

Intelligent Tutoring Systems with Conversational Dialogue

Arthur Graesser, University of Memphis,
Kurt VanLehn, Carolyn Rosé and Pamela Jordan,
University of Pittsburgh,
Derek Harter, University of Memphis

Corresponding Author:

Art Graesser
Department of Psychology
202 Psychology Building
University of Memphis
Memphis, TN 38152-3230
(901) 678-2742
(901) 678-2579 (fax)
a-graesser@memphis.edu

Abstract

Many of the Intelligent Tutoring Systems that have been developed during the last 20 years have proven to be quite successful, particularly in the domains of mathematics, science, and technology. They produce significant learning gains beyond classroom environments. They are capable of engaging most students' attention and interest for hours. We have been working on a new generation of Intelligent Tutoring Systems that hold mixed-initiative conversational dialogues with the learner. The tutoring systems present challenging problems and questions to the learner, the learner types in answers in English, and there is a lengthy multi-turn dialogue as complete solutions or answers evolve. This article presents the tutoring systems that we have been developing. AutoTutor is a conversational agent, with a talking head, that helps college students learn about computer literacy. Andes, Atlas, and Why2 help adults learn about physics. Instead of being mere information delivery systems, our systems help students actively construct knowledge through conversations.

KEYWORDS: Intelligent tutoring systems, conversational agents, pedagogical agents

Introduction

Intelligent Tutoring Systems (ITSs) are clearly one of the successful enterprises in artificial intelligence. There is a long list of ITSs that have been tested on humans and have proven to facilitate learning. There are well-tested tutors of algebra, geometry, and computer languages (such as PACT: Koedinger, Anderson, Hadley, and Mark, 1997), of physics (such as Andes: Gertner and VanLehn 2000, VanLehn 1996), and of electronics (such as SHERLOCK: Lesgold, Lajoie, Bunzo, and Eggan 1992). These ITSs use a variety of computational modules that are familiar to those of us in the world of AI: production systems, Bayesian networks, schema-templates, theorem proving, and explanatory reasoning. According to the current estimates, the arsenal of sophisticated computational modules inherited from AI produce learning gains of approximately .3 to 1.0 standard deviation units compared with students learning the same content in a classroom (Corbett, Anderson, Graesser, Koedinger, and VanLehn 1999).

The next generation of ITSs are expected to go one step further by adopting conversational interfaces. The tutor will speak to the student with an agent that has synthesized speech, facial expressions, and gestures, in addition to the normal business of having the computer display print, graphics, and animation. Animated conversational agents have now been developed to the point that they can be integrated with ITSs (Cassell and Thorisson 1999; Johnson, Rickel, and Lester 2000; Lester, Voerman, Townes, and Callaway 1999). Learners will be able to type in their responses in English in addition to the conventional point and click. Recent developments in computational linguistics (Jurafsky and Martin 2000) have made it a realistic goal to have computers comprehend language, at least to an extent where the ITS can respond with something relevant and useful. Speech recognition would be highly desirable, of course, as long as it is also reliable.

At this point, we are uncertain whether the conversational interfaces will produce incremental gains in learning over and above the existing ITSs (Corbett et al. 1999). But there are reasons for being optimistic. One reason is that human tutors produce impressive learning gains (between .4 and 2.3 standard deviation units over classroom teachers), even though the vast majority of tutors in a schools system have modest

domain knowledge, have no training in pedagogical techniques, and rarely use the sophisticated tutoring strategies of ITSs (Cohen, Kulik, and Kulik 1982; Graesser, Person, and Magliano 1995). A second reason is that there is at least one success case, namely the AutoTutor system that we will discuss in this article (Graesser, K. Wiemer-Hastings, P. Wiemer-Hastings, Kreuz, and the Tutoring Research Group 1999). AutoTutor is a fully automated computer tutor that has tutored approximately 200 college students in an introductory course in computer literacy. One version of AutoTutor simulates conversational patterns of unskilled human tutors, without any sophisticated tutoring strategies. This version of AutoTutor improves learning by .5 standard deviation units (i.e., about a half a letter grade), when compared to a control condition where students reread yoked chapters in the book. Thus, it appears that there is something about conversational dialogue that plays an important role in learning. We believe that the most effective tutoring systems of the future will be a hybrid between normal conversational patterns and the ideal pedagogical strategies in the ITS enterprise.

This article will describe some of the tutoring systems that we are developing to simulate conversational dialogue. We will begin with AutoTutor. Then we will describe a series of physics tutors that vary from conventional ITS systems (the Andes tutor) to agents that attempt to comprehend natural language and plan dialogue moves (Atlas and Why2).

AutoTutor

The Tutoring Research Group (TRG) at the University of Memphis developed AutoTutor to simulate the dialogue patterns of typical human tutors (Graesser et al. 1999; Person, Graesser, Kreuz, Pomeroy, and TRG in press). AutoTutor tries to comprehend student contributions and to simulate dialog moves of either normal (unskilled) tutors or sophisticated tutors. AutoTutor is currently being developed for college students who are taking an introductory course in computer literacy. These students learn the fundamentals of computer hardware, the operating system, and the Internet.

Figure 1 is a screen shot that illustrates the interface of AutoTutor. The left window has a talking head that acts as a dialogue partner with the learner. The talking head delivers AutoTutor's dialog moves with synthesized speech, intonation, facial expressions, nods, and gestures. The major question (or problem) that the learner is working on is both spoken by AutoTutor and is printed at the top of the screen. The major questions are generated systematically from a curriculum script, a module that will be discussed later. AutoTutor's major questions are not the fill-in-the blank, true/false, or multiple choice questions that are so popular in the US educational system. Instead, the questions invite lengthy explanations and deep reasoning (such as why, how, and what-if questions). The goal is to encourage students to articulate lengthier answers that exhibit deep reasoning, rather than to deliver short snippets of shallow knowledge. There is a continuous multi-turn tutorial dialog between AutoTutor and the learner during the course of answering a deep-reasoning question. When considering both the learner and AutoTutor, it typically takes 10 to 30 turns during the tutorial dialog when a single question from the curriculum script is answered. The learner types in his/her contributions during the exchange by keyboard, as reflected in the bottom window. For some topics, as in Figure 1, there are graphical displays and animation, with components that AutoTutor points to. AutoTutor was designed to be a good conversation partner that comprehends, speaks, points, and displays emotions, all in a coordinated fashion.

Insert Figure 1 about here: A Screenshot of AutoTutor.

An example AutoTutor -learner dialogue. Figure 2 shows a dialogue between a college student and AutoTutor. Prior to this question, the student had been asked and attempted to answer 6 previous questions about the Internet. The Internet was the macrotopic and students were tutored by answering several deep-reasoning questions about the Internet. It should be noted that this is not a fabricated toy conversation. It is a *bona fide* dialogue from our corpus of approximately 200 AutoTutor-student dialogues in a computer literacy course.

AutoTutor begins this exchange by asking a how-question in turn 1:
What hardware do you need to take photos and send them over the

Internet? But AutoTutor doesn't merely pop the question out of the blue. It first presents a discourse marker that signals a change in topic (*Alright, let's go on*), presents a context to frame the question (*You want to take photos and send them over the Internet.*), and then presents a discourse marker that signals the questions (*Consider this problem*). Therefore, AutoTutor monitors different levels of discourse structure and functions of dialogue moves. AutoTutor inserts appropriate discourse markers that clarify these levels and functions to the learner. Without these discourse markers, learners are confused about what AutoTutor is doing and what they are supposed to do next. A Dialogue Advancer Network (DAN) has been designed to manage the conversational dialogue (Person et al. in press). The DAN is a finite state automaton that can handle different classes of information that learners type in. The DAN is augmented by production rules that are sensitive to learner ability and several parameters of the dialogue history.

