Resource-bounded Modelling and Analysis of Human-level Interactive Proofs

Christoph Benzmüller, Marvin Schiller, and Jörg Siekmann

1 Introduction

Mathematics is the *lingua franca* of modern science, not least because of its conciseness and abstractive power. The ability to prove mathematical theorems is a key prerequisite in many fields of modern science, and the training of how to do proofs therefore plays a major part in the education of students in these subjects. Computer-supported learning is an increasingly important form of study since it allows for independent learning and individualised instruction.

Our research aims at partially automating intelligent tutoring of mathematical proofs. This research direction is interesting not least because of the large number of potential users of such systems, including students who in addition to an introductory university lecture want to exercise their theorem proving skills, learners without access to university courses, and engineers who want to freshen their skills. Furthermore, the research direction is interesting because of the non-trivial challenge it poses to artificial intelligence, computational linguistics and e-learning: in order to achieve a powerful and effective intelligent proof tutoring system many research problems that are central to these areas have to be addressed and combined.

In the SFB 378 project DIALOG [13, 7, 9] (see also the paper [44] in this volume) we have revealed and addressed foundational research challenges that are crucial for realising intelligent computer-supported proof tutoring based on a flexible, natural

Christoph Benzmüller
FR 6.2 Informatik, Saarland University, Campus, Geb. E11, 66123 Saarbrücken, Germany e-mail: chris@ags.uni-sb.de

Marvin Schiller
FR 6.2 Informatik, Saarland University, Campus, Geb. E11, 66123 Saarbrücken, Germany e-mail: schiller@ags.uni-sb.de

Jörg Siekmann
DFKI GmbH and Saarland University, Campus, Geb. E11, 66123 Saarbrücken, Germany e-mail: siekmann@dfki.de

language-based dialogue between student and computer. In the proof tutoring scenario as studied in the project the student communicates proof steps to the tutorial system by embedding them in natural language utterances. The language used is a mixture of natural-language and mathematical expressions ("mathural" [28]). Proof construction is performed in a stepwise fashion, and the system responds to utterances with appropriate didactically useful feedback or also with hints. The student is free to build any valid proof of the theorem at hand.

To support the generation of appropriate feedback each proposed proof step needs to be analysed by the system in the context of the partial proof developed so far. For this reason, automating proof tutoring requires dynamic techniques that assess the student's proof steps on a case-by-case basis in order to generate the appropriate feedback. The feedback can take the form of confirming correct steps, drawing the student's attention to errors, and offering domain specific hints when the student gets stuck. In case the tutor system is asked to give a hint, the hint is generated in the context of the current proof, and it has to be exactly tailored to the situation in which the hint was requested. The ability to dynamically construct proofs, to dynamically analyse new proof steps, and to complete partial proofs to full proofs is thus an essential prerequisite for intelligent proof tutoring.

The scenario we finally envisage integrates the flexible, dialogue-based proof tutoring system we are aiming at with an interactive e-learning environment for mathematics. An example of an interactive e-learning environment is ActiveMath [17]. ActiveMath is a third generation e-learning system for school and university level learning as well as for self-study that offers new ways to learn mathematics. In ActiveMath the learner can, for example, choose among several learning scenarios, receive learning material tailored to her/his needs and interests, assemble individual courses her/himself, learn interactively and receive feedback in exercises, use interactive tools, and inspect the learner model and partially modify the system's beliefs about the student's capabilities and preferences. The flexible, dialogue-based proof tutoring system which we aim at shall ideally cooperate with such an e-learning environment. A learner taking an interactive course in the e-learning system shall be able to call it in order to exercise his/her theorem proving skills within the trained mathematical domain. Ideally, both the e-learning environment and the proof tutoring system share the formal mathematical content, the didactic goals and the student model. The exercise within the proof tutoring environment will then exploit this information and confirm, modify or refine the student model.

The combination of expertise from computational linguistics and from deduction systems made the research in the DIALOG project particularly interesting. Expertise from the former area was needed because of the choice of the flexible mathural language as communication means between student and system. Expertise in the latter area was needed for the development of techniques for dynamic proof management and dynamic proof step evaluation.

The remainder of the paper is organised as follows: In Section 2, we illustrate the initial position of our project, where the relative lack of data prompted empirical investigations. Based on the collected data, we formulate research challenges for proof tutoring in Section 3. A central role among those challenges is dynamic proof

step evaluation, which is approached in Section 4. Mathural processing is the subject of the article by Wolska *et al.* [44] in this book and human-oriented theorem proving in ΩMEGA is explained in the article by Autexier *et al.* [5]. This paper can be seen as a bridge between these two articles. Section 5 elaborates on didactic strategies, dialogue modelling, feedback generation and hints. Section 6 concludes the article and relates our work to other approaches in the field.

2 The Need for Experiments and Corpora

In order to make a start in this research direction, experiments were needed to obtain corpora that could guide our foundational research. Not only was little known about the type of natural language utterances used in student–tutor dialogues on proofs but also there was little work available about automating proof tutoring based on flexible student–tutor dialogues. To collect a corpus of data, which now forms the basis of our investigations into dialogue-based proof tutoring, we conducted two experiments in the Wizard-of-Oz style, which included the work of Magdalena Wolska (see the twin paper [44] in this volume).

