

# Integrating Task Information into the Dialogue Context for Natural Language Mathematics Tutoring\*

**Mark Buckley**

Department of Computational Linguistics  
Saarland University  
66041 Saarbrücken, Germany  
buckley@coli.uni-sb.de

**Dominik Dietrich**

Department of Computer Science  
Saarland University  
66041 Saarbrücken, Germany  
dietrich@ags.uni-sb.de

## Abstract

Natural language dialogue systems model the state of the dialogue in order to be able to react appropriately to a user's utterances. In addition to such a dialogue model, natural language tutorial dialogue systems depend on having access to expert knowledge about the state of the task the student is solving. This is especially important for complex domains such as mathematical theorem proving. In this paper we present a preliminary model of dialogue context which includes an explicit model of the state of the task at hand. The task model contributes information to the choice of tutor actions.

## 1 Motivation

Natural language dialogue systems which form the interface to a computer application must be able to call on domain dependent knowledge in order to function as conversational agents (McTear, 2002). In this situation the dialogue system provides a front-end to a source of domain knowledge such as a timetabling database (Aust et al., 1995; Eckert et al., 1993). Interfacing with a timetabling database is an example of an information-seeking dialogue, in which the goal of the dialogue is to look up and deliver information from the database. In task-oriented dialogue, however, delivering the information from the domain-dependent backend is not the dialogue goal, but rather this information supports and contributes to achieving the task at hand (Rickel et al.,

2000). This motivates the need for modelling the state of the task at hand within the dialogue system (Flycht-Eriksson, 1999; McTear, 2002).

For demanding tasks the source of domain knowledge is an expert system which maintains both a task and a domain model. However general purpose expert systems may model the task at a very detailed level. A solution is to model the task in an abstract way within the dialogue system, using information from the expert system.

One dialogue genre which relies heavily on task and domain modelling is tutorial dialogue. Development is motivated by the effectiveness of natural language in tutorial interactions (Moore, 1993). The expert system is just one part of a large system which also includes for example natural language analysis, generation, and tutorial expertise. Tutorial dialogue systems have been developed for a range of domains including geometry (Aleven et al., 1999), physics (Jordan et al., 2006; Litman and Silliman, 2004), computer literacy (Graesser et al., 1999) and electronics (Zinn et al., 2002). An important common characteristic of these systems is the use of a separate dedicated expert system to model the task at hand. This in turn informs the choice of tutorial behaviour (Zhou et al., 1999), for instance about what feedback should be given to the student.

In this paper we propose a preliminary model of dialogue context which includes an explicit task model for the domain of tutoring mathematics proofs. The task model is abstracted from the task model stored in a mathematical assistance system, which plays the role of expert system. Mathematical theorem proving is open ended, since there are typically many possibly unknown solutions to a given

---

\* This research is supported by the Collaborative Research Centre on *Resource-Adaptive Cognitive Processes*, SFB 378.

problem. This means solutions or solution graphs can not be preauthored and used as a scaffolding for the interaction. From a corpus of student/tutor dialogues we find that there is a large range of possible student inputs and possible tutor reactions to these in a given situation. To be able to generate the types of tutor actions we observe in the corpus in a context sensitive way we need not only a representation of the dialogue (as an interaction between dialogue participants) but also a representation of certain aspects of the task at hand which are necessary to produce pedagogically appropriate behaviour. Our model thus includes the subtask currently being solved, evaluation of latest student contribution, and a record of the tutoring state.

The structure of the paper is as follows. Section 2 introduces the scenario of mathematics tutoring and presents some linguistic data. In Section 3 we outline the modules we assume in the architecture. Section 4 presents the proposed model of dialogue context, including the abstracted task model. Section 5 shows a step-through of an example from the corpus. We briefly outline the implementation status in Section 6 and mention some related approaches in Section 7 before concluding the paper.