Insert Figure 2 about here

How does AutoTutor handle the student's initial answer to the question? After AutoTutor asks the question in the TUTOR-1 turn, the student gives an initial answer in the STUDENT-1 turn. The answer is very incomplete. A complete answer would include all of the points in the summary at the final turn (TUTOR-30). So what does AutoTutor do with this incomplete student contribution. AutoTutor doesn't simply grade the answer (e.g., good, bad, incomplete, a quantitative score), as many conventional tutoring systems do. Instead, AutoTutor stimulates a multi-turn conversation that is designed to extract more information from the student and to get the student to articulate pieces of the answer. Thus, instead of being an information delivery system that bombards the student with a large volume of information, AutoTutor is a discourse prosthesis that attempts to get the student to do the talking and that explores what the student knows. AutoTutor adopts the educational philosophy that students learn by actively constructing explanations and elaborations of the material (Chi, de Leeuw, Chiu, and LaVanher, 1994; Conati and VanLehn 1999).

How does AutoTutor get the learner to do the talking? AutoTutor has a number of dialogue moves for that purpose. For starters, there are open-ended pumps that encourage the student to say more, such as

What else? in the TUTOR-2 turn. Pumps are very frequent dialogue moves after the student gives an initial answer, just as is the case with human tutors. The tutor pumps the learner for what the learner knows before drilling down to specific pieces of an answer. After the student is pumped for information, AutoTutor selects a piece of information to focus on. Both human tutors and AutoTutor have a set of expectations about what should be included in the answer. What they do is manage the multi-turn dialogue to cover these expected answers. A complete answer to the example question in Figure 2 would have four expectations, as listed below.

Expectation-1: You need a digital camera or regular camera to take the photos.

Expectation-2: If you use a regular camera, you need to scan the pictures onto the computer disk with a scanner.

Expectation-3: A network card is needed if you have a direct connection to the Internet.

Expectation-4: A modem is needed if you have a dial-up connection.

AutoTutor decides which expectation to handle next and then selects dialogue moves that flesh out the expectation. The dialogue moves vary in directness and information content. The most indirect dialogue moves are hints, the most direct are assertions, and prompts are in between. Hints are often articulated in the form of questions, designed to lead the learner to construct the expected information. Assertions directly articulate the expected information. Prompts try to get the learner to produce a single word in the expectation. For example the tutor turns 3, 4, 5, and 6 in Figure 2 are all trying to get the learner to articulate Expectation 3. Hints are in the TUTOR-3 turn (*For what type of connection do you need a network card?*) and the TUTOR-5 turn (*How does the user get hooked up to the internet?*). Prompts are in TUTOR-4 (*If you have access to the Internet through a network card, then your connection is...--* with a hand gesture encouraging the learner to type in information). Assertions are in TUTOR-5 and TUTOR-6 (*A network card is needed if you have a direct connection to the Internet.*). AutoTutor attempts to get the learner to articulate any given expectation E by going through two cycles of hint-prompt-assertion. Most students manage to articulate the expectation within the 6 dialogue moves (hint-prompt-assertion-hint-prompt-assertion).

AutoTutor exits the 6-move cycle as soon as the student has articulated the expected answer. Interestingly, sometimes students are unable to articulate an expectation even after AutoTutor spoke it in the previous turn. After expectation E is fleshed out, AutoTutor selects another expectation.

How does AutoTutor know whether a student has covered an expectation? AutoTutor does a surprisingly good job evaluating the quality of the answers that learners type in. AutoTutor attempts to “comprehend” the student input by segmenting the contributions into speech acts and matching the student’s speech acts to the expectations. Latent semantic analysis (LSA) is used to compute these matches (Landauer, Foltz, and Laham 1998). When expectation E is compared with speech act A, a cosine match score is computed that varies from 0 (no match) to 1.0 (perfect match). AutoTutor uses a *max* function that considers each speech act and all combinations of speech acts that the learner gives in their turns during the evolution of an answer to a major question; the value of the highest cosine match is used when computing whether expectation E is covered by the student. LSA is a statistical, corpus-based method of representing knowledge. LSA provides the foundation for grading essays, even essays that are not well formed grammatically, semantically, and rhetorically. LSA-based essay graders can assign grades to essays as reliably as experts in composition (Landauer et al. 1998). Our research has revealed that AutoTutor is almost as good as an expert in computer literacy in evaluating the quality of student answers in the tutorial dialog (Graesser, P. Wiemer-Hastings, K. Wiemer-Hastings, Harter, Person, and TRG 2000).

How does AutoTutor select the next expectation to cover? AutoTutor uses LSA in conjunction with various criteria when deciding which expectation to cover next. After each student turn, AutoTutor updates the LSA score for each of the four expectations listed above. An expectation is considered covered if it meets or exceeds some threshold value (e.g., .70 in our current tutor). One selection criterion uses the zone of proximal development to select the next expectation; this is the highest LSA score that is below threshold. A second criterion uses coherence, the expectation that has the highest LSA overlap with the previous expectation that was covered. Other criteria that are currently being implemented are preconditions and pivotal expectations. Ideally,

AutoTutor will decide to cover a new expectation in a fashion that both blends in the conversation and that advances the agenda in an optimal way. AutoTutor generates a summary after all of the expectations are covered (e.g., the TUTOR-30 turn).

How does AutoTutor give feedback to the student? There are three levels of feedback. First, there is backchannel feedback that acknowledges the learner's input. AutoTutor periodically nods and says *uh-huh* after learners type in important nouns, but is not differentially sensitive to the correctness of the student's nouns. The backchannel feedback occurs on-line, as the learner types in the words of the turn. Learners feel that they have an impact on AutoTutor when they get feedback at this fine-grain level. Second, AutoTutor gives evaluative pedagogical feedback on the learner's previous turn, based on the LSA values of the learner's speech acts. The facial expressions and intonation convey different levels of feedback, such as negative (e.g., *not really* while head shakes), neutral negative (*okay* with a skeptical look), neutral positive (*okay* at a moderate nod rate), and positive (*right* with a fast head nod). Third, there is corrective feedback that repairs bugs and misconceptions that learners articulate. Of course, these bugs and their corrections need to be anticipated ahead of time in AutoTutor's curriculum script. This mimics human tutors. Most human tutors anticipate that learners will have a variety of particular bugs and misconceptions when they cover particular topics. An expert tutor often has canned routines for handling the particular errors that students make. AutoTutor currently splices in correct information after these errors occur, as in turn TUTOR-8. Sometimes student errors are ignored, as in TUTOR-4 and TUTOR-7. These errors are ignored because AutoTutor has not anticipated them by virtue of the content in the curriculum script. AutoTutor evaluates student input by matching it to what it knows in the curriculum script, not by constructing a novel interpretation from whole cloth.

