

An Approach to Facilitating Reflection in a Mathematics Tutoring System

Dimitra TSOVALTZI

*Department of Computational Linguistics
Saarland University
dimitra@coli.uni-sb.de*

Armin FIEDLER

*Department of Computer Science
Saarland University
afiedler@cs.uni-sb.de*

Abstract. In this paper we present an approach which enables both reflection on the student's own line of reasoning and active learning. We combine the categorisation of the student answer against the expert domain knowledge, a hinting process that engages the student in actively reflecting upon his reasoning in relation to the experts reasoning, and clarification subdialogues in natural language initiated by the tutor or the student on the task.

1 Introduction

Despite empirical evidence that natural language dialogue capabilities are necessary for the success of tutorial sessions [1], only few state-of-the-art tutorial systems use natural-language style interaction that requires menu-based input or exact wording of the input [2, 3, 4]. In the DIALOG project [5], we aim at a mathematical tutoring dialogue system that employs an elaborate natural language dialogue component. To tutor mathematics, we need a formally encoded mathematical theory, means of classifying the student's input in terms of the knowledge of the domain demonstrated, and a theory of tutoring.¹

To that end, we enhanced the mathematical ontology of the state-of-the-art theorem prover Ω MEGA [6] by making explicit relations that can potentially be used in the tutoring. Moreover, to classify the student's input, we developed a categorisation scheme for student answers, which draws on the mathematical ontology.

As far as tutoring is concerned, we aim at a mathematical tutoring system that is able to tutor 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 generalise and apply the taught strategies on their own later on. Adhering to the psychological evidence for the high educational effect of hinting [7, 8], we propose to establish those tutoring aims by making use of the *socratic* tutoring method², whose decisive characteristic is the use of hints in order to achieve self-explanation [2, 8, 9]. Our work builds on the, little, systematic research done to date in the area [11, 12, 13].

¹ We will refer to the student as 'he' and to the tutor as 'she' for clarity. No gender preference is assumed.

² This method bears similarities to *contingent help* [10].

In order to model hinting, we have been developing a taxonomy of hints for the naive set theory, which is based on the previously mentioned mathematical ontology. This taxonomy is used by a hinting algorithm which models the socratic tutoring method by means of calling different hint categories according to an implicit student model [14].

The implicit student model of the hinting algorithm makes use of the student answer categorisation and of information on previous hints produced. It is different from standard user modelling, which we are not concerned with in this paper. Its aim is to use information from the running tutoring and the current progress of the student on the particular problem under consideration in order to produce hints. Those hints, in turn, intend to address the cognitive state of the student and cause him to reflect upon his answers and his reasoning.³

In this paper, we first describe the domain ontology in Section 2. Then, in Section 3 we look at a scheme for categorising the student answer and we outline the use of the domain ontology and subdialogues in categorising the student's input. Next, in Section 4 we give a brief overview of the taxonomy of hints and of the hinting algorithm. In Section 5 we discuss our work with regard to the notion of reflection. Finally, we discuss some relevant work in Section 6 and conclude the paper.

2 A Domain Ontology for Tutoring

In this section, we first show a part of Ω MEGA's [6] mathematical database. Then, we give some of the relations we have defined for the needs of the hinting process, for instance, for realising hint categories (cf. [16]) and for categorising the student's answer (cf. Sections 3 and 4).

A Mathematical Database In Ω MEGA's database, assertions are encoded in a simply-typed λ -calculus. In this paper, we concentrate on mathematical concepts from the naive set theory. These concepts draw on the types *sets* and *inhabitants* of sets. We give two examples of definitions of mathematical concepts in intuitive terms as well as in more formal terms paraphrasing but avoiding the λ -calculus encoding of Ω MEGA's database.

Let U, V be sets and let x be an inhabitant.

Element: The *elements* of a set are its inhabitants:

$x \in U$ if and only if x is an inhabitant of U .

Intersection: The *intersection* of two sets is the set of their common elements:

$U \cap V = \{x \mid x \in U \text{ and } x \in V\}$.

