Building Hint Specifications in an NL Tutorial System for Mathematics

Dimitra Tsovaltzi, Helmut Horacek, and **Armin Fiedler**
Department of Computer Science
Saarland University
{tsovaltzi,horacek,afiedler}@ags.uni-sb.de

Abstract

NL interaction and skillful hinting are known as cornerstones for successful tutoring. Despite these insights, a combination of these two factors is widely under-represented in the field. Building a tutorial system for teaching mathematical proof techniques, we aim at an elaborate hinting algorithm which integrates problem solving, discourse contexts and accurate domain knowledge into a socratic hinting strategy.

Introduction

The mode of interaction between the tutor and the student is a crucial factor in tutorial systems. Specifically, natural language dialog capabilities are necessary for the success of tutorial sessions, as it has been shown empirically (Moore 2000). Moreover, hinting has been demonstrated to be a beneficial ingredient in tutoring (Chi *et al.* 1994; Rosé *et al.* 2001), since it helps the student build a deeper understanding and improve their performance. Despite these insights, only few state-of-the-art tutoring systems use natural-language interaction and hinting in an elaborate way. Typically, no natural language interaction is used, (Graesser *et al.* 2003) being one of the rare exceptions.

We suspect a fundamental reason for the limited use of natural language interaction lies in the architectures used in existing systems, in which domain knowledge, tutoring and pedagogical knowledge, and knowledge about dialog management are tightly intertwined. In contrast, we aim at a tutoring system in the domain of mathematics in which domain knowledge, dialog capabilities, and tutorial strategies can clearly be identified. Our investigations are part of the DIALOG project (Benzmüller *et al.* 2003a). The aim of the project is the development of a mathematical tutoring system that is able to teach proving 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 to better understand the problem at hand, as well as to generalize and apply the solving techniques on their own later on.

In order to accomplish this ambitious goal, we envisage a tutorial system that employs an elaborate natural language dialog component. Furthermore, to tutor mathematics, we need a formally encoded mathematical theory including definitions and theorems along with their proofs, a theory of tutoring, and means of classifying the student's input in terms of the knowledge of the domain demonstrated.

We propose to meet these requirements in the following way. Since it is in general impossible to precompile all possible proofs for a given theorem, we make use of the state-of-the-art theorem prover ΩMEGA (Siekmann *et al.* 2002) with its mathematical knowledge base. To classify the student's input, we have developed a categorization scheme for student answers, which draws on ΩMEGA's mathematical ontology. As far as the tutoring model is concerned, we have been developing a taxonomy of hints for the naive set theory domain. This taxonomy is used by a hinting algorithm which models the socratic tutoring method by means of selecting different hint categories according to an implicit student model (Fiedler & Tsovaltzi 2003). In this paper we focus on the hinting category selection and the generation of its content specifications. More specifically, we talk about the enhancement of the ΩMEGA mathematical ontology and discuss an abstraction technique for hint content specification.

The paper is organized as follows. First, we briefly motivate our approach to hinting. Then we introduce the taxonomy of hint categories. The first major section is devoted to outlining the hinting algorithm, which is responsible for the hint category selection. In the other major section, we describe the hint content determination. Finally, we discuss related work and future activities.

Our Approach to Hinting

Adhering to psychological evidence for the high educational effect of hinting (Rosé *et al.* 2001) and taking into consideration such psychological evidence as cognitive load theory, schema acquisition and motivational theory (Lim & Moore 2002; Owen, Sweller, & Olson 1985; Keller 1987), we propose to establish our tutoring aims by making use of a *socratic* tutoring method, whose decisive characteristic is the use of hints in order to achieve active learning. Our work builds on the little systematic research done to date in the area (Hume *et al.* 1996; Fiedler & Horacek 2001; Tsovaltzi 2001; Tsovaltzi & Matheson 2002).