How does AutoTutor handle mixed-initiative dialogue? We know from research on human tutoring that it is the tutor who controls the lion's share of the tutoring agenda (Graesser et al. 1995). Students rarely ask information-seeking questions and introduce new topics. However, when learners do take the initiative, AutoTutor needs to be ready to handle these contributions. AutoTutor does a moderately good job in

managing mixed-initiative dialogue. AutoTutor classifies the learner's speech acts into the following categories:

- Assertion (*Ram is a type of primary memory*)
- WH-question (*What does bus mean?* and other questions that begin with *who, what, when, where, why, how*, etc.)
- YES/NO question (*Is the floppy disk working?*)
- Metacognitive comment (*I don't understand*)
- Metacommunicative act (*Could you repeat that?*)
- Short Response (*okay, yes*)

Obviously, AutoTutor's dialog moves on turn N+1 need to be sensitive to the speech acts expressed by the learner in turn N. When the student asks a *What does X mean?* question, the tutor answers the question by giving a definition from a glossary. When the learner makes an Assertion, the tutor evaluates the quality of the Assertion and gives short evaluative feedback. When the learner asks *What did you say?*, AutoTutor repeats what it said in the last turn. The Dialogue Advancer Network manages the mixed-initiative dialogue.

The curriculum script. AutoTutor has a curriculum script that organizes the content of the topics covered in the tutorial dialog. There are 36 topics, one for each major question or problem that requires deep reasoning. Associated with each topic are a set of expectations, a set of hints and prompts for each expectation, a set of anticipated bugs/misconceptions and their corrections, and (optionally) pictures or animations. It is very easy for a lesson planner to create the content for these topics because they are English descriptions rather than structured code. Of course, pictures and animations would require appropriate media files. We are currently developing an authoring tool that makes it easy to create the curriculum scripts. Our ultimate goal is to make it very easy to create an AutoTutor for a new knowledge domain. First, the developer creates an LSA space after identifying a corpus of electronic documents on the domain knowledge. The lesson planner creates a curriculum script with deep-reasoning questions and problems. The developer then computes LSA vectors on the content of the curriculum scripts. A glossary of important terms and their definitions is also prepared. After that, the built-in modules of AutoTutor do all of the rest. AutoTutor is currently implemented in

Java for Pentium computers, so there are no barriers to widespread usage.

Andes: A physics tutoring system that does
not use natural language

The goal of the second project is to use natural language processing technology to improve an already successful intelligent tutoring system named Andes (Conati, Gertner, VanLehn, & Druzdzel 1997; Conati & VanLehn 1996; Gertner, Conati, & VanLehn 1998; Gertner 1998; Gertner & VanLehn 2000; Schulze, Correll, Shelby, Wintersgill, & Gertner 1998; Schulze, Shelby, Treacy, & Wintersgill 2000; Shelby et al. in prep.; VanLehn 1996). Andes is intended to be used as an adjunct to college and high-school physics courses to help students do their homework problems.

Figure 3 shows the Andes screen. A physics problem is presented in the upper left window. Students draw vectors below it, define variables in the upper right window, and enter equations in the lower right window. When students enter a vector, variable or equation, Andes will color the entry green if it is correct and red if it is incorrect. This is called immediate feedback and is known to enhance learning from problem solving (Anderson et al. 1995; Kulik & Kulik, 1988).

Insert Figure 3 about here: The Andes tutoring system

How does Andes hint and give help? Students may ask Andes for help by either clicking on the menu item *what do I do next?* or by selecting a red entry and clicking on the menu item *what's wrong with that?* Whenever a student asks for help, Andes prints in the lower left window a short message, such as the one shown in Figure 3. The message is only a hint about what is wrong or what to do next. Often a mere hint suffices, and the students are able to correct their difficulty and move on. However, if the hint fails, then the student can ask for help again. Andes generates a second hint that is more specific than the first. If the student continues to ask for help, Andes' last hint will essentially tell the student what to do next. This technique of giving help is based on human-authored hint sequences. Each hint is represented as a template. It is filled in with text that is specific to the

situation where help was requested. Such hint sequences are often used in intelligent tutoring systems and are known to enhance learning from problem solving (McKendree 1990).

In order to give immediate feedback and hint sequences, Andes must understand the student's entries no matter how the student tries to solve the problem. Thus, it must be able to solve the problem in all possible ways. Moreover, every entry must be explainable in terms that the student will understand, in case the student asks for a hint on how or why to do it. It would be tedious or impossible for human experts to provide such information, so an expert system is used instead. It has rules for solving physics problems in all possible ways, and it annotates the solutions with the goals and rules that are relevant to each entry of each solution path. Andes compares the student's entries to those in its model of expert problem solving. It gives immediate negative feedback if the student's entry does not mention one of the solutions from the expert model. For this reason, Andes and similar tutoring systems are known as model tracing tutors. They follow the student's reasoning by comparing it to a trace of the model's reasoning.

Andes contains a second kind of knowledge that determines what to say when the student asks for help. This expertise is partly about physics and partly about teaching. For instance, if a student asks what is wrong with an equation, it must determine which correct equation is most likely to be the one that the student was trying to enter. It uses a Bayesian network to help with this (Conati et al., 1997; Gertner & VanLehn 2000). Then Andes analyzes the difference between the student's equation and the correct one. It might find, for example, that the student has left out an addend in a sum. It must then decide what might have caused that flaw. If it finds that the student has forgotten or misapplied some physics knowledge, it constructs a hint sequence that should get the student to apply the correct piece of knowledge. For example, the first hint might be, *Are you sure you have identified all the forces acting on the automobile?* The second hint, which is given only if the first one fails and the student asks for help again, would be *You seem to have neglected the friction force on the automobile.* The final hint is given only when both earlier hints failed. It tells the student to draw the missing force (if it is not drawn already) or to include it in the equation. A large amount of pedagogical expertise is needed to

determine what correct entries match the student's entries, what could cause the errors, and which hint sequence would be appropriate for handling them.

Although Andes is still being improved, evaluations in the fall of 1999 at the US Naval Academy indicate that it is already effective. Students using Andes scored about a third of a letter grade higher on the midterm exam than students in a control group (Gertner & VanLehn 2000; Shelby et al. in preparation).

Other intelligent tutoring systems use similar model tracing, immediate feedback and hint sequences techniques, and many have also been shown to be effective (e.g., Anderson et al. 1995; McKendree, Radlinski, & Atwood 1992; Reiser, Copen, Ranney, Hamid, & Kimberg in press). A new company, Carnegie Learning (www.carnegielearning.com), is producing such tutors for use in high-school mathematics classes. As of fall 2000, approximately 10% of the algebra I classes in the United States will be using one of the Carnegie Learning tutors. Clearly, this is a rapidly maturing AI technology. The main contribution of the Andes project is to extend the technology to a scientific task domain, namely physics, because most of the earlier work handled only mathematical task domains.

Criticisms of Andes and other similar tutoring systems. The pedagogy of immediate feedback and hint sequences has sometimes been criticized for failing to encourage *deep learning*. The following four criticisms are occasionally raised by colleagues.

- (1) If students don't reflect on the tutor's hints, but merely keep guessing until they find an action that gets positive feedback, they can learn to do the right thing for the wrong reasons, and the tutor will never detect the shallow learning (Aleven, Koedinger, & Cross 1999; Aleven, Koedinger, Sinclair, & Snyder 1998).
- (2) The tutor does not ask students to explain their actions, so students may not learn the domain's language. Educators have recently advocated that students learn to "talk science." Talking science allegedly is part of a deep understanding of the science. It also

facilitates scientific writing, working collaboratively in groups, and participating in the culture of science.

(3) In order to understand the students' thinking, the user interface of such systems requires students to display many of the details of their reasoning. This design doesn't promote stepping back to see the "basic approach" one has used to solve a problem. Even students who have received high grades in a physics course seldom describe these basic approaches and detect when two problems have similar basic approaches (Chi, Feltovich, & Glaser 1981).

