MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Using a Model of Collaborative Dialogue to Teach Procedural Tasks

Jeff Rickel, Neal Lesh, Charles Rich, Candace L. Sidner, Abigail Gertner

TR2000-40 December 2000

Abstract

Previous research on building intelligent tutoring systems has not taken full advantage of 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, based on the apprenticeship model of learning, built using an application-independent collaboration manager, called Collagen. A 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.

10th International Conference on Artificial Intelligence in Education,San Antonio, Texas, May 2001

Submitted in December 2000.

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
[3] MITRE Corporation, 202 Burlington Road, Bedford, MA, 01730
gertner@mitre.org

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, 19, 21]) 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, 11, 20]), 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, 12, 22]) 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 [15], a middleware system based on a long line of research on collaborative discourse [8, 6, 7, 5, 11]. 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 chooses 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 [16, 17]. 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 6 and 12-28 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 [16]. 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 6 and 12-28). 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

(1) **Paco** "Let's try another scenario. Let's engage engine one."

(2) **Student** "OK."

(3) **Paco** "You take it from here."

(4) **Student** "We should start generator one."

(5) **Paco** "Good."

(6) **Paco** "First, press the alarm reset button to reset the alarm light.

(7) **Student** Presses the alarm reset button.

(8) **Paco** "Right."

(9) **Paco** "You take it from here."

(10) **Student** Presses the on button.

(11) **Paco** "Good job."

(12) **Paco** "The next step of engaging engine one is to stop engine two."

(13) **Student** "OK."

(14) **Paco** "First, press the disengage button on engine two."

(15) **Student** Presses the disengage button.

(16) **Paco** "Great."

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

(18) **Student** "OK."

(19) **Paco** "Go ahead."

(20) **Student** Closes fuel valve one on engine two.

(21) **Paco** "Nice."

(22) **Student** Closes fuel valve two on engine two.

(23) **Paco** "Good."

(24) **Student** "OK."

(25) **Paco** "Next, press the off button on engine two to stop the engine."

(26) **Student** Presses the off button.

(27) **Paco** "Great."

(28) **Paco** "We succeeded in stopping engine two."

(29) **Student** "OK."

(30) **Paco** "You take it from here."

(31) **Student** Opens fuel valve one on engine one.

(32) **Paco** "Nice."

(33) **Student** Opens the bleed valve on engine one.

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

(35) **Paco** "Let's return to opening the fuel valves on engine one."

(36) **Student** Opens fuel valve two on engine one.

(37) **Paco** "Right."

(38) **Student** Presses the engage button on engine one.

(39) **Paco** "Whoops. You can't do that yet."

(40) **Student** "What next?"

(41) **Paco** "You need to press the on button."

(42) **Student** Presses the on button.

(43) **Paco** "Good job."

(44) **Paco** "You take it from here."

(45) **Student** Presses the engage button on engine one.

(46) **Paco** "Excellent."

(47) **Paco** "We succeeded in engaging engine one. We're done with this scenario."

Figure 1: An example dialogue in which Paco teaches steps in the current task (e.g., lines 6, 12, 17), teaches that subtasks are finished (e.g., lines 28, 47), provides help when requested (e.g., line 41), passes initiative to the student when she should know what to do next (e.g., lines 3, 9, 19), provides positive feedback (e.g., lines 5, 8, 46), diagnoses and critiques incorrect actions (e.g., line 39), and helps the student stay focused on the subtask currently being taught (e.g., line 35).

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.

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

Engaging engine one.
1 Paco says "Let's engage engine one."
2 Student says "OK."
3 Paco says "You take it from here."
Done successfully starting generator one.
4 Student says "We should start generator one."
5 Paco says "Good."
Done student successfully pressing the alarm reset button to reset the alarm light.
6 Paco says "First, press the alarm reset button to reset the alarm light."
7 Student presses the alarm reset button to reset the alarm light.
8 Paco says "Right."
9 Paco says "You take it from here."
Done student successfully pressing the on button to start generator one.
10 Student presses the on button to start generator one.
11 Paco says "Good job."
Stopping engine two.
12 Paco says "The next step of engaging engine one is to stop engine two."

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

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 [11] with plan recognition, which can recognize when an act contributes to a DSP through one or more implicit acts [10].

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 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

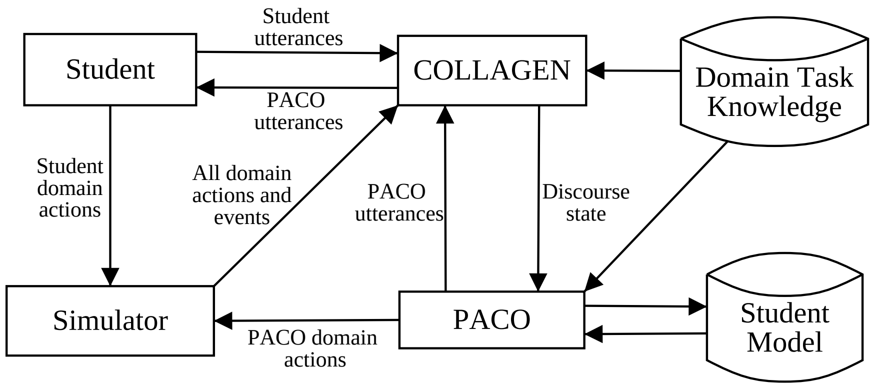

Figure 3: Paco's Architecture

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 [18]. 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 each of Paco's tutorial behaviors is 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 which 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 [18] 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). Our future work includes making these utterances more instructive (in particular, the Teach Step utterances will be more verbose).

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

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

Tutorial Action 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.
Teach step *(rank 6)*	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 7)*	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 7)*	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 7)*	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

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 33 in Figure 1), then Paco will add a Correct Focus action (e.g., line 35) to the agenda. Paco might choose to execute a higher-ranked element on the agenda first (e.g., line 34) 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 39). Finally, when the student asks for help (e.g., line 40) this pushes a discourse purpose of helping the student onto the stack which remains there until the agent provides the help (e.g., line 41).

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 28). 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.

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 Conclusion

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. For example, we expect future versions of Paco to support richer discussion of non-primitive actions, parameter values, causal dependencies among steps in a plan, and the ramifications of incorrect actions. 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 [9, 13, 14], 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.

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

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

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] Jaime 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] Allan Collins, John Seely Brown, and Susan E. Newman. Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics. In L.B. Resnick, editor, *Knowing, Learning, and Instruction: Essays in Honor of Robert Glaser*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1989.

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

[4] Ira 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] Barbara J. Grosz and Candace L. Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204, 1986.

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

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

[10] Neal Lesh, Charles Rich, and Candace 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.

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

[12] Natalie K. Person, Arthur C. Graesser, Roger J. Kreuz, Victoria Pomeroy, and the Tutoring Research Group. Simulating human tutor dialog moves in autotutor. *International Journal of Artificial Intelligence in Education*, 12, 2001. Forthcoming.

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

[14] Andrew Ravenscroft and Rachel 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.

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

[16] Jeff 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.

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

[18] Candace 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.

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

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

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

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