We have been developing a taxonomy of hints, which draws on a mathematical ontology and on abstraction con-

	active	passive
domain-relation	elicit-antithesis elicit-duality	give-away-antithesis give-away-duality
domain-object	give-away-antithesis elicit-basic-knowledge elicit-more-general-knowledge	give-away-relevant-concept give-away-more-general-knowledge give-away-basic-knowledge
inference rule	give-away-relevant-concept elaborate-domain-object elicit-inference-rule	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 reduce-parentheses variables-closer	give-away-performable-step
pragmatic	ordered-list elicit-discrepancy	take-for-granted

Table 1: A fragment of the taxonomy of hints

siderations, as well as on empirical data from the BE&E corpus (Moore *et al.* 2000) and our own corpus (Benzmüller *et al.* 2003b; 2004). This taxonomy is used by a hinting algorithm which models the socratic tutoring method by means of selecting different hint categories according to an implicit student model (Fiedler & Tsovaltzi 2003). The actual content determination of hinting specifications incorporates the use of relations from our enhanced mathematical ontology, as well as abstraction mechanisms which enable us to select from a repertoire of hints at varying degrees of specificity.

A Taxonomy of Hint Categories

The hint taxonomy depicted in Table 1 shows all the dimensions and classes of the full taxonomy, but only illustrative examples of the hint categories in them. The taxonomy captures the underlying function of hints that is mainly responsible for the educational effect of hints. The structure of the hint taxonomy also reflects the function of the hints with respect to the information that the hint addresses or is meant to trigger.

In order to capture the different functions of a hint we have defined hint categories across two dimensions. The first dimension distinguishes between the active and passive function of hints. The passive refers to the information provided each time to the student. The active function refers to the information that the hint aims at triggering in the student's current cognitive state, that is, the information elicited. The second dimension distinguishes between different classes of hints. Hint categories are grouped in classes according to the kind of information they address in relation to the domain and the formal proof step under consideration. The classes are ordered according to the amount of information they give away.

By and large, the hints of the passive function of a class in the second dimension are part of the hints of the active function of its immediately subordinate class. For example, the passive hint *give-away-relevant-concept* of the class *domain-object* is also an active hint of its subordinate class, namely, *inference-rule*. In providing this hint the system is trying to elicit the inference rule, which would help the student proceed with the proof. In an example from our corpus, the tutor pointed to the inference rule that eliminates an implication by producing the following realization of the hint *give-away-relevant-concept* (an English translation is included):

T2: Sie müssen als erstes die wenn-dann-Beziehung betrachten.
[*You first have to consider the if-then-relation.*]

For a detailed discussion of the taxonomy see (Fiedler & Tsovaltzi 2003).

Hint Category Selection

Selecting a hint category relies on modeling of the context which mainly manifests itself in terms of previous student answers. The hinting algorithm's central part is the function `socratic`. A specific feature of the algorithm is subtask handling. We devote a section to each of these issues.

General Session Modeling

In this section we look into a few aspects that are used in choosing the right hint for a student, which constitutes the session modelling. As we will see in the following subsection, the hinting algorithm takes as input the categories of the student's answers. We categorize the student's answer in terms of its *completeness* and *accuracy* with respect to the expected answer. The *expected answer* is any correct proof step that fits into the proof the student has pursued so far. In order to produce the proof that includes the student's answer, we use the state-of-the-art theorem prover ΩMEGA (Siekmann *et al.* 2002). This makes it possible for the system to give guidance according to the proof that the student is attempting without imposing one of the alternatives.

Our definitions of completeness and accuracy make use of the concept of a *part*, that is, a premise, the conclusion, or the inference rule of a proof step.

We say that an answer is *complete* if and only if all desired parts of the answer are mentioned. We say that a part of an answer is *accurate* if and only if the propositional content of

the part is the true and desired one. Based on these notions, we define the following student answer categories:

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.

Irrelevant: An answer, which is in principle correct, but does not contribute to the proof under consideration.

Non-Attempt: The student does not attempt an answer.

Since we do not make use of all of the above categories for the algorithm, we collapse the categories *complete-partially-accurate*, *complete-inaccurate* and *incomplete-partially-accurate* to one category, namely, *inaccurate*. Moreover, for the purposes of this paper we treat irrelevant and non-attempt answers as wrong answers. For more on the student answer categories see (Tsovaltzi & Fiedler 2003).