(4) When students learn quantitative skills, such as algebra or physics problem solving, they are usually not encouraged to see their work from a qualitative, semantic perspective. As a consequence, they fail to induce versions of the skills that can be used to solve qualitative problems and to check quantitative ones for reasonableness. Even physics students with high grades often score poorly on tests of qualitative physics (Hestenes, Wells, & Swackhamer 1992).

Many of these objections can be made to just about any form of instruction. Even expert tutors and teachers have difficulty getting students to learn deeply. Therefore, these criticisms of intelligent tutoring systems should only encourage us to improve them, not reject them.

There are two common themes in the list above. First, all four involve integrating problem-solving knowledge with other knowledge, namely: (1) principles or rationales, (2) domain language, (3) abstract, basic approaches and (4) qualitative rules of inference. Second, the kinds of instructional activities that are currently used to tap these other kinds of knowledge make critical use of natural language. Although one can invent graphical or formal notations to teach these kinds of knowledge on a computer, they might be more confusing to the students and instructors than the knowledge that they are trying to convey. Moreover, students and instructors are likely to resist learning a new formalism, even a graphical one, if they will only use it temporarily.

Atlas: A natural-language enhancement for model-tracing tutors

We believe that tutoring systems must use natural language if they are to become more effective at encouraging deep learning. Therefore, we have begun building *Atlas*, a module that can be added to Andes or other similar model tracing tutoring systems in order to conduct natural language dialogs and thereby promote deep learning. Atlas uses NLP technology that was originally developed for CIRCSIM tutor (Freedman & Evens 1996), the Basic Electricity and Electronics (BEE) tutor (Rose, DiEugenio, & Moore 1999), and the COCONUT model of collaborative dialog (DiEugenio, Jordan, Thomason, & Moore in press). A number of NLU authoring tools have been developed, including the LC-FLEX parser (Rose 2000; Rose & Lavie in press).

Initial versions of Atlas will support only a simple form of interaction. Most of the time, the students interact with Andes just as they ordinarily do. However, if Atlas notices an opportunity to promote deep learning, it takes control of the interaction and begins a natural language dialog. Although Atlas can ask students to make Andes actions as part of the dialog (e.g., it might have the student draw a single vector), most of the dialog is conducted in a scrolling text window, which replaces the hint window shown in the lower left of Figure 3. When Atlas has finished leading the student through a directed line of reasoning, it signs off and lets the student return to solving the problem with Andes.

The dialogs are called knowledge construction dialogs because they are designed to encourage students to infer or construct the target knowledge. For example, Andes might simply tell the student *When an object moving in a straight line is slowing down, its acceleration is in the opposite direction to its velocity*. Atlas will instead try to draw that knowledge out of the student with a dialog like the one shown in Figure 4, where the student derived the target principle from a deeper one. Knowledge construction dialogs (KCDs) are intended to provide deeper knowledge by connecting principles, relating them to common sense knowledge, and giving the student practice in talking about them.

<p>Insert Figure 4 about here: A hypothetical dialog between Atlas and a student.</p>

Knowledge construction dialogs to teach principles. So far, Atlas conducts just one kind of KCD, namely those that teach domain principles. Currently, we are concentrating on only a small portion of physics, so only 55 principles are covered. Even so, building so many knowledge construction dialogues was daunting enough that we built tools to help us.

The primary design concept is to represent knowledge construction dialogs (KCDs) as recursive finite state networks. States correspond to tutor utterances (usually questions) and arcs correspond to student responses. A few arcs are special in that they either call a subdialog or return from one. Such recursive finite state networks are often used in spoken language dialog systems (Jurafsky & Martin 2000), so it makes sense to start with them and see where they break down.

Our primary tool is the knowledge construction dialog (KCD) editor (see Figure 5). In the upper left window, the author selects a topic, which is deceleration in this case. This causes a shorthand form of the recipes (dialog plans) to appear in the upper right window. Selecting a tutor-student interaction brings up windows for seeing the tutor's contribution (as with the lower right window) and the student's expected answers (middle windows). The left middle window is for correct answers. As in AutoTutor, the student's expected answer is represented as a set of expectations (Left and Opposite in this case). The right middle window is for incorrect answers. When one of these is selected, a subdialogue for handling it is displayed in the lower left window. Notice that the author enters natural language text for both the tutor contribution, the expectations and almost everything else.

Insert Figure 5 about here: The KCD editor
--

In a limited sense, the KCDs are intended to be better than naturally occurring dialogs. Just as most text expresses its ideas more clearly than informal oral expositions, the KCD is intended to express its ideas more clearly than the oral tutorial dialogs that human tutors generate. Thus, we need a way for expert physicists, tutors and educators to critique the KCDs and suggest improvements. Since the underlying finite state network can be complex, it is not useful to merely print it

out and let experts pencil in comments. The second tool facilitates this by allowing expert physicists and psychologists to navigate around the network and enter comments on individual states and arcs (see Figure 6). It presents a dialog in the left column, and allows the user to enter comments in the right column. Since there are many expected responses per tutorial contribution, the user can select a response from a pull down menu. This causes the whole dialog to adjust, opening up new boxes for the user's comments. This tool runs in a web browser, so it can be used by experts remotely.

Insert Figure 6 about here: The KCD commenter

At any point in its development, a KCD can be compiled into executable code. The code is interpreted by a reactive planning engine (Freedman 1999). The engine does not simply follow the finite state network. Instead, it has rudimentary (but growing) capabilities for treating the network as a plan for the conversation that it will adapt as necessary. For instance, in the conversation of Figure 4, suppose the student said at line 2, "The acceleration makes the velocity vector longer, so the elevator should be going faster." The reactive planner should recognize that the student has skipped ahead in the conversation plan, so it should have Atlas say line 7 instead of line 3.

Although the authors only see and edit natural language text, we cannot expect students to type in exactly the responses that the authors enter. The compiler uses CARMEL (Rose 2000) to translate the expected student responses into semantic structures. Thus, it should recognize the expected responses even if they are not expressed with the same words and syntax as the author-provided versions.

Current status and future directions. The current KCDs have been critiqued by some experts, but we feel we need more of this feedback in order to get the strong educational impact we are seeking. We are evaluating the system during September 2000 in order to augment the natural language processing parts of the system. We need to see what responses students actually make so that we can extend the set of expected responses (arcs) for each tutor contribution (state). In the short run, these can be added via the KCD editor. In the long run, we would like to automate as many of the components as possible. Secondly, the conventional wisdom is that a finite state network, even when interpreted by a reactive planner, does not provide sufficient flexibility for managing complex dialogs. We need to see if and how it breaks down. Lastly, we have deliberately left the Andes system's two major knowledge sources alone, so Andes is still responsible for solving physics problems and for deciding which hint sequence is appropriate. This implies that KCDs are used mostly to replace hint sequences. We are not sure if this simple design will allow pedagogically useful dialogs, or whether we will need to port some of Andes' knowledge to Atlas.

Why2: Tutoring qualitative explanations

All tutoring systems have students perform a task, and they help the students do it. Some tutoring systems, such as Andes and Atlas, have the student solve problems. Other tutoring systems, such as AutoTutor, ask the student deep questions and help them formulate a correct, complete answer. Recent work with human tutors (Chi, Siler, Jeong, Yamauchi, & Hausmann in press) suggests that a good activity for teaching is to have students explain physical systems qualitatively. Although it is possible to have them express their explanations in formal or graphical languages (such as Cyclepad: Forbus et al. 1998), we believe that they will learn more if they can express their explanations in natural language. Thus, the goal of the Why2 project is to coach students as they explain physical systems in natural language.

