In *Proc. Sixth International Conference on Intelligent Tutoring Systems,* pp. 542-551, Springer-Verlag, June 2002.

Collaborative Discourse Theory as a Foundation for Tutorial Dialogue

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. Research on intelligent tutoring systems has not leveraged general models of collaborative discourse, even though tutoring is inherently collaborative. Similarly, research on collaborative discourse theory has rarely addressed tutorial issues, even though teaching and learning are important 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 a domain-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 [20] 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]), with an equally long history, focuses on computational models of human dialogue for collaborative tasks.

Effective instructional dialogue is best understood as a mixed-initiative interaction between tutor and student, in which the student works to solve problems while the tutor monitors the student's progress, providing feedback and hints when needed. Good human tutors aim to maintain progress in problem solving while supporting the student's sense of control and self-confidence, a balancing act that requires a high degree of interactivity and collaboration. Thus, it is natural to expect that a model of collaborative dialogue would provide a solid framework on which to build a conversational ITS. Furthermore, the flexibility of such a framework in supporting a wide range of collaborative interactions allows for the implementation of a variety of pedagogical strategies.

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., [4, 12, 21]) 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, 16], a middleware system based on a long line of research on collaborative discourse [8, 7, 6, 11]. Collagen maintains a model of the discourse state shared by the user (e.g., student) and the computer agent (e.g., tutor). Agents constructed using Collagen use the discourse state to generate an agenda of candidate *discourse acts*, including both utterances and domain actions, and then choose one to utter or perform. Our domain-independent tutorial agent, Paco (Pedagogical Agent for Collagen), has already been used in several independent research projects [1, 3, 19].

Paco teaches students procedural tasks in simulated environments, building on ideas from earlier tutoring systems. While Paco engages in slightly more sophisticated conversations than previous such tutors, our primary contribution is to show how various tutorial behaviors can be expressed as rules for generating candidate discourse acts in Collagen. Translating tutorial behaviors into the framework of CDS is a first step towards building tutoring agents that can leverage advances in collaborative discourse theory. A secondary goal of this work is to assess Collagen's value for building tutorial agents, both in terms of the theory it reflects and the software architecture it supports.

2 Pedagogical Approach

Paco supports simulation-based training, in which students learn tasks by performing them in a simulation of the real work environment. Paco repeatedly chooses a scenario, i.e., a task to perform starting from a particular simulation state, and then works through it with the student. Ideally, students should learn to flexibly apply well-defined procedures in a variety of situations.

Figure 1 shows an example dialogue from one implemented application 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 taught the student how to engage an engine. Now, Paco is teaching the same procedure under slightly more complex conditions: (1) a high vibration alarm has occurred on the gas turbine generator, shutting down the generator, 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 the desired engine. The remainder of the paper will use this example dialogue to illustrate aspects of our design.

Our pedagogical approach is based on the apprenticeship model of learning [2]. To implement this approach, Paco uses a student model to dynamically interleave demonstration and coached practice, using the approach introduced by Rickel [17]. As the student and Paco progress through a task, Collagen repeatedly identifies the set of valid next steps. Paco consults the student model to see whether the student has sufficient knowledge to choose one of them. (Any action that appears in the set of valid next steps is considered correct if performed by the student). If so, it expects the student to take the next step, and provides assistance only if the student requests it or makes a mistake. If not, Paco intervenes and teaches the student what to do next (e.g., lines 8-12 and 18-

(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."

Fig. 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., line 34), and keeps the student focused on the current subtask (e.g., line 42).

35). Thus, as Paco and the student work through tasks, initiative passes back and forth between them based on the student's prior experience. Whenever Paco decides that the initiative should shift, it lets the student know through verbal comments (e.g., "You take it from here").

Paco is fundamentally a tutor for supporting coached practice, but there are certain strategic pedagogical decisions within this approach that are flexible and easily changed. For example, while it is capable of demonstrating actions on the simulation, in Figure 1 Paco does not demonstrate any of the steps but rather prompts the student to perform them. Whether or not to demonstrate is a strategic decision that can be made based on properties of the domain and the student population. The contribution of this work is not to argue for or against particular instructional strategies, but rather to show that such decisions can easily be implemented in a collaborative dialogue system using straightforward discourse generation rules.

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 that define a partial order over them. 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 this level of generality.