## 2 Tutorial Dialogue on Mathematical Proofs

Our research is motivated by results from the DIALOG project (Benzmüller et al., 2003), which has the final goal of natural tutorial dialogue between a student and a mathematical assistance system. The student's task is to build a proof of a mathematical theorem. The student does this by conducting a dialogue with the tutor in which he/she performs utterances which may contain proof steps. Each correct proof step extends the current partial proof. The task is completed when the student has constructed a complete proof of the theorem.

It is clearly necessary to call on separate modules which provide domain specific support to the dialogue manager. Proof steps are analysed by a domain reasoner which can give an evaluation of the step with respect to the current partial proof attempt. When the student is stuck or shows non-understanding of domain concepts, pedagogical expertise is applied to generate domain-specific hints.

- T1:** Please show: If  $A \subseteq K(B)$ , then  $B \subseteq K(A)$ !
- S1:**  $A \subseteq B$ .
- T2:** That's incorrect. You should consider the if-then relation.
- S2:**  $A \subseteq K(K(A))$ .
- T3:** That's correct, but not interesting right now. Do you know what to do with an if-then relation?
- ...

Figure 1: Excerpt from the corpus.

For this a tutorial module decides on pedagogical strategies. A natural language processing module performs an analysis of student utterances.

### 2.1 Some Linguistic Data

We base our analysis on a corpus of such tutorial dialogues (Wolska et al., 2004). We now show an example (translated from German) from the corpus, given in Figure (1). Here  $K$  refers to the complement operation on sets. In **T1** the tutor sets the task of proving the given theorem in the domain of naive set theory. The student begins by asserting that  $A \subseteq B$ , which does not hold. The tutor signals this, and additionally gives a hint to the student to try to do something with the if-then relation. The student responds in **S2** by ignoring the hint and stating  $A \subseteq K(K(A))$ , which is correct but irrelevant to the current proof. The tutor now decides to signal the irrelevance of the previous step and to query the student's knowledge of the concept of if-then relation.

### 2.2 Requirements for the Model

The example shows that the actions of the tutor rely on information about the state of the dialogue and the state of the task itself. After utterance **S1** the tutor is obliged to respond to the student's proof step, and decides, based on the fact that the step was evaluated as incorrect, to inform the student of this fact. In order to additionally issue the hint the tutor needs to know that the next (in this case first) step in the proof involves the if-then relation. Both of these pieces of information should be contained in a model of the theorem proving task.

Utterance **T3**, with its reference to the domain concept if-then relation, can only be generated if the

tutor has access to the fact that the previous hint also addressed the if-then relation. For this reason the model must also contain a representation of what tutorial actions have taken place and how they were presented.

### 3 Elements of the Architecture

We now describe the modules and their interfaces that we will assume are present in the architecture we are developing, shown in Figure 2. The dialogue manager maintains the dialogue context and enables communication between modules for input/output, language interpretation and generation, and domain reasoning. It therefore has the overall control of system execution. The user interface is a simple textual input and output window in which the user can type utterances and see the dialogue history.

#### 3.1 Domain Reasoning

The domain reasoning component of the system is made up of the  $\Omega$ MEGA mathematical assistance system (Siekman et al., 2006) and the proof manager.  $\Omega$ MEGA uses proof planning to either interactively or automatically build proofs of mathematical theorems. Proofs are built within a theory of mathematics, which has a set of inferences which can be applied in the proof planning process. The domain reasoner has a twofold role in the tutoring system: It must maintain and incrementally build up the state of the proof that the student is building and it must check the correctness and relevance of the student's proof steps with respect to this partial proof.