Why2 is intended to be a successor of one of the first intelligent tutoring systems in the literature, the Why system. Why was envisioned and partially implemented by Albert Stevens and Alan

Collins in days when computers had only 64K of memory (Stevens & Collins 1977). They studied experts helping students articulate such explanations, and tried to embed their tutorial strategies in the Why system. Stevens and Collins discovered that students had a great many misconceptions about nature. These misconceptions would only surface when students expressed their ideas qualitatively, because they could solve textbook quantitative problems correctly (Halloun & Hestenes 1985). Since that time, considerable effort has been expended by physics educators to discover, catalogue and invent remedies for student misconceptions. The remedies are usually intended for classroom or laboratories, and have had only moderate success (Hake under review). By adapting them to the tutorial setting, and embedding the tutorial strategies uncovered by Collins, Stevens and others, Why2 may be much more successful.

The basic idea of Why2 is to ask the student to type in an explanation for a simple physical situation, like the battery-bulb circuit shown in Figure 7. Why2 analyzes the student's explanation (line 1 in Figure 7) to see if the student has any misconceptions. If it detects a misconception, it invokes a knowledge construction dialog (KCD), such as the one shown in lines 2 through 9. During this dialog, further misunderstandings may arise, which can cause another KCD to be selected and applied (see lines 10 onward).

Why2 is a joint project involving both the AutoTutor and Atlas groups. It began recently and is still in the design stages. A corpus of explanations from students has been collected and is being analyzed to see what kinds of misconceptions and language the students are using. Our plan is to use a combination of the LSA technology from AutoTutor and the semantic composition technology from Atlas. The KCDs of Atlas will be generalized to incorporate elements of the DANs of AutoTutor.

Our dialogue technology may be stressed by the complexity of the language and discourse we anticipate from the students. However, if we can make it work, the pedagogical payoffs will be enormous. Repairing the qualitative misconceptions of physics is a difficult and fundamentally important problem.

Insert Figure 7 about here: A hypothetical dialog between a student and Why2.

Conclusions

We have discussed three projects that have several similarities. AutoTutor, Atlas and Why2 all endorse the idea that students learn best if they construct knowledge themselves. Thus, their dialogs try to elicit knowledge from the student by asking leading questions. They only tell the student the knowledge as a last resort. All three projects manage dialogs by using finite state networks. Since we anticipate building hundreds of such networks, the projects are building tools to let domain authors enter those dialogs in natural language. All three projects use robust natural language understanding techniques—LSA for AutoTutor, CARMEL for Atlas, and combination of the two for Why2. All three projects began by analyzing data from human tutors, and are using evaluations with human students throughout their design cycle.

Although the three tutoring systems have the common objective of helping students perform activities, the specific tasks and knowledge domains are rather different. AutoTutor's students are answering deep questions about computer technology, Atlas's students are solving quantitative problems, and Why2's students are explaining physical systems. We may ultimately discover that the conversation patterns need to be different for these different domains and tasks. That is, dialogue style may need to be distinctively tailored to particular classes of knowledge domains. A generic dialogue style may prove to be unsatisfactory. Whatever discoveries emerge, we suspect they will support one basic claim: Conversational dialogue substantially improves learning.

Acknowledgements

The AutoTutor research was supported on grants from the National Science Foundation (SBR 9720314) and the Office of Naval Research

(N00014-00-1-0600). The Andes research was supported by Grant N00014-96-1-0260 from the Cognitive Sciences Division of the Office of Naval Research. The Atlas research is supported by Grant 9720359 from the LIS program of the National Science Foundation. The Why2 research is supported by Grant N00014-00-1-0600 from the Cognitive Sciences Division of the Office of Naval Research.

Biographical Sketches

Dr. Arthur Graesser is a Professor of Psychology and Computer Science at the University of Memphis, Co-Director of the Institute for Intelligent Systems, and Director of the Center for Applied Psychological Research. He has conducted research on tutorial dialogue in intelligent tutoring systems and is current editor of the journal *Discourse Processes*.

Dr. Kurt VanLehn is a Professor of Computer Science and Intelligent Systems at the University of Pittsburgh, Director of the Center for Interdisciplinary Research on Constructive Learning Environments, and a Senior Science at the Learning Research and Development Center. His main interests are applications of AI to tutoring and assessment. He is a senior editor for the journal *Cognitive Science*.

Dr. Carolyn Rosé is a Research Associate at the Learning Research and Development Center at the University of Pittsburgh. Her main research focus is on developing robust language understanding technology and authoring tools to facilitate the rapid development of dialogue interfaces for tutoring systems.

Dr. Pamela Jordan is a Research Associate at the Learning Research and Development Center at the University of Pittsburgh. Her main interest is in interactive natural language dialogue and effective communication strategies.

Derek Harter is a Ph.D. candidate in Computer Science at the University of Memphis, and holds a B.S. in Computer Science from Purdue University and a M.S. in Computer Science and Artificial Intelligence from Johns Hopkins University. His main interests are in

dynamical and embodied models of cognition and neurologically inspired models of action selection for autonomous agents.

References

- Aleven, V., Koedinger, K. R., & Cross, K. (1999). Tutoring answer-explanation fosters learning with understanding, *Artificial Intelligence in Education* (pp. 199-206). Amsterdam: IOS Press.
- Aleven, V., Koedinger, K. R., Sinclair, H. C., & Snyder, J. (1998). Combating shallow learning in a tutor for geometry problem solving. In B. P. Goettle, H. M. Halff, C. L. Redfield, & V. J. Shute (Eds.), *Intelligent Tutoring Systems, Proceedings of the Fourth International Conference* (pp. 364-373). Berlin: Springer-Verlag.
- Cassell, J.; and Thorisson, K.R. 1999. The Power of a Nod and a Glance: Envelope vs. Emotional Feedback in Animated Conversational Agents. *Applied Artificial Intelligence*, 13: 519-538.
- Chi, M. T. H., Feltovich, P., & Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. *Cognitive Science*, 5, 121-152.
- Chi, M. T. H.; de Leeuw, N.; Chiu, M.; and LaVancher, C. 1994. Eliciting Self-explanations Improves Understanding. *Cognitive Science*, 18: 439-477.
- Chi, M. T. H., Siler, S., Jeong, H., Yamauchi, T., & Hausmann, R. G. (in press). Learning from tutoring: A student-centered versus a tutor-centered approach. *Cognitive Science*.
- Clark, H.H. (1996). *Using Language*. Cambridge: Cambridge University Press.
- Cohen, P. A.; Kulik, J. A.; and Kulik, C. C. 1982. Educational Outcomes of Tutoring: A Meta-analysis of Findings. *American Educational Research Journal*, 19: 237-248.
- Conati, C., Gertner, A., VanLehn, K., & Druzdzel, M. (1997). On-line student modeling for coached problem solving using Bayesian networks. In A. Jameson, C. Paris, & C. Tasso (Eds.), *User Modeling: Proceedings of the Sixth International conference, UM97*. New York: Springer Wien.
- Conati, C., & VanLehn, K. (1996). Probabilistic plan recognition for cognitive apprenticeship. In J. Moore & J. Fain-Lehman (Eds.), *Proceedings of the Eighteenth Annual Meeting of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum.

