, and answers questions.

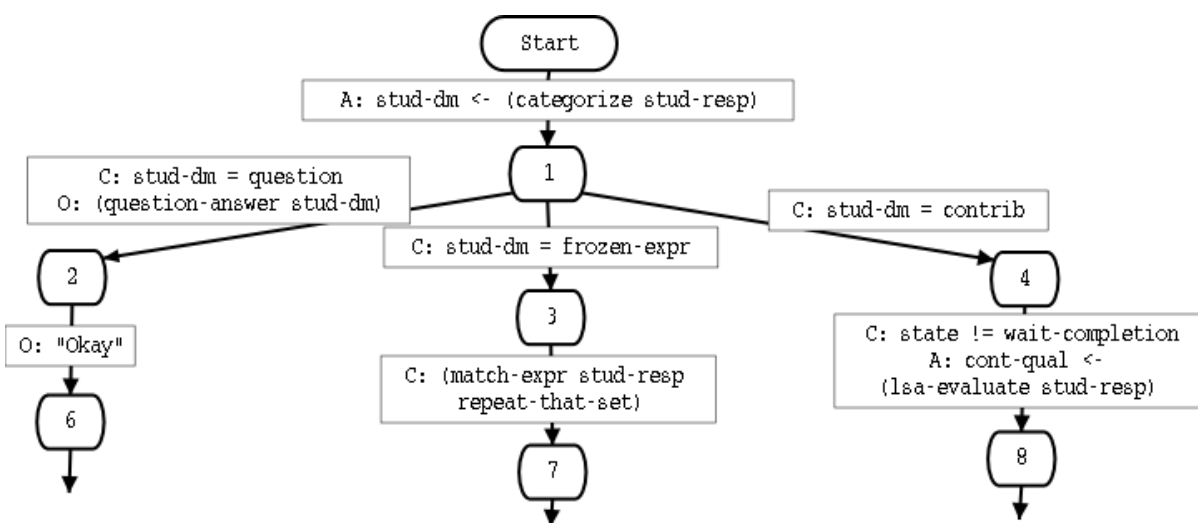


Figure 2: A partial decision network

The DATN is a domain-independent mechanism for determining which type of response the tutor will give. A DATN graphically depicts the conditions, actions and system output types. Figure 2 shows a segment of RMT's DATN. When a student response is received, the diaman invokes the DATN and processing begins at the Start state. The paths through the network eventually join back up at the Finish state, not shown here.¹ On the arcs, the items marked C are the conditions for that arc to be chosen. The items labeled A are actions that will be performed. The predicates and functions that the DATN invokes are either domain-independent, or are

¹If a traversal of the DATN does not end at the Finish state, then the DATN is incomplete.

represented as abstractions which can be instantiated in a domain-dependent method by the curriculum script (described below).

For example, on the arc from the start state in Figure 2, the DATN invokes a domain-independent method for categorizing the student response based on its form, and puts the result in a register called `stud-dm`. The link from node 1 to node 2 is followed if the value in that register is `question`, and produces the output returned by the *domain-dependent* `question-answer` method. The items marked O are outputs — what the tutor will say next. Because this graph-based representation controls utterance selection, the tutor's behavior can be modified by simply modifying the graph. Although this technique has long been used for parsing natural language sentences, RMT is the only dialog-based ITS that uses it for determining the tutor's next utterance. In relation to the AutoTutor system, this mechanism provides the same computational power for determining the next dialog act, but it is much more flexible. The tutor's behavior can be altered simply by modifying the DATN graph.

On Dialog Planning

AutoTutor and RMT have taken a simpler approach to dialog planning than several other dialog-based tutoring systems [13, 4, 18]. These systems are based on the premise that there are particular subtle points of reasoning which can best be taught by saying just the right thing (or asking just the right question) at just the right time, and they use sophisticated dialog planner and domain reasoning to determine the appropriate thing to say. The assumption with RMT and AutoTutor is that this depth of reasoning is seldom, if ever, needed for effective tutoring. Instead, they take a curriculum-script-based approach which is based on studies of human tutors [1]. The idea is that the tutor comes in to a tutoring session with a set of questions, problems and/or examples that they are prepared to discuss with the student. The tutor selects a topic, and initiates a discussion of it. The student and tutor together offer pieces of the “puzzle”, until enough information has been brought out.

The main types of speech acts that human tutors (and AutoTutor and RMT) use are pumps (e.g. “What else? ”), hints (essentially sub-questions), and prompts (sentence completions). With this approach, a sophisticated dialog planner is not needed. The tutor simply evaluates student responses by comparing them with expected answers and uses the curriculum script as a source of relevant things to say next.

