

Towards a Principled Approach to Tutoring Mathematical Proofs

Christoph Benz Müller¹, Armin Fiedler¹, Malte Gabsdil², Helmut Horacek¹,
Ivana Kruijff-Korbayová², Dimitra Tsovaltzi², Bao Quoc Vo¹, Magdalena Wolska²

¹Fachrichtung Informatik ²Fachrichtung Computerlinguistik
Universität des Saarlandes, Postfach 15 11 50, D-66041 Saarbrücken, Germany
{chris,afiedler,horacek,bao}@ags.uni-sb.de, {gabsdil,korbay,dimitra,magda}@coli.uni-sb.de

Abstract. Studies comparing human and computer-based tutoring have identified natural language communication about the tutorial goal as a major source for the increased learning success in human tutoring. However, due to the inherent difficulty of natural language processing in non-tightly restricted tasks, which tutoring is even in simple domains, only few state-of-the-art systems use natural-language style interaction. Addressing tutoring in a principled way, we are developing a system that teaches proving skills in mathematics, by incrementally enhancing a testable Wizard-of Oz environment with tutoring components, existing domain reasoning systems and natural language processing components. The first component integrated is an elaborate hinting algorithm. Experiments carried out recently provided valuable insights in relevant phenomena and demands for tutorial and task-specific enhancements of available mathematical reasoning and natural language processing components.

1 Introduction

Empirical studies have demonstrated that natural language dialog capabilities are necessary for the success of tutorial sessions. However, due to the inherent difficulty of natural language processing in non-tightly restricted tasks, which tutoring is even in simple domains, only few state-of-the-art systems use natural-language style interaction, but most of them require menu-based input or exact wording of the input [15, 2, 10]. This is insufficient in view of Moore’s empirical findings showing that flexible natural language dialog is needed to support active learning [14]. As one of a few exceptions, the latter approach is taken in the CIRCSIM-Tutor project [13] which aims to build a natural language-based tutoring system for first-year medical students to learn about the reflex control of blood pressure.

Motivated by the empirical findings about tutoring, we aim at developing a tutoring system with flexible natural language dialog capabilities to support interactive mathematical problem solving. Thereby, we specifically attack one of the problems that current tutorial systems have with establishing flexible natural language dialog: The organization of knowledge in existing systems tightly interleaves tutorial strategies, domain knowledge and dialog management, instead of clearly separating the different kinds of knowledge in appropriately structured knowledge bases. Therefore, we employ a modular approach keeping a strict separation between the different kinds of knowledge involved in the processing, relying on existing domain inference components, such as theorem provers and transformation tools.

Since there exists virtually no empirical data about tutorial dialogs about proving mathematical theorems, we have carried out an experiment with a simulated tutorial system. Apart from extended capabilities attributed to existing domain reasoning tools for the sake of this experiment, a hinting algorithm has been specifically designed, and its usefulness has been tested in the course of the experiment. The design of the system components going beyond pure proving is informed by the analysis of a corpus of this tutorial dialog data. The overall enterprise is part of the DIALOG project¹ [16], which is a joint activity of the computer science and computational linguistics departments at Saarland University.

The outline of this paper is as follows. We first present the aims of our project and the overall environment. Next we describe our hinting algorithm. We follow with a description of an experiment in which we collected a corpus of natural language tutorial dialogs. Finally, we analyze the phenomena observed in this experiment and the resulting requirements for an ambitious tutorial system.

¹ The DIALOG project is part of the Collaborative Research Center on *Resource-Adaptive Cognitive Processes* (SFB 378) at University of the Saarland [17].

2 The DIALOG Project

The goal of the DIALOG project is (i) to empirically investigate the use of flexible natural language dialog in tutoring mathematics, and (ii) to develop an experimental prototype system gradually embodying the empirical findings. The experimental system will engage in a dialog in written natural language (and later also in multimodal forms of communication based on diagrams, spoken language and animated mathematical function displays) to help a student understand and construct mathematical proofs. The overall scenario for the system is illustrated in Figure 1, including the following components:

- *Learning Environment*: In our scenario, the student takes an interactive course in some field of mathematics within a web-based learning environment. We use ACTIVE MATH [12], a generic web-based learning system that dynamically generates interactive (mathematical) courses adapted to the student's goals, preferences, capabilities, and knowledge.
- *Mathematical Proof Assistant*: It is used for the problem-solving in the mathematical domain underlying the dialogs, which is based on reasoning carried out by the Ω MEGA system [19]. This involves the verification (or falsification) of user specified inference steps and checking whether the application of an inference step leads to a proof state from which a complete proof can be obtained. Mathematical tutorial dialogs thus require (i) stepwise interactive as well as (ii) automated proof construction at a human-oriented level of abstraction.
- *Proof Manager*: In the course of the interactive tutorial session, the user may explore alternative proofs, or make various attempts at constructing a valid proof, involving both valid and invalid inference steps.
- *Dialog Manager*: We employ the Information-State (IS) Update approach to dialog management developed in the TRINDI and SIRIDUS projects [18]. The dialog resources include (i) dialog move selection rules (i.e., rules that determine what dialog move the system will make next, given the current information state and a communicative goal), and (ii) dialog information-state update rules (i.e., rules that dynamically change the information state depending on the dialog moves the user or the system have successfully made). We distinguish between domain independent, generic dialog moves, such as meta-communication moves (e.g., clarification and correction), and domain-specific ones, such as various kinds of hinting moves [8, 20], further specialized for tutoring in the mathematics domain.
- *Knowledge Resources*: The static knowledge in our scenario includes *pedagogical knowledge* (generic and domain-specific teaching strategies), and *mathematical knowledge* (in our MBase system [11]). The dynamic knowledge includes the student model, as well as the information state maintained by the dialog manager.

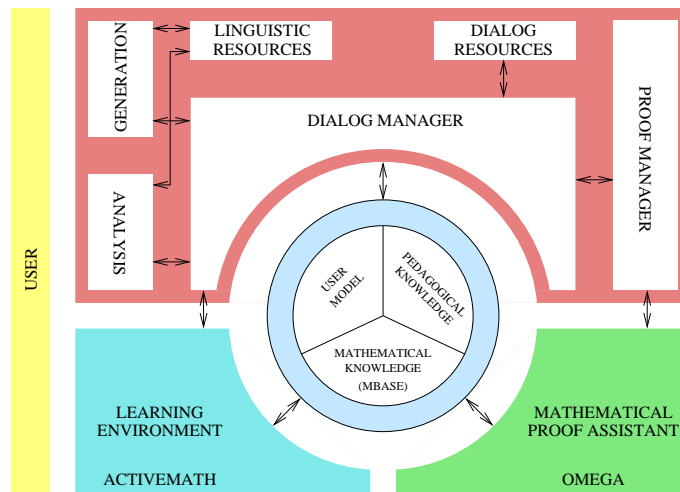


Fig. 1. DIALOG project scenario.

3 The Hinting Algorithm

A mathematical tutoring system must be able to tutor proofs in a way that not only helps the student understand the current proof, but also allows for a high learning effect. What is meant by the latter is the ability of the students not only to better understand the problem at hand, but also to generalize and apply the taught methodology without help later on. We propose to establish those tutoring aims by making use of the *socratic* tutoring method [15, 4]. The decisive characteristic of the socratic method is the use of hints to achieve self-explanation, as opposed to answers and explanations being provided constantly by the tutor, which is characteristic of the *didactic* method [4, 20].

Hinting is a method that aims at encouraging active learning. It can take the form of eliciting information that the student is unable to access without the aid of prompts, or information which he can access but whose relevance he is unaware of with respect to the problem at hand. Alternatively, a hint can point to an inference that the student is expected to make based on knowledge available to him, which helps the general reasoning needed to deal with a problem [3].

The hinting strategy in DIALOG is based on categorizations of two items: the hints themselves, and student answers. We defined the hint categories based on the needs in the domain. The structure of the hint taxonomy reflects the function of the hints with respect to the information that the hint addresses or is meant to trigger. To capture different functions of a hint we define hint categories across two dimensions:

1. active vs. passive
2. domain-relation vs. domain-object vs. inference-rule vs. substitution vs. meta-reasoning vs. performable-step