- Conati, C.; and VanLehn, K. 1999. Teaching Metacognitive Skills: Implementation and Evaluation of a Tutoring System to Guide Self-explanation while Learning from Examples. In, *Artificial Intelligence in Education*, eds., S.P. Lajoie and M. Vivet, 297-304, Amsterdam: IOS Press.
- Corbett, A.; Anderson, J.; Graesser, A.; Koedinger, K.; & VanLehn, K. 1999. Third Generation Computer Tutors: Learn from or Ignore Human Tutors? In *Proceedings of the 1999 Conference of Computer-Human Interaction*, 85-86. New York: ACM Press.
- DiEugenio, B., Jordan, P. W., Thomason, R. H., & Moore, J. D. (in press). The agreement process: An empirical investigation of human-human computer-mediated dialogues. *International Journal of Human-Computer Studies*.
- Forbus, K. D., Everett, J. O., Ureel, L., Brokowski, M., Baher, J., & Kuehne, S. E. (1998). Distributed coaching for an intelligent learning environment, *proceedings of QR98*. Cape Cod.
- Freedman, R. (1999). Atlas: A plan manager for mixed-initiative, multimodal dialogue, *AAAI-99 Workshop on Mixed-Initiative Intelligence*. Los Altos, CA: AAAI.
- Freedman, R., & Evens, M. W. (1996). Generating and revising hierarchical multi-turn text plans in an ITS. In C. Frasson, G. Gauthier, & A. Lesgold (Eds.), *Intelligent Tutoring Systems: Proceedings of the 1996 Conference* (pp. 632-640). Berlin: Springer.
- Gertner, A., Conati, C., & VanLehn, K. (1998). Procedural help in Andes: Generating hints using a Bayesian network student model., *Proceedings of the 15th national Conference on Artificial Intelligence*.
- Gertner, A. S. (1998). Providing feedback to equation entries in an intelligent tutoring system for Physics. In B. P. Goettl, H. M. Half, C. L. Redfield, & V. J. Shute (Eds.), *Intelligent Tutoring Systems: 4th International Conference* (pp. 254-263). New York: Springer.
- Gertner, A. S.; and VanLehn, K. 2000. Andes: A Coached Problem Solving Environment for Physics. In *Intelligent Tutoring Systems: 5th international Conference, ITS 2000*, eds. G. Gauthier, C. Frasson, & K. VanLehn (Eds.), 133-142. New York: Springer.
- Graesser, A.C.; Person, N.K.; & Magliano, J.P. 1995. Collaborative Dialog Patterns in Naturalistic One-on-one Tutoring. *Applied Cognitive Psychology*, 9: 359-387.

- Graesser, A.C.; Wiemer-Hastings, K.; Wiemer-Hastings, P.; Kreuz, R.; and the TRG 1999. AutoTutor: A Simulation of a Human Tutor. *Journal of Cognitive Systems Research*, 1: 35-51.
- Graesser, A.C.; Wiemer-Hastings, P.; Wiemer-Hastings, K.; Harter, D.; Person, N.; and the TRG 2000. Using Latent Semantic Analysis to Evaluate the Contributions of Students in AutoTutor. *Interactive Learning Environments*, 8: 129-148.
- Hake, R. R. (under review). Interactive-engagement vs. traditional methods: A six-thousand student survey of mechanics test data for introductory physics courses. *American Journal of Physics*.
- Halloun, I. A., & Hestenes, D. (1985). Common sense concepts about motion. *American Journal of Physics*, 53(11), 1056-1065.
- Hestenes, D., Wells, M., & Swackhamer, G. (1992). Force concept inventory. *The Physics Teacher*, 30, 141-158.
- Johnson, W. L.; Rickel, J. W.; and Lester, J.C. 2000. Animated Pedagogical Agents: Face-to-face Interaction in Interactive Learning Environments. *International Journal of Artificial Intelligence in Education*.
- Jurafsky, D.; and Martin, J.E. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Upper Saddle River, NJ: Prentice Hall.
- Koedinger, K.R.; Anderson, J.R.; Hadley, W.H.; and Mark, M.A. 1997. Intelligent Tutoring Goes to School in the Big City. *Journal of Artificial Intelligence in Education* 8: 30-43.
- Kulik, J. A., & Kulik, C.-L. C. (1988). Timing of feedback and verbal learning. *Review of Educational Research*, 58, 79-97.
- Landauer, T.K.; Foltz, P.W.; and Laham, D. 1998. An Introduction to Latent Semantic Analysis. *Discourse Processes*, 25: 259-284.
- Lesgold, A.; Lajoie, S.; Bunzo, M.; & Eggan, G. 1992. SHERLOCK: A Coached Practice Environment for an Electronics Troubleshooting Job. In *Computer-assisted Instruction and Intelligent Tutoring Systems*, eds. J. H. Larkin and R. W. Chabay., 201-238. Hillsdale, NJ: Erlbaum.
- Lester, J.C.; Voerman, J.L.; Townes, S.G.; and Callaway, C.B. 1999. Deictic Believability: Coordinating Gesture, Locomotion, and Speech in Life-like Pedagogical Agents. *Applied Artificial Intelligence*, 13: 383-414.

- McKendree, J. (1990). Effective feedback content for tutoring complex skills. *Human-Computer Interaction*, 5, 381-413.
- McKendree, J., Radlinski, B., & Atwood, M. E. (1992). The Grace Tutor: A qualified success. In C. Frasson, G. Gauthier, & G. I. McCalla (Eds.), *Intelligent Tutoring Systems: Second International Conference* (pp. 677-684). Berlin: Springer-Verlag.
- Person, N.K.; Graesser, A.C.; Kreuz, R.J.; Pomeroy, V.; and the TRG (forcoming). Simulating Human Tutor Dialog Moves in AutoTutor. *International Journal of Artificial Intelligence in Education*.
- Reiser, B. J., Copen, W. A., Ranney, M., Hamid, A., & Kimberg, D. Y. (in press). Cognitive and motivational consequences of tutoring and discovery learning. *Cognition and Instruction*.
- Rickel, J.; and Johnson, W.L. 1999. Animated Agents for Procedural Training in Virtual Reality: Perception, Cognition, and Motor Control. *Applied Artificial Intelligence*, 13: 343-382.
- Rose, C.P. 2000. A Framework for Robust Semantic Interpretation. *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics*.
- Rose, C.P., DiEugenio, B., and Moore, J. 1999. A Dialogue Based Tutoring System for Basic Electricity and Electronics. In *Proceedings of AI in Ed*.
- Rose, C. P., & Lavie, A. (in press). Balancing robustness and efficiency in unification-augmented context-free parsers for large practical applications. In J. C. Junqua & G. V. Noord (Eds.), *Robustness in Language and Speech Technology*: Kluwer Academic press.
- Schulze, K. G., Correll, D., Shelby, R. N., Wintersgill, M. C., & Gertner, A. (1998). A CLIPS problem solver for Newtonian physics force problems. In C. Giarratano & G. Riley (Eds.), *Expert Systems Principles and Programming*. Boston, MA: PWS Publishing Company.
- Schulze, K. G., Shelby, R. H., Treacy, D. J., & Wintersgill, M. C. (2000). Andes: A coached learning environment for classical Newtonian physics, *Proceedings of the 11th International conference on College Teaching and Learning*. Jacksonville, FL.
- Shelby, R. N., Schulze, K. G., Treacy, D. J., Wintersgill, M. C., Gertner, A. G., & Vanlehn, K. (in prep.). The Andes Intelligent Tutor: an Evaluation.
- Stevens, A., & Collins, A. (1977). The goal structure of a Socratic tutor, *Proceedings of the National ACM Conference*. New York: ACM.