This approach has significant implications for portability of the tutoring systems to different domains. The planning-based approach requires a detailed conceptual representation of the domain which must be constructed by a knowledge engineer. But a curriculum script can be created by a subject area expert without detailed knowledge of the nuts and bolts of the tutoring system [19].

The important question here is if the simpler curriculum-script approach is as effective for learning as the planning-based approach. Unfortunately, this is rather difficult to evaluate. In order to test this claim, it would be necessary to implement tutoring systems that are equivalent in every way except the power of their reasoning systems. This would obviously be a very difficult undertaking. On the other hand, if a tutoring system were to exist which included a full dialog-planning approach as well as the basic elements of a tutoring system, then the comparison could be made — if, of course, the developers want to risk finding out that all the effort that they put into the planner and the knowledge representation was in vain.

Student modeling

Within the field of intelligent tutoring systems, a large amount of effort has gone into student modeling [20, 21]. A standard technique is to create a structured representation of the desired knowledge in the tutoring area and then layer on top of that the concepts that the student has demonstrated [22]. But human tutors appear to have only very shallow knowledge of what their students know [23]. For this reason, RMT follows AutoTutor's lead in tracking only the basic information about student performance: average response quality, topics covered, and a measure of student initiative (how much they drive the conversation). It is possible that more knowledge would help the tutor help the student better, but that has not yet been demonstrated.

Logging

For data collection purposes, RMT borrows a piece of wisdom from a very successful reading tutor called Project LISTEN, “Log everything” [24]. As it interacts with a student, RMT stores in a database information

about each interaction. The database collects and relates the individual utterances and a variety of other variables, for example, the type and quality of a student response. The database also contains information about the students and the tutoring conditions that they are assigned to. Thus, in addition to providing data for our current experiments (described elsewhere [25]), we will be able to perform *post hoc* analyses by selecting relevant tutoring topics. (For example, “Is there a difference in student response quality on Mondays and Fridays?”)

As mentioned above, this information could also be used pedagogically. RMT could use information about the student’s previous use of the system to guide it when selecting new topics. And the tutor could also use the student’s pretest scores to focus on problem areas (although it currently does not do this).

Talking heads

As AutoTutor does, RMT uses an animated agent with synthesized speech to present the tutor’s utterances to the student. In principle, this allows the system to use multiple modes of communication to deliver a richer message. For example, the tutor can avoid face-threatening direct negative feedback, but still communicate doubt about an answer with a general word like “Well” with the proper intonation. Furthermore, in relation to text-only tutoring, the student is more likely to “get the whole message” because they can not simply skim over the text.

There are also potential disadvantages to using talking heads. The synthesized speech may not be comprehensible to the students. It can be augmented with a “speech bubble” that simultaneously shows the words, but then the student might not notice any facial expressions. Furthermore, the rate at which the talking head speaks the text may be either too slow or too fast to accommodate the students patience and/or speed of comprehension.

RMT can also run in text-only mode via the internet (in addition to the console version mentioned above). This allows us to explore exactly what (if anything) the animated agents add to the tutoring situation.

Testing

As a complement to the online tutor, we have implemented a dynamic testing system which presents multiple choice and short-answer essay questions and logs the student answers. Test questions are stored in XML format, so they are easily created and modified by instructors. For multiple-choice items, the correct answer is also encoded, so we can give immediate feedback to the students on their results. The testing component is also linked to the logger, so we can ensure that the student has completed enough of the discussion topics before they get a post-test, for example.