The existing database also includes lemmata and theorems that use concepts, as well as inference rules. Since every lemma and theorem can be rewritten as an inference rule and vice versa, we identify lemmata and theorems with inference rules henceforth.

Enhancing the Ontology The mathematical database implicitly represents many relations that can be used in tutorial dialogues. Further useful relations can be found when comparing the definitions of concepts with respect to common patterns. We

³ DIALOG uses ACTIVEMATH [15] as an external facility for user modelling.

consider relations between mathematical concepts, between mathematical concepts and inference rules, and among mathematical concepts, formulae and inference rules. By making these relations explicit we convert the mathematical database into an enhanced ontology that can be used in tutoring. Let us look at two examples of relations that we defined between mathematical concepts.

Let σ, σ' be mathematical concepts. We define the relations:

Antithesis: σ is in *antithesis* to σ' if and only if it is its opposite concept (i.e., its logical negation).

Examples: $\text{antithesis}(\in, \notin), \text{antithesis}(\subset, \not\subset)$

Hypotaxis: σ is in *hypotaxis* to σ' if and only if σ' is defined using σ . We say, σ is a *hypotaxon* of σ' , and σ' is a *hypertaxon* of σ .

Examples: $\text{hypotaxon}(\subseteq, \subset), \text{hypotaxon}(\in, \cap)$

3 Student Answer Categorisation

In this section we present a scheme for categorising student answers. We are only concerned here with the parts of the answer that address domain knowledge. The scheme provides a classification of the answer with regard to an expected answer, which is always approximated for the student's own line of reasoning. The line of reasoning is the formal proof that the student is performing with all proof steps and all transitions from one step to the next, as well as all potential explanations for the particular proof and for each step. The system's response is based on that categorisation, as in all traditional tutoring systems. The scheme becomes relevant as the output of the classification constitutes part of the input to the hinting algorithm, which models the hinting process. The process itself aims at encouraging the student to reflect on his line of reasoning and learn actively.

We have defined student categories based on their completeness and accuracy with respect to the expected answer. The *expected answer* is the proof step that is expected next by the student, according to the proof which has been produced by the system for the problem at hand. It includes premises, inference rule and conclusion.

In the following we define completeness and accuracy, as well as the parts of the answer that these terms apply to.

Proof Step Matching In order to allow the student to proceed with his own line of reasoning, we make use of that reasoning in helping him with the task. In order to follow the student's line of reasoning, we match his answer to one of the possible next steps in one proof of a set of possible proofs. Since the set of possible proofs might be too large to be manageable, we achieve that by having Ω MEGA [6] check the correctness and relevance of the student's answer. If the answer was not correct, Ω MEGA returns the step that is closest to the one the student intended to make. That step constitutes our expected answer. We can then evaluate the student's answer against this expected answer. Hence, the system gives guidance according to the proof that the student is attempting without super-imposing one of the alternatives.

Completeness vs. Accuracy We call an answer *complete* if and only if all parts of the expected answer are mentioned. We call a part of an answer *accurate* if and only if the propositional content of the part is the true and expected one.

From this definition of completeness it follows that completeness is dependent on domain objects, but not on the domain ontology. That is, the expected answer, which is the basis of the evaluation of the completeness of a student answer, necessarily makes use of objects in the domain. However, the relations of the objects in the domain are irrelevant to evaluating completeness. Completeness is a binary predicate, and only the presence or absence of objects in the student answer is relevant to it.

Accuracy, contrary to completeness, is dependent on the domain ontology. It refers to the appropriateness of the object in the student answer with respect to the expected object. An object is accurate, if and only if it is the exact expected one. We are using accuracy as a binary predicate in the same way that we do with completeness. However, we intend to extend the categorisation to include different degrees of accuracy. The relation in the domain hierarchy, which holds between the expected concept and the one the student provides, will constitute an additional criterion for the accuracy of an answer.

