AI-ED 2001

May 20, 2001

WORKSHOP PAPERS

Tutorial Dialogue Systems

AIED-2001 Workshop on Tutorial Dialogue Systems

Sunday, May 20, 2001

Organizing committee

Vincent Aleven
Human Computer-Interaction Institute
Carnegie Mellon University
aleven@cs.cmu.edu

Mark Core
Human Communication Research Centre
University of Edinburgh
markc@cogsci.ed.ac.uk

Jérôme Lehuen
Equipe Langue et Dialogue
Laboratoire d'Informatique
Université du Maine
Jerome.Lehuen@lium.univ-lemans.fr

Rachel Pilkington
Computer Based Learning Unit,
The University of Leeds,
R.M.Pilkington@cbl.leeds.ac.uk

Carolyn Penstein Rose
Learning Research and Development Center
University of Pittsburgh
rosecp+@pitt.edu

Florence M. Reeder
George Mason University /
The MITRE Corporation
freeder@mitre.org

Jeff Rickel
USC Information Sciences Institute
rickel@ISI.EDU

Peter Wiemer-Hastings
Human Communications Research Centre
University of Edinburgh
peterwh@cogsci.ed.ac.uk

Beverly Park Woolf
Department of Computer Science
University of Massachusetts
bev@cs.umass.edu

Table of Contents

Introduction

Human one-on-one tutoring is the most effective form of instruction. Although the best intelligent tutoring systems have been shown to be more effective than classroom instruction, they are only half as effective as human tutors. Much of the success of human tutors seems to hinge on their ability to engage students in dialog. It is therefore an interesting and promising hypothesis that intelligent tutoring systems will be more effective if they engage students in dialog or support effective dialog between learners. This raises a number of broad research questions:

- what is good tutorial dialog?
- why is it effective?
- what kind of architectures can support good tutorial dialog?

The workshop will deal with all issues related to these broad questions, including (but not limited to) empirical studies of tutorial discourse, the use of natural language understanding and generation technologies, the representation of pedagogical strategies and knowledge, the use of dialog and text planning, and studies of the effectiveness of tutorial dialog systems.

Although the field of AI & Education has a long-standing interest in these questions, they are more in the foreground now than before, due to advances in technologies such as natural language processing, knowledge representation, virtual reality, and multi-modal interfaces. The recent AAAI Fall Symposium on the topic of tutorial dialog systems reflects the surge of interest in both the AI & Education and computational linguistics communities.

Special Focus: Understanding The Trade-Offs Between Architectural Complexity And Pedagogical Effectiveness

The workshop will focus on understanding the trade-offs between the complexity of a tutorial dialog system and its pedagogical effectiveness. Tutorial dialog systems tend to be complex. They contain not just the components found traditionally in intelligent tutoring systems, but many other components as well, such as a parser, semantic analyzer, dialog planner, text planner, natural language realization component, and a virtual reality module. In the face of this complexity, it is good to ask, where is the biggest bang for the buck? What level of architectural complexity gives the greatest pedagogical pay-off? Is adding complexity always a good thing? What minimum level of complexity is required? Complexity can be measured, among other ways, in terms of development effort or the elaborateness of the architecture. Effectiveness on the other hand can be measured as the range of dialog phenomena that the system supports, the generality of the approach, the ease of maintenance, or, ultimately, the students' learning gains.

It may well be that the trade-offs differ depending on the application domain, the overall pedagogical approach of the system, and the purpose for which it uses natural language processing. Nonetheless, it is likely that some common trade-offs, and ways of dealing with them, can be found that hold across domains.

Using a Model of Collaborative Dialogue to Teach Procedural Tasks

Jeff Rickel,[1] Neal Lesh,[2] Charles Rich,[2] Candace L. Sidner[2] and Abigail Gertner[3]
[1] USC Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, CA, 90292
rickel@isi.edu, http://www.isi.edu/isd/rickel
[2] Mitsubishi Electric Research Laboratories, 201 Broadway, Cambridge, MA, 02139
lesh,rich,sidner@merl.com, http://www.merl.com/projects/collagen
[3] MITRE Corporation, 202 Burlington Road, Bedford, MA, 01730
gertner@mitre.org, http://www.mitre.org/resources/centers/it/g068

Abstract

Previous research on building intelligent tutoring systems has not leveraged general models of collaborative discourse, even though tutoring is an inherently collaborative and often discourse-based activity. Similarly, previous research on collaborative discourse theory has rarely addressed tutorial issues, even though teaching and learning are crucial components of collaboration. We help bridge the gap between these two related research threads by presenting a tutorial agent, called Paco, that we built using an application-independent collaboration manager, called Collagen. Our primary contribution is to show how a variety of tutorial behaviors can be expressed as rules for generating candidate discourse acts in the framework of collaborative discourse theory.

1 Introduction

Our research objective is to develop computer tutors that collaborate with students on tasks in simulated environments. Towards this end, we seek to integrate two separate but related research threads: intelligent tutoring systems (ITS) and collaborative dialogue systems (CDS). Research on ITS (e.g., [1, 21, 23]) focuses on computer tutors that adapt to individual students based on the target knowledge the student is expected to learn and the presumed state of the student's current knowledge. Research on CDS (e.g., [8, 13, 22]), with an equally long history, focuses on computational models of human dialogue for collaborative tasks.

Unfortunately, there has been a surprising lack of cross-fertilization between these two research areas. Work on tutorial dialogue for intelligent tutoring systems (e.g., [3, 14, 24]) has not leveraged general models of collaborative dialogue. Similarly, research on collaborative dialogues has focused on modeling conversations between peers or between an expert and novice, but has rarely addressed tutorial issues.

To help integrate ITS and CDS, we developed a tutorial agent in Collagen [17], a middleware system based on a long line of research on collaborative discourse [8, 6, 7, 5, 13]. Collagen maintains a model of the discourse state shared by the user (e.g., student) and the computer agent (e.g., tutor). The discourse state includes information about the current focus of attention and the collaborators' mutually believed plans. Agents constructed using Collagen use the discourse state to generate an agenda of candidate *discourse acts,* including both "physical" actions and utterances, and then choose one to perform or utter.

Our tutorial agent, Paco (Pedagogical Agent for Collagen), teaches students procedural tasks in simulated environments, building on ideas from earlier tutoring systems [18, 19]. While Paco can engage in slightly more sophisticated conversations than previous such tutors, our primary contribution is to show how a variety of tutorial behaviors can be expressed as rules for generating candidate discourse acts in Collagen. Translating behaviors developed in ITS into the framework of CDS is a first step towards building tutoring agents that can leverage advances in collaborative discourse theory. Also, since Paco is domain-independent, its tutorial actions can be added to the set of candidate discourse acts of any agent built with Collagen, allowing such agents to tutor in addition to their normal role as assistants. Finally, a third goal of this work is to report on Collagen's value for building tutorial agents, both in terms of the theory it reflects and the software architecture it supports.

2 Pedagogical Approach

We designed Paco to support simulation-based training, in which students learn tasks by performing them in a simulation of the real work environment. (Of course, if the target work environment is actually a software application, that application can serve as the simulator.) The computer tutor's instruction and assistance are situated in the performance of domain tasks in the simulated world. That is, the tutor chooses a scenario (task to perform starting from a particular simulation state), works through it with the student, and then repeats until all scenarios have been mastered.

Our pedagogical approach is based on the apprenticeship model of learning [2], which requires two capabilities. First, the tutor must be able to perform and explain the task. Second, it must be able to monitor the student as she performs the task, providing assistance when needed as well as critique or positive feedback when appropriate. As the student gains proficiency, the assistance provided should decrease. Ideally, students should learn to flexibly apply well-defined procedures in a variety of situations.

Figure 1 shows an example dialogue with our current implementation of Paco that illustrates some of the key features we support. Paco is teaching the student how to operate the gas turbine engines that propel naval ships. Paco has previously worked through a simple scenario in which the student engaged one of the turbine engines. Now, Paco is going to teach the same procedure under slightly more complicated conditions: (1) a high vibration alarm has occurred on the gas turbine generator, shutting the generator down, so the student will have to reset the alarm before starting the generator; and (2) a second engine is already running, so the student will have to stop it before starting up the desired engine. The remainder of the paper will use this example dialogue to illustrate aspects of our design.

If there were no overlap among tasks and scenarios, Paco could be implemented in an obvious way: the tutor would first demonstrate the entire task, then repeatedly let the student practice the task, providing assistance where necessary. However, different tasks often share common subtasks or actions, and different scenarios often require variants of the same task. Therefore, at any moment, a student's level of mastery may differ across the different parts of a task. For example, a new scenario may require branches of a task that the student has not yet seen (e.g., lines 8-12 and 18-35 in the example dialogue) while also requiring steps and subtasks that have been mastered already.

To address this issue, Paco uses a student model to dynamically interleave demonstration and coached practice, using the approach introduced by Rickel [18]. As the student and Paco progress through a task, Collagen will repeatedly identify the set of valid next steps in the plan to solve the current task. Paco consults the student model to see whether the student has sufficient knowledge to choose the next step. If so, it will expect the student to take the next step, and will provide assistance only if the student requests it or makes a mistake. If not, Paco will intervene and teach the student what to do next (e.g., lines 8-12 and 18-35). Thus, as Paco and the student work through tasks, initiative will pass back and forth between them based on the student's prior experience. Whenever Paco decides that the initiative should shift, it will let the student know through verbal comments (e.g., "You take it from here").

Paco represents the procedures it will teach using Collagen's declarative language for domain-specific procedural knowledge. This knowledge serves as a model of how domain tasks should be performed. Each task is associated with one or more *recipes* (i.e., procedures for performing the task). Each recipe consists of several elements drawn from a relatively standard plan representation. First, it includes a set of steps, each of which is either a primitive action (e.g., press a button) or a composite action (i.e., a subtask). Composite actions give tasks a hierarchical structure. Second, there may be ordering constraints among the steps; these constraints define a partial order over the steps. Third, a task and its steps can have parameters, and a recipe can specify constraints (bindings) among the parameters of a task and its steps. Finally, steps can have preconditions (to allow Collagen to determine whether a step can be performed in the current state) and postconditions (to determine whether the effects of a step have been achieved).

3 Collagen as a Foundation for Teaching Procedural Tasks

Collagen's main value for building tutoring systems is that it provides a general model of collaborative dialogue based on well-established principles from computational linguistics. The model includes two main parts: (1) a representation of discourse state and (2) a discourse interpretation algorithm that uses plan recognition to update the discourse state given the actions and utterances of the user and agent. Previous tutoring systems for procedural tasks do not include dialogue managers with the same level of generality.

Based on the work of Grosz and Sidner [6], Collagen partitions the discourse state into three interrelated components: the linguistic structure, the attentional state, and the intentional structure. The linguistic structure, implemented as a *segmented interaction history* in Collagen, groups the dialogue history into a hierarchy of discourse segments. Each segment is a contiguous sequence of actions and utterances that contribute to some *purpose* (e.g., performing a task or subtask). For example, Figure 2 shows the segmented interaction history for a portion of the example dialogue.

The attentional state, i.e., what the user and agent are talking about and/or working on *now*, is represented by a stack of discourse purposes called the focus stack [6]. When a new discourse segment is begun, its purpose is pushed onto the stack. When a discourse segment is completed or discontinued, its purpose is popped off the stack. The stack mechanism is crucial for the proper treatment of interruptions. Additionally, the attentional state maintained by Collagen includes an extension to the original model of Grosz and Sidner to capture which participant holds the conversational initiative. This allows Paco to decide when to explicitly pass the initiative to the student (e.g., "You take it from here.").

An explicit representation of attentional state is important in tutoring systems so that the tutor does not confuse the student with unexpected shifts in focus. For example, if the student begins work on a subtask and then asks for help, the tutor's advice should relate to that subtask, even if other actions outside that subtask could be done next. For tutoring systems that follow a strict sequence through domain tasks, a focus stack may not be needed. However, to support mixed-initiative dialogues where the student and tutor have more freedom to choose the execution order for tasks and initiate topics for discussion, a focus stack helps ensure the coherence of the tutor's responses.

(1) Paco sets the state of the simulator.

(2) Paco "I've reset the simulation so that we can try engaging an engine under slightly different conditions."

(3) Paco "Let's engage engine one."

(4) Student "Ok."

(5) Paco "You take it from here."

(6) Student "Let's start the generator."

(7) Paco "Good."

(8) Paco (Pointing) "Now, notice that the alarm light is on."

(9) Paco "Before starting a generator, always make sure that the alarm light is off so that you'll know when a new alarm appears."

(10) Paco (Pointing) "Press the alarm reset button on the generator in order to extinguish the alarm."

(11) Student presses the alarm reset button on the generator.

(12) Paco "Good."

(13) Student "Ok."

(14) Paco "You take it from here."

(15) Student presses the on button on the generator.

(16) Paco "Great."

(17) Paco "We succeeded in starting the generator."

(18) Paco (Pointing) "Now, notice that engine two is running."

(19) Paco "Before starting an engine, always make sure that the other engine is stopped."

(20) Paco "Let's stop engine two."

(21) Paco (Pointing) "First, press the disengage button on engine two in order to disengage the clutch."

(22) Student presses the disengage button on engine two.

(23) Paco "Right."

(24) Paco "The next step of stopping engine two is to close the fuel valves on engine two."

(25) Student "Ok."

(26) Paco "You take it from here."

(27) Student closes fuel valve one on engine two.

(28) Paco "Good."

(29) Student closes fuel valve two on engine two.

(30) Paco "Good job."

(31) Paco (Pointing) "The next step is to press the off button on engine two."

(32) Student presses the off button on engine two.

(33) Paco "Great."

(34) Paco (Pointing) "The indicator light tells us that engine two is stopped."

(35) Paco "So, we succeeded in stopping engine two."

(36) Student "Ok."

(37) Paco "You take it from here."

(38) Student opens fuel valve one on engine one.

(39) Paco "Good."

(40) Student opens the bleed valve on engine one.

(41) Paco "That was a correct action, but I wasn't expecting you to do it now."

(42) Paco "Let's go back to opening the fuel valves on engine one."

(43) Student opens fuel valve two on engine one.

(44) Paco "Great."

(45) Student presses the engage button on engine one.

(46) Paco "Whoops, you can't do that yet."

(47) Student "What next?"

(48) Paco (Pointing) "Press the on button on engine one."

(49) Student presses the on button on engine one.

(50) Paco "Good."

(51) Student "I think I should set the throttle speed."

(52) Paco "Right."

(53) Student "What should the speed be?"

(54) Paco "The speed should be stop."

(55) Student sets the throttle speed to stop.

(56) Paco "Good."

(57) Student presses the engage button on engine one.

(58) Paco "Good job."

(59) Paco "We succeeded in engaging engine one."

Figure 1: An example dialogue in which Paco teaches steps in the current task (e.g., lines 21, 31), teaches that subtasks are finished (e.g., lines 17, 35), provides help when requested (e.g., line 48), passes initiative to the student when she should know what to do next (e.g., lines 5, 14, 26), provides positive feedback (e.g., lines 7, 16, 23), diagnoses and critiques incorrect actions (e.g., line 46), explains why steps needs to be done (e.g., line 9), discusses postconditions for actions (e.g., lines 8, 34), and helps the student stay focused on the current subtask (e.g., line 42).

```
    Engaging engine one
3   Paco says "Let’s engage engine one."
4   Student says "Ok."
5   Paco says "You take it from here."
    Done successfully starting the generator.
6     Student says "Let’s start the generator."
7     Paco says "Good."
      Done user successfully pressing the alarm reset button on the generator.
8       Paco says "Now, notice that the alarm light is on."
9       Paco says "Before starting a generator, always make
                   sure that the alarm light is off so that
                   you’ll know when a new alarm appears."
10      Paco says "Press the alarm reset button on the
                   generator in order to extinguish the alarm."
11      Student presses the alarm reset button on the generator.
12      Paco says "Good."
13      Student says "Ok."
14      Paco says "You take it from here."
15    Student presses the on button on the generator.
16    Paco says "Great."
17    Paco says "We succeeded in starting the generator."
```

Figure 2: The segmented interaction history for a portion of the dialogue in Figure 1.

While the linguistic structure and attentional state closely reflect the actual temporal order of actions and utterances in the dialogue, the intentional structure represents the decisions that have been made as a result of those actions and utterances, independent of their order. Collagen represents the intentional structure as *plan trees*, which are a partial implementation of SharedPlans [7, 5]. Nodes in the tree represent mutually agreed upon intentions (e.g., to perform a task), and the tree structure represents the subgoal relationships among these intentions. Plan trees also record other types of decisions, such as whether a recipe has been chosen for a task, whether any of its parameters have been determined, and who is responsible for performing the task (e.g., student, agent, or both).

The heart of Collagen is the discourse interpretation algorithm, which specifies how to update the discourse state given a new action or utterance by either the user or agent. Its objective is to determine how the current act contributes to the collaboration. For example, the act could contribute to the current discourse segment’s purpose (DSP) by directly achieving it (e.g., pressing a button when that action is the current DSP), proposing how it can be achieved (i.e., suggesting a recipe), proposing or performing a step in its recipe, or proposing a value for one of its unspecified parameters. Collagen extends Lochbaum’s discourse interpretation algorithm [13] with plan recognition, which can recognize when an act contributes to a DSP through one or more implicit acts [12].

Collagen’s discourse interpretation algorithm proceeds as follows. If the current act contributes to the current DSP, it is added to the segment and the plan tree is updated accordingly. If not, Collagen searches up through the plan tree to see if an act contributes to any other action in the plan; if so, and if the act is a valid next step, it represents a shift in focus. Collagen pops all purposes off the stack that are not parents of the matched step, then pushes any necessary purposes on until the act is in focus. Finally, if nothing in the plan tree matches the current act, it is treated as an interruption and pushed onto the stack without popping anything.

Collagen has recently been extended to perform “near-miss” plan recognition if it cannot find a correct interpretation of an act. It systematically searches for extensions to the plan tree that would explain the current act if some constraint were relaxed. For example, it can recognize acts that would violate an ordering constraint, unnecessarily repeat a step that was already performed, or perform a step that should be skipped because its effects are already satisfied. Thus, near-miss

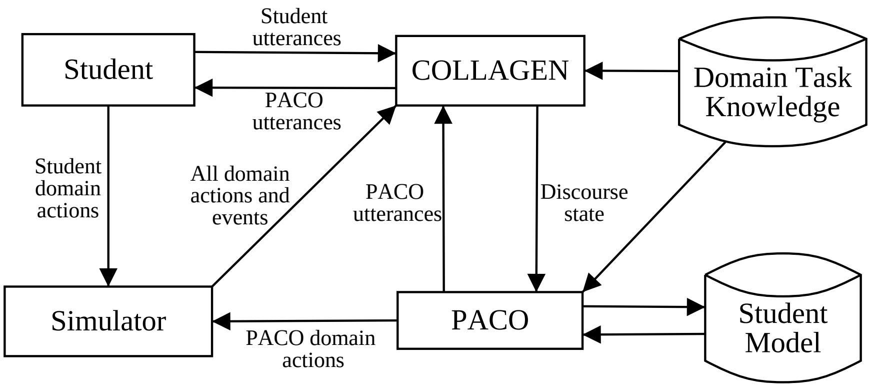


Figure 3: Paco's Architecture

plan recognition attempts to find plausible interpretations of student errors, providing a domain-independent capability for student diagnosis. It is also extensible, allowing a domain author to define new types of errors or even add explicit buggy recipes. Additionally, Collagen is being extended to use causal information in recipes to repair plans after an incorrect action, or external event, occurs.

3.1 Architecture

Figure 3 shows how Paco fits into the general Collagen architecture. The three software components in this architecture are the simulator, Collagen, and the agent (e.g., Paco). Collagen makes very few assumptions about the simulator. Primarily, it assumes that the user (e.g., student) and agent (e.g., Paco) can both perform domain actions (e.g., open a fuel valve) and can observe the actions taken by each other. Collagen provides an API for such event messages, so that it will be able to interpret them. Collagen makes no assumptions about the simulator's user interface. The simulator can, however, optionally specify a screen location for domain actions, which allows the agent to use a pointing hand to draw the user's attention to an object or indicate that the agent is performing an action.

Collagen represents utterances using an artificial discourse language derived from earlier work by Sidner [20]. The language is intended to include the types of utterances that people use when collaborating on tasks. Currently, Collagen's language includes utterance types for agreeing ("yes" and "OK") and disagreeing ("no"), proposing a task or action (e.g., "Let's engage engine one"), indicating when a task has been accomplished (e.g., "We succeeded in stopping engine two"), abandoning a task, asking about or proposing the value of a parameter to a task or action, asking or proposing how a task should be accomplished, and asking what should be done next ("What next?"). Current work is extending Collagen's language to include additional elements from Sidner's language, especially to support negotiation about task decisions.

To bypass natural language understanding issues, Collagen provides a window to allow the user to construct utterances and to display the agent's utterances. In both windows, it converts its internal discourse language into English (or other language) strings, using a combination of domain-independent and (optional) domain-specific text templates. In the user window, users construct utterances by selecting from a menu of utterances and utterance types, and they can modify any utterance by selecting any phrase within it (representing a field in the original text template) and choosing a replacement phrase. Optionally, Collagen can also use speech recognition software to allow the user to speak these utterances rather than creating them through the GUI, and it can use speech synthesis software to allow the agent to speak its utterances.

4 Tutorial Behaviors as Collaborative Discourse Acts

Table 1 is a summary of our progress in integrating ITS and CDS: it lays out in detail how Paco's tutorial behaviors are generated from Collagen's discourse state representation and Paco's student model. The first column of the table is a ranked list of the tutorial act types. The second column describes procedures that generate zero or more instances of each act type from the current discourse state and student model. When it is Paco's turn, it constructs a prioritized agenda by evaluating the procedures for each act type and then selects the highest ranked act in this agenda.[1] The third column of the table shows the semantics of each act type in Sidner's [20] artificial discourse language, which determines how the act will be interpreted by Collagen's discourse interpretation algorithm. Several of the act types have subcases, shown in the fourth column, which share the same basic semantics, but differ in how they are rendered into English (fifth column).

Paco uses several elements of the discourse state to generate its discourse acts including the focus of attention, the initiative, and plan trees. The focus of attention is used, for example, to avoid teaching a step unless its purpose is in focus. The focus stack also indicates when the student has interrupted the current task, which causes Paco to generate a discourse act which would end the current interruption. In addition to the shared focus maintained by Collagen, Paco also maintains a *private focus* because it prefers to finish teaching an action before moving on. If the student starts working on another part of the plan (thus popping the current focus from the shared focus stack) while there are still legal steps within Paco's private focus (e.g., line 40 in Figure 1), then Paco will add a Correct Focus action (e.g., line 42) to the agenda. Paco might choose to execute a higher-ranked element on the agenda first (e.g., line 41) but will re-generate the Correct Focus action unless the student returns to the previous subtask by herself.

The various conditions for generating discourse acts are easy to compute given the data structures maintained by Collagen. For example, several of the acts operate on the *valid next actions*, which refers to the plan steps that can be executed next based on precondition and ordering constraints.[2] Collagen computes this information during discourse interpretation. Additionally, Collagen's near-miss recognition computes the conditions needed to generate the various subcases of Negative Feedback (e.g., line 46). Finally, when the student asks for help (e.g., line 47) this pushes a discourse purpose of helping the student onto the stack which remains there until the agent provides the help (e.g., line 48).

Using the generic capabilities of Collagen to record information about a user, Paco maintains a simple overlay model [4] that records, for each step in a recipe, whether the student has been exposed to it. In Table 1, the condition "the student knows step ω" means that the student has been taught this step before. The condition "student knows step ω needs to be done" means the student has been taught all the steps that connect ω to the root of the current plan. Finally, Paco's student model also records which actions the student has been told that she has completed (e.g., line 17). The condition "the student knows when ω is complete" means that the tutor has told the student when ω was complete, at least once before.

We use Collagen's generic representation for recipes to store domain-specific knowledge about why actions need to be performed. That is, Paco's domain knowledge includes recipes that achieve the subgoal of explaining why an action, or more specifically a step of a recipe, should be performed.

[1] An agent that was more of an assistant might also include acts in Collagen's default agenda in its ranking.

[2] Paco also uses information about the preferred order of executing actions to determine which actions to teach.

Tutorial act type	**Add instance to agenda for ...**	**Semantics**	**Subcases (if any)**	**Example gloss**
Positive feedback *(rank 1)*	the user's most recent action α if it was, or proposed, a valid next action and has not yet received feedback	$accept(should(\alpha))$	α finished subtask	Great job.
			α wasn't proposed by tutor	Nice.
			α caused unnecessary focus shift	That was a correct action, but I wasn't expecting you to do it now.
			α finished top-level goal	We're done with this scenario.
			none of above	Good.
Negative feedback *(rank 1)*	the user's most recent action α if it was, or proposed, an invalid next action and has not yet received feedback	$reject(should(\alpha))$	α was already done	Whoops, you already did that.
			α's purpose was already achieved	Whoops, you didn't need to do that.
			α has an unsatisfied precondition	Whoops, you can't do that yet.
			executing α violates an ordering constraint	Whoops, it's too soon to do that.
End interrupt-ion *(rank 2)*	each step ω that is an unstopped interruption on the focus stack	$propose(\neg should(\omega)$	ω has known purpose	Let's stop closing the fuel valves.
			ω has unknown purpose	That is not relevant to our current task.
Teach complete *(rank 3)*	each non-primitive ω in the current plan $s.t.$ ω is complete and the student does not know when ω is complete	$propose(achieved(\omega))$		We succeeded in closing the fuel valves.
Correct Focus *(rank 4)*	step ω if it is the tutor's private focus but not the action on top of the focus stack	$propose(should(\omega))$		Let's return to opening the fuel valves.
Give initiative *(rank 5)*	any valid next plan step ω that the student knows needs to be done, if the tutor has initiative and the student has not requested help	$propose($ $initiative = user)$	tutor has just proposed ω	Go ahead.
			tutor has not just proposed ω	You take it from here.
Explain Why *(rank 6)*	every plan step ω that is teachable (see Teach Step) and is currently unexplained and has an explanation recipe	*first step of explanation recipe*		Before starting an engine, always make sure that the other engine is stopped.
Teach step *(rank 7)*	every valid next plan step ω that the student does not know and whose parent is in focus	$propose(should(\omega))$	ω is primitive	Now, you should press the on button.
			ω is non-primitive	The next step of engaging the engine is to open the fuel valves.
Remind step *(rank 8)*	every valid next plan step ω that the student knows and whose parent is in focus	$propose(should(\omega))$		You need to press the on button.
Propose new scenario *(rank 8)*	purpose ω, if the current plan is complete, where ω is the next task to work on	$propose(should(\omega))$		Let's try another scenario. Let's engage engine one.
Shift Focus *(rank 8)*	every plan step ω that is not currently on top of the focus stack and the student knows has to be done and has a child c that is a valid next plan step and c is not known by the student	$propose(should(\omega))$		Let's open the fuel valves.

Table 1: Tutorial discourse acts

Typically, these recipes are composed of one or more utterances of text written by a domain expert, but in principle, explanation recipes can contain any type of primitive or abstract actions.[3] Whenever Paco generates a candidate discourse act to teach a step, it also checks to see if an explanation recipe exists for that step. If so, and if the step has not already been explained, Paco generates a candidate discourse act of executing the first step of the recipe.

The conditions for generating discourse acts represent necessary, but not sufficient, conditions for Paco to perform the act. An advantage of making explicit all necessary conditions for a discourse act is to make it easier to extend Paco with new discourse acts or extend other agents with the ability to perform Paco's tutorial actions. However, this approach leaves open the question of how to choose which act to perform. Paco chooses which act to perform based on the rankings of the discourse acts, given in the first column of Table 1. For example, Paco prefers to give initiative when the student knows what to do next rather than teach or remind her what to do next. We hypothesize that different rankings or other methods for choosing an act from the agenda will produce different tutoring styles.

5 Discussion

To facilitate comparison between Paco and other tutorial dialogue systems, the following list outlines some of the main dimensions along which such systems can be compared, categorizes Paco along these dimensions, and provides some of the motivations and trade-offs involved in our design choices:

- Our work focuses on the pragmatics of natural language understanding, i.e., the use of a discourse interpretation algorithm and a rich representation of discourse state. Our claim is that tutorial dialogues will be more natural for students if computer tutors follow the principles of human collaborative dialogues, on which much research in computational linguistics has focused. We do not yet have strong evidence to substantiate this claim, but investigating that hypothesis is the primary focus of our work.

- Paco performs relatively sophisticated domain reasoning, based on the application of Collagen's domain-independent algorithms to a domain-specific task model (recipe library). Specificially, Collagen decides which domain actions can be done next based on its recipe library, its knowledge of which actions and utterances have been performed so far, and its knowledge of the current simulation state. Its reasoning does not yet include a full planner, as can be found in Rickel and Johnson's Steve tutor [19], but we recognize that such planning capabilities are important in many domains, and we are currently extending Collagen in that direction. Collagen's advantage over Steve is that it requires less domain knowledge (specifically, it does not require causal links among task steps), but this limits its ability to recover from some student errors (e.g., that would require repeating earlier actions) and to recognize when some steps can be skipped (e.g., because they only establish preconditions for later steps whose postconditions are already satisfied).

- Paco currently uses a simple overlay student model. The student model is crucial for Paco's approach to interleaving demonstration and coached practice. We do not currently use a bug library, which would allow Paco to recognize common errors and provide more specific feedback aimed directly at those errors, but Collagen's near-miss capability is capable of exploiting such knowledge if it is provided.

[3]Collagen's facilities for executing recipes in a collaborative setting can be used to complete the explanation

- Text and gesture generation are currently relatively simple in Paco, but this is only a matter of research focus. Collagen uses text templates for text generation, and it uses a pointing hand to direct the student's attention to elements of the simulator. We believe this approach suffices for simple 2D simulations. However, more sophisticated text generation would certainly improve Paco, and elsewhere we have elaborated on the costs and benefits of more fully embodied pedagogical agents [10]. As for graphics, we assume that the simulator will provide appropriate graphics for the simulated world, and some additional graphics could be useful in helping students understand the inner workings of equipment they are learning to operate [9], but this has not been a focus of our work.

- Paco does not include a conventional dialogue planning module, i.e., an explicit search for a sequence of utterances that will achieve a desired mental state in the student. The agent's utterances are selected (using a simple priority scheme) from the candidate discourse acts that follow naturally from the current discourse state. We are interested in investigating more sophisticated dialogue planning, but we have no strong evidence yet that it will be required for teaching procedural tasks. One intermediate position that we are currently exploring is the use of "tutorial recipes," which can be viewed as cached dialogue plans. Collagen can use such recipes to guide its interaction with students using the same mechanisms by which it uses domain recipes. Also, Collagen's plan trees can be viewed as plan-like structures that encode expectations for future utterances and actions that will complete the current task, including ordering constraints and subgoal relationships among these discourse acts. Thus, while Paco does not plan its dialogue acts in a traditional sense, its plan trees play a similar role.

- Paco does not allow free-form student utterances, so it does not include any parsing or semantic interpretation of sentences. Instead, the student constructs utterances through a GUI. This is mainly because we are focusing on the dialogue manager; we are not making any claims about the utility of natural language understanding for teaching procedural tasks. However, we do believe that a GUI will be adequate for teaching many procedural tasks, although full natural language understanding would certainly be better if it could be achieved.