The first dimension distinguishes between the *active* and *passive* function of hints. The difference lies in the way the information to which the tutor wants to refer is approached. The active function of hints looks forward and seeks to help the student in accessing a further bit of information, by means of *eliciting*, that will bring him closer to the solution. The *passive* function of hints refers to the small piece of information that is provided each time in order to bring the student closer to some answer. The tutor *gives away* some information, which he has normally unsuccessfully tried to elicit previously. Due to that relation between the active and passive function of hints, the passive function of one hint class in the second dimension consists of hint categories that are included in the active function in its subordinate class. In the second dimension, we ordered the classes with respect to their degree of informativeness. We say, that a class is *subordinate* to another one if it reveals more information. Each of these classes consists of single hint categories that elaborate on one of the attributes of the proof step under consideration. The hint categories are grouped in classes according to the kind of information they address in relation to the domain and the proof. The following table shows an excerpt of our hint taxonomy.

	active	passive
domain-relation	elicit-specialization elicit-generalization	give-away-specialization give-away-generalization
domain-object	give-away-specialization give-away-generalization	give-away-relevant-concept give-away-hypotactical-concept
inference rule	give-away-relevant-concept give-away-hypotactical-concept	give-away-inference-rule
substitution	give-away-inference-rule elicit-substitution	spell-out-substitution
meta-reasoning	spell-out-substitution	explain-meta-reasoning
performable-step	explain-meta-reasoning confer-to-lesson	give-away-performable-step
pragmatic	ordered-list elicit-discrepancy	take-for-granted point-to-lesson

Fig. 2. An excerpt of our hint taxonomy

We briefly explain some of the categories referred to in this table. *Domain-relation* hints address the relations between mathematical concepts in the domain. The passive function of domain-relation hints is the active function of domain-object hints, that is, they are used to elicit domain objects. *Domain-object* hints address an object in the domain. The hint *give-away-relevant-concept* names the most prominent concept in the proposition or formula under consideration. This might be, for instance, the concept whose definition the student needs to use in order to proceed with the proof, or the concept that will in general lead the student to understand which inference rule he has to apply. The passive function of domain-object hints is used to elicit the applicable inference rule, and, therefore, is part of the active function of the respective class. Finally, the class of *pragmatic* hints is somewhat different from other classes in that it makes use of minimal domain knowledge. It rather refers to pragmatic attributes of the expected answer. The active function hints include *ordered-list*, which specifically refers to the order in which the parts of the expected answer appear, and *elicit-discrepancy*, which points out that there is a discrepancy between the student's answer and the expected answer. *Take-for-granted* asks the student to just accept something as a fact either when the student cannot understand the explanation or when the explanation would require making use of material that is not in the focus of the lesson. *Point-to-lesson* points the student to the lesson in general and asks him to read it again when it appears that he cannot be helped by tutoring because he does not remember the study material.

The student's answer is evaluated by use of an expected answer. The *expected answer* is the proof step which is expected next according to the formal proof which the system has chosen for the problem at hand. We want to make use of the student's own reasoning in helping him with the task and avoid super-imposing a particular solution. We model that by trying to match the student's answer to a proof step in one of a set of proofs. To this end we use the state-of-the-art theorem prover Ω MEGA [19].

We define the following student answer categories, which, although being exhaustive, need further refinement:

Correct: An answer which is both complete and accurate.

Complete-Partially-Accurate: An answer which is complete, but some parts in it are inaccurate.

Complete-Inaccurate: An answer which is complete, but all parts in it are inaccurate.

Incomplete-Accurate: An answer which is incomplete, but all parts that are present in it are accurate.

Incomplete-Partially-Accurate: An answer which is incomplete and some of the parts in it are inaccurate.

Wrong: An answer which is both incomplete and inaccurate.

We shall now sketch an algorithm that implements the socratic strategy. In intuitive terms, the algorithm aims at having the student find the proof by himself. If the student does not know how to proceed or makes a mistake, the algorithm prefers hinting at the right solution in order to elicit the problem solving instead of giving away the answer. An implicit student model makes the algorithm sensitive to students of a different level by providing increasingly informative hints.

The algorithm takes as input the number and kind of hints produced so far, the number of wrong answers by the student and the current and previous student answer category. The particular input to the algorithm is the category that the student answer has been assigned, based on our student answer categorization scheme, and the domain knowledge employed in the answer. The category of the student answer in combination with the kinds of hints already produced and the use of required entities of the mathematical ontology together inform the algorithm of the level of the student's knowledge.

Moreover, the algorithm computes whether to produce a hint and which category of hint to produce, based on the number of wrong answers, the number and kind of hints already produced, as well as the domain knowledge demonstrated by the student. The algorithm computes the appropriate hint category to be produced next from the taxonomy of hints.