Our Hinting Algorithm

The algorithm takes into account the current and previous student answers. The particular input to the algorithm is the category to which the student answer has been assigned, based on our student answer categorization scheme and the domain knowledge employed in the answer. Moreover, the algorithm computes whether to produce a hint and which category of hint to produce, based on the number of wrong answers, as well as the number and kind of hints already produced. We will now have a closer look at the algorithm. The bulk of the work in the algorithm which implements hinting is done by the main function `socratic` (cf. Figure 1), which we outline here. The function takes as an argument the category C of the student's current answer. If the origin of the student's mistake is not clear, a clarification dialog is initiated, before `socratic` is called. Further algorithm input includes H, which denotes the number of hints produced so far, and C_{-1}, which denotes the category of the student's previous answer. The function `socratic` calls several other functions that roughly correspond to hint classes of the hint taxonomy, which we do not look into in this paper. Both the function `socratic` and the functions called within it, produce as output single hint categories. After having produced a hint category, the function `socratic` requires the analysis of the student's answer to that hint. If the student's answer is still not right the function `socratic` is recursively called. For a more detailed description of the algorithm see (Fiedler & Tsovaltzi 2003).

Subtasks

Inspired by our corpus data and the need to add more structure to the hinting process that captures a meta-reasoning process, we have built a way of handling subtasks into the hinting algorithm. More specifically, the algorithm as presented so far models hinting at the one main line of reasoning. We elaborate the hinting process by adding more levels, in the form of subtasks, to the line of reasoning that can be hinted at. Subtasks aim at facilitating the main line of reasoning itself. They open and close at the same point in the line of reasoning and may be nested.

To illustrate the function of subtasks we include in Figure 2 a series of possible hint realizations, based on hints from our taxonomy and their use in subtasks. The task is to prove that $A \cap B \in \mathcal{P}(A \cup C) \cap (B \cup C))$ and for the first step the definition of $\mathcal{P}$ (powerset) needs to be applied. That is $A \in \mathcal{P}(B) \Rightarrow A \subseteq B$. Numbers signify sequences of hints within the same task level. Primes signify task levels, which are embedded. There is additional indication for when one level of a task starts and ends. The names in brackets correspond to hint categories from the taxonomy. We look at a full unfolding of the subtasks as we assume that the hints produced have not elicited the correct answer from the student.

Case $H = 0$
  if C is wrong or inaccurate then call `elicit`
  if C is incomplete-accurate then produce an active pragmatic hint {*that is, ordered-list, or unordered-list*}
Case $H = 1$
  if C is wrong
  then if C_{-1} is wrong or incomplete-accurate
    then call `up-to-inference-rule`
    if C_{-1} is inaccurate then call `elicit-give-away`
    if C_{-1} is correct then call `elicit`
  else call `elicit`
Case $H = 2$
  if C is wrong
  then if this is the third wrong answer
    then produce *explain-meta-reasoning*
    else if previous hint was an active substitution hint
      then produce *spell-out-substitution*
      else if previous hint was *spell-out-substitution*
        then produce *give-away-performable-step*
        else call `up-to-inference-rule`
  else call `elicit-give-away`
Case $H = 3$
  if C is wrong and it is at least the third wrong answer
  then produce *point-to-lesson* and stop
    {*The student is asked to read the lesson again. Afterwards, the algorithm starts anew.*}
  else produce *explain-meta-reasoning*
Case $H \geq 4$
  give away the answer and switch to didactic strategy; switch back after three consecutive correct answers with all counters reset

{*After three hints, the algorithm starts to guide the student more closely to avoid frustration.* }

Figure 1: The Function `Socratic`

Hint Content Determination

The content generation of hints relies on ontological information which goes beyond what is represented in ΩMEGA's knowledge base. Among others, this enhanced ontology is exploited for abstraction techniques, which are important features for expressing hints.

1: (`elicit-inf-rule`) Can you find a rule now that connects $\mathcal{P}$ and subset.
(*Subtask initiated*)
1′: (`elicit-more-gen-know`) What is the condition of belonging to/being an element of $\mathcal{P}$?
(*Subsubtask initiated*)
1″: (`elicit-basic-know`) What is the condition of belonging to a set in general?
2″: (`give-away-basic-know`) A subset of a set belongs to the set.
(*End of subsubtask*)
2′: (1′ repeated) What is the condition of belonging to/being an element of $\mathcal{P}$?
3′: (`give-away-more-gen-know`) A set A is an element of $\mathcal{P}(B)$ if the set is a subset of B.
(*End of subtask*)
2: (1 repeated) Do you know which rule you need to use?
3: (`give-away-inf-rule`) You need to apply the definition of $\mathcal{P}$, namely $A \in \mathcal{P}(B) \Rightarrow A \subseteq B$.