The proof manager acts as a wrapper controlling the interaction between the dialogue manager and the domain reasoner and has access to the proof representation of  $\Omega$ MEGA. We consider the interface to the proof manager to be defined by an API of functions that the dialogue manager can call to get the domain information it needs to enact tutorial strategies.  $\Omega$ MEGA in turn provides an API to the proof manager which includes for example verifying types of proof steps and testing relevance. Given a proof step, the proof manager can evaluate the correctness of the step by checking if the domain reasoner can verify it.  $\Omega$ MEGA verifies proof steps by doing a depth-bounded BFS proof search and keeping exactly those new proof states which are a model of

the student's cognitive state (Dietrich and Buckley, 2007). Relevance of a proof step can be checked by testing whether the step contributes to some successful proof attempt.

A further responsibility of the proof manager is to interface with the domain reasoner to supply the dialogue manager with possible next steps in the proof.  $\Omega$ MEGA attempts to automatically find a proof of the theorem starting from the current proof situation. From the representation of this completion of the proof the proof manager can extract the "next step" that the domain reasoner performed while finding the proof. This next step is described by, for instance, the inference rule that was applied and the formula it was applied to.

#### 3.2 Dialogue Management

The dialogue manager has the role of maintaining the system's dialogue context. It continuously updates the dialogue context to reflect the current state of the interaction between student and tutor. The dialogue context provides the basis for deciding on system action. The dialogue manager therefore contains general conversational expertise as well as knowledge about appropriate pedagogical behaviour.

However it is not enough to just have information about the dialogue. The dialogue manager also needs to have contextual information about the task at hand in order to produce tutorial behaviour such as that shown in Figure 1. Information such as the correctness of the previous step or suggested next steps in the proof are necessary to instantiate the respective tutorial strategies. For these reasons the dialogue manager must store a dialogue context which includes both dialogue level and task level information. For instance, the dialogue manager can reveal a subset of the content of the suggested next step by pointing to the concept addressed, formula rewritten, or inference applied in the step, or some combination of these.

#### 3.3 Analysis of Student's Utterances

For this model we abstract away from the surface form of utterances by assuming that the dialogue manager receives the set of dialogue moves that rep-

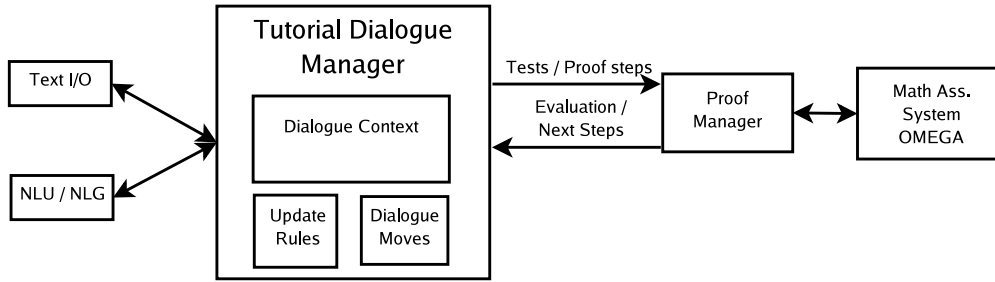


Figure 2: Architecture of the tutorial system.

resents the function of the utterance<sup>1</sup> (see below). In the case that the utterance was a contribution to solving the theorem proving task we have in addition the well-formed formula that the student intended to utter and the type of proof step that the student intended.

The corpus shows that proof steps are embedded in utterances which carry information about the type of the proof step which is important for its verification, such as the phrase “Thus it holds that” or “Hence”. The types of proof step we consider are forward reasoning steps, which deduce new facts, backward reasoning steps, which reduce goals to subgoals, and “let” steps, which introduce new local variables.

#### 4 Integrating the Task Model

We now present the model by way of the example instantiation shown in Figure 3. It shows the state of the dialogue excerpt in example (1) after utterance **T1** has been performed. The content of the dialogue context is split into two parts. The representation of the task is stored in `TASKCONTEXT`, which is domain-dependent and has been defined here for the task of tutoring mathematical proofs. We store the theorem which is to be shown (`TASK`) and the current status of the proof (`TASKSTATUS`), for instance whether it is initialised, partially proved or solved. Since theorem proving is an inherently hierarchical activity, the tutor must have a representation of the subtasks that the student is tackling or will tackle in the future (Rickel et al., 2001). `SUBTASKSTATUS` is a stack of subgoals that have been introduced but not

yet completed.