Parts of Answers We now define the relevant units for the categorisation of the student answer. A *part* is a premise, the conclusion and the inference rule of a proof step. The two former are mathematical formulae and must be explicitly mentioned for the proof to be complete. The inference rule can either be referred to nominally, or it can be represented as a formula itself. In the latter case, we just consider the inference rule as another premise. It is up to the student to commit to using the rule one way or the other. A *formula* is a higher-order predicate logic formula. Every symbol defined in the logic is a function. Formulae can constitute of subformulae to an arbitrary degree of embedding. *Constants* are 0-ary functions that constitute the lowest level of entities considered.

The Categories Let us now enumerate the categories of student answers based on our definitions of completeness and accuracy and with regard to the expected answer. Note, however, that although the following list is exhaustive further refinement is due.

Correct: An answer which is both *complete* and *accurate*.

Complete-Partially-Accurate: A complete answer with some inaccurate parts.

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

Incomplete-Accurate: An incomplete answer, with accurate present parts.

Incomplete-Partially-Accurate: An incomplete answer with some inaccurate parts.

Wrong: An answer which is both incomplete and inaccurate.

Using the Domain Ontology The ontology we presented in Section 2 is evoked, among other things, in categorising the student's answer. That is, the algorithm takes as input the analysed student answer. In analysing the latter, we compare it to the expected answer and look for the employment of necessary concepts. These necessary concepts are defined in terms of the ontology. The algorithm checks for the student's level of understanding by tracking the use of these concepts in the student answer to be addressed next. The hint to be produced is then picked according to the knowledge demonstrated by the student. Note that this knowledge might as well have already been provided by the system itself, in a previous hinting turn when dealing

with the same proof step. Since the algorithm only checks for generic descriptions of those concepts, we use the present ontology in order to map the descriptions onto the actual concepts relevant to the particular context.

Subdialogues The tutor can initiate subdialogues either in case of ambiguity or in case of inability to classify the student's answer. The students are given the opportunity to correct themselves, provide additional information on their reasoning and give essential information about the way they proceed with the task. Let us look at some examples.

Subdialogues are used to treat any irrelevant answers, which are not wrong in principle but do not contribute anything to the current proof, since they do not address the expected step. In these cases, the tutor informs the student of the status of his answer.

Another instance of subdialogue initiation is in case of potential typos. Since we do not want to count the latter towards the student model, we need to treat them differently from conceptual domain mistakes, such as wrong instantiations. We can prevent that by asking the student what he really meant to say. Our assumption is that if the student made a typo and not a conceptual domain mistake, he will realise it and correct it. If he does not correct it, we categorise the answer taking into account the domain mistake. We identify this kind of subdialogue with *alignment* [17], and an instance of it is the example in Figure 1. (The examples are taken from a corpus on mathematics tutorial dialogues in German, which we recently collected [18]. Translations are included, where necessary.) The tutor could not make sense of the student's utterance unless he substituted $A \cap B = \emptyset$ for $A \neq B$ and $B \cap K(B) = \emptyset$ for $B \neq K(B)$. The student could not correct himself, although he realised from the tutor's question that he had used the wrong symbol.

S5: wenn $A \subseteq K(B)$, dann $A \neq B$, weil $B \neq K(B)$ [if $A \subseteq K(B)$, then $A \neq B$, because $B \neq K(B)$]

T6: meinen Sie wirklich \neq oder etwas anderes? [Do you really mean \neq or something else?]

S6: $\not\subseteq$

Figure 1: Subject 13

Every time the tutor cannot categorise the student's answer because no matching can be found, she initiates a clarification subdialogue. This is done up to two times for each answer. If the answer cannot be matched to anything, it is categorised as wrong. A clarification dialogue of this kind can be seen in Figure 2. The student seems to be applying a rule, which is not recognised. Thus, the tutor asks the student to specify the rule.

S2: $P((A \cup C) \cap (B \cup C)) = P(A) \cap P(C) \cup P(B) \cap P(C)$

T2: Welche Regel haben Sie hier angewendet? [Which rule are you using here?]

Figure 2: Subject 21

Subdialogues can also be initiated by the student when, for instance, he is not content with the evaluation of his level. In Figure 3, the tutor provides a *give-away-inference-rule* hint, judging that the student does not have the knowledge that the hint provides. The student initiates a subdialogue and informs the tutor of his erroneous judgement. The tutor accepts that and encourages the student to apply his knowledge.