Figure 2: Subtasks

An Enhanced Domain Ontology

ΩMEGA's knowledge base is organized as an inheritance network. The semantics of mathematical objects is expressed in terms of λ-calculus expressions which constitute precise and complete logical definitions required for proving purposes. Inheritance is merely used to percolate specifications efficiently, to avoid redundancy and ease maintenance, but not even hierarchical structuring is imposed. Meeting purposes of hinting, in contrast, does not require access to the complete logical definitions, but does require several pieces of information that go beyond what is represented explicitly in ΩMEGA's knowledge base. In order to meet these requirements, the domain objects are reorganized in a specialization hierarchy, and semantic relations that exist implicitly in ΩMEGA's knowledge base are expressed explicitly.

We consider relations between mathematical concepts, relations between mathematical concepts and inference rules, and relations among mathematical concepts, formulae and inference rules. By making these relations explicit we enhance the existing mathematical database into an ontology for tutoring mathematics, as these concepts and relations can be used to hint at anchoring points for proving, providing, but not imposing, students with heuristics and proving models that facilitate learning (Lim & Moore 2002; Owen, Sweller, & Olson 1985).

In the following, we shall give some examples of such explicitly defined relations. We use the following inference rule in our examples:

$$\frac{U \subseteq V \quad V \subseteq U}{U = V}\text{Set=}$$

where U and V are sets.

Let σ, σ' be mathematical concepts, R be an inference rule and $\varphi_1, \ldots, \varphi_n, \psi$ formulae, where $\varphi_1, \ldots, \varphi_n$ are the premises and ψ the conclusion of R.

Antithesis: σ is in *antithesis* to σ' if and only if it is its opposite concept (i.e., one is the logical negation of the other).
Examples: $\in$ and $\notin$ are in antithesis

Relevance: σ is *relevant* to R if and only if R can only be applied when σ is part of the formula at hand (either in the conclusion or in the premises).
Examples: $\subseteq$,$\supseteq$ and $=$ are relevant to rule Set=

Introduction and Elimination: Rule R *introduces* σ if and only if σ occurs in the conclusion ψ, but not in any of the premises $\varphi_1, \ldots, \varphi_n$. Rule R *eliminates* σ if and only if σ occurs in at least one of the premises $\varphi_1, \ldots, \varphi_n$, but not in the conclusion ψ.
Examples: Set= introduces $=$ and eliminates $\subseteq$ and $\supseteq$

This ontology is evoked primarily for the determination of the content of hint categories chosen by the socratic algorithm. Because we want to avoid the standard static gamed-based generation of hints, every hint category is chosen with respect to the session model of the student. This means that for every student and for his current performance on the proof being attempted, the hint category is chosen, which must be realized in a different way based on, for instance, the discourse structure and the dialog management. This, in turn, means that hint categories must constitute descriptive place holders for the actual realizations. The realization itself requires collaboration of the hinting algorithm with the dialog manager (cf. (Tsovaltzi & Karagjosova 2004)), on the one hand, and with the knowledge databases, on the other.

Taking these consideration into account, we define hint categories based on generic descriptions of domain objects or relations. The role of the ontology is to map the generic descriptions on the actual objects or relations that are used in the particular context, that is, in the particular proof at hand and the proof step in it. These concepts or relations are used for the actual generation of the hints which point to them as relevant anchoring points.