Experiment 1

A first experiment [8] served to collect a corpus of tutorial dialogues in the domain of proof tutoring. It investigated the correspondence between domain-specific content and its linguistic realisation, and the use, distribution and linguistic realisation of dialogue moves in the mathematics domain. It also investigated three tutoring strategies, a *Socratic* tutoring strategy (cf. [33]), a *didactic* strategy and a *minimal feedback* strategy (where the subjects only obtained very brief feedback on the correctness of their attempts).

Setup. A tutorial dialogue system was simulated in the Wizard-of-Oz paradigm [25] i.e., with the help of a human expert. Twenty-four university students were instructed to evaluate the dialogue system. Their task was to solve exercises from naive set theory in collaboration with the system. The communication between student and tutor, who was hidden in a separate room, was mediated with a software tool DiaWoZ [19], which was specifically designed for that purpose. A comfortable and usable interface is important for three reasons; (i) in a Wizard-of-Oz setting, the tutor is more efficient in constructing responses and thus better able to conceal his identity, (ii) the system as such appears more mature (and thus plausible) to the student, which helps to further disguise the Wizard-of-Oz setup, (iii) in general, it allows a more fluent interaction with the system (and thus the student to concentrate on the learning activity). We like to thank the department of computer linguistics who provided the tools and facilities for this kind of experiments.

DiaWoZ supports complex dialogue specifications, which were needed in order to specify a particular hinting algorithm used in the *Socratic* tutoring condition. Di-

aWoZ allows keyboard-to-keyboard interaction between the wizard and the student. The interfaces consist mainly of a text window with the dialogue history and a menu bar providing mathematical symbols. Furthermore, the wizard can assign dialogue state transitions and speech act categories to student turns w.r.t. the underlying dialogue model. The DiaWoZ interface allowed a free mix of natural language text with mathematical symbols. However, we still found a number of aspects of the system which we were able to improve. For example, the participants suggested the use of the keyboard instead of the mouse for inserting mathematical symbols. Therefore, we have gradually improved our experiment software w.r.t. its interface components.

Corpus. The corpus of dialogues between students and the tutor contains data of 22 subject, i.e. 22 experiment sessions. Each session is subdivided into three subdialogues w.r.t to three different exercises that the students were required to work on in sequence. These 66 subdialogues contain on average 12 dialogue turns. In total, the corpus consists of 1115 dialogue turns (of which 393 are student turns), where an average student turn consists of one sentence, and an average tutor turn consists of two sentences. The example dialogue excerpt in Fig. 1 shows tutoring-related annotations (discussed further in [43] and [7]). The tutor turns are annotated with hint categories, the student turns are annotated with the student answer categories.

T1 Please show: If $A \subseteq K(B)$, then $B \subseteq K(A)$!
S1 (`wrong`) $A \subseteq B$
T2: (`give-away-relevant-concept`) That is not correct! First you have to consider the if-then-relation.
S2: (`wrong`) $A \subseteq K(K(A))$
T3: (`elaborate-domain-object`) That is correct, but at the moment not interesting. Do you know how to deal with the if-then-relation?

Fig. 1 Dialogue excerpt from the first experiment. (K denotes a set complement.)

Experiment 2

A second experiment investigated mathematical domain reasoning tasks and linguistic phenomena in tutorial dialogues. In contrast to the first experiment, it imposed less constraints on the wizards' tutoring and assumes a rather simple dialogue model.

Setup. Thirty-seven students from Saarland University interacted with the mock-up dialogue system, simulated with our software environment DiaWOz-II and four experts[1], who took the role of the wizard in turn. As a minimal requirement, students were required to have completed at least one university-level mathematics course.

[1] The experts consisted of the lecturer of a course *Foundations of Mathematics*, a maths teacher, and two maths graduates with teaching experience.

The students were instructed to solve mathematical exercises in collaboration with the system. The exercises were taken from the domain of relations, and were centred around the concepts of relation composition and relation inverse. Because of the advanced character of the exercises, the participants had to fulfil the prerequisite of having taken part in at least one mathematics course at university level prior to the experiment. At first, the subjects were required to fill out a questionnaire, collecting data about previous experiences with dialogue systems and their mathematics background. Subjects were also given study material with the mathematical definitions that were required to solve the exercises. The largest part of the two-hour experimental session was allotted to the interaction between the student and the simulated system.

Our WOZ environment DiaWOz-II [11] enables dialogues where natural language text is interleaved with mathematical notation, as is typical for (informal) mathematical proofs. The interface components of DiaWOz-II are based on the *what-you-see-is-what-you-get* scientific text editor TEXMACS[2] [22]. DiaWOz-II provides one interaction window for the user and one for the wizard, together with additional windows displaying instructions and domain material for the user, and additional notes and pre-formulated text fragments for the wizard. All of these windows allow for copying freely from one to the other. Furthermore, our DiaWOz-II allows the wizard to annotate user dialogue turns with their categorisation. DiaWOz-II is also connected to a spell-checker for checking both the user's and the wizard's utterances.