The use of Collagen as a dialogue manager for a tutorial system, as an alternative to building such a system from scratch, also presents some trade-offs. To connect an application and agent to Collagen, one must make several commitments. First, one must write a software module that maps application events into Collagen discourse acts and vice versa. However, a similar module is required to connect any tutor to an external simulator, and Collagen provides a nice interface for making such connections. Second, Collagen requires a recipe library that encodes domain task knowledge, but, again, something similar will be required for any intelligent tutoring system for procedural tasks. One important commitment is that the domain task knowledge must be expressed in Collagen's recipe library representation, as opposed to having the freedom to express it procedurally (e.g., as production rules) or through a custom declarative language. The biggest disadvantage this poses is that Collagen may not exploit some types of knowledge (e.g., causal links or temporal constraints) that are important in a domain, or its semantics (e.g., the definition and implications of ordering constraints) may not be appropriate for some domains. Similarly, one must map all student and tutor utterances into Collagen's act types, although this may not be a serious limitation since Collagen allows new act types to be added. The benefit of providing a recipe library and mapping to Collagen's act types is that it maintains the discourse state based on principles from collaborative discourse theory, and it includes both normal and near-miss plan recognition.

We are interested in several areas of future work. Paco thus far has been primarily a reimplementation (on a new foundation) of fairly standard ITS behaviors. As the next step, we plan to better leverage Collagen's rich discourse state representation to implement aspects of tutorial dialogue that have not been treated in a fully general way in previous ITS work. We are also interested in broadening the types of tutorial discourse acts we consider to include those used in recent analyses of human tutorial dialogues [11, 15, 16], and we are especially interested in exploring the relationship of "hinting" strategies to collaborative discourse theory. Some of these issues may require integrating information in Paco's student model into Collagen's discourse interpretation algorithm. Finally, we would like to experimentally evaluate Paco's ability to teach procedural tasks.

6 Conclusion

In conclusion, we believe that building Paco has been a demonstration of successful cross-fertilization between research in intelligent tutoring and collaborative dialogue systems in at least three respects. First, we showed how a variety of tutorial behaviors can be expressed as rules for generating candidate discourse acts in the framework of CDS. This allows us to immediately apply many notions from CDS in our tutorial agents.

Second, building Paco has given us the opportunity to evaluate the suitability of a particular piece of CDS technology, namely Collagen, for building ITS systems. Our experience has been that using Collagen as the starting point for implementing Paco was a great improvement over programming tutorial agents "from scratch," as we have done in the past. Also, using Collagen led us to design Paco as a composition of a generator of candidate discourse acts and a set of preferences for selecting from these acts. This approach makes it easier to understand, explain, and share tutorial behaviors.

Third, building a tutorial agent in Collagen has revealed some implicit biases in how Collagen operates. As a result, we are exploring various generalizations and extensions to Collagen to better support the full spectrum of collaboration.

References

[1] J. R. Carbonell. AI in CAI: An artificial-intelligence approach to computer-assisted instruction. *IEEE Transactions on Man-Machine Systems*, 11(4):190–202, 1970.

[2] A. Collins, J. S. Brown, and S. E. Newman. Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics. In L. Resnick, editor, *Knowing, Learning, and Instruction: Essays in Honor of Robert Glaser*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1989.

[3] R. K. Freedman. *Interaction of Discourse Planning, Instructional Planning and Dialogue Management in an Interactive Tutoring System*. PhD thesis, Northwestern University, 1996.

[4] I. P. Goldstein. Overlays: A theory of modelling for computer-aided instruction. Artificial Intelligence Laboratory Memo 495, Massachusetts Institute of Technology, Cambridge, MA, 1977.

[5] B. J. Grosz and S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.

[6] B. J. Grosz and C. L. Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204, 1986.

[7] B. J. Grosz and C. L. Sidner. Plans for discourse. In P. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communication*, chapter 20, pages 417–444. MIT Press, 1990.

[8] B. J. Grosz [Deutsch]. The structure of task oriented dialogs. In *Proceedings of the IEEE Symposium on Speech Recognition*, Pittsburgh, PA, April 1974. Carnegie-Mellon University. Also available as Stanford Research Institute Technical Note 90, Menlo Park, CA.

[9] J. D. Hollan, E. L. Hutchins, and L. Weitzman. Steamer: An interactive inspectable simulation-based training system. *AI Magazine*, 5(2):15–27, 1984.

[10] W. L. Johnson, J. W. Rickel, and J. C. Lester. Animated pedagogical agents: Face-to-face interaction in interactive learning environments. *International Journal of Artificial Intelligence in Education*, 11:47–78, 2000.

[11] S. Katz, G. O'Donnell, and H. Kay. An approach to analyzing the role and structure of reflective dialogue. *International Journal of Artificial Intelligence in Education*, 11:320–343, 2000.

[12] N. Lesh, C. Rich, and C. L. Sidner. Using plan recognition in human-computer collaboration. In *Proceedings of the Seventh International Conference on User Modeling*, pages 23–32, Banff, Canada, 1999.

[13] K. E. Lochbaum. A collaborative planning model of intentional structure. *Computational Linguistics*, 24(4):525–572, 1998.

[14] N. K. Person, A. C. Graesser, R. J. Kreuz, V. Pomeroy, and the Tutoring Research Group. Simulating human tutor dialog moves in autotutor. *International Journal of Artificial Intelligence in Education*, 12, 2001. Forthcoming.

[15] K. Porayska-Pomsta, C. Mellish, and H. Pain. Aspects of speech act categorisation: Towards generating teachers' language. *International Journal of Artificial Intelligence in Education*, 11:254–272, 2000.

[16] A. Ravenscroft and R. M. Pilkington. Investigation by design: Developing dialogue models to support reasoning and conceptual change. *International Journal of Artificial Intelligence in Education*, 11:273–298, 2000.

[17] C. Rich and C. L. Sidner. COLLAGEN: A collaboration manager for software interface agents. *User Modeling and User-Adapted Interaction*, 8(3-4):315–350, 1998.

[18] J. Rickel. An intelligent tutoring framework for task-oriented domains. In *Proceedings of the International Conference on Intelligent Tutoring Systems*, pages 109–115, Montréal, Canada, June 1988. Université de Montréal.

[19] J. Rickel and W. L. Johnson. Animated agents for procedural training in virtual reality: Perception, cognition, and motor control. *Applied Artificial Intelligence*, 13:343–382, 1999.

[20] C. L. Sidner. An artificial discourse language for collaborative negotiation. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 814–819, Menlo Park, CA, 1994. AAAI Press.

[21] D. Sleeman and J. Brown, editors. *Intelligent Tutoring Systems*. Academic Press, 1982.

[22] D. R. Traum. *A Computational Theory of Grounding in Natural Language Conversation*. PhD thesis, Department of Computer Science, University of Rochester, Rochester, NY, 1994.

[23] E. Wenger. *Artificial Intelligence and Tutoring Systems*. Morgan Kaufmann, Los Altos, CA, 1987.

[24] B. P. Woolf. *Context-Dependent Planning in a Machine Tutor*. PhD thesis, Department of Computer and Information Science, University of Massachusetts at Amherst, 1984.

AMANDA - An Intelligent Dialog Coordination Environment

Marco A. Eleuterio[1]
PUC-PR/UTC
marcoa@hds.utc.fr

Jean-Paul Barthès
UTC, France
barthes@utc.fr

Flávio Bortolozzi
PUC-PR, Brazil
fborto@ppgia.pucpr.br

Celso A. Kaestner
PUC-PR, Brazil
kaestner@ppgia.pucpr.br

[1] Sponsored by CNPq, Brazil

Abstract

This paper describes AMANDA[2] - an intelligent system intended to coordinate collective dialog sessions in distance learning environments. The overall objective of AMANDA is to help tutors achieve better results from group discussions and improve knowledge transfer among the participants. This is done by integrating the collective dialog as a disciplined and well-coordinated activity in distance learning situations. For this purpose, the dialog is represented as an argumentation tree, a structured collection of questions, alternatives and arguments which evolves along sequential dialog cycles. The intelligent behavior of the system is due to its coordination actions taken in response to reasoning over the dialog. We describe how AMANDA coordinates the dialog process by generating a sequence of dialog cycles based on a set of coordination parameters. In this paper we briefly describe AMANDA's functional modules, internal structures and coordination algorithms. The knowledge models that support system reasoning are described, as well as our practical experience in domain modeling. We have tested the system in actual training situations, for which we chose a test course and modeled the corresponding domain knowledge. Although some modules of the system are still under development, specially those related to semantic reasoning, we discuss the application of semantic parameters and identify some techniques which may improve the coordination algorithm.

[2] AMANDA - Agent de Modélisation et ANalyse de Dialogues Argumentés - is a joint R&D effort between the Pontifical University of Paraná, Brazil (PUC-PR), the Technology University of Compiègne, France (UTC) and their respective partners Siemens Telecomunicações, Brazil and Cegos, France.

1. Introduction

Collaborative learning is about promoting knowledge transfer among the apprentices through a series of learning interactions. We recall a well-known knowledge management theory (Nonaka, 99) in which a knowledge transfer environment is composed of four knowledge-transfer spaces, namely the *socialization* space, the *dialoguing* space, the *systematization* space and the *internalization* space. In each of these spaces, a specific implicit↔explicit knowledge conversion occurs. By applying this approach to a collaborative learning environment, as detailed in (Eleuterio, 1999a), we categorize AMANDA as a *dialoguing* space in which the articulation of knowledge is the key for knowledge transfer. In traditional distance learning environments, this dialoguing space is normally implemented by discussion forums.

Our experience with discussion forums in Eureka (Eleuterio, 1999b), a web-based environment developed in partnership with Siemens and extensively used in academic and professional training contexts, shows that traditional discussions forums often fail to promote group learning. They either grow two much to be efficiently followed up by the tutor or suffer from the lack of participation and coordination. Similar problems are described in (Leary, 1998) when identifying common problems in discussion groups of knowledge management systems. From our observations, the two main reasons why discussion forums often fail are (i) the lack of discipline due to the poor integration of the discussion process into the regular activities of the course and (ii) the lack or articulation and coordination of the discussion.

With the purpose of overcoming the identified problems, we propose a dialog framework that covers both aspects, i.e. automatically coordinates the dialog while engaging the participants by generating dialog activities.

We identify three main differences between AMANDA and a traditional discussion forum. Firstly, the presence of domain models in AMANDA's architecture enables a certain degree of semantic reasoning over the dialog. Secondly, its coordination mechanism relieves the tutor from time-consuming coordination tasks, such as finding relations between users' inputs, measuring the degree of commitment of the participants, detecting disagreement topics and measuring the coverage of discussion topics. Thirdly, the system manages the dialog by generating discussion cycles, in which the participants express their supporting and opposing ideas in relation to another participant's input, thus creating a suitable context for the articulation and confrontation of ideas and points of view.

The proposed coordination mechanism allows various degrees of knowledge representation without impairing dialog control. It means that, if the system has no knowledge models, it can coordinate the dialog as well, gracefully degraded, by considering only structural parameters. This is possible due to the separation between structural and semantic aspects in the coordination mechanism (see section 4). This separation allows applying AMANDA to situations where knowledge modeling is neither feasible, e.g. open domain discussions, nor desirable, e.g. short-term courses.

Merging Two Complementary Approaches
Tutorial dialog has been subject of important research efforts, such as the CoLLeGE architecture (Ravenscroft & Pilkington, 2000) which analyzes dialog moves, conceptual changes and world models as the basis of the dialog process. Such work deeply inspects the tutor-apprentice interaction, but doesn't give much emphasis on the *collective* aspect of the dialog. On the other hand, the argumentative discourse environment (Karacapilidis, 1998) describes an argumentation framework applied to multi-agent decision making, which is fully devoted to formalize argumentative discourses. Our objective is to merge both approaches, which seem to be complementary, in a single dialog coordination system applied to collaborative distance learning environments.

2. System Overview

AMANDA is an autonomous domain-independent intelligent dialog coordination system applied to collective discussions. By *domain-independent* we mean that domain-dependent behavior is achieved by providing the corresponding domain knowledge models. By *intelligent coordination system* we mean that AMANDA takes coordination actions by reasoning over the structure and the semantics

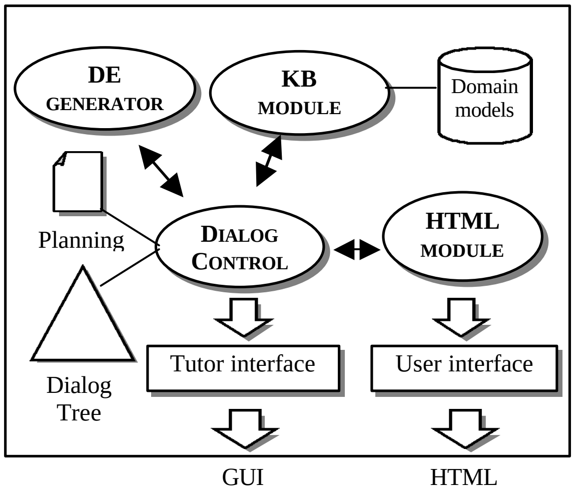


Figure 1a: System overview

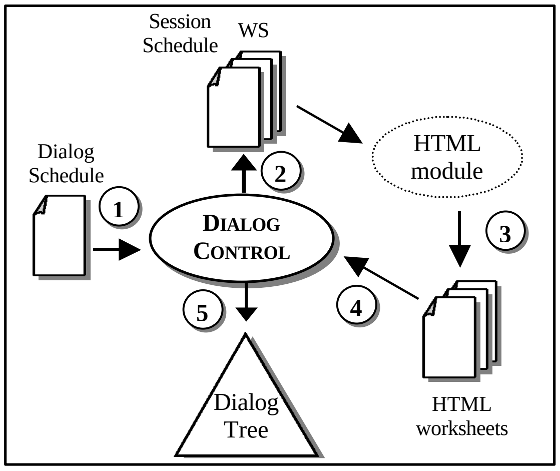


Figure 1b: Dialog control – simplified

of the dialog. The *autonomous* feature of AMANDA is due to its capability of coordinating the dialog without direct interference of the human tutor. Figure 1a shows the main modules of the system and the paragraphs below describe the modules, structures and processes that take part in the dialog coordination.

2.1. Dialog Control Module

This module is AMANDA's central coordination mechanism. Its principle is to organize the dialog in sequential periods called *sessions*, each one representing a time interval in which a certain number of discussions will be carried on. During each session, the system triggers a number of *dialog cycles* in order to update the dialog tree with input from the participants.

In the setup stage, the Dialog Control module reads the dialog schedule ①, where all sessions are described. It then repeatedly generates dialog cycles by producing worksheets ③ until a satisfactory degree of agreement is achieved. Each time the system receives input from the participants ④, the Dialog Control module analyzes and updates the dialog tree ⑤ and decides upon producing a new cycle or closing the dialog. The items below detail the structures handled by the Dialog Control module.

2.1.1. Dialog Planning

The dialog planning is represented by the *dialog schedule* and the *session schedule*.

Dialog Schedule

The dialog schedule is the overall planning of the dialog. It specifies the dialog sessions, the corresponding start/end dates and the respective domain of discourse (Figure 2).

Session	SD	ED	DS (domain of discourse)
S-1	Sd	Ed	(c1 … cm)
S-2	Sd	Ed	(c1 … cn)
			:
S-n	Sd	Ed	(c1 … cp)

S-n: the nth session of the dialog		DS:	a set of concepts from the domain ontology
SD: start date; ED: end date			

Figure 2: The dialog schedule Session

Session Schedule

The *session schedule*, on the other hand, is a dynamic structure automatically produced and updated by the system during a given session (Figure 3). Each entry of the session schedule is a *dialog cycle* which specifies a *dialog task* to each participant. A dialog task is the set of all nodes from the dialog tree (see item 2.1.2) which are assigned to the same participant at a certain dialog cycle.

A dialog task is represented by a *worksheet assignment* of the type (id, list-of-WEs), in which a list of worksheet elements (we) is assigned to a particular participant (id). Worksheet elements map directly to specific nodes of the dialog tree.

Cycle	SD	ED	WS assignment
C-1-x	Sd	Ed	((id (we-y-1 … we-y-n)) …)
C-2-x	Sd	Ed	((id (we-y-1 … we-y-n)) …)
			: :
C-n-x	Sd	Ed	((id (we-y-1 … we-y-n)) …)

C-n-x:	the nth dialog cycle of session x	WS:	worksheet, a set of ordered pairs of the type (id-x we-y)
SD:	start date	id:	the ID of the participant
ED:	end date	we:	worksheet element

Figure 3: The session schedule

2.1.2. Dialog Tree

The dialog tree, shown in Figure 4 is the structure that represents the dialog. Its internal nodes can be of five types: DIALOG, SESSION, DE, ALT and ARG. Its internal structure was adapted from the argumentation model (Karacapilidis, 1998). The paragraphs below describe each type of node and their corresponding relations to the dialog process.

DIALOG node

The DIALOG node is the uppermost node of the tree. It contains a reference to a number of dialog sessions. When a dialog is created, this node is initialized with the information contained in the dialog schedule (Figure 2.a).

SESSION node

The SESSION node is the uppermost node of a dialog session. Dialog sessions are intended to organize the discussion into separate time periods, each one assigned to a certain *domain*

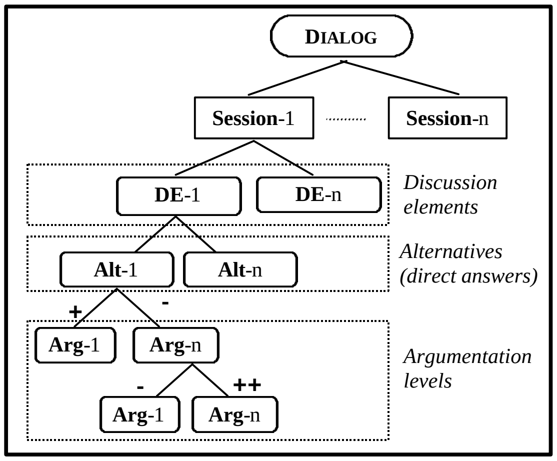


Figure 4: The dialog tree

of discourse. The SESSION node contains a reference to all discussion elements (DEs) which are scheduled for discussion within this session.

DE node

The DE node represents a discussion element, i.e. a natural language question that will originate a specific discussion. Examples of DEs are: "which are the elements of a training budget?" or "what types of connection elements exist in a computer network?".

A DE can be classified as a *content-expected interrogative speech act* (Porayska-Pompa, 2000), for which we expect an answer with a certain "content" as response. According to the argumentation model of (Karacapilidis, 1998), a DE node is an *issue* to be debated.

ALT node

The ALT node is an answer to a question. It is an *alternative* response to a certain DE. The answer contained in an ALT node is the "content" expected by its corresponding DE node. In Karacapilidis' model, an ALT node is a *position* over an *issue*.

ARG node

The ARG node, or *argumentation* node, represents a *supporting* or *opposing* reaction from a given participant over a dialog element placed by another participant. An ARG node can either refer to an ALT node or to another ARG node.

Argumentation nodes are key elements of the dialog. When analyzed as a whole, they represent the level of collective agreement over a given position. Each ARG node conveys a supporting or opposing intention, or *polarity*. This intention is expressed by four levels: *total agreement* (++), *partial agreement* (+), *partial disagreement* (-) and *total disagreement* (--).

A substantial coordination effort of AMANDA is concentrated in analyzing the effects of the ARG nodes over the dialog tree (more details in item 4).

2.1.3. Dialog Control Interface

The Dialog Control module has a graphical interface which allows us to view the dialog tree and perform editing and follow-up functions over the dialog. This interface, primarily designed to follow up the dialog, can also be used to simulate dialog situations and evaluate the coordination algorithms.

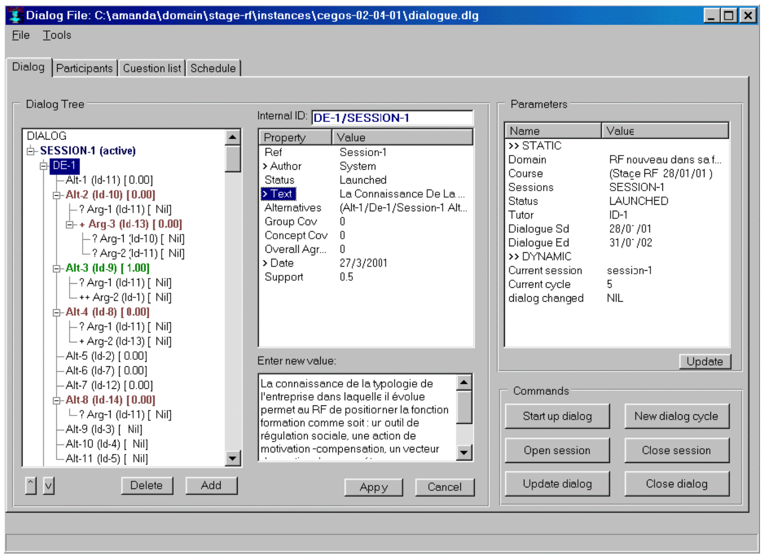
Figure 5 shows the Dialog Control interface. It allows to (i) view the dialog tree, (ii) edit its nodes, (iii) view the internal parameters of the dialog and (iv) simulate a dialog by means of control buttons.

2.2. KB Module

This module is responsible for managing the knowledge model and providing semantic parameters to the Dialog Control module. The central knowledge representation is the *domain ontology*, but other structures may be added, such as the domain task structure. The KB module evaluates the dialog from the semantic point of view, by calculating a certain number of parameters, such as the semantic proximity between two text-based messages, the conceptual distance between ontology concepts or the conceptual coverage of a certain dialog session.

2.2.1. Domain Models

AMANDA requires domain models to perform semantic reasoning over the dialog. In order to enable different types of domain models to be "plugged" into the KB module, we decided to use an ontology-centered approach. This allows to build various models, such as conceptual maps and task structures, which refer to the ontology concepts when applicable.



```
DIALOG
SESSION-1 (active)
  DE-1
    Alt-1 (Id-11) [0.00]
    Alt-2 (Id-10) [0.00]
      ? Arg-1 (Id-11) [ Nil]
      + Arg-3 (Id-13) [0.00]
        ? Arg-1 (Id-10) [ Nil]
        ? Arg-2 (Id-11) [ Nil]
    Alt-3 (Id-9) [1.00]
      ? Arg-1 (Id-11) [ Nil]
      ++ Arg-2 (Id-1) [ Nil]
    Alt-4 (Id-8) [0.00]
      ? Arg-1 (Id-11) [ Nil]
      + Arg-2 (Id-13) [ Nil]
    Alt-5 (Id-2) [0.00]
    Alt-6 (Id-7) [0.00]
    Alt-7 (Id-12) [0.00]
    Alt-8 (Id-14) [0.00]
      ? Arg-1 (Id-11) [ Nil]
    Alt-9 (Id-3) [  Nil]
    Alt-10 (Id-4) [  Nil]
    Alt-11 (Id-5) [  Nil]
```

Internal ID: DE-1/SESSION-1

Property	Value
Ref	Session-1
> Author	System
Status	Launched
> Text	La Connaissance De La ...
Alternatives	(Alt-1/De-1/Session-1 Alt...
Group Cov	0
Concept Cov	0
Overall Agr...	0
> Date	27/3/2001
Support	0.5

Enter new value:

La connaissance de la typologie de l'entreprise dans laquelle il évolue permet au RF de positionner la fonction formation comme soit : ur outil de régulation sociale, une action de motivation -compensation, un vecteur

Name	Value
>> STATIC	
Domain	RF nouveau dans sa f...
Course	(Stage RF 28/01/01)
Sessions	SESSION-1
Status	LAUNCHED
Tutor	ID-1
Dialogue Sd	28/0˙/01
Dialogue Ed	31/0˙/02
>> DYNAMIC	
Current session	session-1
Current cycle	5
dialog changed	NIL

Figure 5: The Dialog Control interface

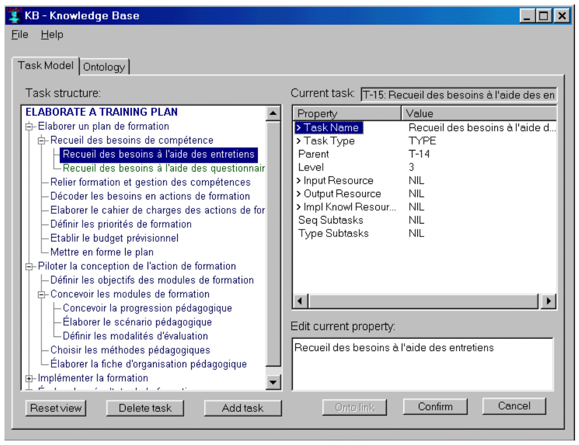
Domain Ontology

The domain ontology is AMANDA's central knowledge representation. Its role is to organize domain concepts so as to enable reasoning. Apart the various definitions found in the knowledge representation literature, it is a consensus that ontologies are conceptual models that explicit the nature of the concepts. The basic type of ontology, the "terminological ontology", or "level 1" ontology (Mizoguchi, 2000), contains primarily is-a links. In some cases, formal definitions are needed to completely reason over a concept. In such cases, more powerful ontologies, like interpretable or executable ontologies, are required.

In our system, since the ontology is used mainly for terminological purposes, we adopted a simple structure which organizes concepts by means of *is-a* and *part-of* links. We decided to merge is-a and part-of links for practical reasons. We were faced with situations in which a concept would be better represented by a part-of decomposition than by a taxonomy relation. In fact, in certain domains, the use of part-of links is the only way to construct ontologies, as in the case of the PLINIUS project (Van der Vet & Mars, 1998). However, depending on the rigor demanded by the ontology application, merging is-a and part-of links may result in tangled hierarchies and confuse the reasoning mechanisms. A formal approach to this problem is described in (Guarino, 2000).

Task Structure

Due to the inherent task-oriented nature of the test course, we used a *task structure* as a complementary knowledge model. It represents the decomposition of a task by means of two types of links: the *sequence* link and the *type* link. *Sequence links* decompose a complex task in a sequence of more detailed sequential subtasks, while *type links* specify different methods of performing a certain task. A detailed description of task structures can be found in (Chandrasekaran, 1992) and (Decker, 1995). Figure 6 shows the Task Model section of the KB interface.



Task structure:

```
ELABORATE A TRAINING PLAN
Elaborer un plan de formation
  Recueil des besoins de compétence
    Recueil des besoins à l'aide des entretiens
    Recueil des besoins à l'aide des questionnair
  Relier formation et gestion des compétences
  Décoder les besoins en actions de formation
  Elaborer le cahier de charges des actions de for
  Définir les priorités de formation
  Etablir le budget prévisionnel
  Mettre en forme le plan
Piloter la conception de l'action de formation
  Définir les objectifs des modules de formation
  Concevoir les modules de formation
    Concevoir la progression pédagogique
    Élaborer le scénario pédagogique
    Définir les modalités d'évaluation
  Choisir les méthodes pédagogiques
  Élaborer la fiche d'organisation pédagogique
Implémenter la formation
```

Current task: T-15: Recueil des besoins à l'aide des en

Property	Value
> Task Name	Recueil des besoins à l'aide d...
> Task Type	TYPE
Parent	T-14
Level	3
> Input Resource	NIL
> Output Resource	NIL
> Impl Knowl Resour...	NIL
Seq Subtasks	NIL
Type Subtasks	NIL

Edit current property:

Recueil des besoins à l'aide des entretiens

Figure 6: KB interface – *Task Model* section

2.3. DE Generator

This module produces natural language questions based on the available knowledge models. Questions are generated by the system in order to include a given domain topic to the dialog. In practice, this is done to move the focus of the dialog to a desired sub-domain. The content of the questions are based on the links and concepts available in the knowledge models.

Suppose an ontology in the "computer network" domain containing an *is-a* link from the concept <*connection element*> toward the concepts <*hub*> and <*router*>. The semantics of this relation is: *"hubs and routers are types of connection elements"*. For the DE Generator, this link would produce a sentence of the type "what differs *hubs* from *routers* since both are *connection elements?*". This sentence conveys a pre-defined intention to find out the *identity criteria*, or a distinguishing property, between two concepts belonging to the same parent. We could generalize this principle by stating: *"if there is a taxonomic distinction between two concepts, there must be a set of properties capable to distinguish them"* (Guarino, 2000). For each type of semantic relation contained in the knowledge models, we can define a set of generic principles that can be used for sentence generation.

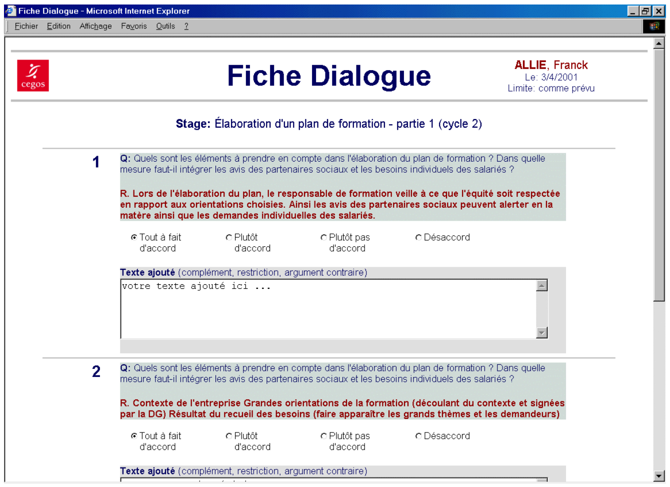
As in the propositions of (Ravenscroft, 2000), the sentences produced by the DE Generator carry a specific intention in the discourse. In our case, they are meant to investigate the domain along five different axes, each one assigned to a specific semantic link of the knowledge model. The ontology contributes with two axes: (i) the nature of the concepts (*is-a* links) and (ii) the elements of a composed concept (*part-of* links). The task model contributes with the remaining three axes: (i) the use of the concepts by a certain task (*resource* link); (ii) the decomposition of a complex task into sub-tasks (*sequence* link); and (iii) different ways of performing a task (*type* link). Each of these axes maps to a set of *sentence structures* of the type shown in the example above. The DE Generator can thus be considered the *linguistic level* of the knowledge models.

2.4. The HTML Module

This module is responsible for the interface between AMANDA and the participants of the dialog. This is done by the dynamic generation of *worksheets* in HTML format (see figure 7).

These worksheets are accessed by the participants, filled in and sent back to the system. Once the worksheets are returned, the Dialog Control module updates the dialog tree.

The HTML module was implemented by a PHP script running on an HTTP server. The communication between the HTML module and the Dialog Control module (see Figure 1.b ④) is done by intermediate files.


Fiche Dialogue - Microsoft Internet Explorer
Fichier Edition Affichage Favoris Outils ?
cegos

Fiche Dialogue

ALLIE, Franck
Le: 3/4/2001
Limite: comme prévu

Stage: Élaboration d'un plan de formation - partie 1 (cycle 2)

1 **Q:** Quels sont les éléments à prendre en compte dans l'élaboration du plan de formation ? Dans quelle mesure faut-il intégrer les avis des partenaires sociaux et les besoins individuels des salariés ?

R. Lors de l'élaboration du plan, le responsable de formation veille à ce que l'équité soit respectée en rapport aux orientations choisies. Ainsi les avis des partenaires sociaux peuvent alerter en la matère ainsi que les demandes individuelles des salariés.

[✓ Tout à fait d'accord] Plutôt d'accord Plutôt pas d'accord Désaccord

Texte ajouté (complément, restriction, argument contraire)

votre texte ajouté ici ...

2 **Q:** Quels sont les éléments à prendre en compte dans l'élaboration du plan de formation ? Dans quelle mesure faut-il intégrer les avis des partenaires sociaux et les besoins individuels des salariés ?