Abstraction Techniques

In the experiments underlying our corpus study, it turned out that the hints produced by the tutor according to the algorithm used were not as helpful as we had hoped. We believe that one of the problems for this lies in the complexity of the domain, since performing proof steps requires understanding at several levels of granularity. In contrast, the underlying system ΩMEGA as well as the hinting algorithm used for the collection of the data mainly provided information about domain objects and relations, but not about the reasoning involved in manipulating these objects at varying degrees of detail. In order to make hints more effective on the basis of the available domain knowledge, we have added abstractions in the hint presentations, thereby also enhancing the repertoire by adding hints addressing the problem from another perspective, which we call *functional* complementing the *conceptual* one. We conjecture that these measures in combination with the added subtask structure of the hinting algorithm, which can make use of them, can improve the student performance. The reason for it is that the varying degree of abstraction allows for a sort of meta-reasoning over the problem, which facilitates skill-transferability. Nonetheless, students are reluctant to phrase any meta-reasoning without such prompting (Conati & VanLehn 1999).

For building abstractions of hint specifications, we made a distinction between a *conceptual* view, which emphasizes

a relation to a mathematical concept, and a *functional* view, which emphasizes the effect imposed on the conclusion of an assertion (i.e., axiom, definition or theorem) under consideration. Hints in the conceptual view refer to assertions relating a central property of a mathematical concept to some assertion by an implication. Depending on the direction of the implication, that assertion expresses a condition for the property of the mathematical concept under consideration, or a consequence. In the functional view, the applicability of some descriptions is tested by comparing structural properties of the premises and conclusion of an inference rule, such as, the number of operators, the number of parentheses, and the appearance of a variable. Possible deriving hint content determinations are "simplify", "reduce the number of parentheses", and "eliminate a variable", respectively. Other descriptions are defined based on patterns that appear in the conclusion but not in the premises of the assertion. These patterns make use of wild card characters which represent abstractions of subformulae. For example, the pattern for isolating a variable is '$X*$', to which the conclusion but not the premises must be reducible by abstraction.

Next, the generation of the hint specification undergoes an abstraction process. The operations for this process depend on the view taken on the inference in question. For the conceptual view, the conceptual elements appearing in that rule are generalized, according to the underlying domain taxonomy. This abstraction can be used in a pure form, as the basis for a rather general hint. Moreover, subformulae of the concrete instance can be embedded in the abstracted form, to be used as more specific follow-up hints. For the functional view of an inference, the structure of the formula is subject to abstraction (in our domain). More concretely, atomic elements in a formula (variables, constants, and operators) are successively abstracted into a wild card character place holder. Moreover, two immediately adjacent wild card characters can be combined into a single one. This way, building formulae reduced to their structural embedding is possible, with selected variables still present.

Let us illustrate the potential uses of this method by an example. The theorem to be proved is the following ($\mathcal{P}$ stands for power set): $(A \cap B) \in \mathcal{P}((A \cup C) \cap (B \cup C))$.

When focusing on the conclusion obtained after applying distributivity to the expression in the scope of the power set operator, which is $(A \cap B) \cup C$, some of the possible and most useful abstractions for the present operation are: $((A \cup B) \cap *)$, $((A \cup B)*)$, $((A * B)*)$, $((*) \cap *)$. The last two fit the hint descriptions "Try to bring A and B closer together" (the last-but-one), and "Reduce the number of parentheses" or "Simplify" (the last one).

Focusing on the relation between the two sets related on top level in the theorem above, the underlying rule can be formulated as $x \in \mathcal{P}(y) \Rightarrow x \subseteq y$, which is essentially how the notion of power set is defined. The most accurate conceptualization for this relation is "the condition for belonging to a power set". Through generalization, this can be weakened into "the condition for belonging to a set", and simply "the condition".

Conversely, these expressions can be augmented by the expressions to which x and y are instantiated in this example: "the condition for $A \cap B$ belonging to $\mathcal{P}((A \cup C) \cap (B \cup C))$", and simply "the condition for $A \cap B$ being related to $(A \cup C) \cap (B \cup C)$".

Related Work

Several other tutoring systems tackle hinting, in one form or the other. We will only mention here the ones that we judge to be most related to our work.

Ms. Lindquist (Heffernan & Koedinger 2000), a tutoring system for high-school algebra, has some domain specific types of questions which are used for tutoring. Although there is some mention of hints, and the notion of gradually revealing information by rephrasing the question is prominent, there is no taxonomy of hints or any suggestions for dynamically producing them.