The mathematical content of the latest student utterance (`LASTPROOFSTEP`) is represented by the formula(s) which were used in the step and the type of the step. These two fields are filled with the results of the natural language analysis of the student’s utterance. As we described above, the domain reasoning component of the system performs an analysis of the proof content of each student utterance. We represent the results of this analysis in `EVALUATION`. We maintain a history of proof steps in `PROOFSTEPHISTORY`.

Information relevant to the tutoring state of the system is stored in the field `TUTORING`. The system stores information about its previous pedagogical behaviour, since this forms the context for future tutor actions. For instance, the system records the number wrong answers given so far by the student so that it can react to student frustration. The number of hints is also recorded. One of the links between task and dialogue information is stored in `HINTHISTORY`; it contains the IDs of the utterances in which hints were given, and thereby makes the type of hints performed available.

The actual dialogue model is stored in `DMODEL`. We use a very simple model of dialogue which only records the last utterance (`LU`), a history of utterances performed by both tutor and student (`HISTORY`), and a stack of dialogue moves that the system intends to perform (`AGENDA`). The last utterance is described by the utterance itself as a string (`UTTERANCE`), the speaker who performed it (`SPEAKER`) and the set of dialogue moves that represents the utterance’s function (`MOVES`). The link between task information and dialogue information in the dialogue context is captured in the `PROOFSTEP`

<sup>1</sup>We refer to (Horacek and Wolska, 2005) for some approaches to the analysis of natural language contributions in a tutorial setting.

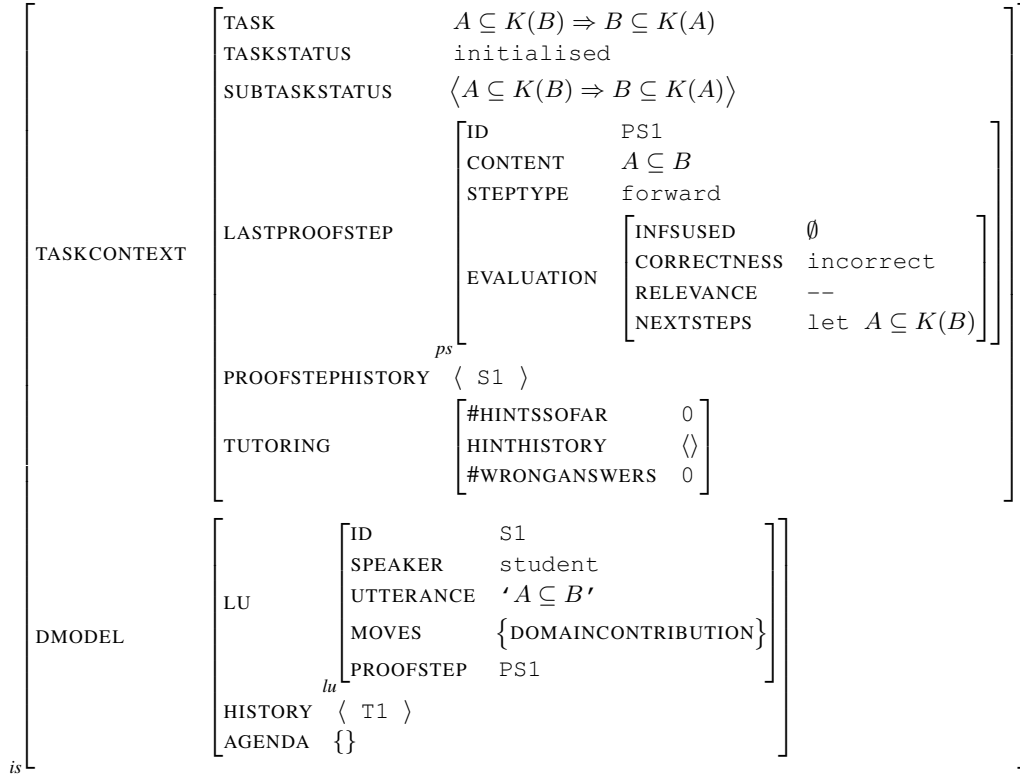


Figure 3: The dialogue context.