T5: Sie müssen die wenn-dann-Beziehung auflösen, indem Sie die Wahrheit der Voraussetzung annehmen. [*You have to eliminate the if-then-relation, by assuming that the presupposition is true.*]
S5: schon klar [*I already know.*]
T6: Dann tun Sie das bitte. [*Then, please, do it.*]

Figure 3: Subject 20

4 Hinting

Hint Taxonomy We have defined hints, which are produced by the hinting algorithm, based on the needs in the domain as they revealed through the domain ontology. The hint taxonomy captures the underlying function of hints that can be common for different surface realisations. This underlying function 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 in the hint taxonomy we have defined hint categories across two dimensions.

The first dimension distinguishes between the active and passive function of hints. The former refers to the information provided each time and the latter to the information that the hint aims at triggering in the student's current cognitive state, that is, the information elicited. Since some information often needs to be given away to elicit some other information, most hints have both the active and passive function.

The second dimension distinguishes between different classes of hints. Each of these classes consists of single hint categories that elaborate on one of the attributes of the proof step under consideration. Hint categories are grouped in classes according to the kind of information they address in relation to the domain and the proof. The classes are ordered with respect to the amount of information that they give away.

By and large, the hints of the passive function of a class in the second dimension constitute 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. A realisation of a *give-away-relevant-concept* hint when the tutor wants to point to the inference that eliminates an implication is given in Figure 4. This hint could precede the hint in Figure 3, which informs the student what to do with the if-then-relation, as the latter gives more information away.

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

Figure 4: Subject 23

For the complete hint taxonomy and a detailed discussion see [14].

A Hinting Algorithm A tutorial system ideally aims at having the student find the solution to a problem by himself. Only if the student gets stuck should the system intervene. There is pedagogical evidence [7, 8] that students learn better if the tutor does not give away the answer but instead gives hints that prompt the student for the correct answer. Accordingly, based on the work by Tsovaltzi [13] we have derived an algorithm that implements an eliciting strategy that is user-adaptive by choosing hints tailored to the students. Only if hints appear not to help does the algorithm switch to an explaining strategy, where it gives away the answer and explains it. We follow

Person and colleagues [2] and Rosé and colleagues [8] in calling the eliciting strategy *socratic* and the explaining strategy *didactic*.

The algorithm makes use of an implicit student model and takes into account the current and previous student answers. The particular input to the algorithm is the category that the student answer has been assigned, based on the student answer categorisation scheme, and the domain knowledge employed. Moreover, the algorithm computes whether to produce a hint and of which category that hint should be, based on the number of wrong answers, as well as the number and kind of hints already produced (cf. Sections 3 and 5).

We now examine the algorithm and the way the student's level is taken into account.

The Algorithm We present in this paper the main function of the algorithm which implements hinting. We derived this algorithm from empirical data, namely from the corpus collected in the BE&E project [8], with additional normalisations motivated by educational theories [13]. The function `socratic` produces hints directly or calls several other functions, which we do not examine in this paper. The functions check the domain knowledge available to the student and decide on producing the least informative hint.

For a more detailed description of the algorithm see [14].

The Function `socratic` The bulk of the work is done by the function `socratic`, which we only 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 dialogue is initiated (cf. Section 3). Note, however, that the function stops if the student gives the correct answer during that clarification dialogue, as that means that the student corrected himself. Otherwise, the function produces a hint in a user-adaptive manner.

In the following, H denotes the number of hints produced so far and C_{i-1} the category of the student's previous answer. Furthermore, the student answer category *inaccurate* is shorthand for one of the categories *complete-partially-accurate* or *complete-inaccurate* or *incomplete-partially-accurate*. A hint is then produced as follows:

```

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 four hints, the algorithm starts to guide the student more closely to avoid frustration. If the
  student is able to follow again the tutor's plan for addressing the task, the algorithm switches back
  to the socratic strategy and lets the student take over. If the student carries on giving correct answers
  the main algorithm guarantees that the tutor just accepts the answer and does not intervene further.
  Only if the student makes a mistake again will the hinting start anew with all counters reset.}

```

After having produced a hint the function `socratic` analyses the student's answer to that hint. If the student's answer is still not right the function `socratic` is recursively called.