An analysis of hints can also be found in the CIRCSIM-Tutor (Hume *et al.* 1996), an intelligent tutoring system for blood circulation. Our work has been largely inspired by the CIRCSIM project both for the general planning of the hinting process and for the taxonomy of hints. CIRCSIM-Tutor uses domain specific hint tactics that are applied locally, but does not include a global hinting strategy that models the cognitive reasoning behind the choice of hints. We, instead, make use of the hinting history in a more structured manner. Our algorithm takes into account the kind of hints produced previously as well as the necessary pedagogical knowledge, and follows a smooth transition from less to more informative hints. Furthermore, we have defined a structured hint taxonomy with refined definition of classes and categories based on the passive vs. active distinction, which is similar to the active-passive continuum in CIRCSIM. These classes roughly correspond to algorithm functions, which resemble CIRCSIM tactics, but are again more detailed and more clearly defined. CIRCSIM-Tutor (Lee, Seu, & Evens 2002) uses an ontology at three levels: the knowledge of domain concepts, the computer context of tutoring and the meta-language on attacking the problem. Our ontology is not concerned with the second level. The first level corresponds to our existing knowledge base. The third level can be viewed as a simplified attempt to model tutoring, which we do via hinting. CIRCSIM-Tutor does, however, use its domain ontology in categorizing the student answer and fixing mistakes.

Matsuda and VanLehn (2003) research hinting for helping students with solving geometry proof problems. They orient themselves towards tracking the student's mixed directionality, which is characteristic of novices. We, on the contrary, follow psychological findings in providing a model, which assists the learning process with specific reference to the directionality of a proof, thus elevating unnecessary cognitive load and assisting schema acquisition (Owen, Sweller, & Olson 1985).

Conclusion and Future Work

In the DIALOG project, we use an adaptive hinting algorithm based on session modeling, and aim at dynamically producing hints that fit the needs of the student with regard to the particular proof. Additionally, we do not restrict ourselves to the use of a gamed of static hints, but use a taxonomy of hints based on domain content and cognitive func-

tion. That allows us an equally dynamic realization of hints that can take into account dialog and discourse management, as well as domain independent pedagogical knowledge.

The choice of hints proposed in this paper describes only the pedagogically motivated selection and the content determination of the hint categories. The realization of hints in terms of dialog moves and the interconnection between the dialog management and hint generation are currently being investigated (Tsovaltzi & Karagjosova 2004). Moreover, we plan to realize the hints and the other dialog moves by extending the system *P.rex*, a full-fledged natural language generation system that presents and explains mathematical proofs (Fiedler 2001).

References

Benzmüller, C.; Fiedler, A.; Gabsdil, M.; Horacek, H.; Kruijff-Korbayová, I.; Pinkal, M.; Siekmann, J.; Tsovaltzi, D.; Vo, B. Q.; and Wolska, M. 2003a. Tutorial dialogs on mathematical proofs. In *Proceedings of the IJCAI Workshop on Knowledge Representation and Automated Reasoning for E-Learning Systems*, 12–22.

Benzmüller, C.; Fiedler, A.; Gabsdil, M.; Horacek, H.; Kruijff-Korbayová, I.; Pinkal, M.; Siekmann, J.; Tsovaltzi, D.; Vo, B. Q.; and Wolska, M. 2003b. A Wizard-of-Oz experiment for tutorial dialogues in mathematics. In *Proceedings of the AIED Workshop on Advanced Technologies for Mathematics Education*, 471–481.

Benzmüller, C.; Fiedler, A.; Gabsdil, M.; Horacek, H.; Karagjosova, E.; Kruijff-Korbayová, I.; Tsovaltzi, D.; Vo, B. Q.; and Wolska, M. 2004. Annotating a corpus on mathematical tutorial dialog. In *Proceedings of the NAACL/HTL Conference Workshop CFP: Frontiers in Corpus Annotation 2004*. Submitted.

Chi, M. T. H.; de Leeuw, N.; Chiu, M.-H.; and Lavancher, C. 1994. Eliciting self-explanation improves understanding. *Cognitive Science* 18:439–477.

Conati, C., and VanLehn, K. 1999. Teaching metacognitive skills: implementation and evaluation of a tutoring system to guide self-explanation while learning from examples. In *Proceedings of AIED '99, 9th World Conference of Artificial Intelligence and Education*, 297–304.