If the hinting does not effect correct student answers after several hints the algorithm switches to a didactic strategy, and, without hinting, explains the steps that the student cannot find himself. Nevertheless, the algorithm continues to ask the student for the subsequent step. If the student gives correct answers again and, thus, the tutor need not explain anymore, the algorithm switches back to the socratic strategy. In effect, hints are provided again to elicit the step under consideration.

Details about the hinting strategy can be found in [8,9]. The algorithm is implemented and integrated in the environment used for carrying out the empirical studies described in the following section.

4 Empirical Study

Dialogs in formal domains, such as mathematics, are characterized by a mixture of telegraphic natural language text and embedded formal expressions. Due to the lack of empirical data for such environments, we have collected a corpus of dialogs with a simulated tutoring system for teaching proofs in naive set theory. The analysis of this corpus enabled us to identify genre-specific variants of linguistic phenomena which impose specific requirements on natural language dialog management.

For this purpose, we conducted a *Wizard-of-Oz (WOz)* experiment. In such an experiment, the subject interacts through an interface with a human “wizard” simulating the behavior of a system [5]. The WOz methodology is commonly used to investigate human-computer interaction in systems under development. One of the reasons for using a WOz setting rather than a human tutor is that it has been observed that humans interact differently with computers than with other humans. Another reason is that the tutor should follow the specific algorithm(s), which we are implementing in our system. In this way the dialog data we collect (i) represents the users’ behavior in interactions following these algorithms and (ii) provides early feedback on the algorithms. In subsequent experiments, implemented components can substitute for some of the tasks now carried out by the wizard, while preserving the overall experimental setup.

For carrying out the experiment, we have implemented a tool, DiaWOz, to support the wizard in his actions and to collect data on-line [7]. DiaWOz consists of two parts which are responsible for dialog authoring and dialog execution, respectively. Dialog specification (authoring) is more powerful than in standard approaches, since it combines a finite state machine with information states. Information states are conceived as sets of local and global variables. Moreover, transitions are more elaborate than usual insofar as they have preconditions and effects, in terms of information states. Dialog execution basically follows these specifications. It provides two separate choosers: one for analyzing the utterance of the subject, and the other for generating the next system action. By this design, the tool allows the integration of existing software components and thus supports progressive refinement.

We invited 24 subjects to participate in the experiment. They were students with educational background in humanities (e.g., law, economy, languages, psychology) or sciences (e.g., biology, chemistry, computer science, computational linguistics). Their prior mathematical knowledge ranged from little to fair. For each subject, the experiment consisted of the following phases (each of which had a fixed maximum duration): (1) *Preparation and pre-test*: First, the subject filled in a background questionnaire. Then he/she studied written lesson material, explaining basic concepts and providing a collection of six lemmata about properties of sets and eleven lemmata about properties of powersets.² Finally he/she was asked to prove (on paper) the theorem $K(A) \in P(K(A \cap B))$. (2) *Tutoring session*: The subject was asked to evaluate a tutoring system with natural language dialog capabilities. He/She was given three theorems to prove: The theorem $K((A \cup B) \cap (C \cup D)) = (K(A) \cap K(B)) \cup (K(C) \cap K(D))$ was used first to let the subject familiarize himself/herself with the system’s interface. Then two more complex theorems were presented (in different order to different subjects): (a) $A \cap B \in P((A \cup C) \cap (B \cup C))$, and (b) If $A \subseteq K(B)$, then $B \subseteq K(A)$. The interface enabled the subject to type text or insert mathematical symbols by clicking on buttons; it also displayed the complete dialog with both the tutor’s and the subject’s utterances. The subject was instructed to enter partial steps of a proof rather than the complete proof as a whole, in order to enable a dialog with the system. (3) *Post-test and evaluation questionnaire*: The subject was asked to write down (on paper) a proof for the theorem $K(A \cup B) \in P(K(A))$.³ At the end, he/she was asked to fill in a questionnaire addressing various aspects of the system and its usability.

The tutor-wizard’s task was to respond to the student’s utterances following the given hinting algorithm. The wizard first interpreted the subject’s utterance and classified its completeness, accuracy, and relevance with respect to a valid proof of the theorem at hand. Then, the wizard decided what dialog moves to make next and verbalized them. Depending on the tutoring strategy employed by the wizard for a given subject, the dialog move options included informing the subject about completeness, accuracy, and relevance of the utterance, giving hints on how to proceed further, explaining a step under consideration, prompting for the next step, or entering into a clarification dialog. The wizard was free to mix text with formulas.