5 Reflection

In order to strike a balance between guiding the student and allowing him to proceed with his own line of reasoning, we guide the student through hints that aim at causing him to reflect on what is missing or wrong in his answers and line of reasoning. Proof step matching makes it possible to follow the student's line of reasoning by choosing between different possible expected answers. The categorisation scheme provides the necessary input for updating the implicit student model of the hinting algorithm. It gives the degree that the student's answer fits the expected answer. A second input for the implicit student model is whether the student possesses certain domain knowledge necessary for the task. The second dimension of the hint taxonomy captures that, as the hints in it provide gradually more of that domain knowledge. They are produced based on the input for the implicit student model.

The hinting process with the input described together with proof step matching aim at making the student reflect about his own line of reasoning by pointing out what is useful in it (relevant mathematical concepts, inference rules etc.) and by providing a hint in order to advance in the task. Therefore, the hinting process gives him both the opportunity to reconsider his answer in an immediate and encouraging mode, and to take the system's guidance into account in order to learn actively. The student can thus benefit in three ways. First, he can reflect on his own reasoning by getting feedback on it. Second, it becomes cognitively easier to learn since learning presumably takes

place based on the structures that gave rise to the particular line of reasoning. Third, he gets the feeling of achievement, which is pedagogically encouraging.

6 Discussion

There are several approaches to categorising the student's input. Some of them are mostly relevant to parsing the input. We briefly discuss work similar to ours only with respect to classifying the domain knowledge demonstrated by the student, as we are not dealing with natural language processing in this paper.

AutoTutor [2] uses a sophisticated LSA (Latent Semantic Analysis) approach. It computes the *truth* of an input based on the maximum match between it and the training material results. It computes *relevance* by comparing it to expected answers. The latter are derived from the curriculum scripts. It further uses the notions of *completeness* and *compatibility*. Both these notions are defined as percentages of the relevant aspect in the expected answer.

Statistical methods, however, are insensitive to recognising linguistic phenomena crucial to the evaluation, such as negation, and LSA is insufficient for evaluating the kind of short answers representative of our domain. Moreover, we need a definition of completeness and accuracy that gives us insight to the part of the student answer that is problematic and hence needs to be addressed by appropriate hinting. Therefore, defining them in terms of percentages is not enough. We need to represent the relevant domain entities that can serve as the basis for the hint production.

Why-Atlas [19] uses theorem proving techniques based on abduction for classifying student answers. Elaborate inferences can be made about the student's reasoning with this approach. This is similar to our approach with the difference that we use deductive instead of abductive reasoning. Abductive reasoning, however, could prove to be useful to detect misconceptions.

CIRCSIM [20, 21] and PACT-Geometry tutor [22] both use reasoning based on domain knowledge, similar to the approach presented in this paper. They evaluate the conceptual content of the student's response by use of domain ontologies. CIRCSIM uses a fine-grained classification. Unfortunately, the rationale of the classification is not documented. That makes comparison difficult. However, a list of ways of responding after a particular classification is available. Still no flexibility is allowed of the kind we are attempting by automating hinting.

PACT-Geometry [22] tutor uses a cognitive model of the ideal student based on production rules, which is used to estimate the student's skills. A semantic representation of the student's input is built that makes use of a domain ontology in order to resolve concepts used and their relations. The correctness of the reasoning used is based on a hierarchy of explanation categories that includes legitimate reasoning steps in the domain. Our work is similar to this approach in the use of organised domain knowledge to evaluate the student's answer and in providing hints. PACT-Geometry tutor aims at helping students with self-explanation. It does not have a structured approach of the kind that we have presented in this paper, but it uses a more elaborate student model to understand what exactly, if possible, the student needs help with.