Curriculum Script

A number of studies have shown that human tutors use a “curriculum script”, or a rich set of topics which they plan to cover during a tutoring session [1, 26, 27]. RMT’s curriculum script serves the same function. It is the repository of the system’s knowledge about the tutoring domain. In particular, it contains the topics that can be covered, the questions that the tutor can ask, the answers that it expects it might get from the students, and a variety of dialog moves to keep the discourse going. RMT’s curriculum script currently contains approximately 2500 items in 5 topics. We believe that this gives us a reasonable starting point for using the tutoring system throughout a significant portion of a quarter-long class.

```
<PICTURE-QUESTION-ANSWER NAME="Lighting1" TYPE="ANALYTIC">
  <INFO>Now I'll give you a description of an experiment, and ask
    you some questions related to the validity of the experiment. The
    description is in this scenario. </INFO>
  <PICTURE TYPE="HTML">An experimenter wants to test whether an
    increase in lighting can help test-taking ability. It is the
    researcher's hypothesis ... </PICTURE>
  <QUESTION>The concept of internal validity in research ... What
    is one way that other things besides the lighting may have
```

```

    influenced the test scores?</QUESTION>
  <GOOD>
    <TARGET>Students took the exam at different times of the
      day.</TARGET>
    <ELAB>The 7th grade students took the exam in the morning, and
      the 8th grade students took the exam after lunch.</ELAB>
    <HINT>
      <CONTENTS>When did each group take the exam? </CONTENTS>
      <HINTC>The 7th graders took it in the morning, and the 8th
        grades took it after lunch.</HINTC>
    </HINT>
    <PROMPT>
      <CONTENTS>The two different groups took the test at
        different</CONTENTS>
      <PROMPTC>times of the day.</PROMPTC>
    </PROMPT>
  </GOOD>
  ...
  <SUMMARY>Confounds are outside factors that vary along with the
    independent variable that keep us from knowing if the independent
    variable ...</SUMMARY>
</PICTURE-QUESTION-ANSWER>

```

Figure 3: A segment of the curriculum script

A portion of the (xml-formatted) curriculum script is shown in figure 3. The main questions are named, and can include ordering information to control the relative order of presentation. We have included three types of questions:

1. **Conceptual:** geared at getting descriptions of concepts or simple relationships in the domain,
2. **Analytic:** critical analysis of example scenarios, and
3. **Synthetic:** design-oriented tasks for testing given hypotheses.

This item includes an information delivery item which is presented before the question to “ground” it. It also includes a “picture” which can either be a graphic, or a textual description that supports the question. The GOOD items contain the different aspects of an ideal answer that the tutor is trying to evoke from the student. There are 4 – 7 GOODs per question. Each GOOD contains one or more TARGETs that the student answer is compared to. If the LSA match is sufficiently high (above the current empirically determined threshold of 0.5), that aspect is marked as covered, and is not discussed (although it may come up in the summary). The highest sub-threshold match is chosen by the tutor to discuss. The tutor will choose one of the associated dialog moves (marked as ELAB, HINT, and ELAB) to try to elicit the information from the student. Hints and prompts have associated completions which are specific strings that are matched against the ensuing student answer with both LSA and a keyword matching technique. If the match is sufficiently high, the aspect is marked as covered, and the tutor chooses a new one. If not, the tutor chooses a different dialog move from the same aspect.

Discussion

Operational experience

Because RMT is designed to be used in conjunction with classes on an everyday basis, there are many significant technical issues to overcome. In addition to the logging mechanism described above, user management (logins and passwords) and user support (installation and dealing with errors) are significant issues. Unfortunately, our biggest problem so far has been getting students to use the system. We have made RMT available in the Winter and Spring quarters of 2004, and have gotten initial high interest from the students, but only a handful have used the system more than once or twice. In the first term, use of the system was completely

voluntary. In the second term, extra credit was given to students who completed the five topics and took the pre- and post-test. In future terms, one or more members of our research group will be teaching the research methods course, so we will be able to more closely integrate the use of the tutor with the class.

On the positive side, the students who have used the system have given favorable reports on it. They felt it would help them learn, and appreciated the opportunity to discuss the topics with the tutor. Experimental results have been mixed so far, but have produced significant learning gains for some topics [25].

The other significant issue that we have identified for use of the tutoring system is the installation of the agent-based version. The Microsoft Agents system can only be used with particular configurations of Internet Explorer, and requires the students to install the software on their personal computers. We suspect that some students are either unwilling or unable to install the software themselves, and are looking into ways of facilitating the software installation.

Comparison with other dialog-based tutors

In the introduction to this paper, RMT was described as a second-generation descendant of the AutoTutor system. In this section, we will describe what that means. RMT is a completely new system, but is informed by the experience of the AutoTutor project. The main goals of the new implementation are to make it flexible and modular so that it can be used to test out many different styles or modalities of tutoring. It was written in Lisp instead of Java so that it could take advantage of Lisp's fast-prototyping capabilities. It uses web-based delivery as AutoTutor does, but, with the exception of the agent software, the client side is just a browser window. In text-only mode, no software must be installed.

For the subcomponents of the system, we trimmed some of the functionality of AutoTutor that did not appear to significantly influence its effectiveness. For example, AutoTutor uses a neural network to classify student responses as questions, contributions, or frozen expressions. RMT uses a simple function that seems to work well. (We have not yet separately evaluated it because it is not the focus of our current research.) RMT does not include the fuzzy logic mechanism for dynamically determining successive topics. They are simply chosen randomly, subject to ordering constraints specified in the curriculum script.