² In this first experiment the lesson material was still presented on paper, not through the ACTIVE MATH system.

³ The comparison of the student’s performance of the pre-test and post-test proofs serve to evaluate their learning gain from the tutoring session.

References and ambiguities	<p>(1) Potenzmenge enthält alle Teilmengen, also auch $(A \cap B)$ <i>Power set contains all subsets, hence also $(A \cap B)$</i></p> <p>(2) $K((A \cup B) \cap (C \cup D)) = K(A \cup B) \cup K(C \cup D)$ de Morgan Regel 2 auf beide Komplemente angewendet <i>de Morgan rule 2 applied to both complements</i></p> <p>(3) de Morgan Regel 1 gilt auch für $K(C \cup D)$ de Morgan Regel 2 besagt $K(A \cap B) = K(A) \cup K(B)$. In diesem Fall z.B. $K(A) =$ dem Begriff $K(A \cup B)$ und $K(B) =$ dem Begriff $K(C \cup D)$. Deshalb ist dann $K((A \cup B) \cap (C \cup D)) = (K(A) \cap K(B)) \cup (K(C) \cap K(D))$ <i>de Morgan rule 1 also holds for $K(C \cup D)$ de Morgan rule 2 means $K(A \cap B) = K(A) \cup K(B)$. In this case e.g. $K(A) =$ the term $K(A \cup B)$ and $K(B) =$ the term $K(C \cup D)$. Therefore $K((A \cup B) \cap (C \cup D)) = (K(A) \cap K(B)) \cup (K(C) \cap K(D))$</i></p> <p>(4) $(A \cup B)$ muß in $P((A \cup C) \cap (B \cup C))$ sein, da $(A \cap B) \in (A \cap B) \cup C$ <i>$(A \cup B)$ must be in $P((A \cup C) \cap (B \cup C))$, since $(A \cap B) \in (A \cap B) \cup C$</i></p> <p>(5) $(B \cup A) \subseteq C$ $(B \cup A) \subseteq D$. Wenn A Teilmenge von C und B Teilmenge von C dann müssen beide Mengen zusammen ebenfalls eine Teilmenge von C sein. Gleiches gilt mit D $K(C \cap D) \cup K(A \cap B)$ Anwendung der de Morgan Regeln. $((B \cup A) \subseteq C$ $(B \cup A) \subseteq D$. <i>If A is a subset of C and B a subset of C, then both sets together must also be a subset of C. The same holds for D. $K(C \cap D) \cup K(A \cap B)$ applying the de Morgan rules. $((B \cup A) \subseteq C$ $(B \cup A) \subseteq D$.</i></p> <p>(6) $A \in P(A \cup C), B \in P(B \cup C)$ $A \cup B \in P(A \cup C) \cap (B \cup C)$</p> <p>(7) $P(A \cap B), P(C) \subseteq P((A \cap B) \cup C)$</p>
Inference steps	<p>(8) Als nächstes stelle ich die rechte Seite unter Anwendung der Eigenschaften von Mengenoperationen so um, daß $P(A \cap B)$ vereinigt mit einer anderen Menge herauskommt. <i>Next, I will recast the right side by applying properties of set operations in such a way that this results in $P(A \cap B)$ union some other set</i></p> <p>(9) Zerlegen der Potenzmenge: $P((A \cup C) \cap (B \cup C)) = P(A \cup C) \cap P(B \cup C)$ <i>Splitting the power set: $P((A \cup C) \cap (B \cup C)) = P(A \cup C) \cap P(B \cup C)$</i> Tutor: Das ist richtig. <i>Tutor: That is correct.</i> Anwenden der Distributivität: $P((A \cup C) \cap (B \cup C)) = P(C \cup (A \cap B))$ <i>Applying distributivity: $P((A \cup C) \cap (B \cup C)) = P(C \cup (A \cap B))$</i> Tutor: Ist das noch derselbe Lösungsweg wie in der vorigen Antwort? <i>Tutor: Is this still the same solution path as in the previous response?</i> Nein, ich habe mich umentschieden. Ich zerlege jetzt die Potenzmenge $P(C \cup (A \cap B)) \supseteq P(C) \cup P(A \cap B)$ <i>No, I have changed my mind. I am now splitting the power set: $P(C \cup (A \cap B)) \supseteq P(C) \cup P(A \cap B)$</i></p>

Fig. 3. Examples of dialog utterances. The predicates P and K stand for power set and complement, respectively.