Finally, STyLE-OLM [23] uses an interactive open student model. The learner has access to his model and can negotiate about it and alter it. It uses a graphical representation of the student's beliefs and the theory of dialogue games for the

dialogue management. The learner's beliefs are approximated by comparing his behaviour to the domain knowledge required. We also aim at getting the student to reflect on his answer and at engaging him in subdialogues that allow us to evaluate the student answer in a more accurate way. However, our approach differs in that we are building a natural language tutorial dialogue system that uses the process of hinting as a method of making the student active in performing the task. Hints intend to address the cognitive state of the student and cause him to reflect upon his reasoning, compare it gradually to the desired reasoning and augment or correct it. Automating the hinting process provides flexibility with the hope of fitting the student's needs better. It allows capturing discourse structure in the generation of hints through natural language. This facilitates the student's understanding further [1].

7 Conclusion and Future Work

We presented an approach which actively involves the student in the learning process by making him reflect upon his reasoning and constitutes the basis for a high learning effect. We examined some of the prerequisites for modelling this approach, namely (i) the enhancement of a domain ontology for the naive set theory in mathematics, (ii) a hint taxonomy, which was derived with the aim of automating the hint categories defined in it and (iii) a hinting algorithm.

Moreover, we presented a student answer categorisation scheme which uses the notions of expected answer, completeness and accuracy in evaluating the student's domain knowledge. This categorisation is further elaborated for evaluating the student's input by clarification subdialogues initiated by the tutor, as well as by allowing the student himself to initiate them.

In the future we plan to make further developments in our domain ontology to improve the evaluation of the student's answer, for instance, by capturing different degrees of accuracy of the parts in the answer. We will also refine our student answer categories and model more subdialogues to better treat the refined categories. All these improvements will, in turn, help us augment the hinting algorithm. To meet these aims, we will use the empirical data from the corpus on mathematics tutorial dialogues that we recently collected [18].

References

1. Johanna Moore. What makes human explanations effective? In Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society, pages 131-136. Hillsdale, NJ. Earlbaum, 2000.
2. Natalie K. Person, Arthur C. Graesser, Derek Harter, Eric Mathews, and the Tutoring Research Group. Dialog move generation and conversation management in AutoTutor. In Rosé and Freedman [24], pages 45-51.
3. Vincent Aleven and Kenneth Koedinger. The need for tutorial dialog to support self-explanation. In Rosé and Freedman [24], pages 65-73.
4. Neil Heffernan and Kenneth Koedinger. Intelligent tutoring systems are missing the tutor: Building a more strategic dialog-based tutor. In Rosé and Freedman [24], pages 14-19.
5. Manfred Pinkal, Jörg Siekmann, and Christoph Benz Müller. Projektantrag Teilprojekt MI3 DIALOG: Tutorieller Dialog mit einem mathematischen Assistenzsystem. In Fortsetzungsantrag SFB 378—Ressourcenadaptive kognitive Prozesse, Universität des Saarlandes, Saarbrücken, Germany, 2001.
6. Jörg Siekmann, Christoph Benz Müller, Vladimir Brezhnev, Lassaad Cheikhrouhou, Armin Fiedler, Andreas Franke, Helmut Horacek, Michael Kohlhase, Andreas Meier, Erica Melis, Markus Moschner, Immanuel Normann, Martin Pollet, Volker Sorge, Carsten Ullrich, Claus-Peter Wirth, and