R. Contexte de l'entreprise Grandes orientations de la formation (découlant du contexte et signées par la DG) Résultat du recueil des besoins (faire apparaître les grands thèmes et les demandeurs)

[✓ Tout à fait d'accord] Plutôt d'accord Plutôt pas d'accord Désaccord

Texte ajouté (complément, restriction, argument contraire)

Figure 7: Worksheets in HTML format

3. The Dialog Process

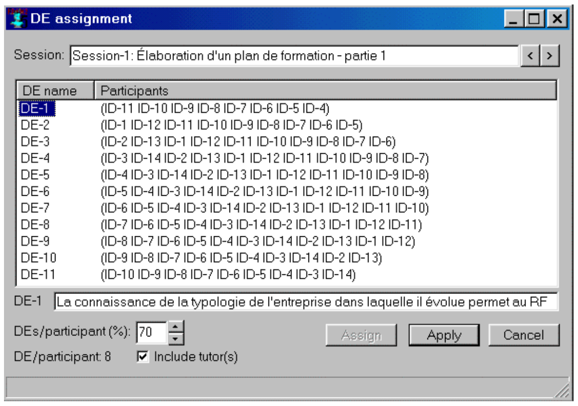
This item explains how AMANDA starts and conducts the dialog process, as well as the related algorithms.

3.1. Dialog Setup

The dialog starts with the creation of a *session schedule* based on the available *dialog schedule* (Figures 2 and 3). Once the dialog session is established, i.e. the SESSION node and the related DE nodes are created, the system can trigger the first dialog cycle.

3.2. First Dialog Cycle

The first dialog cycle, identified as the ALT level in the dialog tree, is intended to distribute the DEs among the participants. To do so, AMANDA takes the set of DEs, as well as the set of participants, and executes the DE-assignment algorithm. This algorithm generates DE assignments of the type (DE, list-of-ids) and can be parameterized according to the desired load of DE/participant and the presence/absence of the tutor(s) in the discussion (see Figure 8).



Session: Session-1: Élaboration d'un plan de formation - partie 1

DE name	Participants
DE-1	(ID-11 ID-10 ID-9 ID-8 ID-7 ID-6 ID-5 ID-4)
DE-2	(ID-1 ID-12 ID-11 ID-10 ID-9 ID-8 ID-7 ID-6 ID-5)
DE-3	(ID-2 ID-13 ID-1 ID-12 ID-11 ID-10 ID-9 ID-8 ID-7 ID-6)
DE-4	(ID-3 ID-14 ID-2 ID-13 ID-1 ID-12 ID-11 ID-10 ID-9 ID-8 ID-7)
DE-5	(ID-4 ID-3 ID-14 ID-2 ID-13 ID-1 ID-12 ID-11 ID-10 ID-9 ID-8)
DE-6	(ID-5 ID-4 ID-3 ID-14 ID-2 ID-13 ID-1 ID-12 ID-11 ID-10 ID-9)
DE-7	(ID-6 ID-5 ID-4 ID-3 ID-14 ID-2 ID-13 ID-1 ID-12 ID-11 ID-10)
DE-8	(ID-7 ID-6 ID-5 ID-4 ID-3 ID-14 ID-2 ID-13 ID-1 ID-12 ID-11)
DE-9	(ID-8 ID-7 ID-6 ID-5 ID-4 ID-3 ID-14 ID-2 ID-13 ID-1 ID-12)
DE-10	(ID-9 ID-8 ID-7 ID-6 ID-5 ID-4 ID-3 ID-14 ID-2 ID-13)
DE-11	(ID-10 ID-9 ID-8 ID-7 ID-6 ID-5 ID-4 ID-3 ID-14)

DE-1 La connaissance de la typologie de l'entreprise dans laquelle il évolue permet au RF

DEs/participant (%): 70

DE/participant: 8 [✓ Include tutor(s)]

Figure 8: The DE-assignment interface

3.3. Argumentation Cycles

As a result of the first cycle, the system receives a number of answers to the proposed DEs, or so called *alternatives*. These alternatives, represented by ALT nodes in the dialog tree, will be subject of analysis in the argumentation cycles. From this moment on, AMANDA will generate a sequence of dialog cycles in order to expand the tree either in depth or in breadth, until a satisfactory degree of agreement is reached. At this point we distinguish two key concepts: the *dialog level* and the *dialog cycle*.

Dialog level

The dialog level is the *depth* level of the dialog tree, i.e. the distance from a certain node to the root. A large number of dialog levels means that the dialog has grown in depth, i.e. an original answer of a given DE has been subject of many subsequent *argumentation* cycles.

High dialog levels indicate that either the answer has been repeatedly *opposed* or progressively *clarified*, depending on the polarity of the ARG nodes. Certain typical behaviors in argumentative discourse, such as belief change, can only be detected with high dialog levels.

In practice, however, high dialog levels lead to interpretation difficulties that must be handled by the interface design. For example, suppose that a participant receives a discussion element of argumentation level 3, i.e. an Arg-3 node. It means that he is supposed to analyze his parent node (argument Arg-2) that refers to another argument (Arg-1), which in turns refers to an answer (Alt) to a given question (DE). If the user interface is not carefully designed, it's likely that we misinterpret the participant's contribution due to the large number of previous elements. On the other hand, we must present the whole history of the discussion so that the user can trace the ideas and place his contribution. This problem opens a design issue which must not be overlooked.

Dialog cycle

The dialog cycle, on the other hand, is a time period in which the dialog tree expands, possibly in depth but not necessarily. A large number of dialog cycles means that the dialog has evolved through a large number of interactions, but not necessarily that it has grown in depth. This is the distinction between the dialog level and the dialog cycle.

To exemplify, suppose that a certain answer (ALT node) exhibits low local support level (typically negative values) and low participation level (i.e. few lower level ARG nodes). This is the case, for example, when an answer is opposed by some counter-arguments, but has not been broadly discussed within the group. In this case, the system may decide to create a specific dialog cycle to re-launch this answer to be analyzed by other participants. This new dialog cycle will only increase the breadth of the tree, keeping the dialog depth unchanged.

4. Reasoning Over the Dialog

The coordination actions taken by the system are based on a certain degree of reasoning over the dialog tree. Two types of reasoning are proposed: *structural* and *semantic* reasoning.

Structural reasoning concerns to the structural aspect of dialog tree, specially the distribution of the ARG nodes and their corresponding polarities. Semantic reasoning, on the other hand, analyzes the content of the textual information in order to find semantic relations among the nodes.

The separation between structural and semantic reasoning allows AMANDA to coordinate the dialog even in the absence of domain models. The following paragraphs identify and propose some of the parameters to be evaluated in each type of reasoning.

4.1. Structural reasoning

Structural reasoning analyses the structure of the dialog tree, mainly the distribution of ARG nodes and their embedded supporting/opposing intentions, to decide which nodes will be re-launched and to which participants they will be assigned. The main structural parameter is the *support level* of a node in respect to its lower level sub-tree. The items below detail the implementation of this reasoning.

4.1.1. Evaluating the support level

Before initiating a new dialog cycle, AMANDA evaluates the overall agreement level of each DE and decides upon creating a new cycle or closing the discussion tree for the corresponding DE. This decision takes into consideration the concepts of *local* and *transmitted support level*.

Local support level

Each "supportable" node of the dialog tree (i.e. nodes of the type ALT or ARG) can be assigned a local support level (LS). This level represents the degree of consensus of this node regarding its lower level sub-tree. The support levels are calculated by traversing the dialog tree from the leaves to the root and assigning support levels to each ALT or ARG node. The local support level is a real number ranging from –1.0 to +1.0, respectively meaning total disagreement and total agreement. This number is the average level of *transmitted support* from all its direct descendant nodes (see Eq. 1). If the node has no direct child nodes, i.e. in the case of *leaf nodes*, the local support level is assigned the maximum value of +1.0.

The local support level of a node N, LS(N), is expressed by Eq. 1 and exemplified in Figure 9.

$$LS(N) = \begin{cases} \Sigma\,(TS(child(N))/n & \text{if } n > 0 \\ +1.0 & \text{if } n = 0 \end{cases}$$

Where:
- TS is the transmitted support level (Eq. 2),
- child(N) returns the next child of node N
- "n" is the number of child nodes.

Eq. 1: The local support level (LS)

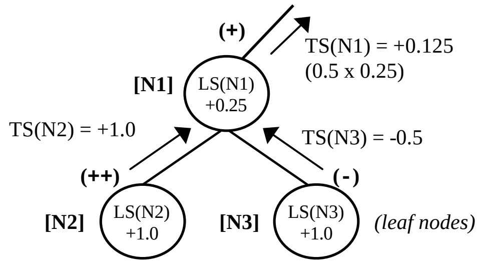


Figure 9 Local and transmitted support levels

The transmitted support level

The principle is that each descendant ARG node transmits to its direct parent a certain level of support – the *transmitted support level*. This level depends on the type of argument (++, +, -, --) and the local support level of the transmitting node itself. The nominal level that a node of type ++/+/-/-- transmits to its parent is respectively +1.0/+0.5/-0.5/-1.0.

For example, an ARG++ transmits to its direct parent a support level of +1.0 multiplied by its own local support level. Analogously, an ARG- node transmits to its direct parent a support level of -0.5 multiplied by its own local support level. In other words, the local support level acts as a "damping" parameter that tends to reduce the transmitted support level if the node does not exhibit total support from its lower levels. The support level TS(N) transmitted by a node N to its direct parent is expressed by Eq.2.

$$TS(N) = \begin{cases} +1.0 \times LS'(N) & \text{if arg-type(N) = "++"} \\ +0.5 \times LS'(N) & \text{if arg-type(N) = "+"} \\ -0.5 \times LS'(N) & \text{if arg-type(N) = "-"} \\ -1.0 \times LS'(N) & \text{if arg-type(N) = "- -"} \end{cases}$$

Where LS'(N) = min(0, LS(N))

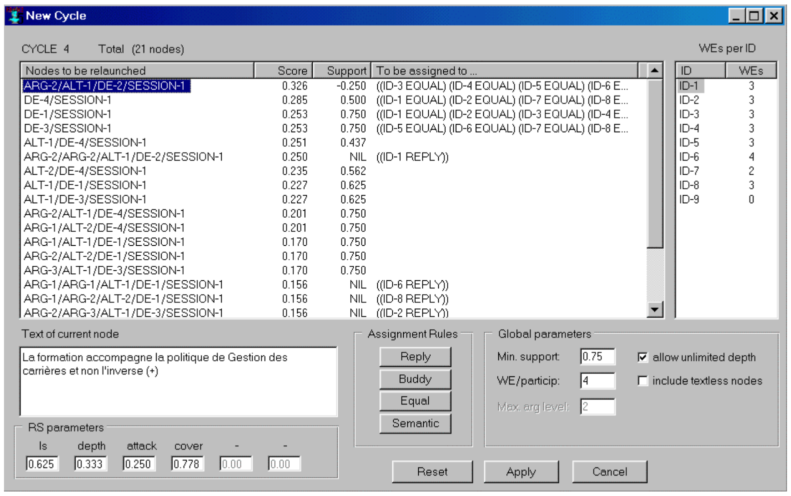
Eq. 2: The transmitted support level (TS)

An important assumption of the algorithm is that nodes with negative LS are disabled to transmit TS level to their parent by being excluded from the set of children in LS calculation. This is done to prevent highly opposed nodes from influencing their respective ascendants. In addition, this is necessary to avoid undesirable situations in which the original polarity of a node (*supporting* or *opposing*) is inverted by its negative LS.

The algorithm starts the evaluation by assigning LS values of +1.0 to all leaf nodes and then "climbs" up the tree by calculating the corresponding LS values for all nodes up to the DE node.

Tests performed in actual dialog situations show that the support levels obtained by this algorithm reflect the collective agreement of a dialog contribution within the discussion. They are used to compute a *priority value* that defines which nodes are to be re-launched in the next dialog cycle.

Figure 10 shows the interface for opening a new dialog cycle. It shows the nodes to be re-launched, their corresponding re-launch score and support levels and the assignment proposed by the system.



CYCLE 4 Total (21 nodes)

Nodes to be relaunched	Score	Support	To be assigned to ...
ARG-2/ALT-1/DE-2/SESSION-1	0.326	-0.250	((ID-3 EQUAL) (ID-4 EQUAL) (ID-5 EQUAL) (ID-6 E...
DE-4/SESSION-1	0.285	0.500	((ID-1 EQUAL) (ID-2 EQUAL) (ID-7 EQUAL) (ID-8 E...
DE-1/SESSION-1	0.253	0.750	((ID-1 EQUAL) (ID-2 EQUAL) (ID-3 EQUAL) (ID-4 E...
DE-3/SESSION-1	0.253	0.750	((ID-5 EQUAL) (ID-6 EQUAL) (ID-7 EQUAL) (ID-8 E...
ALT-1/DE-4/SESSION-1	0.251	0.437	
ARG-2/ARG-2/ALT-1/DE-2/SESSION-1	0.250	NIL	((ID-1 REPLY))
ALT-2/DE-4/SESSION-1	0.235	0.562	
ALT-1/DE-1/SESSION-1	0.227	0.625	
ALT-1/DE-3/SESSION-1	0.227	0.625	
ARG-2/ALT-1/DE-4/SESSION-1	0.201	0.750	
ARG-1/ALT-2/DE-4/SESSION-1	0.201	0.750	
ARG-1/ALT-1/DE-1/SESSION-1	0.170	0.750	
ARG-2/ALT-2/DE-1/SESSION-1	0.170	0.750	
ARG-3/ALT-1/DE-3/SESSION-1	0.170	0.750	
ARG-1/ARG-1/ALT-1/DE-1/SESSION-1	0.156	NIL	((ID-6 REPLY))
ARG-1/ARG-2/ALT-2/DE-1/SESSION-1	0.156	NIL	((ID-8 REPLY))
ARG-2/ARG-3/ALT-1/DE-3/SESSION-1	0.156	NIL	((ID-2 REPLY))

WEs per ID

ID	WEs
ID-1	3
ID-2	3
ID-3	3
ID-4	3
ID-5	3
ID-6	4
ID-7	2
ID-8	3
ID-9	0

Text of current node

La formation accompagne la politique de Gestion des carrières et non l'inverse (+)

RS parameters

ls	depth	attack	cover	-	-
0.625	0.333	0.250	0.778	0.00	0.00

Global parameters

Min. support: 0.75

WE/particip: 4

Max. arg level: 2

☑ allow unlimited depth

☐ include textless nodes

Figure 10: Opening a new dialog cycle

4.2. Semantic Reasoning

Due to the text-based nature of the dialog contributions and the domain dependency of the dialog, it seems reasonable to apply semantic matching techniques to improve the coordination mechanism. We identify two semantic parameters with large potential for this purpose.

The first parameter is the *semantic proximity* between textual inputs, such as direct answers or arguments. This may be useful to discover hidden relations among users' input, specially in extensive dialog trees with large amounts of textual information. The availability of a domain ontology may extend the traditional word-matching by adding concept-based matching, as described in (Honkela, 1995).

The second parameter is the *conceptual coverage*, which aims to detect missing or insufficiently covered topics in dialog sessions. Such topics can be identified by analyzing the occurrence of certain words of domain in a given dialog sub-tree. As a response, specific DEs can be generated with the objective of bringing such subjects back to the dialog (see section 2.3).

Other text techniques, such as ontology-based information retrieval, can be applied for finding related concepts among textual information (Guarino, 1999).

One of the difficulties to apply semantic reasoning is the need for comprehensive and well constructed knowledge models, which are difficult to achieve even by experienced knowledge experts. In addition, lexical diversity may impose difficulties in relating similar concepts from different user inputs. This suggests that semantic reasoning might give better results when applied to very specific domains with low terminology diversity.

5. Conclusion

Our system was developed under empirical observations over distance learning environments, specially over the poor results achieved in traditional discussion forums. The large potential in terms of knowledge transfer of such environments encouraged us to go beyond traditional approaches and to design an environment that takes advantage of the collective discussions.

The real problem that we aim to solve is that successful distance discussion sessions require participants to be highly committed and represent a very time-consuming effort from the tutors. As a result, very few discussion forums end up satisfactorily.

We created a dialog framework that attempts to keep up the commitment of the participants by generating regular dialog activities and relieve the tutor from the dialog coordination task. This framework has been applied in actual distance training situations and has been the test-bed for various algorithms and coordination strategies.

A modular approach for the coordination mechanism, which separates structural from semantic parameters, allows it to be applied to situations where domain models are not available.

The next steps of this work are to implement the semantic reasoning over the dialog and to consolidate the results obtained in actual training situations.

Acknowledgements

We are grateful to CEGOS for its financial support and active participation in the context of the CEGOS-UTC partnership and to Siemens Telecomunicações Brasil for its financial support of the LAMI laboratory in the context of the PUCPR-Siemens partnership.

6. References

Chandrasekaran B., Johnson T., Smith J. Task-structure analysis for knowledge modeling. Communications of the ACM (CACM 35) no.9, 1992, pp. 1124-137.

Decker K. Environment Centered Analysis and Design of Coordination Mechanisms; Ph.D. thesis. Department of Computer Science; University of Massachusetts, Amherst, 1995.

Eleuterio M., Eberspächer H. A Knowledge Management Approach to Virtual Learning Environments. International Workshop on Virtual Education (WISE), 1999.

Eleuterio M., Eberspächer H.,Vasconcelos C., Jamur J. Eureka: um ambiente de aprendizagem cooperativa baseado na Web para Educação à Distância. In: Brazilian Symposium on Informatics in Education (SBIE), 1999.

Guarino N., Masolo C., and Vetere G., OntoSeek: Content-Based Access to the Web, IEEE Intelligent Systems 14(3), May/June 1999.

Guarino N., Welty, C. Ontological Analysis of Taxonomic Relationships. In, Laender, A. and Storey, V., eds, Proceedings of ER-2000: The 19th International Conference on Conceptual Modeling. Springer-Verlag LNCS. 2000.

Honkela T. Self-organizing maps in natural language processing. Ph.D. thesis. Helsinki University of Technology, 1995.

Karacapilidis N., Papadias D. A computational approach for argumentative discourse in multi-agent decision making environments. AI communications 11 (1998) 21-23.

Leary D. Using AI in Knowledge Management: Knowledge Bases and Ontologies; IEEE Intelligent Systems, May/June, 1998.

Mizoguchi R., Bourdeau J. Using Ontological Engineering to Overcome Common AI-ED Problems; International Journal of Artificial Intelligence in Education 2000.

Nonaka I., Toyama R., Konno N. SECI, Ba, and Leadership: A Unified Model of Dynamic Knowledge Creation; D.J. Teece and I. Nonaka (Eds). Oxford University Press, 1999.

Porayska-Pompa K, Pain H. Aspects of Speech Act Categorisation: Towards Generating Teacher's Language. International Journal of Artificial Intelligence in Education, 2000.

Ravenscroft A., Pilkington R. Investigation by design: developing dialog models to support reasoning and conceptual change. International Journal of Artificial Intelligence in Education, 2000.

Van der Vet P., Mars N. Bottom-up Construction of Ontologies. IEEE Transactions on Knowledge Engineering, vol. 10 no. 4, 1998.

The Design and Formative Analysis of a Dialog-Based Tutor

Neil T. Heffernan (neil@cs.cmu.edu)
Kenneth R. Koedinger (koedinger@cmu.edu)
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Symbolization is the ability to translate a real world situation into the language of algebra. We believe that symbolization is the single most important skill students learn in high school algebra. We present research on what makes this skill difficult and report the discovery of a "hidden" skill in symbolization. Contrary to past research that has emphasized that symbolization is difficult due to both comprehension difficulties and the abstract nature of variables, we found that symbolization is difficult because it is the articulation in the "foreign" language of "algebra". We also present *Ms. Lindquist*, an Intelligent Tutoring System (ITS) designed to carry on a tutorial dialog about symbolization. Ms. Lindquist has a separate tutorial model encoding pedagogical content knowledge in the form of different tutorial strategies, which were partially developed by observing an experienced human tutor. We discuss aspects of this human tutor's method that can be modeled well by Ms. Lindquist. Finally, we present an early formative showing that students can learn from the dialogs Ms. Lindquist is able to engage student in. Ms. Lindquist has tutored over 600 students at www.AlgerbaTutor.org.

Introduction

The mission of the Center for Interdisciplinary Research on Constructive Learning Environments (CIRCLE) is 1) to study human tutoring and 2) to build and test a new generation of tutoring systems that encourage students to construct the target knowledge instead of telling it to them (VanLehn et. al., 1998). CAI (Computer Aided Instruction) systems were 1st generation tutors. They presented a page of text or graphics and depending upon the student's answer, put up a different page. Model-tracing ITSs are 2nd generation tutoring systems that allow the tutor to follow the line of reasoning of the student. ITS have had notable success (Koedinger et. al., 1997) despite the fact that human tutoring can look very different (Moore, 1996). One way they are different is that there is a better sense of a dialog in human tutoring and maybe this is important. After analyzing over 100 hours of untrained tutors in naturalistic tutoring sessions Graesser et. al. (in press) believe "there is something about interactive discourse that is responsible for learning gains."

The members of CIRCLE are working on 3rd generation tutoring system that are meant to engage in a dialog with students, using multiple strategies, to allow students to construct their own knowledge of the domain. We have built a new ITS, called *Ms. Lindquist*, which not only is able to model-trace the student's actions, but can be more human-like in carrying on a running conversation, complete with probing questions, positive and negative feedback, follow-up questions in embedded sub-dialogs, and requests for explanation as to why something is correct. In order to build Ms. Lindquist we have expanded the model-tracing paradigm so that Ms. Lindquist not only has a model of the student, but also has a model of tutorial reasoning (e.g. Clancey, 1982). Based on observation of an experienced tutor and cognitive research, this tutorial model has multiple tutorial strategies at its disposal.

The task domain we are working on is symbolization, which is the task of writing an algebraic expression given a real-world problem context, often presented in the form of a word problem. Symbolization is important because if students can't translate problems into algebra, they will not be able to apply algebra to solve real world problems. This domain makes it easy to avoid some difficult natural language issues because we can ask students to write algebraic expressions and those expressions are easy for the computer to "understand". We take advantage of this property of the domain to avoid any serious natural language processing; we also use pull-down menus to allow students to construct explanations. Instead, we focus our energies on modeling tutorial reasoning which includes capturing the *pedagogical content knowledge (Shulman, 1986)* of an experienced human tutor. Pedagogical content knowledge is the knowledge that a good tutor has about how to teach a specific skill. A good tutor is not just one who knows the domain, nor is it simply one who knows generally tutoring rules. A good tutor is one who also has content specific strategies that can help a student overcome common difficulties. We have set out to observe and model some of these strategies for our specific domain of symbolization. Lets look at one example of a content-specific pedagogical strategy for symbolization. The following was collected and transcribed from a one hour long one-on-one tutoring session between an experienced human tutor and an eighth grade student working on the "bike-trip" problem, which we use as one of several running examples.

240. Student: [reads problem[1]] Cathy took a "m" mile bike ride. She rode at a speed of "s" miles per hour. She stopped for a "b" hour break. Write an expression for how long the trip took.
241. S: uhm [writes "s/m+b" but should be "m/s+b"]
242. Tutor: How do you calculate the amount of time it takes you? If you're, if you're, if you're riding at, let's make it simple. If you are riding at 20 miles per hour, OK, and you go 100 miles, how many hours did that take you?
243. S: Um 5
244. T: 5. And how did you get that 5? How did you use the numbers 100 and …
245. S: 100 miles divided by miles per hour
246. T: So you took the miles and divided it by the [garbled, but possibly "speed"]
247. S: Miles divided by s plus b equals time [writes m/s+b]
248. T: Right.

The tutor in the above dialog, appears to have done two things to scaffold this problem. First, the tutor focused on the problem of calculating the time actually on the bikes by decomposing what had been a problem with two operators into a problem that had only one operator. Presumably this is because the student indicated he understood that the goal quantity was found by adding for the amount of the break ("b") to the time actually on the bikes.

The second scaffolding move the tutor did was to change the problem question from a symbolization question to a presumably simpler *compute question* by asking the student to calculate the speed using 100 and 20 rather than "m" and "s". Then in line 244 the tutor asked for the *articulation step* of "How did you get that 5?" Finally, the student is prompted for the *generalization step* of writing the expression using variables.

Our experienced tutor often invited the student to use concrete numbers. We call this strategy the *concrete articulation strategy (Koedinger & Anderson, 1998[2])*. McArthur et. al. (1990) also observed that human tutors often used what he called *curriculum scripts* and *micro-plans*, which often involved a series of questions designed to remediate particular difficulties. We call these scripts *knowledge construction dialogs* to emphasis the fact that that we are trying to build a tutor that encourages students to build their own knowledge by less often *telling* them a hint and more often *asking* them a question.

The impediments to building a third generation tutor is not just technical. We think

[1] Throughout this paper, text in square brackets are comments, and S and T stand for "student" and "tutor" respectfully.

[2] Then called the *inductive support* strategy.

that if you want to build a good ITS for a domain you need to:

- Study what makes that domain difficult, including discovering any hidden skills, as well as determining what types of errors students make.
- Construct a theory of how students solve these problem. (We instantiated that theory in a cognitive model.)
- Observe experienced human tutors to find out what pedagogical content knowledge they have and then build a tutorial model that, with the help of the theory of domain skills, can capture and reproduce some of that knowledge.

We look at these each of these steps in turn.

What Makes Symbolization Difficult?

Symbolization is a difficult task for students. For instance, only 13% of student correctly answered the following question "Anne is in a rowboat in a lake that is 2400 yards wide. She is 800 yards from the dock. She rows back towards the dock at a speed of 40 yards per minute for 'm' minutes. How far is Ann from the dock?" To determine what makes symbolization difficult we conducted two *difficulty factors assessments* (e.g., Koedinger & MacLaren, 1997) which are paper and pencil tests that we gave to groups of 80+ students (Heffernan & Koedinger, 1997 and 1998). First, we identified three hypotheses about what makes symbolization difficult.

The first of these is the *comprehension hypothesis*. Much of the prior research (e.g., Lewis & Mayer, 1987) on word problem solving has focused on students' comprehension abilities. For instance, Nathan, Kintsch, & Young (1992) "claim that [the] symbolization [process] is a highly reading-oriented one in which poor comprehension and an inability to access relevant long term knowledge leads to serious errors.". Kintsch (1991) also states the "the premise of [his work] is that comprehension failures are central to the difficulty of word algebra problems." The general conclusion from the above research is that comprehension rules are key knowledge components students must acquire to become competent problem solvers.

A second hypothesis is the *generalization hypothesis*. According to this hypothesis, symbolization is difficult because students must learn how to use variables to generalize arithmetic procedures..

More recent research by Koedinger and Anderson (1998), and which we confirmed (Heffernan & Koedinger, 1997 and 1998), showed that students could comprehend many problems well enough to find a numerical answer, but they nevertheless failed to correctly symbolize. Although this refutes the comprehension hypothesis it does not refute the generalization hypothesis because the symbolization problems had variables in them. Therefore, we compared students' ability to symbolize a problem that contained a variable (with an answer like "800-40m") to their ability to symbolize a problem with just constants. In the "constants" case the students were asked to write an expression for their answer (i.e. "800-40*3") instead of finding a numerical solution (like "680"). Even if we counted as correct the very few students who did not follow the directions and evaluated the answer, we found that the presence of the variable in the problem did not make problems more difficult. Therefore, the generalization hypothesis was refuted.

So what can explain why symbolization is so difficult? We propose the *articulation hypothesis* which suggests that there is a "hidden" skill that is not obvious to most teachers and researchers. The hidden skill is the ability to produce symbolic sentences in the language of algebra. It appears that many students are able to figure out all the conceptual relations in a problem, but are not able to express those relationships in algebra. If we asked students to translate a story written in English into Greek we would not be surprised if many fail because they don't know Greek. But teachers and researchers often fail to realize that algebra too is a language. And a language that students have had relatively little practice in "speaking" By "speaking" we mean producing sentences of symbols, not verbalizing.

This was demonstrated anecdotally by one of our students who when asked to symbolize a problem with the answer of “(72-m)/4” responded with “72-m=n/4=”. Many commentators have noted that students will incorrectly use an equal sign in a way that makes sense if “=” means “results in.” Sfard et. al. (1993) gives the following example “3*4=12-5=7.” Another example is the student who when working on a problem with an answer of “550/(h-2)” answered with

h-2 à $h\overline{)550}$

This student means to suggest that first she would subtract 2 from “h.” The arrow seems to indicate that this new decremented value of h should be assigned back to the symbol “h”. Then 550 should be divided (indicated with the grade school way of expressing division) by this new value of “h.” Both of these examples indicate students who probably understand the quantitative structure and the sequence of operations that should happen, but nevertheless, failed to express that structure in normative algebra. What does such a student need to learn? A computer scientist or linguist might say that the student needs to learn the correct grammar for algebraic expressions. The novice student knows how to write one-operator expression like “5+7” using the following simple grammar:

```
<expression> = <literal> <operator> <literal>
<literal>  = 1|2|3|4….
<operator>  = “+” | “-“ | “*” | “/”
```

But the competent student knows how to write multiple operator expression indicated by these grammar rules:

```
<expression> = <expression> <operator>
        <expression>
        | “(“ <expression> “)” | <literal>
```

Phrased differently, what the student needs to be told is that “You can always wrap parentheses around an expression and substitute an expression anywhere you normally think a number can go. There are also rules for when you can leave out the parenthesis but you can always put them in to be sure that your expression won’t be misinterpreted.”

We found experimental evidence that supports the articulation hypothesis when we performed the following manipulation (Heffernan & Koedinger, 1997 and 1998). We started with a two-operator problem, like

> **Composed**: Ann is in a rowboat in a lake. She is 800 yards from the dock. She then rows for "m" minutes back towards the dock. Ann rows at a speed of 40 yards per minute. Write an expression for Ann's distance from the dock.

and decomposed the problem into two new separate questions like the following.

> **Decomposed**: A) Ann is in a rowboat in a lake. She is 800 yards from the dock. She then rows "y" yards back towards the dock. Write an expression for Ann's distance from the dock.
>
> B) Ann is in a rowboat in a lake. She then rows for "m" minutes back towards the dock. Ann rows at a speed of 40 yards per minute. Write an expression for the distance Ann has rowed.

Then we compared the ability of a student to answer the composed problem with their ability to get both decomposed parts correct. We found that the composed problems were much harder. Why? We speculated that many students could not compose the two decomposed expressions together; just because you know that you need to first add two quantities together and then multiply them by a number, doesn’t mean you know how to express this correctly in the language of algebra. The following is an example of a student who appeared to be missing just this skill of composing expressions together. This example occurred while the first author was tutoring a student on the following “two-jobs” problem:

T: Debbie has two jobs over the summer. At one job she bags groceries at Giant Eagle and gets paid 5 dollars an hour. At the other job she delivers newspapers and gets paid 7 dollars an hour. She works a total of 30 hours a week. She works "g" hours bagging groceries. Write an expression for the total amount she earns a week. [the correct answer is “5g+7(30-g)”]

S: A=5*g, B=30-g, C=7*B and D=A+C

This student clearly understands the 4 math operations that need to be performed, and the order in which to perform them. This student spontaneously introduced new variables (A, B, C, and D) to stand for the intermediate results. We were surprised to find that this student could not easily put this together and write "5g+7(30-g)". This student appears to be ready for a strategy that will help him on just one skill; combining expressions by substitution. (We also turn this idea into a tutoring strategy which is presented below in the section on *Tutorial Strategies*.)