5 Preliminary analysis of the data

We have identified two major aspects of relevance for enhancing existing system components: (1) Identification and resolution of natural language references to terms and concepts of the domain; (2) Evaluation of the inference step(s) proposed by the student with respect to the master proof(s). We discuss those aspects in the following (see Fig 3, which illustrates varying degrees of the use of natural language across subjects).

5.1 References

The most prominent problems handling references to domain concepts are associated with ambiguities:

- *Discourse references*: Their proper resolution requires domain-specific interpretation, for demonstratives: (e.g., “*this* also holds vice-versa”), for deictic anaphora (“the *above* expression”, “apply *here* the axiom in opposite direction”), and for noun phrases (e.g. “on *the left side*”, “for the *inner parenthesis*”). Expressions (1) and (2) refer to an assertion and an expression, respectively, to be determined according to the discourse context. The references in (3) are incomplete specifications. “Left side” refers to an equation, and “inner parenthesis”, which is metonymic, refers to the expression enclosed by it.
- *Domain relations*: comprise propositional logic junctors, logical derivation, and justifications, which may be the source for scope ambiguity. For instance, (7) is an aggregated (or: abbreviated) description of two subset relations where the subsets share a common superset. Similarly, in (6), the comma is meant to be interpreted as logical “and” followed by the expression that constitutes the logical consequence. Another issue concerns reference to mathematical relations which may be imprecise in the sense that the natural language formulation fits several relations (e.g., within the domain of mathematical sets, “must be in” in (4) can be interpreted as “element” or “subset”; and “both sets *together*” in (5) as union or intersection).
- *Variables*: Ambiguities arise in case an identifier in the text refers to variables in different formulas (e.g., A may refer to a variable used in an axiom or to a variable in an expression instantiating that axiom, as in (3)), where, the two occurrences of A in “ $K(A) = \text{the term } K(A \cup B)$ ” do not co-refer).
- *Domain rules (axioms)*: Problems of ambiguity with domain rules arise due to duality (e.g., distributivity) or due to imprecision in formulation (e.g., “de Morgan rules” in (5)).
- *Descriptions of domain operations*: This includes application of an inference rule, which is often described informally (e.g., “to split” an expression as in (9)). A challenge for the natural language analysis lies in the large number of unexpected synonyms, where some of them have a metaphoric flavor.
- *Descriptions of goal*: occurred in natural language as an abstract form of a formal expression. They typically served to indicate the purpose of building a new expression; see (8).

5.2 Inference Steps

The major task of a student in our tutorial session on proving elementary facts of set theory is the specification of inference steps. The specified inference steps need to be checked and classified in order for a tutorial strategy to be decided upon. The classification has two dimensions: (1) correctness of the inference, and (2) relevance for the proof at hand.

Correctness of expressions: Students may make mistakes or produce expressions that are otherwise confused and thus need to be classified as partially incorrect or wrong. A particular important issue in dealing with mistakes is the identification of near misses. They need to be treated differently in the tutorial context than severe errors that show a misconception or a complete lack of understanding. An inference appears as a “near miss” if the expression inferred differs from the correct one in a single element only. The incorrect element may be a variable or a constant; for example, a typing mistake. It may also be an operator, but the correct operator and the incorrect one must be conceptually related. Examples of commonly confused conceptually related operations in our specific domain include the use of “=” instead of “ \subseteq ” (too specific relation) or “ \in ” instead of “ \subset ” (sort incompatibility on similar relations).

Relevance of expressions: For a correct inference step, its potential contribution to the proof at hand needs to be determined. In order for an inference to be considered relevant, it needs to be an inference step in the proof to be taught. Depending on the previous proof steps, the student may have committed

himself to one or another proof out of the set of potential solution paths known to the system. In case an inference step is incompatible with the prior followed proof path, the intention of the student to revise his proof strategy is checked (see utterance (9) in Figure 3).