- Jürgen Zimmer. Proof development with Ω MEGA. In Andrei Voronkov, editor, *Automated Deduction—CADE-18*, number 2392 in LNAI, pages 144-149. Springer Verlag, 2002.
7. Michelene T. H. Chi, Nicholas de Leeuw, Mei-Hung Chiu, and Christian Lavancher. Eliciting self-explanation improves understanding. *Cognitive Science*, 18:439-477, 1994.
 8. Carolyn P. Rosé, Johanna D. Moore, Kurt VanLehn, and David Allbritton. A comparative evaluation of socratic versus didactic tutoring. In Johanna Moore and Keith Stenning, editors, *Proceedings of the 23rd Annual Conference of the Cognitive Science Society*, University of Edinburgh, Scotland, UK, 2001.
 9. Dimitra Tsovaltzi and Colin Matheson. Formalising hinting in tutorial dialogues. In *EDILOG: 6th workshop on the semantics and pragmatics of dialogue*, pages 185-192, Edinburgh, Scotland, UK, 2002.
 10. Heather A. Wood and David J. Wood. Help seeking, learning and contingent tutoring. In *Computers and Education*, 33:153-169, 1999.
 11. Gregory D. Hume, Joel A. Michael, Rovick A. Allen, and Martha W. Evens. Hinting as a tactic in one-on-one tutoring. *Journal of the Learning Sciences*, 5(1):23-47, 1996.
 12. Armin Fiedler and Helmut Horacek. Towards understanding the role of hints in tutorial dialogues. In *BI-DIALOG: 5th Workshop on Formal Semantics and Pragmatics in Dialogue*, pages 40-44, Bielefeld, Germany, 2001.
 13. Dimitra Tsovaltzi. Formalising hinting in tutorial dialogues. Master's thesis, The University of Edinburgh, Scotland, UK, 2001.
 14. Armin Fiedler and Dimitra Tsovaltzi. 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, 2003.
 15. Erica Melis, Eric Andres, Andreas Franke, George Gogvadse, Michael Kohlhase, Paul Libbrecht, Martin Pollet, and Carsten Ullrich. A generic and adaptive web-based learning environment. In *Artificial Intelligence and Education*, pages 385-407, 2001.
 16. Dimitra Tsovaltzi and Armin Fiedler. Enhancement and use of a mathematical ontology in a tutorial dialogue system. In *Proceedings of the IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Acapulco, Mexico, 2003. In press.
 17. Jean Carletta, Amy Isard, Stephen Isard, Jacqueline C. Kowtko, Gwyneth Doherty-Sneddon, and Anne H. Anderson. The reliability of a dialogue structure coding scheme. *Computational Linguistics*, 23(1):13-32, 1997.
 18. Chris Benz Müller, Armin Fiedler, Malte Gabsdil, Helmut Horacek, Ivana Kruijff-Korbayová, Manfred Pinkal, Jörg Siekmann, Dimitra Tsovaltzi, Bao Quoc Vo, and Magdalena Wolska. A Wizard-of-Oz experiment for tutorial dialogues in mathematics. In *Proceedings of the AIED Workshop on Advanced Technologies for Mathematics Education*, Sidney, Australia, 2003. In press.
 19. Pamela W. Jordan and Kurt VanLehn. Discourse processing for explanatory essays in tutorial applications. In *Proceedings of the 3rd SIGdial Workshop on Discourse and Dialogue*, Philadelphia, USA, 2002.
 20. Zhou Yujian, Reva Freedman, Michael Glass, Joel A. Michael, Allen A. Rovick, and Martha W. Evens. What should the tutor do when the student cannot answer a question? In *Proceedings of the Twelfth International Florida AI Research Society Conference (FLAIRS-99)*, pages 187-191, Orlando, FL, 1999. AAAI Press.
 21. Michael Glass. Processing language input in the CIRCSIM-Tutor intelligent tutoring system. In Johanna Moore, Carol Luckhardt Redfield, and W. Lewis Johnson, editors, *Artificial Intelligence in Education*, pages 210-212. IOS Press, 2001.
 22. Vincent Aleven, Ocar Popescu, and Kenneth R. Koedinger. A tutorial dialogue system with knowledge-based understanding and classification of student explanations. In *Working Notes of 2nd IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Seattle, USA, 2001.
 23. Vania Dimitrova, John Self, and Paul Brna. The interactive maintenance of open learner models. In S. P. Lajoie and M. Vivet, editors, *Artificial intelligence in education*, pages 405-412. IOS Press, Amsterdam, 1999.
 24. Carolyn P. Rosé and Reva Freedman, editors. *Building Dialog Systems for Tutorial Applications Papers from the AAAI Fall Symposium*, North Falmouth, MA, 2000. AAAI press.