Fiedler, A., and Horacek, H. 2001. Towards understanding the role of hints in tutorial dialogues. In *Proceedings of the BI-DIALOG 5th Workshop on Formal Semantics and Pragmatics in Dialogue*, 40–44.

Fiedler, A., and Tsovaltzi, D. 2003. Automating hinting in mathematical tutorial dialogue. In *Proceedings of the EACL-03 Workshop on Dialogue Systems: Interaction, Adaptation and Styles of Management*, 45–52.

Fiedler, A. 2001. Dialog-driven adaptation of explanations of proofs. In Nebel, B., ed., *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI)*, 1295–1300. Seattle, WA: Morgan Kaufmann.

Graesser, A., et al. 2003. AutoTutor improves deep learning of computer literacy: Is it the dialog or the talking head? In *Proceedings of the Conference on Artificial Intelligence in Education*, 47–54.

Heffernan, N. T., and Koedinger, K. R. 2000. Building a 3rd generation ITS for symbolization: Adding a tutorial model with multiple tutorial strategies. In *Proceedings of the ITS 2000 Workshop on Algebra Learning*.

Hume, G.; Michael, J.; Rovick, A.; and Evens, M. 1996. Student responses and follow up tutorial tactics in an ITS. In *Proceedings of the 9th Florida Artificial Intelligence Research Symposium*, 168–172.

Keller, J. M. 1987. Strategies for simulating the motivation to learn. *Performance and Instruction Journal* 26(8):1–7.

Lee, C. H.; Seu, J. H.; and Evens, M. W. 2002. Building an ontology for CIRCSIM-Tutor. In *Proceedings of the 13th Midwest AI and Cognitive Science Conference, MAICS-2002*, 161–168.

Lim, E. L., and Moore, D. W. 2002. Problem solving in geometry: Comparing the effects of non-goal specific instruction and conventional worked examples. *Journal of Educational Psychology* 22(5):591–612.

Matsuda, N., and VanLehn, K. 2003. Modelling hinting strategies for geometry theorem proving. In *Proceedings of the 9th International Conference on User Modeling*.

Moore, J., et al. 2000. The BE&E corpus. `www.hcrc.ed.ac.uk/~jmoore/tutoring/BEE_corpus.html`.

Moore, J. 2000. What makes human explanations effective? In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, 131–136. Hillsdale, NJ. Earlbaum.

Owen, E.; Sweller, J.; and Olson, T. W. 1985. What do students learn while solving mathematical problems? *Journal Educational Psychology* 77:272–284.

Rosé, C. P.; Moore, J. D.; VanLehn, K.; and Allbritton, D. 2001. A comparative evaluation of socratic versus didactic tutoring. In Moore, J., and Stenning, K., eds., *Proceedings 23rd Annual Conference of the Cognitive Science Society*.

Siekmann, J.; Benzmüller, C.; Brezhnev, V.; Cheikhrouhou, L.; Fiedler, A.; Franke, A.; Horacek, H.; Kohlhase, M.; Meier, A.; Melis, E.; Moschner, M.; Normann, I.; Pollet, M.; Sorge, V.; Ullrich, C.; Wirth, C.-P.; and Zimmer, J. 2002. Proof development with ΩMEGA. In Voronkov, A., ed., *Automated Deduction — CADE-18*, number 2392 in LNAI, 144–149. Springer Verlag.

Tsovaltzi, D., and Fiedler, A. 2003. An approach to facilitating reflection in a mathematics tutoring system. In *Proceedings of AIED Workshop on Learner Modelling for Reflection*, 278–287.

Tsovaltzi, D., and Karagjosova, E. 2004. A dialogue move taxonomy for tutorial dialogues. In *Proceedings of 5th SIGdial Workshop on Discourse and Dialogue*. Submitted.

Tsovaltzi, D., and Matheson, C. 2002. Formalising hinting in tutorial dialogues. In *Proceedings of EDILOG: 6th Workshop on the Semantics and Pragmatics of Dialogue*, 185–192.

Tsovaltzi, D. 2001. Formalising hinting in tutorial dialogues. Master's thesis, The University of Edinburgh, Scotland, UK.