slot, which stores the ID of the proof step which was performed in the utterance in question. This information is provided by the graphical user interface and the natural language understanding module.

#### 4.1 Dialogue Moves

A small set of high-level dialogue moves is enough to describe simple interactions in the mathematical tutoring domain, and are presented in Table 1. Students can perform `domaincontribution` to

Dialogue Move	Function
<code>domaincontribution</code>	performs a proof step
<code>requestassistance</code>	requests assistance from tutor
<code>signalcorrect</code>	signals correctness of proof step
<code>signalincorrect</code>	signals incorrectness of proof step
<code>signalirrelevant</code>	signals irrelevance of a proof step
<code>giveawaynextstep</code>	reveals (part of) next step in proof

Table 1: Set of dialogue moves.

make a contribution to the solution being constructed, and `requestassistance` to make a direct request for tutorial help from the tutor. The tutor can signal the correctness, incorrectness or irrel-

evance of the proof step a student has offered, or give away the next step in the solution. The tutor moves can be combined to make a larger turn, such as **T2** in example (1), which realises both `signalincorrect` and `giveawaynextstep`. The tutor moves are taken from a taxonomy of dialogue moves (Fiedler and Tsovaltzi, 2003) tailored to hinting in tutorial applications, which extends the DAMSL taxonomy (Allen and Core, 1997).

#### 4.2 Updating the Task and Dialogue State

The model is updated as a result of utterances performed by either the tutor or the student. Here we give an example of an update rule which illustrates that we need both task and dialogue information.

```
rule( evaluateDomCon,
  [ $/dmodel/lu/speaker = student,
    in($/dmodel/lu/moves, domcon),
    $/taskcontext/lps/steptype = Type,
    $/taskcontext/lps/content = Formulas,
    proofmanager_interface(extendproofstep,
      [Type,Formulas,Evaluation])
  ],
  [ set(/taskcontext/lps/evaluation,
    Evaluation)])
```

Update rules are made up of preconditions which constrain the values in the information state and effects that are carried out when the rule fires. This update rule accesses the proof manager in order to get the evaluation of the student's proof step. It fires in the case that the last utterance was performed by the student and it was a domain contribution. In order to pass the correct information to the proof manager, the rule also needs the type and the content of the proof step. The final constraint is the actual interface to the proof manager. The single effect of the rule is to add the evaluation to the representation of the proof step. This rule shows how dialogue and task information can combine to trigger an information state transition.

## 5 Updating the Dialogue Context for a Fragment of Dialogue

The dialogue context shown in Figure 3 is the state of the dialogue in Figure 1 after utterance **S1** and after the execution of the update rule shown in section 4.2. It shows that the last proof step PS1 was evaluated as incorrect by the domain reasoner. Therefore there is no relevance rating given. The domain reasoner has however provided a suggestion for a possible next step in the proof. The values of the tutoring-relevant slots are still at their initial values. We now show how the dialogue context is updated to reflect the progression of the rest of the dialogue fragment.

The utterance **T2** is selected based on a combination of the dialogue move the student performed, the evaluation of the proof step which was uttered, and the next step suggested by the proof manager. The domaincontribution by the student in **S1** obliges the tutor to address the correctness (or otherwise) of a step. Since the step was incorrect, the tutor must perform the dialogue move `signalincorrect`. Tutorial strategy dictates that a further hint should be provided. This can be triggered for instance by a student model. In addition to the `signalincorrect` the tutor performs a `giveawaynextstep`. The fact that the hint count is 0 leads the tutor to give away as little information as possible; only the domain concept which should be eliminated is given. These rules result in the agenda  $\{\text{signalincorrect}, \text{giveawaynextstep}\}$ , which is verbalised as utterance **T2**.