To see if substitution really is a hidden component skill in symbolization, we designed the following transfer experiment. Thirty-nine students were given one hour of group instruction on algebraic substitution problems like the following:

> Let X= 72-m. Let B= X/4. Write a new expression for B that combines these two steps.

The student were guided in practicing this skill. The students got better at this skill, but that is not the interesting part. By comparing pre-tests and post-tests, we found statistically significant increases in the students ability to do symbolization problems, even though they did not get instruction involving word problems! The students transferred knowledge of the skill of substitution to the skill of symbolization revealing a shared skill of being able to "speak" complicated (more than one-operator) sentences in the foreign language of algebra. This is strong supporting evidence for the articulation hypothesis.

This research has put a new focus on the production side of the translation process. This work also has ramifications for sequencing in the algebra curriculum. If learning how to do algebraic substitution involves a sub-skill of symbolization, perhaps algebraic substitution should be taught much earlier. In many curriculums (e.g. Larson, 1995) it is not taught until students get to *systems of equations* half-way through the year .

Cognitive Student Model

Our student model is similar to traditional student models. We use the Turtle (Anderson & Pelletier, 1991) production system, which is a simplification of the ACT (Anderson, 1993) Theory of Cognition. A production system is a group of if-then rules operating on a set of what are called *working memory elements (wmes)*. We use these rules to model the cognitive steps a student could use to solve a problem. Our student model has 68 production rules. Our production system can solve a problem by being given a set of wme that encodes the problem at a high level.

We model the common errors that students make with a set of "buggy" productions. From our data, we compiled a list of student errors and analyzed what were the common errors. We found that the following list of errors was able to account of over 75% of the errors that students made. We illustrate the errors in the context of the "two-jobs" problem which has a correct answer of "5g+7(30-g)".

1) Wrong operator (e.g. "5g-7(30-g)")
2) Wrong order of arguments (e.g. "5g+7(g-30)")
3) Missing parentheses (e.g. "5g+7*30-g")
4) Confusing quantities (e.g. "7g+5(30-g)")
5) Missing a component (e.g. "5g+7g" or "g+7(30-g)" or "5g+30-g")
6) Omission: correct for a subgoal. (e.g. "7(30-g)" or "5g")
7) Combinations of errors (e.g. "5g+7*g-30" has the wrong order for "g-30" and is missing parenthesis)

These "buggy" productions are used to allow us to make sense of a student's input even if she has made several incorrect steps. We don't want a computer system that can't understand a student if she gives an answer that has parts that are completely correct and parts that are wrong. We want the system to be able to understand as much as possible of what a student says and be able to give positive feedback even when the overall answer to a question might be incorrect.

Traditional model-tracing tutors have a bug message attached to each buggy production that generates a message through the use of a template. We do not do that. We feel such an architecture confuses student reasoning with tutorial reasoning. We instead have the student model report its full diagnosis (which is represented with a set of wmes) to the tutor model that will then decide what to do.

If the student makes several errors, traditional model-tracing tutors are sometimes in a quandary as to what to do. Some ITSs do not deal with multiple bugs and instead rely on breaking down the problem into finer steps. A problem with this approach is that you can't break down a skill like symbolization easily without decreasing the overall difficulty. Another solution is to ask the student what the subgoals should be and then tutor them on the subgoals individually (Corbett & Anderson, 1995.) However, a problem remains about what the ITS should do if the student makes more than one distinct error in a given input. This is addressed below.

The Tutorial Model

As mentioned already, we collected and transcribed one hour of experienced human tutoring. We wanted to observe what experienced tutoring in this domain looked like. The tutor worked as a full time math tutor for over a year before teaching middle school math for 5 years. She was given a list of symbolization problems and told her goal was get the student to learn how to solve such problems.

After transcribing the dialog we have been able to extract some regularities in terms of the tutorial strategies. One caveat: our tutorial model is informed by this observation of human tutoring, but it doesn't model any one individual or make claims to being the most effective model.

Now we will look at the components of the tutorial model shown in Figure 1. A fundamental distinction in the intelligent tutoring system (ITS) is between the student model, which does the diagnosing, and the tutorial models, which chooses the pedagogical plan that best responds to that particular diagnosis. It is composed of a tutorial agenda component, as well as tutorial questions that can be used alone or in combination to make a tutorial strategy. The system currently has 4 tutorial strategies. Through empirical study, we plan to learn which strategies are most effective. The tutorial model is implemented with 77 productions. This approach is similar to Freedman's

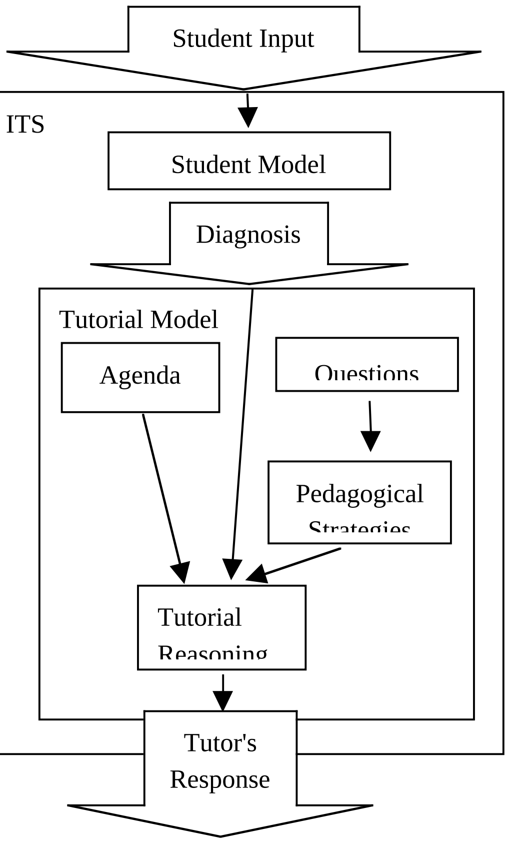


Figure 1: Ms. Lindquist's Architecture

(2000). First, we deal with how Ms. Lindquist decides what to focus problem attention upon.

Dealing with the diagnosis: The Focusing Heuristic

Ms. Lindquist uses a heuristic to decide what to focus the conversation on. In cases when the student model's diagnosis indicates that the student had some correct elements and some incorrect elements. For instance, we considered giving the following positive feedback on an answer like that in line 242 : "Your answer of 's/m+b' has some correct elements; it is true that you need to add the time of the break to the time on the bikes to find the total trip time." This feedback was meant to confirm the "+b" portion of the answer. After looking at what our human tutor did we decided not to give positive feedback unless the student has two operands correct and the correct operator. We give an example of this in the context of the "two-jobs" problem.

T: [problem with answer of 5g+7*(30-g)]

S: 5g+7*g

T: No, but, 5*g does represent the amount Debbie earned bagging groceries. Let me ask you a simpler question. Can you tell me how much she made delivering newspapers?

If the student has made more than one error, the tutor decides to come up with a strategy to deal with each error. The errors are considered in the order they would be encountered in a post-order traversal of the parse tree of the correct answer (i.e visited "bottom-up.") Therefore, the tutor might add multiple questions to the tutorial agenda depending upon the tutorial strategy selected for each error.

If a student says something the student model doesn't understand (e.g. says "5/30-5*7/g" when the answer is "5g+7(30-g)") we will still want a robust ITS to be able to pick a reasonable strategy for a response. This is important because many times the tutor (humans or computers) will not be able to make sense of the student's input. Graesser et. al. (in press) reports in their study of human tutors that they "found that the human tutors and learners have a remarkably incomplete understanding of each other's knowledge base and that many of each other's contributions are not deeply understood… Most tutors have only an approximate assessment of the quality of student contributions." We want our ITS to be able to operate under these same difficult conditions and still be robust enough to say something reasonable.

Tutorial Agenda

Ms. Lindquist has a data structure we called the agenda, that stores the ideas she wants to talk about next. This agenda ordinarily operates like a push down stack, but we give an example of when the stack order is violated below in the section on the Concrete Articulation Strategy.

Tutorial Questions

The tutorial model can ask the following kinds of tutorial questions illustrated with an example of how the question can be phrased:

1) Q_symb : Symbolize a given quantity ("Write an expression for the distance Anne has rowed?")
2) Q_compute: Find a numerical answer ("Compute the distance Anne has rowed?")
3) Q_explain: Write a symbolization for a given arithmetic quantity. This is the articulation step. ("How did you get the 120?")
4) Q_generalize: Uses the results of a Q_explain question ("Good, Now write your answer of 800-40*3 using the variables given in the problem (i.e. put in 'm')")
5) Q_represents_what: Translate from algebra to English("In English, what does 40m represent?" (e.g. "the distance rowed so far"))
6) Q_explain_verbal: Explain in English how a quantity could be computed from other quantities. (We have two forms: The reflective form is "Explain how you got 40*m" and the problem solving form is "Explain how you would find the distance rowed?")
7) Q_decomp: Symbolize a one operator answer, using a variable introduced to stand for a sub-quantity. ("Use A to represent the 40m for the distance rowed. Write an expression for the distance left towards the dock that uses A.")
8) Q_substitute: Perform an algebraic substitution ("Correct, that the distance left is given by 800-A. Now, substitute "40m" in place of A, to get a symbolization for the distance left.")

You will notice that questions 1, 3, 4, and 8 all ask for a quantity to symbolize. Their main difference lies in when those questions are used, and how the tutor responds to the student's attempt. Questions 5 and 6 ask the student to answer in English rather than algebra. To avoid natural language the student is prompted to use pull down menus to complete this sentence "The distance rowed is equal to <noun phrase> <operator> <noun phrase>." The noun phrase menu contains a list of the quantity names for that problem. The operator menu contains "added to", "subtracted from", "multiplied by" and "divided by."

Below we will see how these questions can be combined into strategies.

Tutorial Strategies

We will now look at three different tutorial strategies that Ms Lindquist uses; "Concrete Articulation", "Introduced Variable", and finally

"Explain in English First" (There is a fourth strategy that we don't have space to illustrate. This strategy is that if a students makes an error the tutor tells the student the answer and then asks the student questions to see if the student understands the answer.) The first strategy we will look at is *Concrete Articulation* strategy.

Concrete Articulation Strategy

The concrete articulation strategy usually uses three questions of type Q_compute, Q_explain and Q_generalize. Here is an example of the dialog generated by Ms. Lindquist which we believe parallels the experienced human tutor closely.

T1: [Doing the “bike-trip” problem] Please write an expression for the total time the trip took. [Q_symb]

S1: s/m+b

T2: No. Let me ask you an easier question. Please compute the time actually on bikes if the speed was 20 miles per hour, and the distance ridden was 100 miles? [Q_compute]

S2: 5

T3: Correct. What math did you do to get that five? [Q_explain]

S3: 100/20

T4: Correct. Now write your answer of 100/20 using the variables given in the problem (i.e. put in ‘m’ and ‘s’)? [Q_generalize]

S4: b+m/s

T5: Correct.

We will walk through what Ms. Lindquist did here. First, the student model diagnosed S1 (“s/m+b”) as having the quantity “s/m” with reversed arguments (bug type #2.) The diagnosis also said that the student understood that the total trip time was the time of the break (“b”) plus the time actually on the bikes. This diagnosis was passed to the tutorial model that could have selected among competing strategies. In this case the tutorial model decided to conduct a *concrete articulation* knowledge construction dialog. Therefore, the tutor model generated questions for the *compute, explain/articulation* and *generalization* steps. The tutor put these three questions on the agenda.

At S2, the student answers the compute question correctly, and the tutor decides to continue with its plan. It has to be sensitive to what the student typed. If the student typed 100/20 instead of 5, then both the compute and explain questions would be removed from the agenda and the tutor would skip to the generalize step. An example of this sort of flexibility is demonstrated in T5. T4 was a question to articulate the time actually on bikes. The tutor was hoping the student would type “m/s” but instead the student answered “b+m/s.” Only a pedantic tutor wouldn’t accept that answer as it indicates an understanding that subsumes the understanding required for the sub-goal. This is where the tutorial agenda behaves differently than a stack because Ms. Lindquist pops both of the questions off the agenda.

To show you an example of how the previous situation could have come out differently, lets look at the following dialog

T1: Hello. [Doing the “bike-trip” problem] Please write an expression for the total time the trip took. [Q_symb]

S1: help!

T2: Let me ask you an easier question. Please compute the time actually on bikes if instead the speed was 20 miles per hour, the time of the break was 2 hours, and the distance ridden was 100 miles? [Q_compute]

S2: 7

T3: Correct. What math did you do to get that 7? [Q_explain 7]

S3: 2+5

T4: Okay, but how did you get the 5? [Q_explain 5]

S4: 100/20

T5: Correct Now lets try this again, what math did you do to get the 7? [Q_explain 7]

S5: 2+100/20

T6:Correct. Now write your answer of ‘2+100/20’ using the variables given in the problem (i.e. put in ‘m’, ‘b’ and ‘s’)? [Q_generalize]

S6: b+m/s

T7: Correct.

This example is interesting in the way Ms. Lindquist responds at T4. The system has bug rules to apply to questions other than symbolization questions. In this case, the student model reported that the student's answer of "2+5" is a partially complete articulation but that the student didn't spell out where the "5" came from. So the tutorial model is able to "dig-down" into the student's reasoning with an embedded sub-dialog. This involved adding the additional question of explaining how the student got the "5".

Introduced Variable Strategy

A second example of a tutorial strategy is what we call the *Introduced Variable* strategy, which was alluded to above when we looked at the student who answered the "two-jobs" problem with "A=5*g, B=30-g, C=7*B and D=A+C". Below we present an real example of Ms. Lindquist tutoring an ninth grader with this strategy. This strategy uses both Q_decomp and Q_substitute questions. Question Q_decomp sets the stage for this strategy, and is used to break down the originally composed problem into separate decomposed one-operator problems (as shown below in T2 and T3), introducing new variables when appropriate (as in T3.) The real meat of this strategy is shown at line 4, where the tutor introduces the variable "A" to stand for a sub-quantity. Question Q_substitute is used to scaffold the substitution step as shown in T5.

T1: Hello. Jane is "j" years old. Peg is "p" years old. Mary is Peg's age minus Tom's age. Tom is 3 years younger than Jane. Peg is "x" years older that Mark. Write an expression for Mary's age. [Q_symb]

S1:(p-3)-(3-j)+(p+x) [correct answer is "p-(j-3)"]

T2:Hmm. No. Let me ask you and easier question. Use j and 3 to calculate Tom's age. [Q_decomp]

S2:3-j

T3:Hmm. No. Try again please. What is Tom's age? Hint: pick one of these: j-3, 3-j, j+3, j*3, j/3, or 3/j. [This is a rephrasing as explained in the hint section below]

S3:j-3

T4:Correct. Calculate Mary's age. Use A instead of j-3 for Tom's age. [Q_decomp]

S4:p-a

T5:Correct. You are correct that p-A is Mary's age. Now, can you state this in terms of the numbers given in the problem. [Q_substitute]

S5:p-(j-3)

T6:Correct.

We note that this strategy is not one inspired by looking at human tutors but by our cognitive analysis that informed us that a crucial difficulty occurs when a student has to treat an expression (i.e., "j-3") like a number, so this strategy breaks that hard step into two steps (i.e., T4 and T5).

"Explain in English" Strategy

Mathematical communication is increasingly seen as a goal for students, and in particular translating between an algebraic representation, an English verbal representation, and a concrete representation of numerical instances (Schoenfeld et. al., 1992, Koedinger & Nathan) Other researchers are also interested in wanting an ITS to elicit explanations from students (e.g. Conati et. al., 1997.) We noticed in the dialog we collected from an experienced human tutor that she would often ask the student to explain how to compute a quantity in English and afterward, ask the student to say it in symbols. (Heffernan, 2001). Therefore, we have implemented similar questions in the form of Q_represents_what and Q_explain_verbal as shown below.

T1: Hello. [Doing the "bike-trip" problem] Please write an expression for the total time the trip took. [Q_symb]

S1: m/s

T2: What does "m/s" represent (e.g. the speed on the bike?)[Q_represents_what]

S2: the total time of the trip

T3: No, "m/s" represents the actual time on the bikes. Can you explain how you would find the total time of the trip using the time on the bikes? [Q_explain_verbal]

S3: The total time of the trip is equal to [1st menu] "the amount of time for the break" [2nd menu] "plus" [3rd menu] "the actual time on the bikes". [Composed using three pull down menus.]

T4: Good, now say what the total trip time is in symbols [Q_symb]

S4: b+m/s

T5: Correct

This strategy is based on the hypothesis that students will learn more if they are asked to explain in English how to compute a quantity.

Hints

Giving the students a hint is the simplest tutorial strategy we have and is a common way that a traditional model-tracing tutors gives help. Each question has a *hint chain* which is simply a list of rephrasings of the question. The rephrasings get progressively more explicit, eventually, possibly, telling the student exactly what to do[3].

Formative Evaluation

We have performed a early formative analysis of Ms. Lindquist, by comparing her to a traditional computer-aided instruction(CAI) control condition that simply told students the answer (if they got the problem wrong). This study answers the question of whether the dialogs Ms. Lindquist has with students lead to increased student learning, but ignores the question of whether it is worth the additional time. That is because we controlled for the number of problems, and not the amount of time. Students in the condition that received Ms. Lindquist were expected to take more time.

Procedure

The procedure and the results are more fully explicated in Heffernan's dissertation (2001). In summary, 20 high school students participated in the study. They were given pretests and posttests on symbolization. They were blocked into groups according to their pretests. Students proceeded though the curriculum consisting of 68 problems. Students in the experimental condition received a partially random combination of the different strategies Ms. Lindquist is capable of.

[3] The question of how explicit to make the “bottom-out”, or last hint (i.e. “Do you tell the student the answer?”) is not a question that we, or this architecture, makes a commitment about. Some of our hint chains end with multiple choice questions while others tell the student exactly what to type.

Results

The students in the control condition, as expected, took half the amount of time, since they did not engage in dialogs with Ms. Lindquist. Students performed better on a pretest after having just received instruction from Ms. Lindquist. We performed a ANOVA that showed only a marginally statically significant effect (F(1,37)=2.6,p=.12) but with a respectable effect size of .7 standard deviations.

Discussion

Now that we have established that this dialogs appear to lead to better learning, our next experiment will control for "time on task" in order to determine if the additional time required to have a dialog is worth it.

The following example from this experiment nicely illustrates some of the aspects of Ms. Lindquist, including embedded dialogs (starting on lines 4 and 10), as well as the embedded feedback shown at the beginning of line 8. The number of seconds it takes the student to respond is in square brackets.

1) Problem: Jane is "j" years old. Peg is "p" years old. Mary is Peg's age minus Tom's age. Tom is 3 years younger than Jane. Peg is "x" years older than Mark. Write an expression for Mary's age
2) Tutor: Hello. Please write an expression for Mary's age.
3) Student [143]: **Student hit the help button** *[should be p-(j-3)]*
4) Tutor: Let me try to give you some help. We will come back to Mary's age. Let me try to break this down for you. Please compute Mary's age if Jane's age was "20" years, and "Peg's age" was "40" years?
5) Student: [169]: **23** *[correct]*
6) Tutor: Correct. Using only 40, 20, and 3 write an expression for Mary's age.

7) Student [126]: **(20-3)-40** *[should be 40-(20-3)]*
8) Tutor: No. Your answer of "(20-3)-40" equals "-23", but you wanted to write an expression that equaled "23". Using only 40, 20, and 3 write an expression for Mary's age. Hint: Use 40 for Peg's age and 20-3 for Tom's age.
9) Student [47]: **(40-20)+3** *[should be 40-(20-3)]*
10) Tutor: Hmm. No. Let me ask you a different question. Using only 20, and 3 write an expression for Tom's age.
11) Student [15]: **20-3** *[correct]*
12) Tutor: Correct. Now, lets try this question again: Using only 40, 20, and 3 write an expression for Mary's age. Hint: Use 40 for Peg's age and 20-3 for Tom's age.
13) Student [138]: **40-(20-3)** *[correct]*
14) Tutor: Correct. Now, say that again but this time use the variables from the problem.
15) Student [25]: **p-(j-3)** *[correct]*
16) Tutor*:* Correct

Notice that this example shows a student how could compute an instance (line 5), but had trouble articulating how to write the complete expression (lines 7, and 9), which is consistent with what our cognitive analysis predicted about the relatively difficulty of these tasks. Also consistent is the fact that using a variable, per se, does not appear to have been difficult (line 15). This students performance is consistent with our theory that *articulation*, and not comprehension of word problems or variables per se is what makes symbolizing difficult for students.

Conclusion

McArthur et. al. criticized Anderson's et. al. (1985) model-tracing ITS and model-tracing in general "because each incorrect rule is paired with a particular tutorial action (typically a stored message), every student who takes a given step gets the same message, regardless of how many times the same error has been made or how many other error have been made. … Anderson's tutor is tactical, driven by local student errors (p. 200)" and goes on to argue for the need for a more strategic tutor. Ms. Lindquist meets that criticism. Ms. Lindquist's model of tutorial reasoning is both strategic (i.e. has multi-step plans) and tactical (i.e. reasons to produce output at the single question level.) She also intelligently handles multiple errors and reasons about the order in which to deal with them and then constructs a plan to deal with each of them. Ms. Lindquist is a modest step on the path to making a more dynamic tutor.

We have released Ms. Lindquist onto the web at www.AlgebraTutor.org, and have had over 600 students who have been tutored by Ms. Lindquist, the results of which are now in preparation. In addition she has won various industry awards from teacher related web sites such as *USAToday Education* and the National Council of Teachers of Mathematics. Ms. Lindquist is a system that combines the student modeling of traditional model-tracing tutors with a model of tutorial dialog based on an experienced human tutor. Early analysis reveals Ms. Lindquist can be effective, but more analysis is needed to determine where the biggest "bang for the buck" is to be found.

Acknowledgements

This research was supported by NSF grant number 9720359 to CIRCLE and the Spencer Foundation.

References

Anderson, J. R. (1993). *Rules of the Mind*. Hillsdale, NJ: Erlbaum.

Anderson, J. R., Boyle, D. F., & Reiser, B. J. (1985). Intelligent tutoring systems. *Science*, 228, 456-462.

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995) Cognitive tutors: lessons learned. *The Journal of the Learning Sciences*, 4 (2), 167-207.

Anderson, J. R. & Pelletier, R. (1991) A developmental system for model-tracing tutors. In Lawrence Birnbaum (Eds.) *The International Conference on the Learning Sciences*. Association for the Advancement of Computing in Education. Charlottesville, Virginia (pp. 1-8).

Clancey, W. J., (1982) Tutoring rules for guiding a case method dialog. In D. Sleeman & J. S. Brown (Eds.) *Intelligent Tutoring Systems* London: Academic Press. (pp. 201-226.)

Corbett, A. T., and Anderson, J. R., (1995) Knowledge decomposition and subgoal reification in the ACT programming tutor. in *Proceedings of Artificial Intelligence in Education* (pp. 469-476)

Conati, C., Larkin, J. and VanLehn, K. (1997) A computer framework to support self-explanation. In : du Bolay, B. and Mizoguchi, R.(Eds.) Proceedings of AI-ED 97 World Conference on Artificial Intelligence in Education. Vol.39, pp. 279-276, Amsterdam: IO Press.

Freedman, R. (2000) Using a reactive planner as the basis for a dialogue agent. In *Proceedings of the Thirteenth Florida Artificial Intelligence Research Symposium* (FLAIRS '00), Orlando.

Graesser, A.C., Wiemer-Hastings, P., Wiemer-Hastings, K., Harter, D., Person, N., & the TRG (in press). Using latent semantic analysis to evaluate the contributions of students in AutoTutor. Interactive Learning Environments.

Heffernan, N. T. (2001). *Intelligent Tutoring Systems have Forgotten the Tutor: Adding a Cognitive Model of an Experienced Human Tutor*. Dissertation. Carnegie Mellon University, Computer Science Department. http://gs260.sp.cs.cmu.edu/diss

Heffernan, N. T., & Koedinger, K. R.(1997) The composition effect in symbolizing: the role of symbol production versus text comprehension. *Proceeding of the Nineteenth Annual Conference of the Cognitive Science Society* 307-312. Hillsdale, NJ: Erlbaum.

Heffernan, N. T., & Koedinger, K. R. (1998) A developmental model for algebra symbolization: The results of a difficulty factors assessment. In Proceedings of the Twentieth Annual Conference of the Cognitive Science Society, (pp. 484-489). Hillsdale, NJ: Erlbaum.

Kintsch, W. (1991). A theory of discourse comprehension: Implications for a tutor for word algebra problems. In *Learning and instruction: European research in an international context*. M. Carretero, M. L. Pope and et al. Oxford, England UK, Pergamon Press. 3**:** 235-253.

Koedinger, K. R., Anderson, J.R., Hadley, W.H., & Mark, M. A. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education, 8,* 30-43.

Koedinger, K. R., & Anderson, J. R. (1998). Illustrating principled design: The early evolution of a cognitive tutor for algebra symbolization. In *Interactive Learning Environments*, 5, 161-180.

Koedinger, K. R., & MacLaren, B. (1997). Implicit strategies and errors in an improved model of early algebra problem solving. In Proceedings of the Nineteenth Annual Meeting of the Cognitive Science Society (pp. 382-7). Mahwah, NJ: Erlbaum.

Koedinger, K. R. & Nathan, M. J. (submitted to). The real story behind story problems: Effects of representations on quantitative reasoning. Submitted to *Cognitive Psychology*.

Larson, R., Kanold, T., & Stiff, L. (1995) *Algebra 1: An Integrated Approach.* D.C. Heath. Lexington, MA.

Lewis, A. B. & Mayer, R. E. (1987). *Journal of Educational Psychology, 79(4),* 363-317.

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

Moore, J. D. (1996) Discourse generation for instructional applications: Making computer-based tutors more like humans. *Journal of Artificial Intelligence in Education*, 7(2), 118-124

Nathan, M. J., Kintsch, W. & Young, E. (1992). A theory of algebra-word-problem comprehension and its implications for the design of learning environments. *Cognition & Instruction 9* (4): 329-389.

Sfard, A., & Linchevski, L. (1993). The gain and the pitfalls of reification- the case of algebra. *Educational Studies in Mathematics*, 00: 1-38.

Schoenfeld, A., Gamoran, M., Kessel, C., Leonard, M., Or-Bach, R., & Arcavi, A. (1992) Toward a comprehensive model of human tutoring in complex subject matter domains. *Journal of Mathematical Behavior*, 11, 293-319

Shulman, L. (1986). Those who understand: Knowledge growth in teaching. *Educational Researcher, 15*, 4-14.

VanLehn, K, Anderson, J., Ashley, K., Chi. M., Corbett, A., . Koedinger, K., Lesgold, A., Levin, L., Moore, M., and Pollack, M., NSF Grant 9720359. *CIRCLE: Center for Interdisciplinary Research on Constructive Learning Environments*. NSF Learning and Intelligent Systems Center. January, 1998 to January, 2003.

A Decision-Theoretic Architecture for Selecting Tutorial Discourse Actions

R. Charles Murray
Intelligent Systems Program
University of Pittsburgh
Pittsburgh, PA 15260
rmurray@pitt.edu

Kurt VanLehn
LRDC
University of Pittsburgh
Pittsburgh, PA 15260
vanlehn@pitt.edu

Jack Mostow
Project LISTEN
Carnegie Mellon University
Pittsburgh, PA 15213
mostow@cs.cmu.edu

Abstract

We propose a decision-theoretic architecture for selecting tutorial discourse actions. *DT Tutor*, an action selection engine which embodies our approach, uses a dynamic decision network to consider the tutor's objectives and uncertain beliefs in adapting to the changing tutorial state. It predicts the effects of the tutor's discourse actions on the tutorial state, including the student's internal state, and then selects the action with maximum expected utility. We illustrate our approach with prototype applications for diverse target domains: calculus problem-solving and elementary reading. Formative off-line evaluations assess DT Tutor's ability to select optimal actions quickly enough to keep a student engaged.

1 Introduction

A tutoring system achieves many of its objectives through discourse actions intended to influence the student's internal state. For instance, a tutor might tell the student a fact with the intended effect of increasing the student's knowledge and thereby enabling her to perform a problem-solving step. The tutor might also be concerned with the student's goals, focus of attention, and affective or emotional state, among other internal attributes. However, a tutor is inevitably uncertain about the student's internal state, as it is unobservable. Compounding the uncertainty, the student's state changes throughout the course of a tutoring session–after all, that is the purpose of tutoring. To glean uncertain information about the student, a tutor must make inferences based on observable actions and guided by the tutor's beliefs about the situation. The tutor is also likely to be concerned with observable attributes of the tutoring situation, or tutorial state, including the discourse between tutor and student and their progress at completing tutorial tasks (e.g., solving problems).

The tutor's actions depend not only on the tutorial state, but also on the tutor's objectives. Tutorial objectives often include increasing the student's knowledge within a target domain, helping the student solve problems or complete other tasks, and bolstering the student's affective state (Lepper et al., 1993). Tutors also generally want to be cooperative discourse partners by coherently addressing topics that are relevant to the student's focus of attention. Objectives and priorities may vary by tutor and even for an individual tutor over time. Furthermore, tutors must often strike a "delicate balance" among multiple competing objectives (Merrill et al., 1992, p. 280).

To model the tutor's uncertainty about the student's internal state, probabilistic reasoning is becoming increasingly common. However, almost all probabilistic tutoring systems still model the tutor's objectives implicitly at best, and use heuristics to select tutorial actions. *DT Tutor* uses a decision-theoretic approach to select tutorial actions, taking into account both the tutor's uncertain beliefs and multiple objectives regarding the changing tutorial state. This paper describes DT Tutor's approach along with prototype applications for diverse domains, calculus problem-solving and elementary reading.

2 General Approach

2.1 Belief and Decision Networks

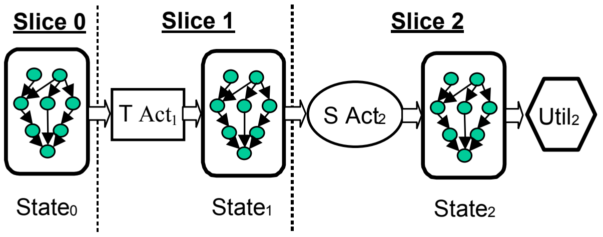


Figure 1. Tutor Action Cycle Network, overview

DT Tutor represents the tutor's uncertain beliefs in terms of probability using Bayesian belief networks. A belief network is a directed acyclic graph with *chance nodes* representing beliefs about attributes and *arcs* between nodes representing conditional dependence relationships among the beliefs. Beliefs are specified in terms of probability distributions. DT Tutor's chance nodes represent the tutor's beliefs about the tutorial state. For each node with incoming arcs, a conditional probability table specifies the probability distribution for that node conditioned on the possible states of its parents. For nodes without incoming arcs, prior probability distributions are specified.

At any particular time, each node within a belief network represents an attribute whose value is fixed. For an attribute whose value may change over time (such as a tutorial state attribute), separate nodes can be used to represent each successive value. Dynamic belief networks do just that. For each time in which the values of attributes may change, a dynamic belief network creates a new *slice*. Each slice is of a set of chance nodes representing attributes at a specific point in time. For tutoring, slices can be chosen to represent the tutorial state after a tutor or student action, when attribute values are likely to change. Nodes may be connected to nodes within the same or earlier slices to represent the fact that an attribute's value may depend on (1) concurrent values of other attributes and (2) earlier values of the same and other attributes.