VanLehn, K. 1996. Conceptual and Meta Learning during Coached Problem Solving. In ITS96: Proceeding of the Third International conference on Intelligent Tutoring Systems, eds. C. Frasson, G. Gauthier, and A. Lesgold. New York: Springer-Verlag.

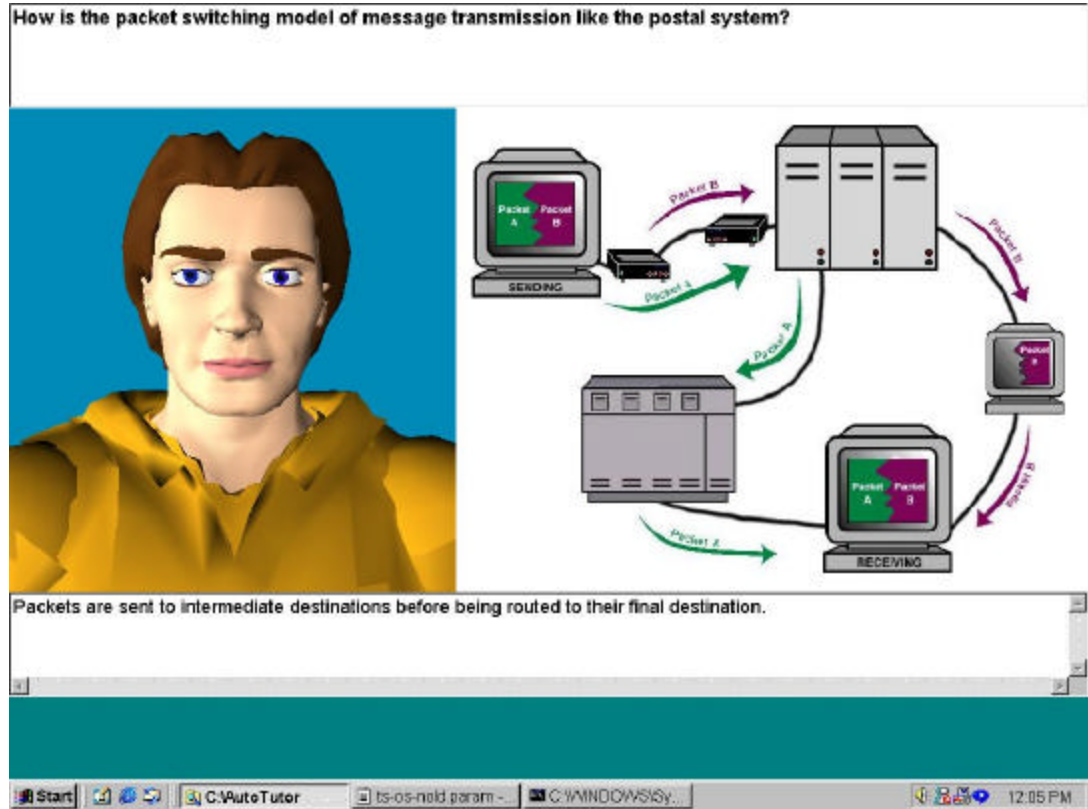


Figure 1: AutoTutor with Marco Talking Head

TUTOR-1: Alright, let's go on. You want to take photos and send them over the Internet. Consider this problem. What hardware do you need to take photos and send them over the Internet?

STUDENT-1: First of all you need to have access to the internet

TUTOR-2: What else?

STUDENT-2: a camera

TUTOR-3: Yeah. For what type of connection do you need a network card?

STUDENT-3: network card

TUTOR-4: Okay. Now, if you have access to the Internet through a network card, then your connection is ...

STUDENT-4: I don't know.

TUTOR-5: Is direct. Okay. A network card is needed if you have a direct connection to the Internet. Okay. How does the user get hooked up to the internet?

STUDENT-5: by a plug.

TUTOR-6: Once again, a network card is needed if you have a direct connection to the Internet. Okay, what does a modem do?

STUDENT-6: connects network card and internet

TUTOR-7: Okay. Try this. A modem is required when there is ...

STUDENT-7: a connection

TUTOR-8: a dial up connection.

.

.

.

TUTOR-30: Let's review. To send your photos on the Internet, you need either a digital camera or a regular camera to take the photos. If you use a regular camera, you need a scanner to scan them onto a computer disk. If you have a direct connection to the Internet, then you need a network card. A modem is needed if you have a dial up connection.

Figure 2 : An Example Tutorial Dialogue Between AutoTutor and a Learner.

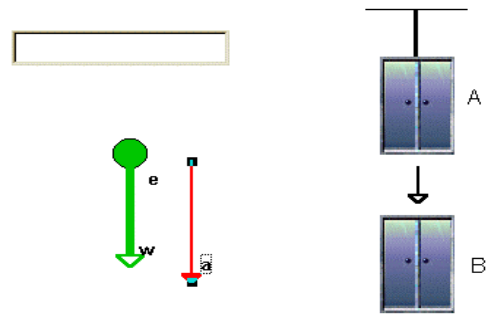
ANDES Physics Workbench - [Ex3-1.fbd]

File Edit Diagram Variable View Help

A 1600Kg elevator moving downward begins slowing down at point A until it comes to a rest at point B.

If the velocity of the elevator at point A is 12m/s and the distance from point A to point B is 42m, Find the constant tension in the supporting cable.

Answer:



Think about the direction of A.

Explain Further How do I do that? Why? Hide

270 degrees NUM 23:43:10

Name	Definition
T0	time when at point A
T1	time when at point B
me	mass of elevator
w	magnitude of the Weight Force on elevator at tir
a	magnitude of the average Acceleration of eleva

Enter scalar equations here: Calculate

- $F = me * a$
- $w = me * g$
-
-
-
-
-
-

Figure 3: The Andes tutoring system

1. Tutor: As you know, you can think of an acceleration vector as changing a velocity vector by pulling on its tip. In this case, the elevator's velocity vector points straight down and you drew its acceleration vector pointing down too. What does your acceleration vector do to the velocity vector?
2. Student: It stretches it downward.
3. Tutor: Does that make it longer?
4. Student: Yes.
5. Tutor: Which means the elevator is speeding up, right?
6. Student: Yeah, but its not really speeding up. Its slowing down.
7. Tutor: Right. So maybe your acceleration vector's direction is wrong. What direction should it be?
8. Student: Up.
9. Tutor: Right. So what have you learned about the direction of acceleration when an object is moving in a straight line and slowing down?
10. Student: It's in the opposite direction from the velocity.
11. Tutor: Yes! That's exactly right. Good job.

Figure 4: A hypothetical dialog between Atlas and a student.