The student now makes a second attempt at starting the proof in **S2**, which is judged to be irrelevant. The evaluation of the step is integrated into the dialogue context as described above, and the `LAST-PROOFSTEP` information becomes

$$ps \left[ \begin{array}{l} \text{ID} \\ \text{CONTENT} \\ \text{STEPTYPE} \\ \text{EVALUATION} \end{array} \right. \left. \begin{array}{l} \text{PS2} \\ A \subseteq K(K(A)) \\ \text{forward} \\ \left[ \begin{array}{l} \text{INFSUSED} \quad \emptyset \\ \text{CORRECTNESS} \quad \text{correct} \\ \text{RELEVANCE} \quad \text{irrelevant} \\ \text{NEXTSTEPS} \quad \text{let } A \subseteq K(B) \end{array} \right] \end{array} \right]$$

which is again a domain contribution. Due to the performance of the tutorial moves by the tutor in the utterance before, the `TUTORING` section of the dialogue context is now

$$\left[ \begin{array}{l} \#HINTSSOFAR \\ HINTHISTORY \\ \#WRONGANSWERS \end{array} \right. \left. \begin{array}{l} 1 \\ \langle T2 \rangle \\ 2 \end{array} \right]$$

Again a domain contribution was performed, and the tutor should give an evaluation. However in addition to the correctness of the step the tutor considers its irrelevance to the current proof attempt. These constraints lead to the tutor choosing the dialogue moves `signalcorrect` along with `signalirrelevant`. The dialogue context also contains the information that two wrong answers have been offered by the student in a row (`#WRONGANSWERS`). Because of this the tutorial strategy decides to perform another `giveawaynextstep` move, but this time not revealing information about the step, but rather querying the student's grasp of the domain knowledge necessary to perform the step. This decision is based on a combination of dialogue information, tutoring information, and information about the state of the task and the student's previous performance in relation to the task, and results in the agenda  $\{\text{signalincorrect}, \text{signalirrelevant}, \text{giveawaynextstep}\}$ .

The final dialogue context after the fragment above contains the tutoring information

$$\left[ \begin{array}{l} \#HINTSSOFAR \\ HINTHISTORY \\ \#WRONGANSWERS \end{array} \right. \left. \begin{array}{l} 2 \\ \langle T3, T2 \rangle \\ 2 \end{array} \right]$$

along with the utterance history  $\langle T3, S2, T2, S1, T1 \rangle$  and the proof step history  $\langle S2, S1 \rangle$ .

## 6 Development Status

We are currently building a prototype of the tutorial dialogue manager using TrindiKit (Larsson and Traum, 2000). TrindiKit is a dialogue move engine based on the information state update approach. It provides a platform in which different theories of dialogue can be formalised and tested, and supports the integration of modules with the dialogue manager.

A dialogue system built in TrindiKit is characterised by an information state, which is a central storage area in which the dialogue context is represented. Conversational expertise is encoded in a set of update rules, which define possible transitions between information states. They fire based on the performance of dialogue moves by the dialogue participants and based on their constraints on the current information state. The model is implemented as a typed feature structure in TrindiKit's information state. Implementation of tutorial strategies in the form of update rules is ongoing.

## 7 Related Work

Our model is similar to other tutorial dialogue systems for mathematics (Aleven et al., 1999; Callaway et al., 2006; Zinn et al., 2002) in that it uses a dedicated domain reasoner, but differs in that we maintain an explicit task model with the dialogue manager. This is also the case for tutorial dialogue systems for other formal tutoring domains such as physics (Litman and Silliman, 2004). In a more general dialogue setting, the MALIN system incorporates domain knowledge into a timetable information system by including a domain knowledge manager with which the dialogue manager interfaces (Flycht-Eriksson and Jönsson, 2000).