Decision theory extends probability theory to provide a normative theory of how a rational decision-maker should behave. Quantitative utility values are used to express preferences among possible outcomes of actions. To decide among alternative actions, the expected utility of each alternative is calculated by taking the sum of the utilities of all possible outcomes weighted by the probabilities of those outcomes occurring. Decision theory holds that a rational agent should choose the alternative with maximum expected utility. A belief network can be extended into a decision network (equivalently, an influence diagram) by adding decision and utility nodes along with appropriate arcs. For DT Tutor, decision nodes represent tutorial action alternatives, and utility nodes represent the tutor's preferences among the possible outcomes.

A dynamic decision network (DDN) is like a dynamic belief network except that it has decision and utility nodes in addition to chance nodes. DDNs model decisions for situations in which decisions, attributes or preferences can change over time. The evolution of a DDN can be computed while keeping in memory at most two slices at a time (Huang et al., 1994).

2.2 General Architecture

DT Tutor's action selection engine uses a DDN formed from dynamically created tutor action cycle networks (TACNs). A TACN consists of three slices, as illustrated in Figure 1. The tutorial state (*State$_s$*) within each slice is actually a sub-network representing the tutor's beliefs about the tutorial state at a particular point in time (slice)[1]. The *T Act$_1$* decision node represents the tutorial action decision, the *S Act$_2$* chance node represents the student turn following the tutor's action, and the *Util$_2$* utility node represents the utility of the resulting tutorial state.

Each TACN is used for a single cycle of tutorial action, where a cycle consists of deciding a tuto-

[1] For sub-network and node names, a numeric subscript refers to the slice number. A subscript of *s* refers to any appropriate slice.

rial action and carrying it out, observing the subsequent student turn, and updating the tutorial state based on the tutor and student actions. During the first phase (deciding upon a tutorial action), slice 0 represents the tutor's current beliefs about the tutorial state. Slice 1 represents the tutor's possible actions and predictions about their effects on the tutorial state. Slice 2 represents a prediction about the student's next turn and its effect on the tutorial state. The DDN update algorithm calculates which tutorial action has maximum expected utility.

In the next phase of the cycle, the tutor executes that action and waits for the student response. The tutor then updates the network based on the observed student action(s).

At this point, the posterior probabilities in $State_2$ represent the tutor's current beliefs. It is now time to select another tutor action, so another TACN is created and the DDN is rolled forward: Posterior probabilities from $State_2$ of the old TACN are copied as prior probabilities to $State_0$ of the new TACN, where they represent the tutor's current beliefs. The old TACN is discarded. The tutor is now ready to begin the next cycle by deciding which action to take next.

With this architecture, the tutor not only reacts to past student actions, but also anticipates future student actions and their ramifications. Thus, for instance, it can act to prevent errors and impasses before they occur, just as human tutors often do (Lepper et al., 1993).

In principle, the tutor can look ahead any number of slices without waiting to observe student actions. The tutor simply predicts probability distributions for the next student turn and the resulting $State_2$, rolls the DDN forward, predicts the tutor's next action and the following student turn, and so on. Thus, the tutor can select an optimal sequence of tutorial actions for any fixed amount of look ahead. However, a large amount of look ahead is computationally expensive with decreasing predictive accuracy.

3 Application Domains

3.1 Calculus Problem-Solving

CTDT (Calculus Tutor, Decision-Theoretic) is a prototype action selection engine for calculus related rates problems (Murray & VanLehn, 2000). Singley (1990) developed a tutoring system for this domain with an interface designed to make student problem-solving actions observable, including goal-setting actions that are normally invisible. CTDT presumes an extension to Singley's interface to make all problem-solving actions observable. This makes it easier to select tutorial actions for two reasons. First, as each problem-solving action is executed through the interface, CTDT has the opportunity to intervene. (However, CTDT can select a *null* action on its turn and thus allow the student to execute multiple actions without tutorial intervention). This means that CTDT can select a response for only a single student action per turn, rather than deciding which of multiple student actions to respond to. Moreover, it is easier to predict a single student action per turn than to predict a combination of multiple actions.

Second, when CTDT can observe all of the student's prior actions, it knows exactly what portion of the problem solution space the student had already completed and thus what steps the student is likely to attempt next. Calculus related rates problems, like problems in many other domains, have a prerequisite structure that induces a partial order in which problem steps may be completed – for instance, the chain rule (e.g., $dx/dy * dy/dz = dx/dz$) cannot be applied until the component equations are in the required form. The student is unlikely to be able to successfully complete problem steps for which prerequisites have not been completed, and is therefore less likely to attempt them. The student is also unlikely to repeat problem-solving steps that have already been completed successfully. This means that the student is most likely to attempt problem steps that (1) have not already been completed, and (2) have no uncompleted prerequisite steps. We call these *ready* steps. Thus, by observing which steps the student has already completed, CTDT can easily determine the set of *ready* steps that the student is most likely to attempt next.

Even so, predicting the next student action is still not trivial, since there may be more than one way to solve a calculus related rates problem (i.e., more than one *solution path*), and there may be multiple orders in which the steps of a solution path can be executed.

3.2 Project LISTEN's Reading Tutor

RTDT (Reading Tutor, Decision-Theoretic) is a prototype action selection engine for Project LISTEN's Reading Tutor, which uses mixed-initiative spoken dialogue to provide reading help for children as they read aloud (Mostow & Aist, 1999). The Reading Tutor has helped to improve the reading of real students in real classrooms (Mostow & Aist, in press). It displays one sentence at a time for the student to read, and a simple animated persona that appears to actively watch and patiently listen. As the student reads, the Reading Tutor uses automated speech recognition to detect when the student may need help, which it provides using both speech and graphical display actions. Thus, the Reading Tutor already has an extensively developed interface. This is in contrast to CTDT, for which we assumed an interface built to our specifications. Inter-operability with existing tutoring systems is a key to extending the applicability of DT Tutor's approach.

RTDT models some of the Reading Tutor's key tutorial action decisions in just enough depth to determine the feasibility of applying DT Tutor to this domain. We targeted two types of unsolicited help: (1) *preemptive help* before the student attempts a sentence, and (2) *corrective feedback* after the student has stopped reading (whether or not the student has completed the sentence). The Reading Tutor provides preemptive help when it believes that the student is likely to misread a word, and corrective feedback when it detects words read incorrectly, skipped words and disfluent reading. To avoid disrupting the flow of reading, the Reading Tutor ignores errors on a list of 36 common function words (e.g., *a, the*) that are unlikely to affect comprehension. For the Reading Tutor's corpus of readings, approximately two-thirds of the words in a sentence are non-function words, or *content* words.

Tutoring reading differs enough from coaching calculus problem-solving to pose challenges for adapting DT Tutor's approach. First, student turns may consist of multiple reading actions, where each action is an attempt to read a word. Therefore, in contrast to CTDT, RTDT must predict and respond to multiple student actions per turn. Student turns may indeed include multiple actions in many target domains, so meeting this challenge is important for extending DT Tutor's generality.

Second, beginning readers often make repeated attempts at words or phrases and sometimes omit words, with the effect of jumping around within a sentence. Even when jumping around, a student may be able to read each individual word. Thus, the order in which beginning readers attempt words is not always sequential, and has very little prerequisite structure. This means that the set of actions that the student is likely to attempt next is less constrained than with CTDT, posing a challenge for predicting the student's next turn. A similar challenge must be faced for tutoring in any target domain with weak structure for the order in which actions may be completed.

4 Tutor Action Cycle Networks in More Detail

4.1 TACN Components

Figure 2 provides a closer look at the major TACN components and their interrelationships. The *State*$_s$ representation in each slice actually consists of several sub-networks. These include the *Knowledge*$_s$, *Focus*$_s$, and *Affect*$_s$ sub-networks which compose the student model, and the *Task Progress*$_s$ and *Discourse State*$_s$ sub-networks. Arcs between corresponding sub-networks in dif-

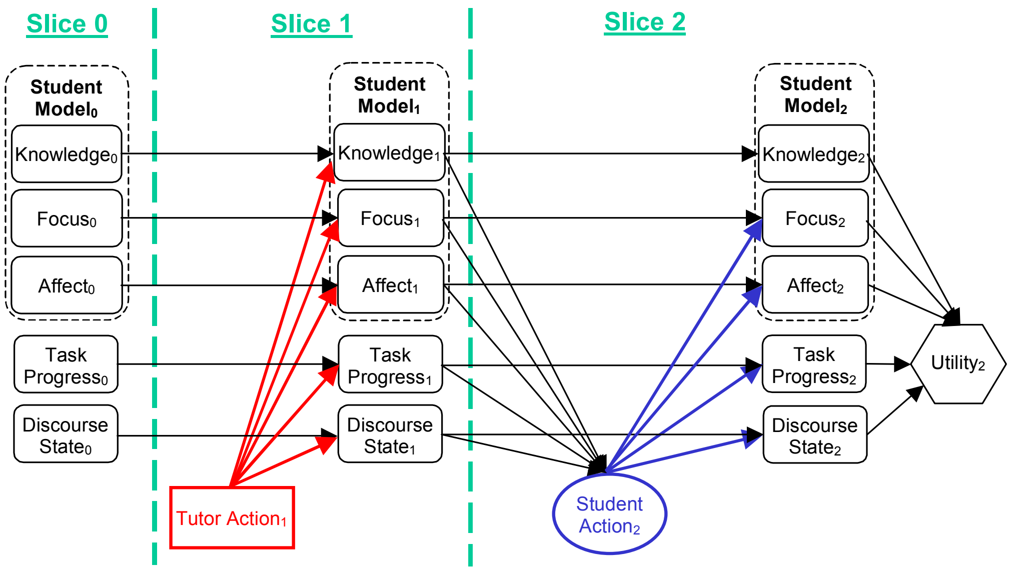


Figure 2. TACN architecture in more detail

ferent time slices represent the stability of attributes over time. For instance, the student's knowledge in slice 1, *Knowledge*$_1$, is likely to be about the same as the student's knowledge in slice 0, *Knowledge*$_0$, except as influenced by the tutor's action, *Tutor Action*$_1$.

The architecture shown in Figure 2 is generic. Depending on the needs of the application, fewer or more components may be required. For instance, the initial implementation of the RTDT prototype lacks a model of the student's affective state because we focused on modeling other tutorial state attributes, such as multiple student actions per turn. Therefore, its TACNs do not include the *Affect*$_s$ sub-networks. However, RTDT also has *Tutor Efficacy*$_s$ sub-networks to model the efficacy of the various tutorial help alternatives. The *Tutor Efficacy*$_s$ sub-networks dynamically tune RTDT's model of the effects of the tutor's actions on the student's knowledge, helping RTDT to avoid repeating ineffective tutorial actions and reducing the need for accurate conditional probabilities regarding the influence of *Tutor Action*$_1$ on *Knowledge*$_1$.

Selected components are described below along with illustrations from CTDT and RTDT.

4.1.1 *Tutor Action*$_1$ Nodes

The purpose of the TACN is to compute the optimal alternative for *Tutor Action*$_1$, which may consist of one or more decision nodes. For CTDT, *Tutor Action*$_1$ consists of two decision nodes, one to specify the *topic* of the tutor action and one to specify the action *type*. The action *topic* is the problem-related focus of the action, such as a problem step or related rule in the target domain. The *type* is the manner in which the topic is addressed, including *prompt, hint*, *teach*, *positive* or *negative feedback*, *do* (tell the student how to do a step) and *null* (no tutor action).

For RTDT, *Tutor Action*$_1$ is currently a single decision node with values *null* (no tutor action), *move_on* (move on to the next sentence), *read_move_on* (read the sentence to the student and then move on), *hint_sentence* (e.g., read the current sentence to the student), and *hint_word_i* for each content word *i* in the current *n*-content-word sentence, $i = \{1, 2, ..., n\}$. The *hint_sentence* and *hint_word_i* alternatives specify the *topic* but not the *type* of the tutorial action – e.g., they don't specify whether the Reading Tutor should hint about a particular word by saying the word itself or by giving a rhyming hint. Deciding among action type alternatives would require information than was not available for the prototype implementation. For instance, information about the student's knowledge of the letter-sound mappings pertinent to a particular word would help RTDT determine the likelihood that a rhyming hint would supply the required knowledge.

CTDT considers tutoring only on *ready* problem steps and related rules, plus the step that the student has just completed (e.g., to give *positive* or *negative feedback*). RTDT considers every action alternative for preemptive help, including hinting on each content word. However, for fast response time on corrective feedback, RTDT does not consider hinting on words that the student has already read correctly, because such hints are less likely to be pedagogically productive.

4.1.2 Student Model $\textit{Knowledge}_s$ Sub-Network

The $\textit{Knowledge}_s$ sub-network represents the tutor's beliefs about the student's knowledge related to the target domain. Each $\textit{Knowledge}_s$ node has possible values *known* and *unknown*. For CTDT, the student's knowledge related to each problem is represented in a belief (sub-)network whose structure is obtained directly from a *problem solution graph*. See Figure 3 for an example. The top two rows of nodes in the figure represent rules licensing each problem step. The remaining nodes represent problem steps, from the givens (the goal *Find dx/dz for z=c* and the facts $x=ay^b$, $y=ez^f$ and $z=c$) through each goal-setting and fact-finding step in all solution paths (this example has only one solution path) until the answer is found ($dx/dz=bay^{b-1}fec^{f-1}$). Arcs represent dependence between nodes. For instance, knowledge of a step depends on knowledge of both its prerequisite steps and the rule required to derive it.

For RTDT, $\textit{Knowledge}_s$ includes nodes to represent the student's knowledge of how to read each content word and the sentence. For each content word *i*, a $\textit{Know\_Word\_i}_s$ node represents the student's knowledge of how to read the word. A $\textit{Know\_Sentence}_s$ node represents the student's knowledge of how to read the sentence as a whole.

In slice 1, each $\textit{Knowledge}_1$ node is influenced by the tutor's action. For instance, a tutorial hint about a particular problem step or word increases the probability that the node corresponding to the knowledge element is *known*. After the student turn has been observed, $\textit{Knowledge}_1$ is updated diagnostically to reflect its causal role in the success of the student's action(s).

$\textit{Knowledge}_2$ is not directly influenced by the student's turn because student actions generally do not influence student knowledge without feedback (e.g., by the tutor). Instead, $\textit{Knowledge}_2$ is influenced by $\textit{Knowledge}_1$, which is diagnostically influenced by the student's turn.

4.1.3 Student Model $\textit{Focus}_s$ Sub-Network

The $\textit{Focus}_s$ sub-network represents the student's focus of attention within the current tutorial task. For CTDT, the focus may be any problem step, so $\textit{Focus}_s$ has the same problem solution graph structure as $\textit{Knowledge}_s$. *Ready* steps are most likely to be in focus. Nodes representing these steps have some distribution over the values *ready* and *in_focus*, where *in_focus* means that the step is in the student's focus of attention. Consistent with a human depth-first problem-solving bias (Newell & Simon, 1972), any such steps that are in the student's current solution path are most likely to be *in_focus*. *Focus aging* is also modeled: the probability that an uncompleted step is *in_focus* attenuates with each passing time slice as other problem steps come into focus.

For RTDT, $\textit{Focus}_s$ models the likelihood of each content word being the first word in the student's focus of attention. $\textit{Focus\_Word\_i}_s$ nodes for each content word *i* in the current sentence have possible values *in_focus* and *out_of_focus*, where *in_focus* means that the word is the first content word in the student's focus of attention.

In slice 1, each $\textit{Focus}_1$ node is influenced by the tutor's action. For instance, if the tutor hints about a problem step or word, the corresponding node is likely to be *in_focus*. For RTDT, a tutor hint about the sentence as a whole increases the probability that the student will attempt to read

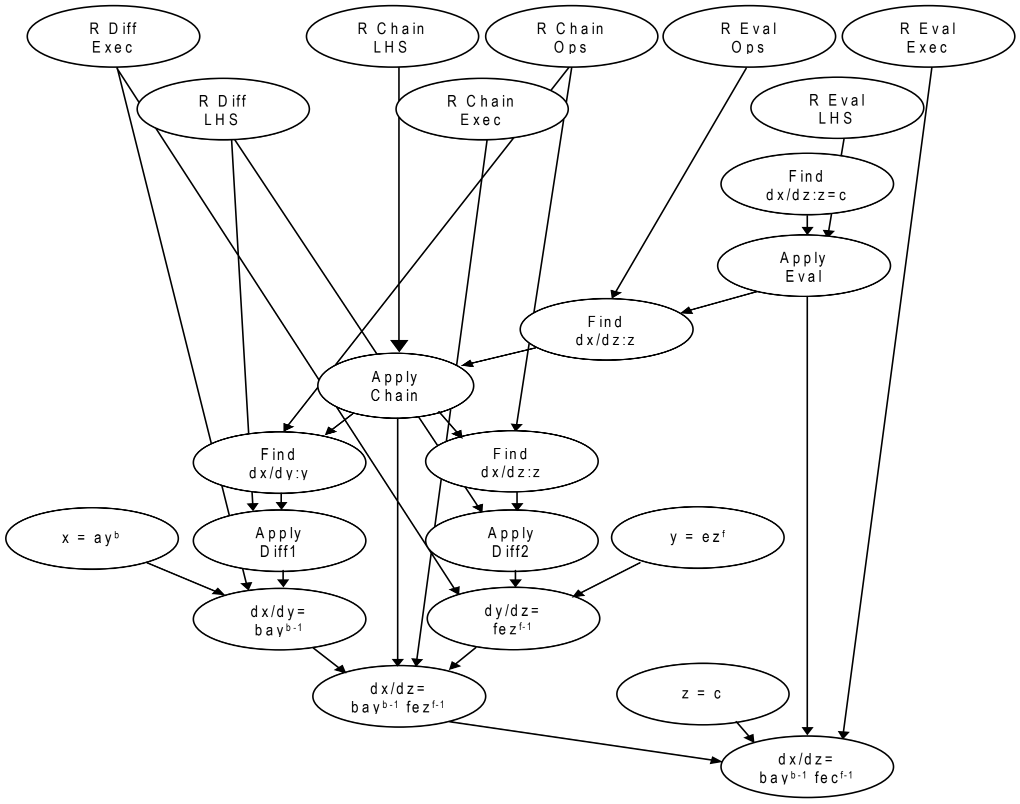


Figure 3. Problem solution graph for CTDT

the entire sentence (starting with the first word), increasing the probability that *Focus_Word_1*$_1$ is *in_focus*. In slice 2, the student action influences the tutor's beliefs about the student's focus of attention (in *Focus*$_2$). For instance, if the student experiences an impasse on a problem step or a word, the corresponding node is more likely to be *in_focus*.

4.1.4 *Student Action*$_2$ Nodes

These nodes represent one or more actions taken on the student's turn. For CTDT, a single student action is assumed. This action is represented by two nodes, one for the action *topic* and another for the action *type*. The action *topic* may be any problem step and the action *type* may be *correct*, *error*, *impasse*, or *null* (no student action).

For RTDT, the student turn may include multiple reading actions, where each action is an attempt to read a word. Student action *Word_i*$_2$ nodes represent the student's reading of each content word *i* as *not_read*, *error*, or *correct*. This representation models student turns ranging from no productive attempt (all words *not_read* – e.g., a silent impasse), to all words read correctly (all words *correct*), to any combination of words *not_read*, read in *error*, and read *correct*ly. In addition, a student action *Sentence*$_2$ node models the student's reading of the sentence as a whole as either *fluent* or *disfluent*.

Both CTDT and RTDT probabilistically predict the next student action. For CTDT, *Focus*$_1$ influences the student action *topic*. Given the action *topic*, whether the action *type* will be *correct*, *error* or *impasse* depends on the student's knowledge. Therefore, both the student action *topic* and *Knowledge*$_1$ influence the student action *type*.

For RTDT, influences on each *Word_i*$_2$ node from the corresponding *Focus_Word_i*$_1$ node probabilistically predict which word the student will attempt first. For any word that the student attempts, an influence from the corresponding *Know_Word_i*$_1$ node predicts whether the reading will be in *error* or *correct*. We assume that if a student reads one word correctly, she is most likely to attempt the next word, and so on, until she gets stuck or makes an error. Therefore, arcs from each node *Word_i*$_2$ to node *Word_i+1*$_2$, $i = \{1, 2, ..., n-1\}$, model the influence of reading word *i* correctly on the likelihood that the student will attempt word *i+1*. For a *fluent* reading of the sentence, each word must be *correct* without pauses in between – i.e., the student must be able to read each word and the sentence as a whole. The *Sentence*$_2$ node is therefore influenced by each *Word_i*$_2$ node and by the *Know_Sentence*$_1$ node.

4.1.5 *Discourse State*$_s$ Sub-Network

For CTDT, a *Coherence* node represents the coherence of the tutor's action in response to the previous student action as either *coherent* or *incoherent*. For instance, *negative feedback* in response to a *correct* student action is *incoherent*. A *Relevance* node, with values *high* and *low*, models how well the tutor cooperates with the student's focus of attention by assessing the extent to which the same problem steps are *in_focus* before and after the tutor's action: Problem steps that are in the student's focus of attention are likely to be *in_focus* in *Focus*$_0$. A tutorial action which addresses a problem step or related rule that is in the student's focus of attention will further increase the probability that the problem step is *in_focus* in *Focus*$_1$. Therefore, if the same problem steps are most likely *in_focus* in *Focus*$_0$ and *Focus*$_1$, *Relevance* is most likely *high*.

For RTDT, *Discourse State*$_s$ is simply the number of discourse turns, counted as a measure of success at avoiding spending too much time on a sentence.

4.1.6 *Utility*$_2$ Nodes

Utility$_2$ consists of several utility nodes in a structured utility model representing tutor preferences regarding tutorial state outcomes. Total utility is a weighted sum of the utilities for each tutorial state component (e.g., student knowledge, focus, and affect; task progress; discourse state). The utility value for each component may in turn be a weighted sum of the utilities for each subcomponent. For instance, *Knowledge*$_2$ rules that are important to the curriculum may be weighted more heavily than certain problem steps.

The tutor's behavior can easily be modified by changing the utilities or their weights. For instance, it may be that the best way for the tutor to improve the student's domain knowledge is to focus on the student's knowledge at the expense of helping the student make progress on tutorial tasks (e.g., solving problems). The tutor will do this automatically if a high weight is assigned to the utility of student knowledge and a low weight is assigned to the utility of task progress.

4.2 Implementation

With input from a problem solution graph (CTDT) or text (RTDT), DT Tutor creates a TACN with default values for prior and conditional probabilities and utilities. Default values are specified by parameter for easy modification. An optional file specifies any prior probability or utility values that differ from the defaults. After creating the initial TACN, DT Tutor recommends tutorial actions, accepts inputs representing tutor and student actions, updates the network, and adds new TACNs to the DDN as appropriate.

We automated construction of the large number of conditional probability table entries using a much smaller number of rules and parameters. For instance, for RTDT, the rule for the probability that a student will remember in slice 2 a word that she knew in slice 1 is:

$$P(Know\_Word\_i_2{=}known \mid Know\_Word\_i_1 = known) = 1.0 - word\text{-}forget\text{-}probability$$

word-forget-probability is a parameter that specifies the probability that the student will forget a known word between slices.

Both of DT Tutor's applications are prototypes for testing the viability and generality of the approach. CTDT does not yet have an interface, and RTDT has not been integrated with the Reading Tutor. Therefore, we used simulated student input for formative evaluations.

5 Formative Evaluation

Our goal was to determine whether DT Tutor's prototype applications can select optimal actions quickly enough to keep a student engaged.

5.1 Response Time

One of the major challenges facing probabilistic systems for real-world domains is tractability. We performed response time testing on a 667-MHz Pentium III PC with 128-MB of RAM. Using Cooper's (1988) algorithm for decision network inference using belief network algorithms, we tested with three algorithms: an exact clustering algorithm (Huang & Darwiche, 1996) and two approximate, sampling algorithms, likelihood sampling (Shachter & Peot, 1989) and heuristic importance (Shachter & Peot, 1989), with 1,000 samples each. Response times reported are the mean over 10 trials. The times for the approximate algorithms were extremely close, with neither holding an advantage in all cases, so they are reported as one below.

For CTDT, only the approximate algorithms had reasonable response times for both problems tested: 1.5 seconds for a 5-step problem and 2.1 seconds for an 11-step problem.

For the Reading Tutor's corpus of readings, sentence length ranges from approximately 5 to 20 words as reading level progresses from kindergarten through fifth grade, with approximately two-thirds content words, so we tested response times for preemptive help on sentences with 2 to 14 content words. Our response time goal was 0.5 seconds or less. For all three algorithms, response times for sentences with up to 7 content words were less than 0.5 seconds, ranging from 0.04 seconds for 2 content words to .49 seconds for 7 content words. Response times for the exact algorithm blew up starting at 10 content words with a time of 12.48 seconds. Response times for the approximate algorithms remained promising (as explained below) for up to 12 content words, ranging from .59 seconds for 8 content words to 3.14 seconds for 12 content words. However, response times for even the approximate algorithms blew up at 13 content words with times of 23-26 seconds. Therefore, response time for preemptive help was satisfactory for students at lower reading levels, did not meet the goal for longer sentences (starting at 8 content words), and was entirely unsatisfactory even with the approximate algorithms for the longest sentences (13-14 content words). Response time would tend to increase if the number of tutor action types is increased (see section 4.1.1), although the amount of increase would be at most linear in the proportion of additional action alternatives considered.

For decision-making purposes, it is sufficient to correctly rank the optimal alternative. When only the rank of the optimal alternative was considered, the approximate algorithms were correct on every trial. While this result cannot be guaranteed, it may make little practical difference if the alternative selected has an expected utility that is close to the maximum value. Moreover, many sampling algorithms have an *anytime* property that allows an approximate result to be obtained at any point in the computation (Cousins et al., 1993), so accuracy can continue to improve until a response is needed. For RTDT, response times for corrective feedback should generally be faster because RTDT does not consider helping with words that have already been read correctly. In any case, faster response times can be expected as computer hardware and probabilistic reasoning algorithms continue to improve. Therefore, the response times reported above for the approximate algorithms show promise that DT Tutor applications for real-world domains will be able to respond accurately enough within satisfactory response time. To handle the more challenging cases (such as the longest sentences faced by RTDT) in the near-term, application-specific adjustments may be required – e.g., abstraction in the knowledge representation within TACN components.

5.2 Action Selections

DT Tutor's decision-theoretic representation guarantees that its decisions will be optimal given the belief structure and objectives that it embodies. Nevertheless, the first step in evaluating a tutoring system is to see if it behaves in a manner that is consistent with strong intuitions about the pedagogical value of tutorial actions in specific situations. Such a sanity check cannot of course be a complete test. The space of network structures and probability and utility values, in combination with all possible student actions, is infinite, so the most we can do is sample from this space. However, if DT Tutor can handle many situations in which our intuitions are strong, we are more apt to have faith in its advice in situations where intuitions are less clear, and this is a prerequisite for testing with human subjects. Therefore, we tested DT Tutor's behavior in clear-cut situations.

First, we used default parameters to initialize TACNs with intuitively plausible probability and utility values. Next, we simulated student action inputs while perturbing probability and utility values to probe dimensions of the situation space. For instance, to test whether CTDT and RTDT would give preemptive help when warranted, we simply perturbed the prior probabilities for student knowledge of one or more domain elements (e.g., problem steps or words) to be most likely *unknown* and then verified that the application would suggest appropriate preemptive help.

The tests showed that DT Tutor is capable of selecting tutorial actions that correspond in interesting ways to the behavior of human tutors. Notable action selection characteristics include the following:

- Preemptively intervenes to prevent student errors and impasses, as human tutors often do (Lepper et al., 1993).
- Does not provide help when the student does not appear to need it. Human tutors often foster their students' independence by letting them work autonomously (Lepper et al., 1993).
- Adapts tutorial topics as the student moves around the task space and predicts the influence of the tutor's actions on the student's focus of attention.
- With equal utilities for knowledge of rules and steps, CTDT tends to address the student's knowledge of rules rather than problem-specific steps (because rule knowledge helps the student complete steps on her own). Effective human tutoring is correlated with teaching generalizations that go beyond the immediate problem-solving context (VanLehn et al., in press).
- CTDT tempers its actions based on consideration of the student's affective state (e.g., avoiding *negative feedback*). Human tutors consider the student's affect as well (Lepper et al., 1993).
- RTDT avoids repeating ineffective tutorial actions.

6 Related Work

Very few tutoring systems have used decision theory. Reye (1995) proposed a decision-theoretic approach for tutoring systems, mentioning an implementation in progress for tutoring SQL. Reye (1996) also proposed modeling the student's knowledge using a dynamic belief network. CAPIT (Mayo & Mitrovic, 2001, to appear), a decision-theoretic tutor for capitalization and punctuation, bases its decisions on a single objective and ignores the student's internal state in order to focus on observable variables. DT Tutor is a domain-independent architecture which considers multiple objectives, including objectives related to a rich model of the student's internal state.

Tutoring is a type of practical, mixed-initiative interaction. Within this broader domain, systems by Horvitz and colleagues (e.g., Horvitz et al., 1998; Horvitz & Paek, 1999) also model the state of the interaction, including the user's state, with connected sets of Bayesian models, and employ decision theory for optimal action selection. Some of these systems (e.g., Horvitz & Paek, 1999) use value-of-information to guide user queries and observation selection, which DT Tutor does not (yet) do. To model temporal evolution, a number of probabilistic approaches have been tried, including dynamic and single-stage network representations (e.g., Horvitz et al., 1998). DT Tutor appears to be alone among systems for mixed-initiative interaction in (1) using a dynamic decision network to consider uncertainty, objectives, and the changing state within a unified paradigm, and (2) explicitly predicting the student's next action and its effect on the interaction.

7 Future Work and Discussion

We are currently selecting the domain for the first full-fledged implementation of DT Tutor's action selection engine in a complete tutoring system, either by combining it with an existing tutoring system (such as the Reading Tutor) or by building our own user interface. We are also investigating applications that are more explicitly dialogue-oriented. Whichever domain we select, our next major milestone will be testing the effectiveness of DT Tutor's approach with students.

Efficiently obtaining more accurate probability and utility values is a priority. However, precise numbers may not always be necessary. For instance, diagnosis (say, of the student's knowledge) in Bayesian systems is often surprisingly insensitive to imprecision in specification of probabilities (Henrion et al., 1996). For a decision system, it is sufficient to correctly rank the optimal decision alternative. Moreover, if the actual expected utilities of two or more alternatives are very close, it may make little practical difference which one is selected.

This work has shown that a decision-theoretic approach can be used to select tutorial discourse actions that are optimal, given the tutor's beliefs and objectives. DT Tutor's architecture balances tradeoffs among multiple competing objectives and handles uncertainty about the changing tutorial state in a theoretically rigorous manner. Discourse actions are selected both for their direct effects on the tutorial state, including the student's internal state, and their indirect effects on the subsequent student turn and the resulting tutorial state. The tutorial state representation may include any number of attributes at various levels of detail, including the discourse state, task progress, and the student's knowledge, focus of attention, and affective state. A rich model of the tutorial state helps DT Tutor to select actions that correspond in interesting ways to the behavior of human tutors. Response time remains a challenge, but testing with approximate algorithms shows promise that applications for diverse real-world domains will be able to respond with satisfactory accuracy and speed.