Based on the work of Grosz and Sidner [7], Collagen partitions the discourse state into three interrelated components: the linguistic structure, the attentional state, and the intentional structure. The linguistic structure 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 subtask). For example, lines 8 through 14 in Figure 1 represent a segment whose purpose is to reset the alarm, and this segment is part of a larger segment (lines 6 through 17) whose purpose is to start the generator. The segmentation scheme does not restrict the nature of the dialogue; Collagen allows segments with an unknown purpose, it allows dialogue about a single purpose to be split into multiple segments when it is interrupted by unrelated dialogue, and a single utterance can even include clauses belonging to different segments.

The attentional state represents what the user and agent are talking about or working on at a given moment. Tutors must track the attentional state in order to interpret the student's actions and utterances (i.e., guide plan recognition [10]), tailor suggestions, choose actions and utterances that serve as natural progressions of the dialogue (or properly mark unexpected shifts with cue phrases [7]), and recognize unnatural focus shifts by the student that may suggest misconceptions. Representing and tracking attentional state is simple for tutors that retain tight control over a fixed topic progression, but it becomes more complex in mixed-initiative dialogues where the student and tutor have more freedom to initiate topics for discussion and choose the execution order for tasks and subtasks.

Following Grosz and Sidner [7], Collagen represents the attentional state as a stack of discourse purposes called the focus stack. 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 allows Collagen to handle interruptions and digressions. 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.").

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. As with attentional state, such information is most needed in mixed-initiative tutoring systems that do not impose a strict ordering on decisions. Collagen represents the intentional structure as *plan trees*, which are a partial implementation of SharedPlans [6]. 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. If it does not contribute to the current DSP, Collagen must determine whether it is a shift in focus or an interruption. 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, 9].

Collagen performs "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. Additionally, a domain author can define new types of errors or add explicit buggy recipes.

Figure 2 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). Paco and Collagen can be used in a new domain given only the appropriate domain task knowledge and an interface to the simulator. Collagen makes very few assumptions about the simulator. Primarily, it assumes that the user and agent can both perform domain actions (e.g., open a fuel valve) and can observe the actions taken by each other. 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 [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 start the generator"), 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 (e.g., "What should the speed be?"), 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.

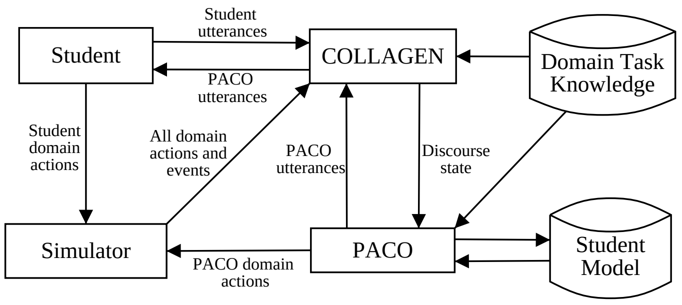

Fig. 2. Paco's Architecture

Since Collagen operates on statements in the artificial discourse language, it does not rely on any particular method of gathering input utterances or generating output utterances. Collagen provides a menu-driven user window that allows the user to construct utterances in English. We also use commercial speech recognition software and grammar rules as a shortcut to the menu-driven interface. To generate output utterances, Collagen uses a combination of domain-independent and (optional) domain-specific text templates. It displays the output strings in a window and speaks them using commercial speech synthesis software.

4 Tutorial Behaviors as Collaborative Discourse Acts

Table 1 summarizes our progress in mapping tutorial behaviors into a CDS framework. Due to space restrictions, it only describes a representative subset of Paco's current behaviors. The first column is a ranked list of 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. The third column 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).

Paco uses several elements of the discourse state to generate its discourse acts, including the focus stack, the initiative, and plan trees. The focus stack is used, for example, to avoid teaching a step before its purpose is in focus. The focus stack also indicates when the student has interrupted the current task prematurely, which causes Paco to generate a candidate 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 task (thus popping the current focus from the shared focus stack) while there are still valid steps within Paco's private focus (e.g., line 40 in Figure 1), Paco will add a Correct Focus action (e.g., line 42) to the agenda, although Paco might choose to execute a higher-ranked element on the agenda first (e.g., line 41).

The conditions for generating discourse acts are easy to compute given the data structures maintained by Collagen. For example, several acts operate on the *valid next actions*, which refers to steps that can be executed next based on precondition and ordering constraints.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 [5] 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.