## 8 Conclusion and Future Work

The work presented here shows that augmenting a simple dialogue model with task-specific information makes it possible to account for some of the tutor behaviour found in a corpus of student/tutor dialogues. This extension was enabled by the use of the information state update approach to dialogue management, which allowed us easily integrate task information which is supplied to the dialogue manager by different system modules.

The proposed extension to a simple dialogue context raises a number of possibilities for future work. We will continue to investigate the integration of task information with a more sophisticated dialogue model, such as one of those which have been formalised within the information state update approach (Poesio and Traum, 1997; Matheson et al., 2000). This could allow us to support more general dialogue phenomena while being able to retain pedagogical expertise. A further goal is to abstract away from representations which are specific to the theorem proving task. We will attempt to adapt our account of dialogue context to the general case of tutorial dialogue. A related issue will be the generalisation of the interface to the task-specific domain reasoner. We further intend to evaluate the model experimentally when enough tutorial expertise has been encoded.

## References

- Vincent Aleven, Kenneth R. Koedinger, and K. Cross. 1999. Tutoring answer explanation fosters learning with understanding. In S. P. Lajoie and M. Vivet, editors, *Artificial Intelligence in Education, Open Learning Environments: New Computational Technologies to Support Learning, Exploration, and Collaboration*, proceedings of AIED-99, pages 199–206, Amsterdam. IOS Press.
- James Allen and Mark Core. 1997. Draft of DAMSL: Dialogue act markup in several layers. *DRI: Discourse Research Initiative*, University of Pennsylvania.
- Harald Aust, Martin Oerder, Frank Seide, and Volker Steinbiss. 1995. The Philips Automatic Train Timetable Information System. In *Speech Communication*, volume 17, pages 249–262.
- Chris Benz Müller, Armin Fiedler, Malte Gabsdil, Helmut Horacek, Ivana Kruijff-Korbayová, Manfred Pinkal, Jörg Siekmann, Dimitra Tsovaltzi, Bao Quoc Vo, and Magdalena Wolska. 2003. Tutorial dialogs on mathematical proofs. In *Proceedings of the IJCAI Workshop on Knowledge Representation and Automated Reasoning for E-Learning Systems*, pages 12–22, Acapulco.
- C. Callaway, M. Dzikovska, C. Matheson, J. Moore, and C. Zinn. 2006. Using Dialogue to Learn Math in the LeActiveMath Project. In *Proceedings of the ECAI Workshop on Language-Enhanced Educational Technology*, pages 1–8, Riva del Garda Italy, August.
- Dominik Dietrich and Mark Buckley. 2007. Verification of Proof Steps for Tutoring Mathematical Proofs. In