As an action-selection engine, DT Tutor plays at most the role of a high-level discourse planner, leaving the specifics of dialogue understanding and generation (parsing, semantic interpretation, surface realization, etc.) to other components of the tutoring application. It performs near-term discourse planning by anticipating the effects of its actions on the student's internal state, the student's subsequent discourse turn, and the resulting tutorial state. To predict how its actions will influence the tutorial state, including the student's internal state, DT Tutor's architecture includes strong domain reasoning and student modeling.

Acknowledgments

This research was sponsored by the Cognitive Science Division of the Office of Naval Research under grant N00014-98-1-0467. For decision-theoretic inference, we used the SMILE reasoning engine contributed to the community by the Decision Systems Laboratory at University of Pittsburgh (http://www.sis.pitt.edu/~dsl). We thank the reviewers for several helpful suggestions.

References

Cooper, G. F. (1988). A method for using belief networks as influence diagrams. In *Workshop on Uncertainty in Artificial Intelligence*, pp. 55-63.

Cousins, S. B., Chen, W., & Frisse, M. E. (1993). A tutorial introduction to stochastic simulation algorithms for belief networks. *Artificial Intelligence in Medicine 5*, pp. 315-340.

Henrion, M., Pradhan, M., Del Favero, B., Huang, K., Provan, G., & O'Rorke, P. (1996). Why is diagnosis in belief networks insensitive to imprecision in probabilities? In *12th Annual Conference on Uncertainty in Artificial Intelligence*, pp. 307-314.

Horvitz, E., Breese, J., Heckerman, D., Hovel, D., & Rommelse, K. (1998). The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In *14th Conference on Uncertainty in Artificial Intelligence*, pp. 256-265.

Horvitz, E., & Paek, T. (1999). A computational architecture for conversation. In *Seventh International Conference on User Modeling*, pp. 201-210.

Huang, C., & Darwiche, A. (1996). Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning 15*, 225-263.

Huang, T., Koller, D., Malik, J., Ogasawara, G., Rao, B., Russell, S., & Weber, J. (1994). Automated symbolic traffic scene analysis using belief networks. In *12th National Conference on Artificial Intelligence*, pp. 966-972.

Lepper, M. R., Woolverton, M., Mumme, D. L., & Gurtner, J.-L. (1993). Motivational techniques of expert human tutors: Lessons for the design of computer-based tutors. In S. P. Lajoie & S. J. Derry (Eds.), *Computers as Cognitive Tools*, pp. 75-105. Lawrence Erlbaum Associates.

Mayo, M., & Mitrovic, A. (2001, to appear). Optimising ITS behaviour with Bayesian networks and decision theory. *International Journal of Artificial Intelligence in Education 12*.

Merrill, D. C., Reiser, B. J., Ranney, M., & Trafton, J. G. (1992). Effective tutoring techniques: A comparison of human tutors and intelligent tutoring systems. *The Journal of the Learning Sciences 2*(*3*), 277-306.

Mostow, J., & Aist, G. (1999). Giving help and praise in a reading tutor with imperfect listening -- because automated speech recognition means never being able to say you're certain. *CALICO Journal 16*(*3*), 407-424.

Mostow, J., & Aist, G. (in press). Evaluating tutors that listen: An overview of Project LISTEN. In K. Forbus & P. Feltovich (Eds.), *Smart Machines in Education: The coming revolution in educational technology*. MIT/AAAI Press.

Murray, R. C., & VanLehn, K. (2000). DT Tutor: A dynamic, decision-theoretic approach for optimal selection of tutorial actions. In *Intelligent Tutoring Systems, 5th International Conference,* pp. 153-162.

Newell, A., & Simon, H. A. (1972). *Human Problem Solving*. Prentice-Hall, Inc.

Reye, J. (1995). A goal-centred architecture for intelligent tutoring systems. In J. Greer (Ed.) *World Conference on Artificial Intelligence in Education*, pp. 307-314.

Reye, J. (1996). A belief net backbone for student modeling. In *Intelligent Tutoring Systems, Third International Conference*, pp. 596-604.

Shachter, R., & Peot, M. (1989). Simulation approaches to general probabilistic inference on belief networks. In *5th Annual Conference on Uncertainty in Artificial Intelligence*, pp. 221-231.

Singley, M. K. (1990). The reification of goal structures in a calculus tutor: Effects on problem solving performance. *Interactive Learning Environments 1*, 102-123.

VanLehn, K., Siler, S., Murray, C., Yamauchi, T., & Baggett, W. B. (in press). Human tutoring: Why do only some events cause learning? *Cognition and Instruction*.

AutoTutor: An Intelligent Tutor and Conversational Tutoring Scaffold

Arthur C. Graesser[1], Xiangen Hu[1], Suresh Susarla[1], Derek Harter[1], Natalie Person[2], Max Louwerse[1], Brent Olde[1], and the Tutoring Research Group[1]
[1]University of Memphis
[2]Rhodes College

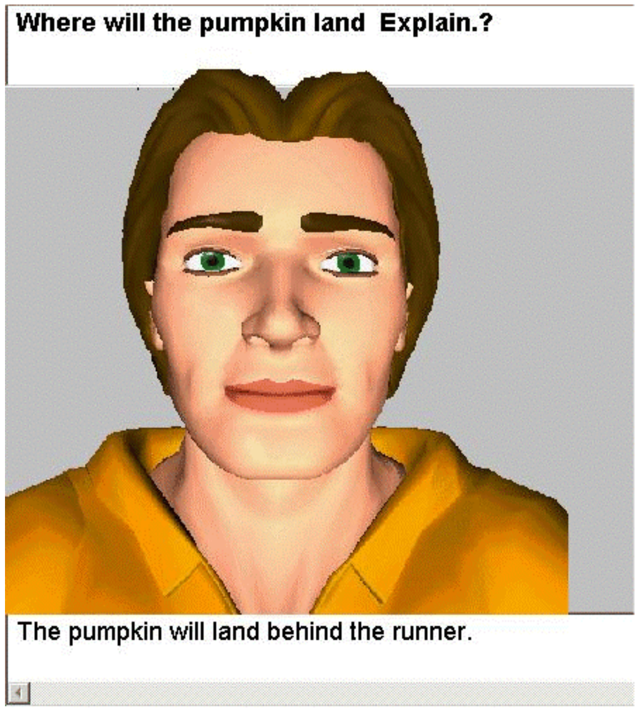
The Tutoring Research Group (TRG) at the University of Memphis has developed a computer tutor (called AutoTutor) that simulates the discourse patterns and pedagogical strategies of a typical human tutor (Graesser, P. Wiemer-Hastings, K. Wiemer-Hastings, Kreuz, & TRG, 1999). The dialog mechanisms of AutoTutor were designed to incorporate conversation patterns that exist in naturalistic tutoring sessions (Graesser, Person & Magliano, 1995), as well as some ideal strategies for promoting learning gains. AutoTutor was originally designed to help college students learn introductory computer literacy, such as the fundamentals of hardware, operating systems, and the Internet. Evaluations of AutoTutor have shown that the tutoring system improves learning and memory of the lessons by .5 to .6 standard deviation units compared to rereading a chapter (Graesser, Person, Harter, & TRG, in press).

Instead of merely being an information delivery system, AutoTutor is a collaborative scaffold that assists the student in actively constructing knowledge by holding a conversation in natural language. A dialog manager coordinates the conversation that occurs between a learner and a pedagogical agent, whereas lesson content and world knowledge are represented in a curriculum script and latent semantic analysis (Landauer, Foltz, & Laham, 1998). LSA and surface language cues guide the evaluation of the quality of student input (Wiemer-Hastingset al., 1999). There is an animated conversational agent with facial expressions, synthesized speech, and some rudimentary gestures. The modules of AutoTutor are uniformly weak rather than strong when considering parsing, semantic interpretation, dialog planning, domain reasoning, student modeling, and discourse production; the weakness of these modules arguably reflects the capability of human tutors. We are currently developing a hybrid version of AutoTutor that incorporates both weak and strong computational modules.

As an example of a weak module, a dialog advancer network (DAN) manages the exchange by specifying appropriate discourse markers (e.g., *Moving on, Okay*), dialog move categories, and frozen expressions within the tutor's turn. The content of selected dialog move category is generated by a separate mechanism, so there is a natural segregation of dialog functions from substantive content. There are the following different categories of dialog moves that AutoTutor generates: main question, short feedback (i.e., positive, neutral, negative), pumps (*uh huh, tell me more*), prompts (*The primary memories of the CPU are ROM and _____*), prompt response *(and RAM*), hints, assertions, corrections, and summaries. The DAN is formally an augmented state transition network because the selection of a dialog move category on tutor turn N+1 is sensitive to a large space of parameters computed from the dialog history. The DAN in AutoTutor-1 does a fairly impressive job in managing the conversation, based on our performance data (Person, Graesser, Pomeroy, Kreuz, & TRG, in press), even though it does not incorporate sophisticated dialog planning capabilities.

AutoTutor was designed to be reusable for other knowledge domains that do not require mathematical precision and formal specification. In order to test the portability of the AutoTutor architecture, we developed a version for the domain of conceptual physics. Together with computer literacy, conceptual physics is one of the fields in which extra tutoring sessions are needed. The target population for the tutor was undergraduate students taking elementary courses in conceptual physics.

In the transition of AutoTutor from computer literacy to physics only three modules needed to be changed for the new subject matter: (1) a glossary of terms and definitions for physics, (2) an LSA space for conceptual physics, (3) a curriculum script with deep reasoning questions and associated answers for physics. The three modules can loosely be affiliated with metacognition, comprehension, and production. Changing the glossary required approximately 15 man hours. This process is relatively easy: definitions from text books need to be included in order to give AutoTutor the possibility of accurately answer metacognitive questions ("What does X mean?"). The majority of AutoTutor's comprehension mechanisms use LSA, so setting his long-term memory representation is an important process. The LSA space needs to be trained with an adequate corpus of texts applicable to the knowledge domain, such as text books, chapters, and technical articles. After the corpus is prepared in an electronic form, we declare the parameters of LSA, such as the number of dimensions and size of document units. The training of the LSA space takes less than an hour (Cleaning up the corpus (removing code, pictures, etc.), however, could be time consuming, in our case approximately 10 hours). Most of the work, however, lies in the curriculum script. Since AutoTutor's architecture only allows one particular format, the curriculum script needs to be carefully changed. Questions need to be defined, ideal answers need to be formulated, hints, prompts and pumps need to be included. AutoTutor's application to computer literacy had three topics, each consisting of 12 deep-reasoning questions. Changing the curriculum script to conceptual physics required approximately 20 hours for 3 questions/problems. However, an authoring tool makes this process less time-consuming than most intelligent tutoring systems because the format of the entries are descriptions in English rather than structured code (e.g., Lisp, Prolog). Several lesson planners can simultaneously work on the transition and do not require sophisticated programming expertise.


Where will the pumpkin land Explain.?

The pumpkin will land behind the runner.

Figure 1. Screen shot of AutoTutor interface

TUTOR	Okay, let's begin. Hello, I'm Marco. I'm one of the computer tutors developed by the Tutoring Research Group, at the University of Memphis. I try to help students learn about Physics, by having a conversation with them. Let's get straight to the demonstration. Suppose a runner is running in a straight line at constant speed, and the runner throws a pumpkin straight up. Where will the pumpkin land? Explain.
STUDENT	The pumpkin lands behind the runner.
TUTOR	Anything else?
STUDENT	I don’t know.
TUTOR	Okay. What is the horizontal velocity of the pumpkin?
STUDENT	The pumpkin has a constant horizontal velocity.
TUTOR	so? The horizontal velocity of the pumpkin is the same as the?
STUDENT	Could you repeat that?
TUTOR	Once again, the horizontal velocity of the pumpkin is the same as the?
STUDENT	runner.
TUTOR	The same as the runner. Okay. Also, the man and the pumpkin have the same horizontal velocity. What can we say about the horizontal velocity of the pumpkin if there is no force acting on the pumpkin in the horizontal direction?
STUDENT	its constant.

Note that this transcript only presents a small part of the rich information (pauses, intonation, gestures) that AutoTutor provides in the interaction with a student.

Table 1. Sample of a student interaction transcript

Acknowledgements

Research on AutoTutor was supported on grants from the National Science Foundation (SBR 9720314) and the Officie of Naval Research (N00014-00-1-0600).

References

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

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., VanLehn, K., Rose, C., Jordan, P., & Harter, D. (in press). Intelligent tutoring systems with conversational dialogue. *AI Magazine*.

Graesser, A.C., Wiemer-Hastings, K., Wiemer-Hastings, P., Kreuz, R., & TRG (1999). AutoTutor: A simulation of a human tutor. *Journal of Cognitive Systems Research*, *1*, 35-51.

Landauer, T.K., Foltz, P.W., Laham, D. (1998). An introduction to latent semantic analysis. *Discourse Processes*, *25*, 259-284.

Person, N.K., Graesser, A.C., Kreuz, R.J., Pomeroy, V., & TRG (in press). Simulating human tutor dialog moves in AutoTutor. *International Journal of Artificial Intelligence in Education*.

Wiemer-Hastings, P., Wiemer-Hastings, K., and Graesser, A. (1999). Improving an intelligent tutor's comprehension of students with Latent Semantic Analysis. In S.P. Lajoie and M. Vivet, *Artificial Intelligence in Education* (pp. 535-542). Amsterdam: IOS Press.

Simple Natural Language Generation and Intelligent Tutoring Systems

Barbara Di Eugenio, Michael Glass, Michael J. Trolio
Electrical Engineering and Computer Science Department
University of Illinois at Chicago
Chicago, IL, 60607 USA
{bdieugen,mglass,mtrolio}@eecs.uic.edu

Susan Haller
Computer Science Department
University of Wisconsin Parkside
Kenosha, WI 53141, USA
haller@cs.uwp.edu

Abstract

In this paper, we report on our approach to adding Natural Language Generation (NLG) capabilities to ITSs. Our choice has been to apply simple NLG techniques to improve the feedback provided by an existing ITS, specifically, one built within the DIAG framework (Towne 1997). We evaluated the original version of the system and the enhanced one with a between subjects experiment. On the whole, the enhanced system is better than the original one, other than in helping subjects remember the actions they took. Current work includes exploiting more sophisticated NLG techniques but still without delving into full fledged text planning. We are also conducting a constrained data collection, in which students and tutors interact via the ITS.

Introduction

Today, many projects aim at providing ITSs with a full-fledged dialogue interface, e.g. see the work at the CIRCLE center (http://www.pitt.edu/˜circle/), or (Hume *et al.* 1996; Moore, Lemaire, & Rosenbloom 1996; Rosé, Di Eugenio, & Moore 1999; Freedman 1999; Graesser *et al.* 2000). On the contrary, our approach to adding NLG capabilities to an Intelligent Tutoring System falls on the weak side of the divide: we are concentrating on simple sentence planning with no or minimal amounts of text planning. Our choice is partly a development strategy, because we set out to rapidly improve the language feedback provided by an existing ITS shell, partly a desire to evaluate how effective the system can be with a relatively small effort. A similar approach — using simple generation techniques for surface realization in tutoring dialogues — is taken in YAG (McRoy, Channarukul, & Ali 2000). Our results so far suggest that simple NLG can help, but the gains are small enough to suggest that moving to somewhat more sophisticated techniques should be beneficial, even if we still don't intend to develop a full fledged NLG interface.

We take this approach for two reasons. First, we want to understand what can be accomplished by interfacing an NL generator to an ITS taken as a blackbox: can the ITS tutoring strategy be left as is, or is there a point in which the dialogue strategies and the original tutoring strategy are at odds with each other? Second, we are interested in finding out what is the "added value" of an NL interface to an ITS. One way to do so is to compare a system that does not use NL techniques to a version of the same system that uses NL. We are aware of only one other experiment in this direction (Trafton *et al.* 1997), in which subjects gave input to a cartographic system using either NL only, direct manipulation only, or a combination of the two. Subjects were given instructions such as "go to intersection X"; time on task and score on map drawing after the session were recorded. In the NL only condition, subjects performed the poorest on the map drawing task. However, it is not clear which conclusions should be drawn from this work, given that the system they describe does not seem to qualify as a real ITS. In general, the evaluation of NL interfaces to ITSs is an area that needs investigation. ITSs are often evaluated in terms of pre/post-test score, however task performance measures may be appropriate as well. To our knowledge, the only ITSs with an NL interface which has been formally evaluated is CIRCSIM (Evens *et al.* 1993; Kim, Glass, & Evens 2000), but the results of the evaluation are not available yet.

We will first discuss DIAG, the ITS authoring shell we are using. We will then discuss the work we have completed; this comprises the aggregation rules we implemented within EXEMPLARS and the formal evaluation we conducted. We will then discuss some current work on generating more coherent feedback by exploiting more sophisticated NLG techniques, and the data collection we have started, to study how tutors verbalize the information that the ITS wants to communicate.

DIAG

DIAG (Towne 1997) is a shell to build ITSs that teach students to troubleshoot complex artifacts and systems, such home heating and circuitry. DIAG in turn builds on the VIVIDS authoring environment (Munro 1994). VIVIDS based tutors deliver instruction and practice in the context of graphical simulations. Authors build interactive graphical models of complex systems, and build lessons based on these graphical models.

A typical session with a DIAG application presents the student with a series of troubleshooting problems of increasing difficulty. DIAG's tutoring strategy steers the student towards performing the tests that have the greatest potential for reducing uncertainty (Towne 1997). Most of the times, a test consists of the visual observation of an *indicator*. DIAG keeps track of the tests the student performs, and the inferences that could be made from the symptoms shown. The student interacts with the application by testing indicators and trying to infer which faulty part (RU) may cause the detected abnormal states. RU stands for *replaceable unit*, because the only course of action open to the student to fix the problem is to replace faulty components in the graphical simulation. Figure 1 shows one of the graphical views in a DIAG application that teaches how to troubleshoot a home heating system. The subsystem being displayed is the furnace system. Some of its components are indicators (e.g., the gauges labeled Burner Motor RPM and Water Temperature). Others are either replaceable units, or other complex modules that contain indicators and replaceable units, e.g. the Oil Burner. Complex components are in turn zoomable.

At any point, the student can consult the built-in tutor in one of several ways. For example, if the student suspects an RU to be faulty, s/he can ask the tutor to specify the likelihood that this part is the cause of the fault. The tutor will also indicate the state of any indicators that the student has explored and try to imply a correlation, positive or negative, between the states of the indicators to the RU in question. By utilizing the tutor's feedback, the student can deduce relationships among the system parts and continually refine his/her solution.

Language Generation in DIAG

After deciding which content to communicate, the original DIAG system (*DIAG-orig*) uses very simple templates to assemble the text to present to the student. The result is that the feedback that DIAG provides is repetitive, both as a sequence of replies to requests for feedback, and within each verbal feedback. In many cases, the feedback presents a single long list of many parts. This problem is compounded by the fact that most DIAG applications involve complex systems with many parts. Although there are different levels of description in the system model, and hierarchies of objects, the verbal feedback is almost always in terms of individual indicators or units. The top part of Figure 2 shows the reply originally provided by DIAG to a request of information regarding the indicator named "Visual Combustion Check".

We set out to improve on DIAG's feedback mechanism by applying aggregation rules. For example, a long list of parts can be broken down by classifying each of these parts in to one of several smaller lists and then presenting the student with this set of lists. The bottom part of Figure 2 shows our aggregation rules at work. The revised output groups the parts under discussion by the system modules that contain them (Oil Burner and Furnace System), and by the likelihood that a certain RU causes the observed symptoms. Notice how the *Ignitor Assembly* is singled out in the revised answer. Among all mentioned units, it is the only one that cannot cause the symptom. This fact is lost in the original answer.

As our sentence planner, we chose EXEMPLARS (White & Caldwell 1998) over better known systems such as FUF (Elhadad 1993) and Penman (Bateman 1994) because of the complexity and learning curve of the latter two. Efficiency and rapid prototyping are among the reasons we chose EXEMPLARS.

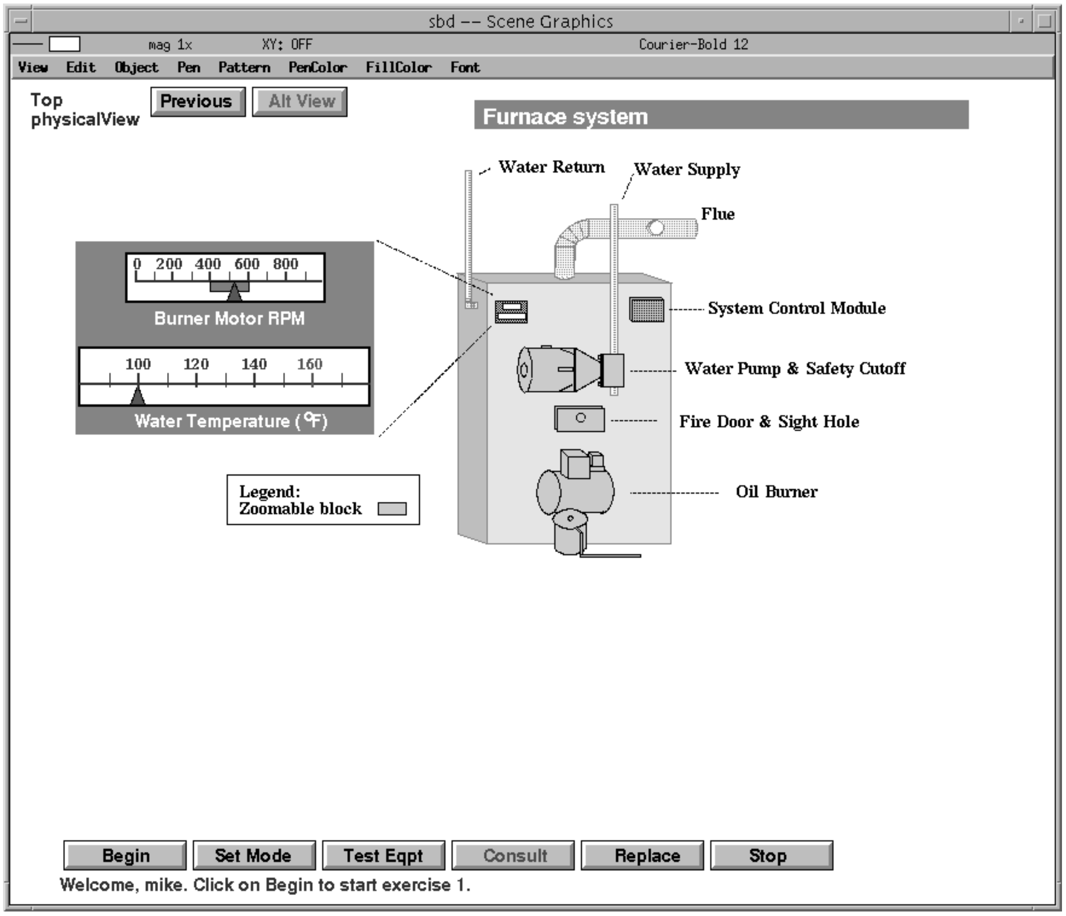
EXEMPLARS is an object-oriented, rule based generator. The rules (called *exemplars*) are similar to schema-like text planning rules because they are meant to capture an exemplary way of achieving a communicative goal in a given communicative context, as determined by the system designer. EXEMPLARS is a hybrid system that mixes template-style and more sophisticated types of text planning. The text planner selects rules by traversing the exemplar specialization hierarchy. The applicability conditions associated with each exemplar are successively evaluated in order to find the most specific exemplar for the current context.

In the enhanced version of the system (*DIAG-NLP*), DIAG passes the information to be communicated to EXEMPLARS (the two systems communicate via a text file). EXEMPLARS performs essentially three tasks:

1. it determines the specific exemplars needed;
2. it adds the chosen exemplars to the sentence planner as a goal;
3. it linearizes and lexicalizes the feedback in its final form, writing it to an external file which is passed back to DIAG for display in the appropriate window.

In *DIAG-NLP*, we concentrated on rules for aggregation, some of which also affect format and layout. Our choices were suggested by the need to relate the language feedback to the hierarchical structure of the physical system. We have two main kinds of rules, description rules and aggregation rules.

Description rules are used when the full description of a part is required, such as whether the part is



Welcome, mike. Click on Begin to start exercise 1.

Figure 1: A screen from a DIAG application on home heating

in a normal state, its current reading, and, if abnormal, what the normal state should be (see the first sentence in the bottom part of Figure 2).

The aggregation rules are used to group large lists of parts into smaller lists. They allow composite aggregation, so that nested lists are created. Among our aggregation exemplars are:

- *AggByContainer*: each part within this DIAG application is contained within a larger block, called a system module. The *AggByContainer* rule accepts a list of parts, classifies each part by its containing module, and then creates a set of lists by module;
- *AggByFufer*: it groups replaceable units according to the likelihood of being at fault for a specific symptom;
- *AggByState*: it groups indicators by their normal / abnormal state.

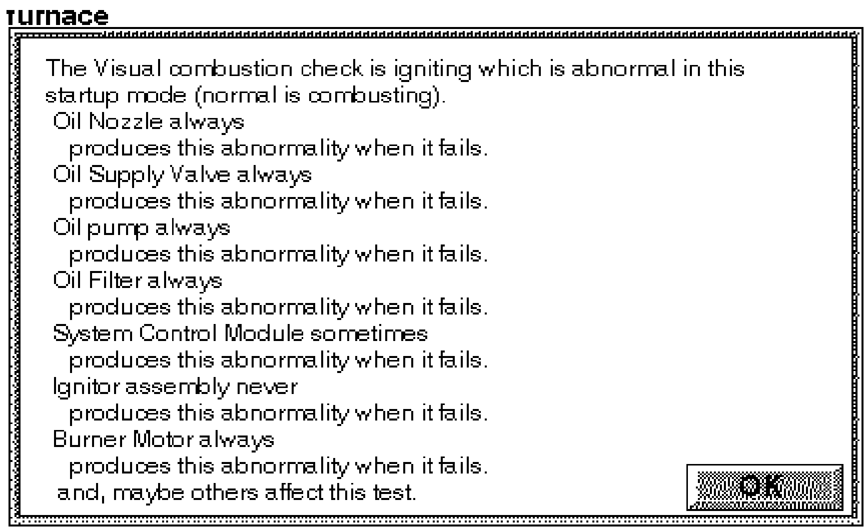
A final exemplar, invoked by the other aggregation rules, deals with formatting, namely, creating vertical lists, spacing, etc.

The most frequent application of the aggregation rules is to group parts according to the system module they belong to, and within each module, to group replaceable units by how likely it is they may cause the observed symptom, as shown in Figure 2.

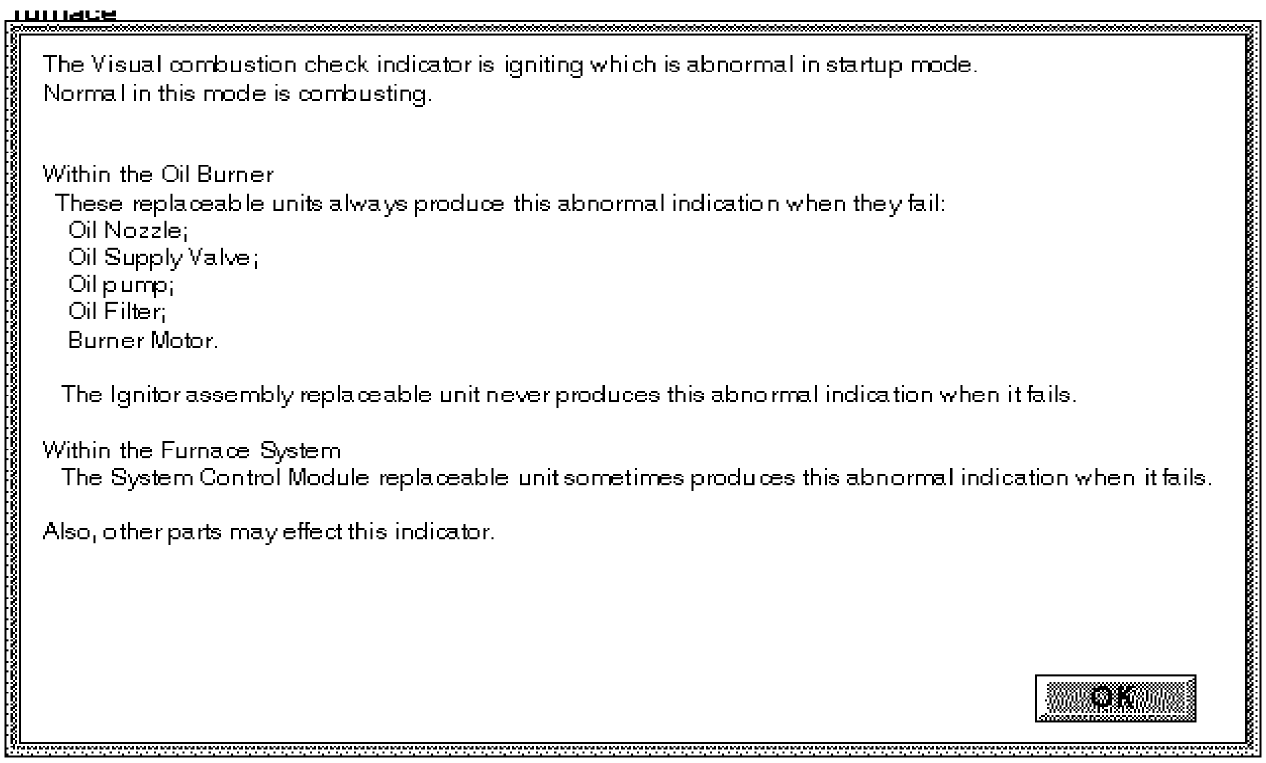
In this version of *DIAG-NLP*, morphology, lexical realization and referring expression generation were all treated ad hoc, i.e., they were directly encoded in the appropriate exemplars.

Experiments

Intuitively, the contrast between the feedback produced by *DIAG-orig* and by *DIAG-NLP* (top and bottom in Figure 2) suggests that even simple aggregation rules dramatically improve the language feedback. To provide a real assessment of this claim, we conducted an empirical evaluation designed as a between-subject study. Both groups interact with the same DIAG application that teaches them to troubleshoot a home-heating system. One group interacts with *DIAG-orig* and the other with *DIAG-*



The Visual combustion check is igniting which is abnormal in this startup mode (normal is combusting).
 Oil Nozzle always
  produces this abnormality when it fails.
 Oil Supply Valve always
  produces this abnormality when it fails.
 Oil pump always
  produces this abnormality when it fails.
 Oil Filter always
  produces this abnormality when it fails.
 System Control Module sometimes
  produces this abnormality when it fails.
 Ignitor assembly never
  produces this abnormality when it fails.
 Burner Motor always
  produces this abnormality when it fails.
 and, maybe others affect this test.



The Visual combustion check indicator is igniting which is abnormal in startup mode.
Normal in this mode is combusting.

Within the Oil Burner
 These replaceable units always produce this abnormal indication when they fail:
  Oil Nozzle;
  Oil Supply Valve;
  Oil pump;
  Oil Filter;
  Burner Motor.

 The Ignitor assembly replaceable unit never produces this abnormal indication when it fails.

Within the Furnace System
 The System Control Module replaceable unit sometimes produces this abnormal indication when it fails.

Also, other parts may effect this indicator.

Figure 2: Original (top) and revised (bottom) answers provided by DIAG to the same *Consult Indicator* query

NLP.