6 Conclusions and Future Research

We briefly presented the overall framework of the DIALOG project which aims at the development of a mathematical tutoring system with flexible dialog. We have sketched a hinting algorithm, which constitutes the heart of tutoring in the *socratic* style. Moreover, we reported on a WOz experiment in which we collected a corpus of tutorial dialogs with 24 subjects on several problems in naive set theory. The corpus also enabled us to identify interesting cases with genre-specific properties which pose challenges for natural language dialog management and handling interaction with domain reasoning components. In the next stages of the project, we will continue to investigate the interaction between various system parts and gradually implement the missing system components.

References

1. *Papers from the 2000 AAAI Fall Symposium on Building Dialogue Systems for Tutorial Applications*. AAAI Press, 2000.
2. V. Aleven and K. Koedinger. The need for tutorial dialog to support self-explanation. In [1], pages 65–73.
3. G. Hume, Gregory, J. Michael, A. Rovick, and M. Evens. Student Responses and Follow Up Tutorial Tactics in an ITS. *Proceedings of the 9th Florida Artificial Intelligence Research Symposium*, pages 168–172, Key West, FL, 1996.
4. C. Rosé, J. Moore, K. VanLehn, and D. Allbritton. A Comparative Evaluation of Socratic versus Didactic Tutoring. *Proceedings 23rd Annual Conference of the Cognitive Science Society*, J. Moore and K. Stenning (eds.), University of Edinburgh, 2001.
5. N. O. Bernsen, H. Dybkjær, and L. Dybkjær. *Designing Interactive Speech Systems — From First Ideas to User Testing*. Springer, 1998.
6. A. Fiedler, A. Franke, H. Horacek, M. Moschner, M. Pollet, and V. Sorge. Ontological Issues in the Representation and Presentation of Mathematical Concepts. *Workshop on Ontologies and Semantic Interoperability at ECAI-2002*, 2002.
7. A. Fiedler and M. Gabsdil. Supporting Progressive Refinement of Wizard-of-Oz Experiments. In *Proceedings of the Sixth International Conference on Intelligent Tutoring Systems—Workshop W6: Empirical Methods for Tutorial Dialogue Systems*, 2002.
8. A. Fiedler and D. Tsovaltzi. Automating Hinting in Mathematical Tutorial Dialogue. *Proceedings of the EACL-03 Workshop on Dialogue Systems: interaction, adaptation and styles of management*, Budapest, 2003.
9. A. Fiedler and D. Tsovaltzi. An Approach to Automating Hinting in an Intelligent Tutorial System. *IJCAI Workshop on Knowledge Representation and Automated Reasoning for E-Learning Systems*, Acapulco, 2003.
10. N. Heffernan and K. Koedinger. Intelligent tutoring systems are missing the tutor: Building a more strategic dialog-based tutor. In [1], pages 14–19.
11. M. Kohlhase and A. Franke. MBase: Representing knowledge and context for the integration of mathematical software systems. *Journal of Symbolic Computation*, 32(4):365–402, 2000.
12. E. Melis et al. ACTIVEMATH: A generic and adaptive web-based learning environment. *Artificial Intelligence in Education*, 12(4), 2001.
13. CIRCSIM-Tutor Project: <http://www.csam.iit.edu/circsim/> (Illinois Institute of Technology).
14. J. Moore. What makes human explanations effective? In *Proc. of the Fifteenth Annual Conference of the Cognitive Science Society*, Hillsdale, NJ. Earlbaum, 1993.
15. N. Person, A. Graesser, D. Harter, and E. Mathews. Dialog move generation and conversation management in AutoTutor. In [1], pages 45–51.
16. M. Pinkal, J. Siekmann, and C. Benzmlüller. Projektantrag Teilprojekt MI3 — DIALOG: Tutorieller Dialog mit einem mathematischen Assistenzsystem. In *Fortsetzungsantrag SFB 378 — Ressourcenadaptive kognitive Prozesse*, Saarbrücken, Germany, 2001. Universität des Saarlandes.
17. SFB 378 web-site: <http://http://www.ling.gu.se/projekt/siridus/>.
18. TRINDI project: <http://www.ling.gu.se/research/projects/trindi/>.
19. J. Siekmann et al. Proof development with Ω MEGA. In *Proceedings of the 18th Conference on Automated Deduction*, LNAI 2392, Copenhagen, DENMARK, 2002. Springer Verlag.
20. D. Tsovaltzi and C. Matheson. Formalising Hinting in Tutorial Dialogues. EDILOG: 6th workshop on the semantics and pragmatics of dialogue, Edinburgh, Scotland, 2002.