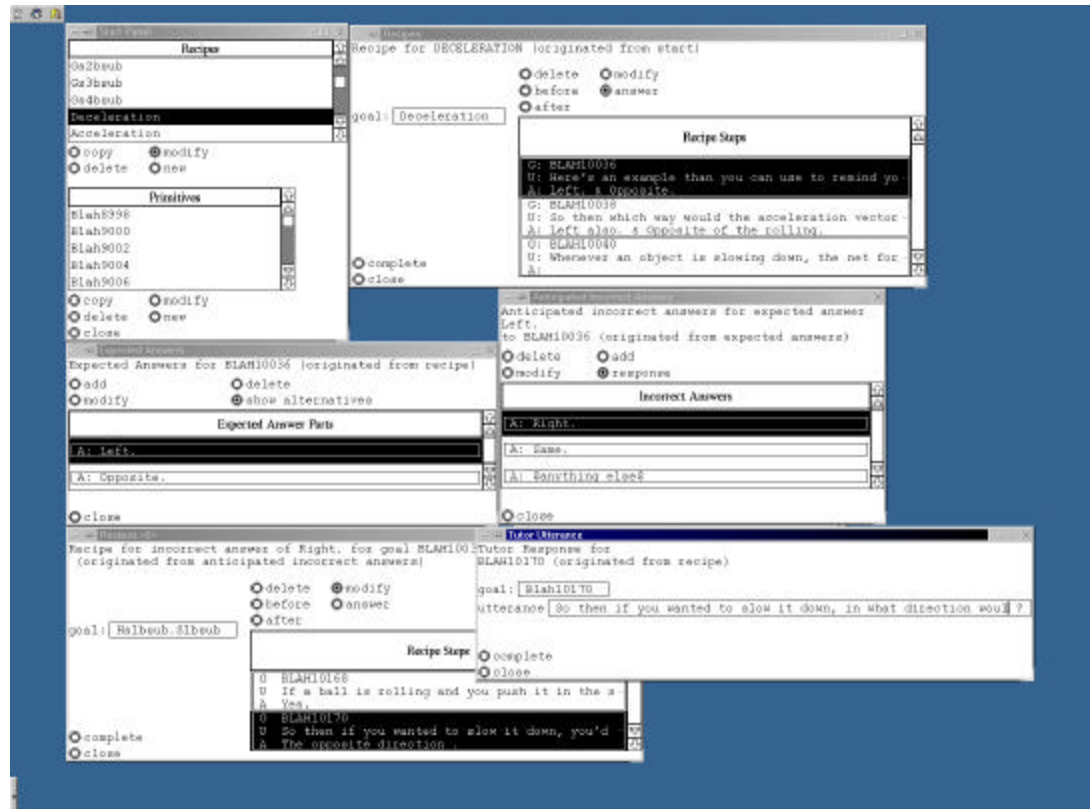


Figure 5: The KCD editor

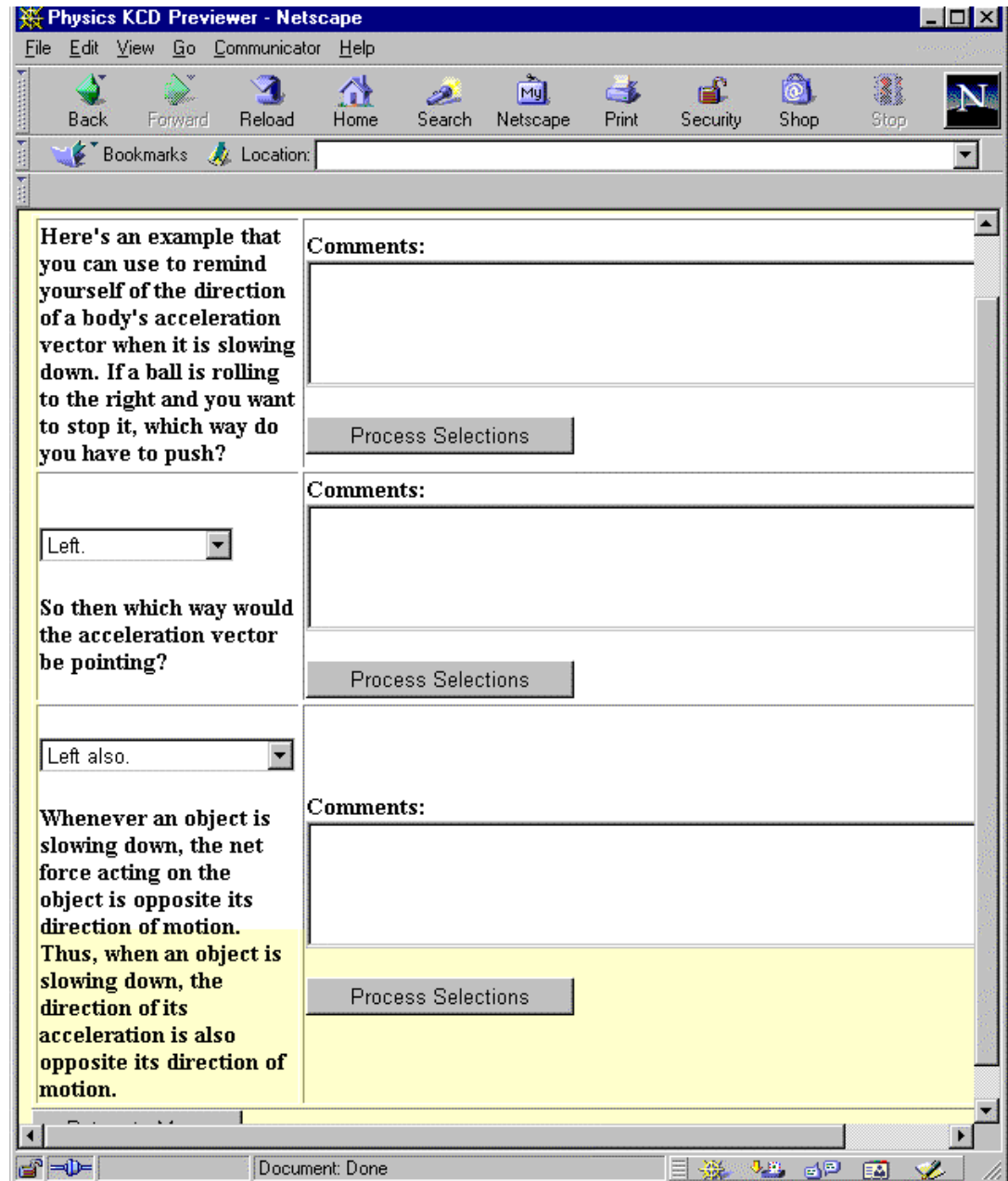
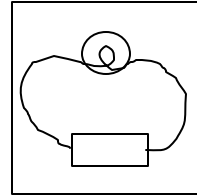


Figure 6: The KCD reviewer

Question: A battery is connected to a bulb by two wires. The bulb lights. Why?



1. Student: There are two currents which come out, one current positive and one negative. They combine in the bulb to produce light and heat.
2. Tutor: I see. Which current is composed of moving electrons?
3. Student: I'm not sure. Both?
4. Tutor: Right, both are. An electrical current is just moving electrons. Which direction do the electrons move in your two currents?
5. Student: It think the electrons move out from the battery to the bulb. The two currents meet there and produce energy.
6. Tutor: Are the electrons actually destroyed in the bulb?
7. Student: I'm not sure.
8. Tutor: Actually, electrons are not destroyed or lost in an electrical current. Instead, they go out one side of the battery, go through the bulb and go back in the other side of the battery. Thus, your two currents are actually the same flow, only the electrons are leaving the battery for one and returning to the battery for the other. Given everything you know about batteries and bulbs, does this make sense?
9. Student: But I thought the battery dies when it runs out of electrons.
10. Tutor: Batteries die when they run out of the chemical energy that they use to pump electrons around the circuit. The battery is like a water pump that produces a powerful current, and the bulb is like a paddle wheel that is driven by the water current. When the water pump runs out of energy, the current slows to a stop. As this happens, does the number of electrons in the battery change?
11. Student: Maybe a little. But it stays about the same.
12. Tutor: Exactly right! I think you understand electrical current better now. See if you can write a better explanation.

Figure 7: A hypothetical dialog between a student and Why2.