Seventeen subjects were tested in each group. Our subject pool comprised 13 undergraduates, 18 graduates, and 3 research staff, all affiliated with our university. Participation in the experiment was restricted to science or engineering majors. Each subject first reads some short material about home heating that we developed. Afterwards, each subject goes through the first problem as a trial run. Each subject then continues through the curriculum on his/her own. The curriculum consists of three problems of increasing difficulty. Subjects are encouraged to interact with DIAG as much as possible. At the end of the experiment, each subject is administered a questionnaire.

Metrics. A detailed log is collected while the subject solves problems. It includes how many problems the subject solved, and, for each problem: total time, and time spent reading feedback; how many and which indicators and RUs the subject consults DIAG about; how many, and which RUs the subject replaces.

Questionnaire. The questionnaire is divided into three parts. The first part tests the subject's understanding of the domain. Because the questions asked are fairly open ended, this part was scored as if grading an essay.

The second part concerns the subjects' recollection of their actions, specifically, of the indicators they consulted the system on and of the RUs they replaced. By taking the log of the subject's actions as the target, we can compute the usual measures of precision and recall. We compute precision as the percentage of correct answers out of the total number of answers the subject gave; whereas recall is the percentage of correct answers they gave with respect to the log of their actions. We also compute the F-measure, $\frac{(\beta^2+1)PR}{\beta^2 P+R}$, that smooths precision and recall off, with $\beta = 1$.

The third part of the questionnaire asks the subject to rate the system's feedback along four dimensions on a scale from 1 to 5 (see Table 3).

Comments on collected measures. As the reviewers of this paper pointed out, almost all the measures we collected, and whose significance is analyzed below, pertain to task performance or user satisfaction, rather than to learning per se — only *Essay score* directly addresses learning. We agree that learning measures should be the ultimate test of the success of the NL interface to the ITS. However we would argue that performance measures are important too: they provide indirect evidence of the effectiveness of the system, including issues of usability. For example, the lower number of indicator consultations in *DIAG-NLP* is evidence in favor of the effectiveness of the aggregated feedback: because the feedback highlights what is important (such as that the Ignitor Assembly can never cause the Visual Combustion check to ignite, see Figure 2), subjects can focus their troubleshooting without asking as many questions of the system. We would argue that an ITS whose NL feedback leads the student more effectively towards the solution of a problem is a better ITS. This holds for usability as well (the four measures in Table 3): presumably, in a real setting, students should be more willing to sit down with a system that they perceive as more friendly and usable than a system that engenders similar learning gains, but is harder to use.

	DIAG-orig	DIAG-NLP
Time	29.8'	28.0'
Feedback Time	6.9'	5.4'
Consultations	30.4	24.2
Indicator consultations	11.4	5.9
RU consultations	19.2	18.1
Parts replaced	3.85	3.33
Essay score	81/100	83/100

Table 1: Performance measures

	DIAG-orig	DIAG-NLP
Indicator Precision	.33	.17
Indicator Recall	.33	.27
Indicator F-measure	.44	.29
RU Precision	.74	.65
RU Recall	.73	.63
RU F-measure	.72	.63

Table 2: Precision / recall

The measures in Table 2 measure something in between learning and performance. One could argue that remembering what you did correlates with learning — e.g., if you remember that to solve a certain problem you checked whether the furnace was combusting (the "Visual Combustion Check" in Figure 2) and that gave you crucial information, you may be able to apply similar knowledge in similar problems. However, it is unlikely that detailed quantitative measures such as those we collected in this experiment are telling in this regard; and in fact, we would be happy to eliminate them, as they actually show an advantage for *DIAG-orig*. However, we collected them because they are relevant to the more general question of the added value of NL interfaces to applications, which we are also interested in.

	DIAG-orig	DIAG-NLP
Usefulness	4.35	4.47
Helped stay on right track	4.35	4.35
Not misleading	4.00	4.12
Conciseness	3.47	3.76
Average score	4.04	4.18

Table 3: Subjective rating of DIAG's feedback

	DIAG-orig	DIAG-NLP
Total Time		✓
Indicator consultations		✓
RU consultations		✓
Parts replaced		✓
Essay score		✓
Usefulness		✓
Helped stay on right track	✓	
Not misleading		✓
Conciseness		✓

Table 4: Successes for *DIAG-orig* and *DIAG-NLP*

Results. Every student solved all the problems, but differences emerge with respect to other measures. Tables 1, 2, 3 show the results for the cumulative measures across the three problems (measures on individual problems show the same trends).

On the whole, Tables 1 and 3 show a cumulative effect in favor of *DIAG-NLP*, whereas Table 2 does not. Focusing first on Tables 1 and 3, differences on individual measures are not statistically significant; the measure that individually comes closest to statistical significance is *indicator consultations*, which exhibits a non-significant trend in the predicted direction (Mann-Whitney test, U=98, p=0.11). We have discussed individual measures at length in (Di Eugenio & Trolio 2000); here, we provide a different statistical analysis to assess whether the *cumulative* effect of these measures shows that *DIAG-NLP* performs better than *DIAG-orig*.

We consider only independent measures (for example, the total number of consultations in Table 1 is clearly not independent from indicator and RU's consultations, given it is the sum of these two measures). For each measure, we decide whether its value indicates a "success" for *DIAG-NLP*. We are not looking at the magnitude of the difference between the two values of the measure, but simply at the fact that the values differ. Every measure in Table 1 is in *DIAG-NLP* favor, and so is every measure apart from *helped stay on right track* in Table 3 (we consider a tie as a success for *DIAG-orig*). We then ask, what is the probability that the *m* successes for *DIAG-NLP* out of the *n* independent measures are simply due to chance? We can answer via $B(m-1, n, 0.5)$, the binomial cumulative distribution function through $m-1$ for sample size n and probability of success p = 0.5: it gives us the probability that of n random trials, the number of successes will fall between 0 and $m-1$, inclusive. Thus, $1 - B(m-1, n, 0.5)$ gives us the probability that m or more successes out of n are due to chance.

As an example, consider Table 4, in which we combine the independent measures from Tables 1 and 3 and note whether they represent a success for *DIAG-orig* or *DIAG-NLP*. The probability of 8 successes out of 9 measures is p = 0.020 ($1 - B(7, 9, 0.5)$). If we leave *Total Time* out because it may not be an independent measure,[1] the probability of 7 successes out of 8 is p = 0.035 ($1 - B(6, 8, 0.5)$). Finally, if instead of using the four subjective measures we use their average (which constitutes a success for *DIAG-NLP*),[2] we obtain p = 0.016, and if we eliminate time in this last case, we obtain p = 0.031. To conclude, in whatever way we combine these measures, we obtain evidence that the better scores *DIAG-NLP* obtains, albeit individually not statistically significant, cumulatively show that *DIAG-NLP* outperforms *DIAG-orig*.

However, we have not discussed Table 2 yet. This table shows that subjects in *DIAG-orig* remember what they did better than those in *DIAG-NLP*. The measures concerning indicators achieve or show trends towards statistical significance: indicator precision and indicator F-measure are significant (t-test, respectively 2.19, p = 0.04 and 2.51, p = 0.02), and indicator recall is marginally significant (Mann-Whitney, U = 93.5, p = 0.08). All in all, this is a puzzling result, especially because subjects in *DIAG-orig* consult the system on indicators almost twice as many times as the subjects in *DIAG-NLP*, thus we would expect them to have more problems remembering what they did. Perhaps this result can be related to (Kintsch 1998), that shows that high-quality text does not necessarily lead to better performance.

Finally, the reader may wonder what happens to the cumulative effect that shows *DIAG-NLP* better than *DIAG-orig* if we take into account the measures in Table 2 as well. By adding to Table 4 two successes for *DIAG-orig*,[3] we compute the probability of obtaining 8 suceesses out of 11 measures

[1]One could argue the time went down because of the smaller number of consultations.

[2]A reviewer pointed out that including all four measures gives much more weight to the subjective measures than e.g. to the single *essay score* learning measure.

[3]Given their definitions, precision and recall cannot be considered as really independent measures, and certainly the F-measure that combines them is not independent from either of them. So we synthesize Table 2 as two successes for *DIAG-orig*, one for indicators, one for RU's.

by chance. We obtain p = 0.113, which shows a non significant trend in the predicted direction. However, recall that we are being conservative: for example, we counted *help stay on right track* in favor of *DIAG-orig* even if it is a tie; if we count it in favor of *DIAG-NLP*, p goes down to 0.033.

Current and future work

The results of the study we just discussed make us confident that it is not necessary to add a full fledged NL generator to an existing ITS or to change the ITS original tutoring strategies to obtain reasonable results. Better language can be added at a relatively low cost (the implementation took one graduate student six months), and it can be effective.

As a consequence, we are now pursuing two lines of research. The first is to add some more sophisticated NL techniques without plunging into full text planning, because we want to see how far the weak approach can go. Second, we are conducting a constrained data collection to help us discover some empirical foundations on which to base the realization of the facts the ITS intends to communicate.

We now discuss both efforts in more detail.

Focusing and rhetorical relations

In the work done so far, we imposed coherence on the tutor turn by means of aggregation rules. However, the turn could be made more coherent by introducing appropriate referential expressions (generated ad hoc so far), and a few domain or rhetorical relations among the facts to be expressed. For example, the fact that the ignitor assembly never causes the abnormal indication in Figure 2 as opposed to the other parts within the oil burner always causing it, could be given more prominence if the relevant propositions were linked by a *contrast* relation[4] rendered via an appropriate cue phrase, such as *but* ((A) and (B) are used later to refer to the appropriate part of the explanation):

(1) *The visual combustion check indicator is igniting which is abnormal in startup mode. Normal in this mode is combusting.*

(A) Within the oil burner, the oil nozzle, oil supply valve, oil pump, oil filter and burner motor always produce this abnormal indication when they fail. (B) But the ignitor assembly never does.

In ongoing work, we have coupled EXEMPLARS to a knowledge base built via the SNePS representation system (Shapiro & Rapaport 1992). SNePS is a semantic network formalism where each node represents a proposition. In general, it is very difficult to access the knowledge about the physical structure of the system and causal relationships in VIVIDS-based tutors. These types of knowledge are often only indirectly present: they are reflected in how changes to graphical objects affect other objects, but this is not sufficient to generate language. When they are present, they are expressed in a very non-symbolic way. In a sense we need to extract some of this knowledge from the existing tutor and represent it in a usable form for the NL generator — this was done in *DIAG-NLP* by representing the required knowledge via Java classes, as EXEMPLARS is written in Java (SNePS is written in LISP, communication between the different systems is achieved via a Java API).

The need to replicate some of the knowledge present in DIAG may be seen as inconsistent with our earlier claim that we treat the ITS as a blackbox. Actually, we intended that claim to apply only to the tutoring strategies the ITS embodies, not to its underlying knowledge. However, it is certainly true that a full fledged blackbox approach cannot work if the ITS is built without taking into account the knowledge needed for communication. For example, the CIRCSIM tutor embodies domain knowledge at three different levels, because it was found that all three levels are necessary for communication (Khuwaja *et al.* 1992), even if only one level is directly relevant to the material to be mastered.

SNePS make it easy to represent and reason about entire propositions, not just about objects. For example, it is straightforward to represent the various individual propositions that underlie Ex. 1 above, and the causal relations between the failure of the individual parts and the abnormal state of the visual combustion check. Moreover, it is also easy to represent the contrast relation between the two complex propositions (A) and (B). Finally, because propositions are full fledged entities in the representation, they can become part of the discourse model, and be referred to with appropriate referential expressions. In this version of the generator, we implemented the GNOME algorithm to generate referential expressions (Kibble & Power 2000).

This revised version of the generator renders the same facts underlying Figure 2 as shown in Figure 3. The deictic *This* is generated by the GNOME algorithm and is used to refer to the proposition representing the abnormal state of the visual combustion check indicator; this cuts down on some of the repetitiveness of the feedback generated by both *DIAG-orig* and *DIAG-NLP*, cf. Figure 2. However, the indefinite articles in Figure 3 are incorrect (the algorithm we implemented does not take into account the visual context, or the fact that there is only one part with that description). The contrastive particle *but* is not included because we have not yet implemented exemplars to generate cue phrases; however,

[4] We are using relations from Rhetorical Structure Theory (Mann & Thompson 1988).

```
A visual combustion check indicator is igniting in startup mode.
The visual combustion check indicator igniting in startup mode is abnormal.
Within the furnace system,
This is sometimes caused when a system control module replaceable unit is inop-
erative.
Within the oil burner,
This is never caused when an ignitor assembly replaceable unit is inoperative.
This is sometimes caused when a burner motor, oil filter, oil supply valve, or
oil nozzle is inoperative.
```

Figure 3: Adding a bit more of sophistication to the generator

as soon as we do so, it will be straightforward to generate it, as the appropriate rhetorical relation is included in the SNePS representation of the message to be conveyed.

First observations of human consulting

The aggregation rules we implemented in EXEMPLARS appear to be plausible, but they have no empirical foundation. To understand how a human tutor may verbalize a collection of facts, we are collecting tutoring dialogues between a student interacting with the same DIAG application we have previously discussed and a human tutor. In this experiment the tutor and the student are in different rooms, sharing images of the same DIAG tutoring screen. When the student exercises the consult function the tutor sees the information that DIAG would use in generating its advice — exactly the same information that DIAG gives to EXEMPLARS in *DIAG-NLP*. The tutor then types a response that substitutes for DIAG's response. Although we cannot constrain the tutor to provide feedback that includes all and only the facts that DIAG would have communicated at that specific moment, we can still see the effects of how the tutor uses the information provided by DIAG. As of this writing, we have preliminary observations of several human tutors, consisting of about 200 human responses to DIAG consult requests, in over 20 sessions.

The two most striking patterns we observe in the human-generated advice are 1) they often eschew syntactic aggregation of part lists and instead describe or name functional aggregations of parts, and 2) they give advice on the problem-solving process, either directly or indirectly. In the following examples, the pairs of utterances show two tutors independently describing the same assemblages of parts and giving similar problem-solving advice:

1. Referring to oil nozzle, supply valve, pump, filter, etc:
   a) "…check the other items on the fuel line" [Tutor 1]
   b) "…follow the path of the oil flow" [Tutor 2]
2. Referring to all the burner parts:
   a) "What are the parts that control the combustion?" [Tutor 1]
   b) "…consider the units that are involved with heating the water" [Tutor 2]

The assemblages we see in the human discourse are not necessarily represented in the training documentation or the functional diagrams on the DIAG screen; it appears the tutors are constructing them. In general the assemblages seem to be fixed collections. But the tutor sometimes constructs an impromptu subset according to the discourse context, as in "the valve is open, so you have to check the point below the filter," which appears to be a reference to the parts in the fuel line "below" the filter.

The problem-solving advice generally conforms to the patterns of "point-to" and "convey-information" hints observed by (Hume *et al.* 1996).

Some of the other phenomena we have observed:

- In contrast with DIAG, tutors less often mention parts that *cannot* be causing the problem (e.g., the ignitor assembly in Figure 2), except when the student consults precisely on those parts.
- Tutors frequently introduce devices for inter-turn coherence. For example, two adjacent turns were introduced by "not a good choice" and "better choice," respectively. Another turn was introduced by "the question is now," indicating the reasoning was in some way following from the previous turn.
- The human tutors occasionally justify a statement, frequently by appealing to causal reasoning. For example, one tutor wrote "The oil filter is normally clean. A dirty and clogged oil filter *blocks the flow of oil* and should be replaced" (emphasis added). By contrast, DIAG merely states whether a broken oil filter can cause the problem, without interpolated explanation.

As our experiments with human tutors continue, we should be able to produce a more complete catalog of language and discourse phenomena. Of particular interest, given our emphasis on aggregation, is the parts assemblages the tutors use, especially how they are described and when they are invoked, and how to organize the knowledge the system needs in order to replicate human tutors.

Acknowledgements. This work is supported by grants N00014-99-1-0930 and N00014-00-1-0640 from the Office of Naval Research, Cognitive, Neural and Biomolecular S&T Division. We are grateful to CoGenTex Inc., in particular to Mike White, for making EXEMPLARS available to us; and to Michael Scott for suggesting the binomial cumulative distribution.

References

Bateman, J. A. 1994. KPML: The KOMET-Penman (Multilingual) Development Environment. Technical report, Institut für Integrierte Publikations- und Informationssysteme (IPSI), GMD, Darmstadt. Release 0.6.

Di Eugenio, B., and Trolio, M. J. 2000. Can simple sentence planning improve the interaction between learners and an intelligent tutoring system? In *Building Dialogue Systems for Tutorial Applications (AAAI Fall Symposium)*. American Association for Artificial Intelligence.

Elhadad, M. 1993. FUF: the universal unifier – user manual version 5.2. Technical Report CUCS-038-91, Columbia University.

Evens, M. W.; Spitkovsky, J.; Boyle, P.; Michael, J. A.; and Rovick, A. A. 1993. Synthesizing tutorial dialogues. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, 137–140. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Freedman, R. K. 1999. Atlas: a plan manager for mixed-initiative, multimodal dialogue. In *AAAI99 Workshop on Mixed-Initiative Intelligence*. Orlando, FL: American Association for Artificial Intelligence.

Graesser, A. C.; Wiemer-Hastings, K.; Wiemer-Hastings, P.; Kreuz, R.; and the Tutoring Research Group. 2000. Autotutor: A simulation of a human tutor. *Journal of Cognitive Systems Research*.

Hume, G. D.; Michael, J. A.; Rovick, A. A.; and Evens, M. W. 1996. Hinting as a tactic in one-on-one tutoring. *Journal of the Learning Sciences* 5(1):23–47.

Khuwaja, R. A.; Evens, M. W.; Rovick, A. A.; and Michael, J. A. 1992. Knowledge representation for an intelligent tutoring system base on a multilevel causal model. In Frasson, C.; Gauthier, G.; and G.I.McCalla., eds., *Intelligent Tutoring Systems, Second International Conference*.

Kibble, R., and Power, R. 2000. Nominal generation in GNOME and ICONOCLAST. Technical report, Information Technology Research Institute, University of Brighton, Brighton, UK.

Kim, J. H.; Glass, M.; and Evens, M. W. 2000. Learning use of discourse markers in tutorial dialogue for an intelligent tutoring system. In *COGSCI 2000, Proceedings of the 22nd Annual Meeting of the Cognitive Science Society*.

Kintsch, W. 1998. *Comprehension. A paradigm for cognition*. Cambridge University Press.

Mann, W. C., and Thompson, S. 1988. Rhetorical Structure Theory: toward a Functional Theory of Text Organization. *Text* 8(3):243–281.

McRoy, S. W.; Channarukul, S.; and Ali, S. 2000. Text realization for dialog. In *Building Dialogue Systems for Tutorial Applications (AAAI Fall Symposium)*. American Association for Artificial Intelligence.

Moore, J. D.; Lemaire, B.; and Rosenbloom, J. A. 1996. Discourse generation for instructional applications: Identifying and exploiting relevant prior explanations. *Journal of the Learning Sciences* 5(1):49–94.

Munro, A. 1994. Authoring interactive graphical models. In de Jong, T.; Towne, D. M.; and Spada, H., eds., *The Use of Computer Models for Explication, Analysis and Experiential Learning*. Springer Verlag.

Rosé, C. P.; Di Eugenio, B.; and Moore, J. D. 1999. A dialogue based tutoring system for basic electricity and electronics. In *AI-ED 99, Proceedings of the 9th International Conference on Artificial Intelligence in Education*.

Shapiro, S., and Rapaport, W. 1992. The SNePS Family. *Computers and Mathematics with Applications, Special Issue on Semantic Networks in Artificial Intelligence, Part 1* 23(2–5).

Towne, D. M. 1997. Approximate reasoning techniques for intelligent diagnostic instruction. *International Journal of Artificial Intelligence in Education*.

Trafton, J. G.; Wauchope, K.; Raymond, P.; Deubner, B.; Stroup, J.; and Marsch, E. 1997. How natural is natural language for intelligent tutoring systems? In *Proceedings of the Annual Conference of the Cognitive Science Society*.

White, M., and Caldwell, T. 1998. Exemplars: A practical, extensible framework for dynamic text generation. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, 266–275.

Pedagogical Content Knowledge in a Tutorial Dialogue System to Support Self-Explanation

Vincent Aleven, Octav Popescu, and Kenneth R. Koedinger

Human-Computer Interaction Institute

Carnegie Mellon University

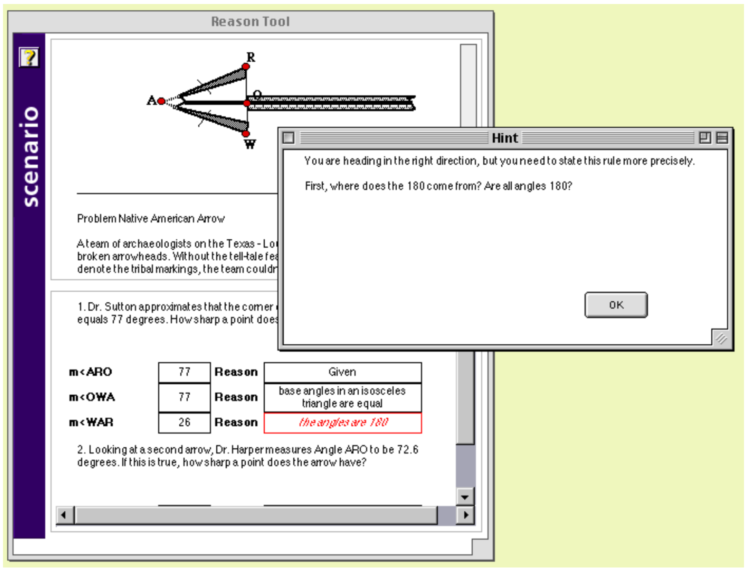
aleven@cs.cmu.edu, octav@cmu.edu, koedinger @cs.cmu.edu

Abstract: We are engaged in a research project to create a tutorial dialogue system that helps students learn through self-explanation. Our current prototype is able to analyze students' general explanations of their problem-solving steps, stated in their own words, recognize the types of omissions that we often see in these explanations, and provide feedback. Our approach to architectural tradeoffs is to equip the system with a sophisticated NLU component but to keep dialogue management simple. The system has a knowledge-based NLU component, which performed with 81% accuracy in a preliminary evaluation study. The system's approach to dialogue management can be characterised as "classify and react". In each dialogue cycle, the system classifies the student input with respect to a hierarchy of explanation categories that represent common ways of stating complete or incomplete explanations of geometry rules. The system then provides feedback based on that classification. We consider what extensions are necessary or desirable in order to make the dialogues more robust.

INTRODUCTION

Self-explanation is an effective metacognitive strategy. Explaining examples or problem-solving steps helps students learn with greater understanding (Chi, et al., 1989; 1994; Berardi-Coletta, et al., 1985; Gagne & Smith, 1962). Yet few students are good self-explainers, even when prompted (Renkl, et al., 1998). So how can we leverage self-explanation to improve learning in actual classrooms? The AI & Education literature provides evidence that self-explanation can be supported effectively by 2nd-generation tutors (Aleven et al 1999; Conati & VanLehn, 2000). However, these systems did not interact in natural language. It is plausible that students will learn even better when they explain in their own words. Natural language allows for flexible expression of partial knowledge: Students can show what they know and the tutor can help them to construct more complete knowledge. Also, articulation forces attention to relevant features. Finally, combining visual and verbal learning modes may create "dual codes" in memory which may facilitate recall (Paivio, 1986). However, it appears that these potential advantages will not be fully obtained without tutoring or giving feedback to students. When students worked with a tutor version that prompted them to explain their steps in their own words, but did not check explanations, they often ignored these prompts and provided almost no good explanations (Aleven & Koedinger, 2000b).

We are preparing to test the hypothesis that students learn better when they explain in their own words and receive feedback on their explanations. To this end we are developing a tutorial dialogue system, the Geometry Explanation Tutor, that assists students as they generate general explanations of their problem-solving steps in their own words. The system engages students in a restricted form of dialogue to help them improve explanations that are not sufficiently precise. We have a working prototype and are starting a phase of pilot testing. The Geometry Explanation



You are heading in the right direction, but you need to state this rule more precisely.

First, where does the 180 come from? Are all angles 180?

Problem Native American Arrow

A team of archaeologists on the Texas - Lo
broken arrowheads. Without the tell-tale fe
denote the tribal markings, the team couldr

1. Dr. Sutton approximates that the corner
equals 77 degrees. How sharp a point does

m<ARO	77	Reason	Given
m<OWA	77	Reason	base angles in an isosceles triangle are equal
m<WAR	26	Reason	*the angles are 180*

2. Looking at a second arrow, Dr. Harper measures Angle ARO to be 72.6 degrees. If this is true, how sharp a point does the arrow have?

Figure 1: The Geometry Explanation Tutor

Tutor is built on top of an existing 2nd-generation system for geometry problem solving, the PACT Geometry Tutor (Aleven et al., 1999), which is currently in use in about five schools in the Pittsburgh area and elsewhere.

In designing the architecture of the system, we are faced with a number of choices. Thus we find ourselves asking the question, as phrased in the call for papers, “Where is the biggest bang for the buck?” A significant architectural decision has been to equip the system with a fairly sophisticated NLU component (Popescu & Koedinger, 2000). A detailed understanding of the explanations is needed if the system is to provide detailed feedback. A second decision has been to keep the system’s dialogue planning and management component as simple as possible, but in Einstein’s words, no simpler than that. We follow the approach taken by Heffernan and Koedinger (2000) in developing Ms. Lindquist, an algebra symbolization tutor, and focus on identifying the *pedagogical content knowledge* needed to help students produce accurate and complete explanations. By pedagogical content knowledge they mean domain-specific strategies that experienced human tutors use to help students deal with common difficulties and to scaffold students’ problem-solving efforts. Pedagogical content knowledge also includes knowledge about students, their typical errors and typical, often rugged, pathways to learning success.

We foresee that a tutor that helps students to generate accurate geometry explanations needs to have knowledge about (1) how to provide good and detailed comments that help students to improve explanations that are incomplete and (2) how to lead students to good explanations if they have difficulty getting started. So far, we have focused on the first need. The analysis of several small corpora of student explanations indicated that students explanations of geometry tend to be incomplete more often than wrong. The system therefore has a hierarchy of explanation categories that represent common ways of stating full and partial explanations of geometry rules. It decides what feedback to give to student explanations primarily by classifying them into this hierarchy. While it is not difficult to see the limitations of the current system, it is not easy to predict what improvements will give the greatest bang for the buck. Thus, in extending the system, we plan to be guided by results of frequent preliminary evaluation and pilot studies, adding more sophisticated mechanisms or strategies only when the data suggest that they will improve students’ learning.

In this paper, we describe the current architecture of the Geometry Explanation Tutor and illustrate its current capabilities by means of dialogue examples. We present results from a

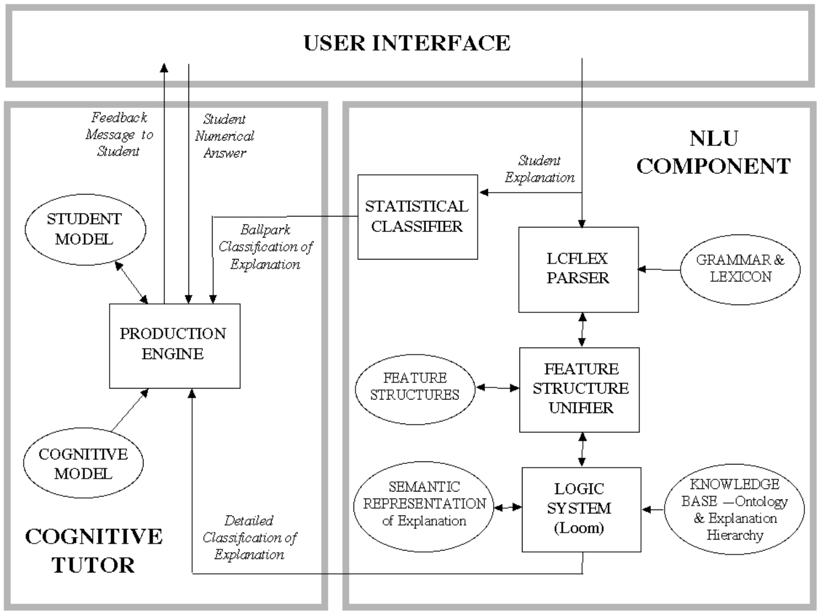


Figure 2: Architecture of the Geometry Explanation Tutor

preliminary evaluation of the accuracy of the system's NLU component. Finally, we discuss what limitations need to be addressed most urgently: What pedagogical content knowledge we will need to add and how far we will have to push the system's dialogue management architecture.

THE GEOMETRY EXPLANATION TUTOR

The Geometry Explanation Tutor covers one of the six units that make up the curriculum of the original PACT Geometry Tutor, namely, the unit that deals with the geometric properties of angles. The Geometry Explanation Tutor provides problem-solving support, just like other Cognitive Tutors (Koedinger, et al., 1997). It monitors students as they work through problems and provides assistance in the form of feedback and context-sensitive hints. Unlike other Cognitive Tutors, the Geometry Explanation Tutor requires that students explain their steps and engages students in a restricted form of dialogue in order to help students state geometry rules accurately (Popescu & Koedinger, 2000; Aleven, et al., in press). The system has been pilot-tested with 20 of our colleagues and staff and with two high-school students.

System Architecture

The Geometry Explanation Tutor is based on the standard Cognitive Tutor architecture (Anderson et al., 1995), augmented with a NLU component (see Figure 2). In each dialogue cycle, the NLU component creates a semantic representation of the student's explanation and *classifies* that representation with respect to the system's hierarchy of explanation categories. The Cognitive Tutor module then checks whether the student's explanation focuses on the right geometry rule and decides how to *react* (i.e., what feedback to give to the student).

An important knowledge source is the hierarchy of explanation categories, which constitutes the system's pedagogical content knowledge. The explanation categories in this hierarchy represent ways of stating each geometry rule correctly, as well as frequently occurring ways of stating rules incorrectly. An excerpt of this hierarchy is shown in Figure 3. Each node represents a class of explanations that have the same meaning, but may have vastly different surface forms. A canonical example of a sentence that falls in each category is shown in each node. Explanation

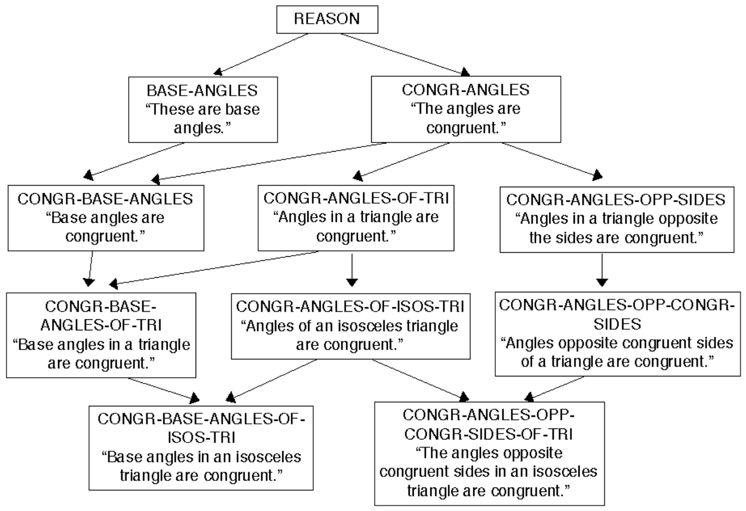

REASON
BASE-ANGLES "These are base angles."
CONGR-ANGLES "The angles are congruent."
CONGR-BASE-ANGLES "Base angles are congruent."
CONGR-ANGLES-OF-TRI "Angles in a triangle are congruent."
CONGR-ANGLES-OPP-SIDES "Angles in a triangle opposite the sides are congruent."
CONGR-BASE-ANGLES-OF-TRI "Base angles in a triangle are congruent."
CONGR-ANGLES-OF-ISOS-TRI "Angles of an isosceles triangle are congruent."
CONGR-ANGLES-OPP-CONGR-SIDES "Angles opposite congruent sides of a triangle are congruent."
CONGR-BASE-ANGLES-OF-ISOS-TRI "Base angles in an isosceles triangle are congruent."
CONGR-ANGLES-OPP-CONGR-SIDES-OF-TRI "The angles opposite congruent sides in an isosceles triangle are congruent."