- Rose Luckin and Ken Koedinger, editors, *Proceedings of the 13th International Conference on Artificial Intelligence in Education*, Los Angeles. To appear.
- W. Eckert, T. Kuhn, H. Niemann, S. Rieck, A. Scheuer, and E. Schukat-Talamazzini. 1993. A Spoken Dialogue System for German Intercity Train Timetable Inquiries. In *Eurospeech*, pages 1871–1874.
- Armin Fiedler and Dimitra Tsovaltzi. 2003. Automating Hinting in Mathematical Tutorial Dialogue. In *Proceedings of the EACL-03 Workshop on Dialogue Systems: Interaction, Adaptation and Styles of Management*, pages 45–52, Budapest.
- Annika Flycht-Eriksson and Arne Jönsson. 2000. Dialogue and Domain Knowledge Management in Dialogue Systems. In *Proceedings of the 1st SIGdial workshop on Discourse and Dialogue*, pages 121–130, Morristown, NJ, USA. Association for Computational Linguistics.
- Annika Flycht-Eriksson. 1999. A Survey of Knowledge Sources in Dialogue Systems. *Electronic Transactions on Artificial Intelligence*, 3(D):5–32.
- Arthur C. Graesser, Katja Wiemer-Hastings, Peter Wiemer-Hastings, and Roger Kreuz. 1999. Autotutor: A simulation of a human tutor. *Cognitive Systems Research*, 1:35–51.
- Helmut Horacek and Magdalena Wolska. 2005. Interpretation of Mixed Language Input in a Mathematics Tutoring System. In *Proceedings of AIED-05 Workshop on Mixed Language Explanations in Learning Environments*, pages 27–34.
- Pamela Jordan, Maxim Makatchev, Umarani Pappuswamy, Kurt VanLehn, and Patricia Albacete. 2006. A Natural Language Tutorial Dialogue System for Physics. In *Proceedings of the 19th International FLAIRS conference*.
- Staffan Larsson and David R. Traum. 2000. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering*, 6(3-4):323–340.
- Diane J. Litman and Scott Silliman. 2004. ITSPOKE: An Intelligent Tutoring Spoken Dialogue System. In *Proceedings of the Human Language Technology Conference: 4th Meeting of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL) (Companion Proceedings)*, Boston, MA.
- Colin Matheson, Massimo Poesio, and David Traum. 2000. Modelling grounding and discourse obligations using update rules. In *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*, pages 1–8, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Michael F. McTear. 2002. Spoken Dialogue Technology: Enabling the Conversational User Interface. *ACM Computing Surveys*, 34(1):90–169.
- Johanna Moore. 1993. What makes human explanations effective? In *Proceedings of the Fifteenth Annual Meeting of the Cognitive Science Society*, pages 131–136, Hillsdale, NJ.
- Massimo Poesio and David Traum. 1997. Conversational actions and discourse situations. *Computational Intelligence*, 13(3).
- Jeff Rickel, Rajaram Ganeshan, Charles Rich, Candace L. Sidner, and Neal Lesh. 2000. Task-oriented tutorial dialogue: Issues and agents. In Carolyn Penstein Rosé and Reva Freedman, editors, *Proceedings of the AAAI Fall Symposium: Building Dialogue Systems for Tutorial Applications*, pages 52–57, North Falmouth, MA. AAAI press.
- Jeff Rickel, Neal Lesh, Charles Rich, Candace Sidner, and Abigail Gertner. 2001. Using a Model of Collaborative Dialogue to Teach Procedural Tasks. In *Proceedings of the AIED Workshop on Tutorial Dialogue Systems*.
- Jörg Siekmann, Christoph Benzmüller, and Serge Autexier. 2006. Computer Supported Mathematics with  $\Omega$ MEGA. *Journal of Applied Logic*, 4(4):533–559.
- Magdalena Wolska, Bao Quoc Vo, Dimitra Tsovaltzi, Ivana Kruijff-Korbayova, Elena Karajosova, Helmut Horacek, Malte Gabsdil, Armin Fiedler, and Christoph Benzmüller. 2004. An annotated corpus of tutorial dialogs on mathematical theorem proving. In *Proceedings of International Conference on Language Resources and Evaluation (LREC 2004)*, Lisbon, Portugal. ELDA.
- Yujian Zhou, Reva Freedman, Michael Glass, Joel A. Michael, Allen A. Rovick, and Martha W. Evens. 1999. Delivering hints in a dialogue-based intelligent tutoring system. In AAAI Press and MIT Press, editors, *Proceedings Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, pages 128–134, Orlando FL. Published in one volume with Eleventh Conference on Innovative Applications of Artificial Intelligence (IAAI-99).
- Claus Zinn, Johanna D. Moore, and Mark G. Core. 2002. A 3-tier Planning Architecture for Managing Tutorial Dialogue. In *Proceedings of Intelligent Tutoring Systems, Sixth International Conference*, volume 2363 of LNCS, pages 574–584, Biarritz, France. Springer.