RMT also includes some extensions to the AutoTutor approach. The dialog manager described above produces the same general behavior as AutoTutor's, but its implementation as a full-fledged transition network allows researchers to easily modify its behavior just by modifying the graphical representation of the network. Additionally, RMT includes an automatic spellchecker. We have not yet evaluated the results of the inclusion of this on the system's understanding of the student utterances, but a quick scan of a set of recent transcripts revealed misspellings or typos in more than 15% of student responses. Because these occur most frequently in content words instead of function words, these would necessarily affect the system's ability to evaluate the responses. RMT's use of XML to represent the curriculum script has two main advantages. First, the representation language can be specified and can be used to validate the scripts after modification. Second, the language can be easily modified to allow inclusion of manipulated alternatives for specific research projects. For example, to explore different motivating factors, we are currently developing a "research assistant" version of the system, where the tutor presents itself as the student's employer on a research project rather than a straight tutor. To allow this, we can simply add elements to the curriculum script which are conditionally presented based on the version of the system (tutor or research assistant). Finally, we have been experimenting with the inclusion of "dialog chains" to allow the system to pursue a specific line of reasoning when triggered by a particular student response. These are similar to the Knowledge Construction Dialogs used in Atlas/Andes [13]. These can also be graphically represented, and are included in the curriculum script with XML markup.

Conclusions

We feel that in the long run, this type of system will be shown to be a valuable adjunct to classroom instruction. With a dialog-based tutoring system, the student can interact in a natural way using their own words. The process of constructing responses to the tutor's questions can in itself help the students "firm up the ideas" in their heads. RMT has been shown to help students learn the rather difficult material covered in Psychology research methods classes. As we continue to develop and refine the system, we hope that it can eventually become another standard mechanism for augmenting the students' education. As a research platform, RMT's

flexibility and modularity should allow us to quickly reconfigure the system to evaluate a variety of research questions in dialog-based tutoring like those described above.

Acknowledgements

This work was supported in part by a grant from the DePaul University Quality of Instruction Council. The curriculum script for research methods in Psychology was created by Elizabeth Arnott and David Allbritton and they have also taken the lead in the evaluations of the system. Oussama BenKhadra and Jesse Efron contributed aspects of the programming for the tutoring system.

References

- [1] Graesser, A.C., Person, N.K., Magliano, J.P.: Collaborative dialogue patterns in naturalistic one-to-one tutoring. *Applied Cognitive Psychology* **9** (1995) 359–387
- [2] Hume, G.D., Michael, J., Rovick, A., Evens, M.W.: Hinting as a tactic in one-on-one tutoring. *The Journal of the Learning Sciences* **5** (1996) 23–47
- [3] Freedman, R., Zhou, Y., Glass, M., Kim, J., Evens, M.: Using rule induction to assist in rule construction for a natural-language based intelligent tutoring system. In: *Proceedings of the 20th Annual Conference of the Cognitive Science Society*, Hillsdale, NJ, Erlbaum (1998) 362–367
- [4] Glass, M.: Processing language input in the CIRCSIM-tutor intelligent tutoring system. In Moore, J., Redfield, C., Johnson, W., eds.: *Artificial Intelligence in Education*, Amsterdam, IOS Press (2001) 210–221
- [5] Aleven, V., Popescu, O., Koedinger, K.R.: Towards tutorial dialog to support self-explanation: Adding natural language understanding to a cognitive tutor. In: *Proceedings of the 10th International Conference on Artificial Intelligence in Education*. (2001)
- [6] Zinn, C., Moore, J.D., Core, M.G., Varges, S., Porayska-Pomsta, K.: The BE&E tutorial learning environment (BEETLE). In: *Proceedings of the Seventh Workshop on the Semantics and Pragmatics of Dialogue (DiaBruck 2003)*. (2003) Available at <http://www.coli.uni-sb.de/diabruck/>.
- [7] Wiemer-Hastings, P., Graesser, A., Harter, D., the Tutoring Research Group: The foundations and architecture of AutoTutor. In Goettl, B., Half, H., Redfield, C., Shute, V., eds.: *Intelligent Tutoring Systems, Proceedings of the 4th International Conference*, Berlin, Springer (1998) 334–343
- [8] Graesser, A., Jackson, G., Mathews, E., Mitchell, H., Olney, A., Ventura, M., Chipman, P., Franceschetti, D., Hu, X., Louwense, M., Person, N., TRG: Why/autotutor: A test of learning gains from a physics tutor with natural language dialog. In: *Proceedings of the 25th Annual Conference of the Cognitive Science Society*, Mahwah, NJ, Erlbaum (2003)
- [9] Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. *Journal of the American Society for Information Science* **41** (1990) 391–407
- [10] Landauer, T., Dumais, S.: A solution to Plato’s problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review* **104** (1997) 211–240
- [11] Landauer, T.K., Laham, D., Rehder, R., Schreiner, M.E.: How well can passage meaning be derived without using word order? a comparison of Latent Semantic Analysis and humans. In: *Proceedings of the 19th Annual Conference of the Cognitive Science Society*, Mahwah, NJ, Erlbaum (1997) 412–417
- [12] Foltz, P.: Latent semantic analysis for text-based research. *Behavior Research Methods, Instruments, and Computers* **28** (1996) 197–202
- [13] Rosé, C., Jordan, P., Ringenber, M., S. Siler and, K.V., Weinstein, A.: Interactive conceptual tutoring in Atlas-Andes. In: *Proceedings of AI in Education 2001 Conference*. (2001)
- [14] Kanejiya, D., Kumar, A., Prasad, S.: Automatic evaluation of students’ answers using syntactically enhanced LSA. In: *Proceedings of the Human Language Technology Conference (HLT-NAACL 2003) Workshop on Building Educational Applications using NLP*. (2003) available at: <http://www.cse.iitd.ernet.in/ eer99010/pub/hlt-naacl03.pdf>.