Figure 3: Excerpts from the explanation hierarchy, represented in the system's knowledge base

categories at the bottom of the hierarchy represent correct and complete ways of stating the isosceles triangle rule. Explanations categories higher up in the hierarchy represent progressively more incomplete ways of stating this rule. The hierarchy also includes information about how to respond to student explanations. Attached to each category is a feedback message that is appropriate when an explanation by the student is classified under that category. We have identified about 140 explanation categories, related to the 25 geometry rules that make up the tutor's Angles unit. A key point is that these categories were driven from observations of real student data, contained in several corpora of student explanations. Thus, this rich network captures categories that occur frequently as learners progress towards success.

The hierarchy is implemented as a Loom knowledge base (MacGregor, 1991). This knowledge base also contains an ontology of the domain, which consists of geometry objects such as angles and lines, as well as relations such as congruency, adjacency, etc. The ontology covers the material of the Angles unit of the tutor curriculum. Currently, the knowledge base contains definitions for about 310 concepts and 90 relations.

The NLU component parses student input using a unification-based approach. We employ the LCFLEX parser, an active chart parser (Rose & Lavie, 1999), in combination with a feature structure unifier. We have developed a grammar of about 200 rules. The parser and unifier build a feature structure encoding the syntax of the sentence. They also direct Loom to build a semantic representation. In the process, the Loom classifier tests the coherence of the semantic representation with respect to semantic constraints expressed in the system's domain ontology. When a coherent semantic representation has been constructed, Loom classifies it with respect to the explanation categories.

The Cognitive Tutor module decides how to respond to the student. First, the tutor determines which geometry rule the student should be explaining. This rule must be one that justifies the current problem-solving step. The tutor determines the set of applicable geometry rules by running its cognitive model of geometry problem solving. This model captures the typical and desired knowledge of geometry students, represented in the form of production rules. The tutor then selects an appropriate feedback message as follows. If the student explanation is a complete statement of a relevant geometry rule, the tutor accepts the explanation. If the explanation is only a partial statement, the tutor selects an appropriate feedback message. Usually, this will be the feedback message associated with the category under which the explanation was classified. If the explanation is only the name of a geometry rule, the tutor asks

Table 1: Example dialogue with the Geometry Explanation Tutor. Shown on the left are a student's successive attempts at explaining the given geometry rule (underlining shows what was added since the previous attempt) and the tutor's responses. Shown on the right is the explanation category under which the tutor classified the student's explanation.

1.	**S:** congruent	NIL
2.	**T:** The Geometry Tutor is not able to understand your explanation. Sorry! Please add to your explanation or type something different. Please state a GENERAL geometry rule. Do not mention the names of the angles in the problem.	
3.	**S:** the base angles are congruent	CONGRUENT-BASE-ANGLES
4.	**T:** You are getting there, but you need to be more precise. Are base angles always congruent?	
5.	**S:** the base angles are congruent here	CONGRUENT-BASE-ANGLES
6.	**T:** You are getting there, but you need to be more precise. Are base angles always congruent?	
7.	**S:** Base angles in a triangle are congruent	CONGRUENT-BASE-ANGLES-OF-TRIANGLE
8.	**T:** Well.... Almost! Are the base angles in EVERY triangle congruent?	
9.	**S:** The base angles in an isosceles triangle are congruent	CONGRUENT-BASE-ANGLES-OF-ISOSCELES-TRIANGLE
10.	**T:** [ACCEPTED as correct explanation. Students get non-verbal positive feedback.]	

the student to state the rule. Finally, if the explanation focuses on the wrong geometry rule, the tutor will inform the student of this fact.

We have begun to experiment with the integration of a statistical Naïve Bayes text classifier (Mitchell, 1997, Ch. 6). When the knowledge-based NLU component fails to produce an analysis, the statistical classifier is used to determine whether the student's explanation is in the ballpark (i.e., focuses on the correct geometry rule). If so, the tutor will print a feedback message saying that the student appears to be on the right track. This helps in dealing with unexpected input, as is illustrated below. We are looking for further ways to leverage the statistical text classifier.

EXAMPLE DIALOGUES WITH THE GEOMETRY EXPLANATION TUTOR

The Geometry Explanation Tutor's capabilities are illustrated in two dialogues shown in Tables 1 and 2. These dialogues are adapted from protocols collected during a session in which about 20 of our colleagues and staff worked with the system. In order to make the dialogs reflect the system's current capabilities, we have replaced the system's responses with responses generated by the *current* system version. The downside is that the dialogues are not 100% realistic.

In the first example, the tutor helps a student in explaining the isosceles triangle theorem, which says (stated somewhat informally) that the base angles in an isosceles triangle are congruent. Starting with a minimum effort explanation (step 1) and testing the tutor's sense of humor along the way (steps 3-5), the student incrementally refines an explanation until the explanation is accepted by the tutor (step 10). The tutor produces appropriate responses using its pedagogical content knowledge, captured in the explanation hierarchy. For example, category CONGRUENT-BASE-ANGLES-OF-ISOSCELES-TRIANGLE represents correct and complete statements of the isosceles triangle rule (see Figure 3, bottom left). Student explanations in this category get the thumbs up from the tutor, provided of course that the isosceles triangle rule justifies the problem-solving step being explained (see step 9 and 10). On the other hand, category CONGRUENT-BASE-ANGLES-OF-TRIANGLE represents statements such as "the

Table 2: Example Dialogue with the Geometry Explanation Tutor

1.	**S:** 180 is what you get when you add the interior angles	EQUAL-180
2.	**T:** Can you be more specific? What is 180? And what tells you that this is so?	
3.	**S:** interior	NIL
4.	**T:** It appears that you are getting at the right rule, the triangle sum rule. But the tutor has difficulty understanding your explanation. Please make small changes .	
5.	**S:** triangle sum	REFERENT
6.	**T:** It looks like you have the right idea. Can you state this geometry rule more completely?	
7.	**S:** triangle sum is 180	SUM-OF-TRIANGLE-180
8.	**T:** Very close. The sum of what exactly is equal to 180? Not of the triangle itself but of its... ?	
9.	**S:** add the interior angles is 180	INTERIOR-ANGLES-180
10.	**T:** That is good, but you need to be more specific. Which interior angles do you mean?	
11.	**S:** interior angles add up to 180	INTERIOR-ANGLES-SUM-180
12.	**T:** Very good. The sum of interior angles of WHAT is 180 degrees?	
13.	**S:** interior angles of the triangle add up to 180	ANGLES-SUM-OF-TRIANGLE-180
14.	**T:** [ACCEPTED as correct explanation. Students get non-verbal positive feedback.]	

base angles of a triangle are congruent," which are incomplete (Figure 3, left, one level from bottom). In response to statements in this category, the tutor generates a feedback message that suggests that the statement is an overgeneralization, provided again that the rule that needs to be explained is the isosceles triangle rule (steps 7 and 8). The feedback message is the one associated to the given category. Generally, the dialogue is smooth. Of course one prefers to avoid tutor responses like "the tutor does not understand," as shown in step 2 or instances where the tutor repeats itself in subsequent dialogue turns, (steps 4 and 6). Under the given circumstances however these responses were quite appropriate.

In the second dialogue (shown in Table 2), the student starts off rather well. The student's first explanation attempt could have been completed simply by adding "of a triangle" at the end, so that the sentence reads "180 is what you get when you add the interior angles of a triangle." Unfortunately, the tutor feedback does not make this clear. After a minimalist strategy in step 3, the student quickly gets on track again and gradually improves the explanation until it is complete in step 13. With the exception of the first tutor message, the tutor's feedback seems appropriate and helpful. The reason that the tutor did not produce a more helpful message in step 2 is that the NLU component currently does not handle the construction "is what you get when you add …"

The example illustrates that the statistical text classifier sometimes enables the tutor to produce a more helpful feedback message than it could if it only had the knowledge-based NLU component. In step 3, the student's answer "interior" is not classified under any explanation category by the knowledge-based NLU component. The statistical classifier however returns TRIANGLE-SUM as the most likely category. This enables the tutor to acknowledge (in step 4) that the student is on the right track ("it appears that you are getting at the right rule, the triangle sum rule"). Without the statistical classifier, the tutor could only have said, "the tutor does not understand your explanation."

Table 3: Classification accuracy of the knowledge-based NLU component

Classification result	*N*	%
Classified correctly		
Explanation was actually complete	194	29.9
Explanation was actually incomplete	227	35.0
Explanation was actually a reference	102	15.7
Classified under overly general category		
Explanation was actually complete	60	9.3
Explanation was actually incomplete	38	5.9
Classified incorrectly	3	0.5
Not classified	24	3.7
Total	648	100

Further, the example dialogue illustrates that the tutor accepts some common forms of abbreviations that students make. Students very often say "the angles are 180" when they mean strictly speaking that the measures of the angles are 180. This is a form of metonymy, the phenomenon of referring to a concept by means of a related concept [Jurafsky & Martin, 2000]. A prime example is "New York called" where it was the guy or girl from that city who called. The tutor accepts common forms of metonymy without complaint. For example, the tutor responds to the sentence "interior angles add up to 180" (step 11) as if the student said "the measures of interior angles add up to 180". Similarly, the sentence "a linear pair is 180" is treated as if the student had said "the measures of the angles in a linear pair are 180" (double metonymy).

However, the tutor is not so accommodating that it accepts all abbreviations or elliptical expressions. Nor should it be. A tutor whose goal it is to help students learn to "speak mathematics" should be helpful but should also insist on a certain level of precision in language. For example, the tutor does not interpret "triangle sum is 180" (step 7) as meaning "the sum of the measures of the angles of a triangle is 180 degrees," even though one might argue that this is what was meant. The tutor does not assume that the sum was implied.

The challenge in dealing with metonymy is to construct a semantic representation that observes semantic constraints even though the input sentence does not. Some forms of metonymy are dealt with in our system by having the NLU component recover the missing structure (Popescu & Koedinger, 2000). The semantic representations that are constructed in this way are the same as those that would have been built if the same sentence had occurred without abbreviations. The abbreviations that the tutor needs to know about are handled differently. The semantic representation that is constructed reflects the abbreviation, so that the tutor can take notice. For example, for the sentence "a triangle sums to 180" the NLU component builds a semantic representation that does not include angles. This gives the tutor the option of insisting on a more complete explanation (step 8). Exactly on what level of precision the system should insist is not easy to determine. The choice should ultimately be driven by what is best for student learning.

PRELIMINARY EVALUATION OF THE NLU COMPONENT

We conducted a preliminary evaluation of the classification accuracy of the knowledge-based NLU component. As test data we used a corpus of 648 explanations collected during a session in which about 20 of our colleagues and staff worked on the system. The explanations were labeled by hand by two authors of the paper, who assigned each explanation to the most specific category in the explanation hierarchy to which it belongs. New categories were invented as needed. A total of 138 categories were used, 92 of which were represented in the data set. The system's knowledge base was then extended to include definitions for many of the new categories. The system was then run to classify the 648 explanations.

Table 4: Example Dialogue with the Geometry Explanation Tutor

9.	**S:** that two angles in a isosceles triangle are equal	CONGRUENT-ANGLES-OF-ISOSCELES-TRIANGLE
10.	**T:** Correct, except for one thing: are all three angles in an isosceles triangle congruent?	
11.	**S:** that two congruant angles in a isosceles triangle are equal	CONGRUENT-ANGLES-OF-ISOSCELES-TRIANGLE
12.	**T:** Correct, except for one thing: are all three angles in an isosceles triangle congruent?	
15.	**S:** that two congruant base angles are equal	CONGRUENT-BASE-ANGLES
16.	**T:** You are getting there, but you need to be more precise. Are base angles always congruent?	
19.	**S:** base angles in a isosceles triangle are equal	CONGRUENT-BASE-ANGLES-OF-ISOSCELES-TRIANGLE
20.	**T:** [ACCEPTED as correct explanation. Students get non-verbal positive feedback.]	

As shown in Table 3, the system classified 81% of the explanations correctly. Of the correctly classified explanations, 80% (421 out of 523) fall under categories of full or partial explanations. The rest were references, meaning that the student stated only the name of a geometry rule. The system classified a further 15% of the explanations under categories that were too general, although not strictly wrong. The remaining 4% of explanations were either not classified at all or under categories that were unrelated to the correct category. Interestingly, when the system classified an explanation as being complete, it was correct 100% of the time.

An accuracy score of 81% is very encouraging, especially given the fact that we are dealing with a very fine-grained classification task, where small differences between categories are the rule rather than the exception. There are two caveats. First, the accuracy results were obtained with a data set that was used during the development of the system. Second, this data set was obtained with subjects who are more advanced than those in the target population (high-school students). More work is needed before we expect to see the same accuracy score with new data and students from the target population. Nonetheless, the results provide a preliminary indication that knowledge-based NLU is an appropriate choice for analysing geometry explanations.

LIMITATIONS OF CLASSIFY AND REACT

Currently, the system's response in each dialogue turn is based only on the classification of the student's last explanation attempt. No further context is taken into account. This way, the tutor can respond to the types of omissions we often see in students' explanations and can sometimes produce a sense of coherent dialogue, as illustrated in the examples. However, one does not have to look far to see the limitations of the approach. For example, the system has no memory of what went on before in the dialogue. It is therefore not able to detect situations where students stagnate or regress and will not respond adequately. Also, the tutor is not able to engage in multi-turn strategies or to lead students through a directed line of reasoning, as human tutors often do. But which of these limitations is most worth addressing? Which will have the greatest impact on learning? In the next section, we illustrate two multi-turn tutorial strategies for the current domain and discuss how we plan to explore their utility. In the current section, we illustrate a form of stagnation and discuss how the tutor can be made to respond in a more helpful manner.

In contrast to the previous examples, the current dialogue example (see Table 4) involves a student of the same age as students in the target population (10th graders), although the student was definitely better than average. Further, the responses shown are the actual system responses.

We skip the first part of the dialogue and omit two steps from the dialogue that contained spelling errors—at the time, the tutor did not have spelling correction, but currently it does. In step 9, the student is very close to the correct explanation ("two angles in an isosceles triangle congruent"). The explanation is missing only the term "base angles". The tutor's response in step 10, "Are all *three* angles in an isosceles triangle congruent?" was designed to hint at that fact but is not quite appropriate. The message writer had not anticipated that the student might use the word "two" in his explanation. This can be fixed simply by crafting a better message.

Next, the student adds the word "congruant" [sic] to the explanation (step 11). This is not an improvement over the previous explanation attempt (step 9). One might say it is worse, because the purported explanation is now a tautology. This may well be a sign that the student does not fully understand what he typed. The tutor however is oblivious to the problem and simply repeats the feedback message that it gave before, in spite of the fact that this message did not help (step 13). This is unsatisfactory. A likely cause of the problem is that the student does not know the concept of base angles or at least does not think of using the term in this context[1]. The tutor should realize that and provide more helpful feedback. For example, the second time around, the tutor should have said: "WHICH angles in an isosceles triangle are congruent? What is the right term to use here?" If that message again does not help, then the tutor should cut to the chase and simply tell the student to use the term "base angles" and explain what the term means.

This problem needs to be addressed. In the current example, the students quickly gets back on track, but this will not happen as easily with all students. The tutor needs to be able to help students over the hump if they get stuck. In order to be able to detect this kind of stagnation, the tutor needs to keep a history list of the categories under which the student's explanation attempts were classified. Further, it needs to have multiple, increasingly specific messages associated with each explanation category. As a further way to help avoid stagnation, the system needs to provide helpful hints when the student clicks the "Help" button. These messages need to be sensitive to the current state of the student's explanation and need to be coordinated with the feedback messages, so that help and feedback can be interleaved in any order. Quite possibly, this can be achieved simply by using the same sequences of messages for help and feedback. We will have to make sure that students do not abuse the feedback messages and help messages to let the tutor provide the explanation without making a real effort to come up with an explanation themselves. (We have observed similar minimum effort tactics in the patterns of students hint use, see Aleven & Koedinger, 2000a.) One way to discourage such tactics is not to make the tutor too forthcoming with information, even if this may conflict with other maxims of good tutoring.

TUTORIAL STRATEGIES TO HELP STUDENTS EXPLAIN

A category of pedagogical content knowledge that is currently not addressed in the tutor is strategies that can be used to lead students to explanations, if they are struggling to produce an explanation. We look at two examples of such strategies.

Using a "procedural replay" as a bridge

We have observed that students often provide a "procedural replay" when asked to provide an explanation of a geometry problem-solving step. That is, they merely restate the arithmetic that was used to come up with a numeric answer. Such an "explanation" of course does little to justify why that was the correct thing to do. However, the prevalence of procedural replays suggest that one way to help students who are struggling to provide an explanation is (a) to ask how they came

[1] One might argue that the student's response in step 11 indicates that he does not know the meaning of the term "congruent", an important term in geometry. However, the use of the term congruent might reflect a superficial repair strategy, a shot in the dark. The student does not know how to improve the explanation in step 9. The tutor has just introduced the term "congruent" (step 10), so why not use that term?

up with their numeric answer (thus inviting them to provide a procedural replay) and then (b) to ask how they knew that this procedure was the right thing to do. This strategy thus grounds the explanation task in an activity that the student has just completed successfully. This strategy is not unlike the "concrete instance" strategy in Ms. Lindquist (Heffernan & Koedinger, 2000). The strategy is illustrated in the following hypothetical dialogue about the triangle sum theorem.

Tutor: How did you find your answer?
Student: subtracted 76 and 61 from 180.
Tutor: So you subtracted two other angles from 180. Which angles?
Student: XZY and YXZ
Tutor: OK, And why did you use the 180? How did you know that this was the right number to use?
Student: Because of the triangle.
Tutor: OK. So there is a triangle and there is the 180. Can you state a general rule about triangles?
Student: The triangle totals 180.
Tutor: …

One of us tried out this tutorial strategy on a few occasions while assisting students during a classroom study with a tutor version that did not provide feedback (Aleven & Koedinger, 2000b). The strategy seemed natural and effective at least for some of the geometry rules. We note that this strategy goes well beyond what the Geometry Explanation Tutor currently can do. First, the dialogue shown above does not conform to the assumption made in the current system that each utterance by the student is an attempt at providing a complete explanation. Further, the dialogue shown above seems to require some form of dialogue planning. Before we add such facilities to the system, we need to know more about how effective this tutorial strategy (and others that human tutors may use) will be in improving students' learning. A Wizard of Oz kind study, in which a human tutor communicates with the student through the computer interface (simulating the smarts of a computer tutor) would be appropriate to explore that question further.

Counterexamples

A second tutorial strategy that is likely to be useful involves the use of counterexamples when students state overgeneralizations. Consider the angle addition theorem, which says: "The measure of an angle formed by two adjacent angles is equal to the sum of the measures of those two angles." This rule appears to be one of the more difficult rules to explain. Let us assume that the student needs to explain the angle addition rule and starts out with the kind of intuitive, informal explanation that students tend to provide with some frequency (shown below). How to get the student to state this geometry rule more formally? The judicious use of counterexamples may be a lot better than just asking "can you state your explanation in more mathematical terms?"

Student: the big angle is equal to the sum of the two small ones.
Tutor: OK - true in this diagram. Now let us look for a general rule. Did you mean that whenever you have a big angle and two small ones, the big one is the sum of the two small ones?
Student: Yeah, like it is in the problem here.
Tutor: Let me change this problem diagram a little bit [in a drawing of the diagram, moves the two adjacent angles apart and draws a third angle]. Here we have a big angle and two small ones. According to your theory, the big angle is the sum of the two small ones. Correct?
Student: no the two small ones need to be next to each other
Tutor: OK, they need to be adjacent. So you are saying that whenever you have two adjacent angles and a third angle, the third angle is equal to the sum of the other two?
Student: Yes.

Tutor: Well, let me change this diagram one more time. [Moves the two angles so that they are adjacent again but keeps a third angle.]

It seems very worthwhile to have the tutor show counterexamples, as this would expose students to a venerable and ubiquitous strategy of mathematical reasoning. On the other hand, it is not clear that this kind of extended dialogue should be carried out in the context of problem-solving exercises. This might distract too much from solving the geometry problem at hand. It may be better to have students engage in activities that focus explicitly on creating and stating definitions based on examples and non-examples of the term to be defined. The (very influential) curricular guidelines of the National Council of Teachers of Mathematics include this kind of mathematical argumentation as an important objective (NCTM, 1989).

The current system is not capable of generating the dialogue shown above, for much the same reasons that it cannot generate the "procedural replay as bridge" dialogue. This is not to say that the current system could not present counterexamples. Certainly, its feedback messages could be modified to do just that. However, within the classify-and-react framework, it may be quite difficult to recover when the student does not understand the counterexample. Also, it may be difficult to stick to the strategy when a first counterexample gets the student to go only halfway (as illustrated in the dialogue shown above). At this point it is not quite clear how important it is to have such capabilities. This question is best explored by means of a Wizard of Oz study.

CONCLUSION

We are involved in a project to develop a tutorial dialogue system that helps students learn through self-explanation. The main purpose is to help students learn geometry problem-solving skills with greater understanding. A secondary purpose is to get students to learn to "speak mathematics", that is, to help students to learn basic math communication skills. With respect to the field of cognitive science, our goal is to test the hypothesis that self-explanation has a greater impact on learning if students explain in their own words, rather than through a structured computer interface, such as a menu.

Our development strategy is to equip the system with a sophisticated NLU component and to keep the dialogue management component simple. Thus, our efforts so far have focused on developing an NLU component that provides detailed analysis of students' explanations. A preliminary evaluation study showed that this component accurately classifies 81 % of student explanations and somewhat reasonable classifications on all but 4% of student explanations. Work on the NLU component continues in order to improve its performance.

Currently, the system's pedagogical content knowledge consists of a hierarchy of explanation categories, which represent common ways of providing complete or partially complete statements of geometry rules. The system uses this knowledge in each dialogue turn to classify the student's explanation and to select appropriate feedback messages. This approach enables the tutor to respond to the types of omissions we often see in students' explanations and produce reasonably effective dialogue. However, some extensions are needed in order to make the dialogue more robust. The tutor must be able it to detect situations where a student stagnates and is not able to improve her explanation even after receiving tutor feedback. The tutor must be able to help students over the hump in such situations. To do so, the system needs to have a dialogue history and multiple levels of feedback messages associated with each explanation category. It will also be necessary to coordinate the hint messages and the feedback messages.

At this point, it is not quite clear that the tutor needs to engage in multi-turn tutorial strategies such as "use procedural replay as bridge" and "counterexamples". To investigate the importance of such strategies, we will follow the 3rd-generation methodology exemplified by many other projects, namely, to study expert human tutors and perform Wizard of Oz studies. More importantly, we will build alternative versions of the tutor and experimentally test whether our changes lead to greater student learning.

Acknowledgements

This research is sponsored by an NSF grant to the Center for Interdisciplinary Research on Constructive Learning (CIRCLE), a joint research center located at the University of Pittsburgh and Carnegie Mellon University.

REFERENCES

Aleven V., Popescu, O. & Koedinger, K. R. (to appear). Towards tutorial dialog to support self-explanation: Adding natural language understanding to a cognitive tutor. To appear in *AI-ED 2001.*

Aleven, V. & Koedinger, K. R. (2000a). Limitations of Student Control: Do Student Know when they need help? In G. Gauthier, C. Frasson, and K. VanLehn (Eds.), *Proceedings ITS 2000* (pp. 292-303). Berlin: Springer Verlag.

Aleven, V. & Koedinger , K. R. (2000b). The Need for Tutorial Dialog to Support Self-Explanation. In C. P. Rose & R. Freedman (Eds.), *Building Dialogue Systems for Tutorial Applications, Papers of the 2000 AAAI Fall Symposium* (pp. 65-73). Menlo Park, CA: AAAI Press.

Aleven, V., K. R. Koedinger, and K. Cross. Tutoring Answer Explanation Fosters Learning with Understanding. In *Proceedings of AIED-99,* edited by S. P. Lajoie and M. Vivet, 199-206. Amsterdam: IOS Press, 1999

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences, 4*, 167-207.

Berardi-Coletta, B., Buyer, L. S., Dominowsky, R. L., & Rellinger, E. R. (1995). Metacognition and Problem-Solving: A Process-Oriented Approach. Journal of Experimental Psychology: Learning *Memory, and Cognition,* 21 (1) 205-223.

Chi, M. T. H., Bassok, M., Lewis, M. W., Reimann, P., & Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science,* 13, 145-182.

Chi, M. T. H., N. de Leeuw, M. Chiu, and C. Lavancher (1994). Eliciting self-explanations improves understanding. *Cognitive Science, 18*, 439-477.

Conati C. & VanLehn K. (2000). Toward Computer-Based Support of Meta-Cognitive Skills: a Computational Framework to Coach Self-Explanation. *International Journal of Artificial Intelligence in Education,* 11.

Gagné, R. M., & Smith, E. C. (1962). A Study of the Effects of Verbalization on Problem Solving. Journal of Experimental Psychology, 63(1), 12-18.

Heffernan, N. T. & Koedinger, K. R. (2000). Intelligent Tutoring Systems are Missing the Tutor: Building a More Strategic Dialog-Based Tutor. In C. P. Rose & R. Freedman (Eds.), *Building Dialogue Systems for Tutorial Applications, Papers of the 2000 AAAI Fall Symposium* (pp. 14-19). Menlo Park, CA: AAAI Press.

Koedinger, K. R., Anderson, J. R., Hadley, W. H., & Mark, M. A. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, *8,* 30-43.

MacGregor, R, (1991). The Evolving Technology of Classification-Based Knowledge Representation Systems. In J. Sowa (ed.), *Principles of Semantic Networks: Explorations in the Representation of Knowledg.* San Mateo, CA: Morgan Kaufmann.

Mitchell, T, 1997. *Machine Learning.* McGraw-Hill.

NCTM, 1989. Curriculum and Evaluation Standards for School Mathematics. National Council of Teachers of Mathematics. Reston, VA: The Council. See also http://standards-e.nctm.org/index.htm.

Paivio, A. (1986). *Mental representations: A dual coding approach.* New York: Oxford University Press.

Popescu, O. & Koedinger , K. R. (2000). Towards Understanding Geometry Explanations. In C. P. Rose & R. Freedman (Eds.), *Building Dialogue Systems for Tutorial Applications, Papers of the 2000 AAAI Fall Symposium* (pp. 80-86). Menlo Park, CA: AAAI Press.

Renkl, A., Stark, R., Gruber, H., & Mandl, H. (1998). Learning from Worked-Out Examples: the Effects of Example Variability and Elicited Self-Explanations. *Contemporary Educational Psychology,* 23, 90-108.

Rose, C. P. & Lavie, A. (1999). LCFlex: An Efficient Robust Left-Corner Parser. User's Guide, Carnegie Mellon University.

Introducing RMT: A dialog-based tutor for research methods

Peter Wiemer-Hastings and Kalloipi-Irini Malatesta
University of Edinburgh

Send correspondence to:

Peter Wiemer-Hastings
Division of Informatics
2 Buccleuch Place
The University of Edinburgh
Edinburgh EH8 9LW
Scotland

peterwh@cogsci.ed.ac.uk

AutoTutor is an intelligent tutoring system that interacts with students in the way that human tutors do: with natural language dialog (Graesser, Wiemer-Hastings, Wiemer-Hastings, Kreuz, & the TRG, 1999). It presents questions and responses with a talking head which uses speech production and gesture to give graded feedback. It understands student replies using surface clues and latent semantic analysis (LSA) (Landauer & Dumais, 1997). It directs a student through a tutoring session using responses from its curriculum script which represents its knowledge of the domain (Wiemer-Hastings, Graesser, Harter, & the Tutoring Research Group, 1998).

AutoTutor has been shown to be effective in aiding student learning. Compared to control subjects who simply reread a chapter, students who used AutoTutor had improved learning and memory of the lessons by .5 to .6 standard deviations (Graesser, Person, Harter, & the TRG, 2001).

The AutoTutor system has limitations however. Despite the fact that the questions in AutoTutor's curriculum script are meant to be "deep reasoning" questions, its approach to the dialog is very shallow. There are two main reasons. First, AutoTutor's language analysis mechanism is limited. LSA tells AutoTutor how similar a particular student answer is to some desired good answer. But if the student answer is not so close, the system does not know where it is lacking. More detailed analysis of the student answer could change the types of responses AutoTutor makes. Instead of just moving on to the next point when the current one was matched sufficiently, an improved understanding mechanism would support more intelligent generation of follow-up questions.

The second limitation to the depth of AutoTutor's conversations is its subject matter. Computer Literacy attempts only to familiarize students with the basic concepts of computers, and does not get into any deep issues. Thus, many of AutoTutor's questions have a short-answer feel; the ideal answers can be summed up in one or two words. A more complicated domain would allow much more interesting discussions.

For these reasons, we are developing RMT, the Research Methods Tutor. RMT is aimed at undergraduate psychology or cognitive science students who are studying research methods. RMT takes a case-based approach. It presents a research question to the student, and asks the student how to go about evaluating it. This domain supports in-depth discussions of the student's approach to addressing the research question. It also allows the system to develop the student's analogical reasoning. RMT brings in related research paradigms to help the student infer both similarities and differences with their approach.

RMT also makes use of the Structured LSA (SLSA) language analysis system (Wiemer-Hastings, 2000). This system uses part-of-speech tagging, anaphora resolution, and shallow parsing to break apart input sentences into their subject, verb, and object segments and to replace pronouns with their antecedents. This technique provides a better match to human similarity judgments than standard LSA does (Wiemer-Hastings & Zipitria, 2001). Additionally, this allows the tutoring system to know what part of the student's answer matched an expected good answer, and what part did not match. This will allow RMT have a more effective dialog with the student by finding the "nugget of truth" in the answer, and leading the student to the complete correct answer.

References

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

Graesser, A., Wiemer-Hastings, K., Wiemer-Hastings, P., Kreuz, R., & the TRG (1999). AutoTutor: A simulation of a human tutor. *Journal of Cognitive Systems Research*, *1*, 35–51.

Landauer, T., & Dumais, S. (1997). A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, *104*, 211–240.

Wiemer-Hastings, P. (2000). Adding syntactic information to LSA. In *Proceedings of the* 22nd *Annual Conference of the Cognitive Science Society*, pp. 989–993 Mahwah, NJ. Erlbaum.

Wiemer-Hastings, P., Graesser, A., Harter, D., & the Tutoring Research Group (1998). The foundations and architecture of AutoTutor. In Goettl, B., Halff, H., Redfield, C., & Shute, V. (Eds.), *Intelligent Tutoring Systems, Proceedings of the 4th International Conference*, pp. 334–343 Berlin. Springer.

Wiemer-Hastings, P., & Zipitria, I. (2001). Rules for Syntax, Vectors for Semantics. In *Proceedings of the* 23rd *Annual Conference of the Cognitive Science Society* Mahwah, NJ. Erlbaum. In press.