Paco's domain knowledge can include recipes that achieve the subgoal of explaining why a task step should be performed. 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. Collagen's facilities for executing recipes in a collaborative setting are used to complete the explanation started by an

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 it 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 it 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 interruption *(rank 2)*	each step ω that is an 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 such that 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

Explain Why action. This general approach could be used to support other types of multi-turn tutorial strategies as well [1].

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 [14]. 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 Evaluation

We assessed our progress on Paco through a formative evaluation with seven users. They had no prior experience with Paco or research in user interfaces, AI, or dialogue systems. Users received a single page of instructions, and then Paco instructed them on a sequence of seven tasks in the Gas Turbine Engine simulator, with a variety of initial conditions. The target task was to start engine one from an initial condition in which engine two is running. Paco first taught four subtasks of this target task, then taught the target task, and then had the users repeat this task twice more. Users completed the tasks in 20 to 30 minutes. Next, they filled out a survey and discussed their opinions with an interviewer.[1] To test Paco's effectiveness, five of the users were asked to teach the target task back to the interviewer.

The results suggest that Paco is a reasonable tutor. Most users commented positively on Paco's overall teaching skills, explanations, feedback, clarity of communication, and ability to understand what the user was doing. One user wanted the ability to ask "why" questions, three users wanted more rationales for actions (although two users praised Paco for providing rationales), and two users commented that Paco should not praise actions that the user should know well. After their session with Paco, users felt somewhat or very confident in their ability to perform the task. Of the users who taught the task back to the interviewer, three made no errors, one made a few minor errors but completed the task, and one forgot to turn off engine two so failed to complete the task.

6 Conclusion

By mapping tutorial acts into the framework of collaborative discourse theory, Paco provides a foundation for cross-fertilization between these two research areas. As our next step, we plan to broaden the types of tutorial acts we consider to include those used in recent analyses of human tutorial dialogues [13]. We are especially interested in the relationship of hinting and prompting to collaborative discourse theory.

[1] We thank Clifton Forlines for running and helping to design the evaluation.

References

1. B. Cheikes and A. Gertner. Teaching to plan and planning to teach in an embedded training system. In *Proceedings of the Tenth International Conference on Artificial Intelligence in Education*, pages 398–409. IOS Press, 2001.
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, 1989.
3. J. Eisenstein and C. Rich. Agents and GUIs from task models. In *Proceedings of the Sixth International Conference on Intelligent User Interfaces*, pages 47–54, 2002. ACM Press.
4. R. K. Freedman. *Interaction of Discourse Planning, Instructional Planning and Dialogue Management in an Interactive Tutoring System*. PhD thesis, Northwestern University, 1996.
5. I. P. Goldstein. Overlays: A theory of modelling for computer-aided instruction. Artificial Intelligence Laboratory Memo 495, MIT, Cambridge, MA, 1977.
6. B. J. Grosz and S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.
7. B. J. Grosz and C. L. Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204, 1986.
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. N. Lesh, C. Rich, and C. Sidner. Collaborating with focused and unfocused users under imperfect communication. In *Proc. 9th Int. Conf. on User Modelling*, pages 64–73, 2001.
10. N. Lesh, C. Rich, and C. L. Sidner. Using plan recognition in human-computer collaboration. In *Proc. 7th Int. Conf. on User Modeling*, pages 23–32, 1999.
11. K. E. Lochbaum. A collaborative planning model of intentional structure. *Computational Linguistics*, 24(4):525–572, 1998.
12. 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*, 12:23–39, 2001.
13. R. Pilkington, editor. *Special Issue on Analysing Educational Dialogue Interaction*, volume 11 of *International Journal of Artificial Intelligence in Education*, 2000.
14. C. Rich, N. Lesh, and J. Rickel. A plug-in architecture for generating collaborative agent responses. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, New York, 2002. ACM Press. Forthcoming.
15. 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.
16. C. Rich, C. L. Sidner, and N. Lesh. Collagen: Applying collaborative discourse theory to human-computer collaboration. *AI Magazine*, 22(4):15–25, 2001.
17. 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.
18. 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.
19. C. L. Sidner and M. Dzikovska. Hosting activities: Experience with and future directions for a robot agent host. In *Proceedings of the Sixth International Conference on Intelligent User Interfaces*, pages 143–150, New York, 2002. ACM Press.
20. E. Wenger. *Artificial Intelligence and Tutoring Systems*. Morgan Kaufmann, 1987.
21. B. P. Woolf. *Context-Dependent Planning in a Machine Tutor*. PhD thesis, Department of Computer and Information Science, University of Massachusetts at Amherst, 1984.