- [15] Wiemer-Hastings, P., Wiemer-Hastings, K., Graesser, A.: Improving an intelligent tutor's comprehension of students with Latent Semantic Analysis. In Lajoie, S., Vivet, M., eds.: *Artificial Intelligence in Education*, Amsterdam, IOS Press (1999) 535–542
- [16] Bloom, B.S.: The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher* **13** (1984) 4–16
- [17] Atkinson, K.: Spell checker test kernel results. Web page (2002) url: <http://aspell.net/test/>, accessed Dec 7, 2003.
- [18] Freedman, R.: An approach to increasing programming efficiency in plan-based dialogue systems. In: *Proceedings of the 10th International Conference on AI in Education (AIED 2001)*, Amsterdam, IOS Press (2001)
- [19] Susarla, S., Adcock, A., Van Eck, R., Moreno, K., Graesser, A.: Development and evaluation of a lesson authoring tool for autotutor. In Alevan, V., Hoppe, U., Kay, J., Mizoguchi, R., Pain, H., Verdejo, F., Yacef, K., eds.: *AIED2003 Supplemental Proceedings*, Sydney, Australia, University of Sydney School of Information Technologies (2003) 378–387
- [20] VanLehn, K., Niu, Z., Siler, S., Gertner, A.: Student modeling from conventional test data: A bayesian approach without priors. In Goettl, B., Halff, H., Redfield, C., Shute, V., eds.: *Intelligent Tutoring Systems, Proceedings of the 4th international conference*, Berlin, Springer (1998) 434–443
- [21] Katz, S., Lesgold, A., Eggan, G., Gordin, M.: Modelling the student in sherlock ii. *International Journal of Artificial Intelligence in Education* **3** (1992) 495–518
- [22] Goldstein, I.: The genetic graph: A representation for the evolution of procedural knowledge. In Sleeman, D.H., Brown, J.S., eds.: *Intelligent Tutoring Systems*. Academic Press (1982) 51–77
- [23] Person, N.K., Graesser, A.C., Magliano, J.P., Kreuz, R.J.: Inferring what the student knows in one-to-one tutoring: The role of student questions and answers. *Learning and Individual Differences* **6** (1994) 205–229
- [24] Mostow, J., Aist, G.: Evaluating tutors that listen. In Forbus, K., Feltovich, P., eds.: *Smart Machines in Education*. AAAI Press, Menlo Park, CA (2001) 169–234
- [25] Wiemer-Hastings, P., Allbritton, D., Efron, J., Arnott, E.: Rmt: A dialog-based research methods tutor with or without a head. In: *Proceedings of the ITS2004 Seventh International Conference*, Maciao, Brazil, Elsevier (2004) available at <http://reed.cs.depaul.edu/peterwh/papers/its2004.pdf>.
- [26] McArthur, D., Stasz, C., Zmuidzinas, M.: Tutoring techniques in algebra. *Cognition and Instruction* **7** (1990) 197–244
- [27] Putnam, R.T.: Structuring and adjusting content for students: A study of live and simulated tutoring of addition. *American Educational Research Journal* **24** (1987) 13–48