Corpus. The collected corpus contains the data of 37 subjects. The thirty-seven experiment sessions include a total of 1917 dialogue turns (980 by the wizards and 937 by the students). The students tried maximally four different exercises each. Unlike in the first experiment, the time spent on an exercise was not strictly limited. However, since the duration of the experiment session was limited to two hours, some students did not have the opportunity to do all exercises. The fourth exercise was considered a "challenge exercise", and therefore we expected that only some students would attempt it. On average, each student attempted 2.7 exercises (i.e., we have collected a total of 100 exercise-subdialogues).

The dialogues were annotated by the wizard during the experiment and recorded by DiaWOz-II. Any student utterance that represents a proof step was classified by the wizards w.r.t. three dimensions; *correctness* (i.e. *correct*, *partially correct* or *incorrect*), the *step size* (i.e. *appropriate*, *too detailed* or *too coarse-grained*), and *relevance* (i.e., *relevant*, *restricted relevance* or *irrelevant*). Statements from the student which did not represent a contribution to the proof (e.g. meta-comments or questions) were annotated with a placeholder *unknown*. A dialogue excerpt from the experiment together with annotations is shown in Fig. 2. In addition to the log-files recorded by DiaWOz-II, screen recordings were made. Furthermore, the participants were encouraged to "think aloud" and they were audio-recorded and filmed. This comprehensive collection of data not only documents the text of the tutorial dialogues, but also allows us to analyse how the participants used the interface and the

[2] www.texmacs.org

T1: Let R and S be relations in an arbitrary set M. It holds that: $(R \circ S)^{-1} = S^{-1} \circ R^{-1}$. Do the proof interactively with the system.
S1: Help me with the proof of the exercise
`(unknown; unknown; unknown)`
T2: In order to show the equality between two sets, one generally shows that the first set is a subset of the second set and vice versa
S2: $R^{-1} \subset R^{-1}$ `(incorrect; appropriate; irrelevant)`
T3: No, you first have to show: $(R \circ S)^{-1} \subseteq S^{-1} \circ R^{-1}$
S3: $(R \circ S)^{-1} \subseteq S^{-1} \circ R^{-1}$
`(unknown; appropriate; relevant)`
T3: Yes, that is to be shown. Therefore take an element from $(R \circ S)^{-1}$ and show that it is also in $S^{-1} \circ R^{-1}$.
S4: $R \subseteq S^{-1} \circ R^{-1}$
`(incorrect; appropriate; irrelevant)`
T5: No, that is not the right approach. Begin like this: Let $(x,y) \in (R \circ S)^{-1}$.
S5: $(x,y) \in (R \circ S)^{-1}$ `(correct; appropriate; relevant)`
T6: Now try to draw conclusions from this!
S6: $(x,y) \in S^{-1} \circ R^{-1}$
`(correct; too coarse-grained; relevant)`
T6: This cannot be concluded so directly. You need some intermediate steps.

Fig. 2 Dialogue excerpt from the second experiment. Annotations indicate the *correctness*, *granularity* and *relevance* of the student's proof step as judged by the tutor.

study material. The resulting corpus exhibits variety in the use of natural language and mathematical style. This variety enabled us – besides studying the task of proof step evaluation, as presented in the next section – to study the influence of the instructions presented to the students on the use of natural language, as illustrated in [10].

3 Main Challenges and Resources for Proof Tutoring

An analysis of our corpora revealed various challenges for the automation of proof tutoring based on flexible student–tutor dialogues. We present some of the main challenges here and point to the resources required by the tutor system to fruitfully address them. The success of mathural dialogue-based proof tutoring depends on:

A The student's knowledge and his learning abilities.
B The tutor system's mathural processing and mathural generation capabilities.
C The tutor system's ability to maintain and manage the dialogue state and the proof under construction.
D The tutor system's capability to dynamically judge about the proof steps uttered by the student.

E The tutor system's capabilities to perform its proof step analysis tasks with respect to a dynamically changing tutorial context.
F The tutor system's capabilities for a fine grained analysis of erroneous proof steps.
G The didactic strategy for feedback generation employed in the tutor system.
H The tutor system's capability to flexibly interleave the above tasks.

We now discuss the challenges for the tutor systems, that is, aspects B-H, in more detail. We take an application perspective on student modelling (challenge A) for addressing some of these aspects, therefore those aspects of student modelling relevant for proof tutoring will be discussed within the frame of challenges B-H.

B: Mathural Processing and Mathural Generation

An essential capability of human tutors in mathematics is their ability to successfully *communicate* with students. This communication process includes the task of processing the student's utterances as well as the generation of feedback understandable by the student. These processing and generation capabilities of the human tutor thus constitute an essential resource with respect to his success as a maths tutor. Analogously, powerful analysis and generation capabilities are amongst the most important resources required for any proof tutoring system which is based on flexible dialogues.

Processing natural language with embedded mathematical content, however, is a highly challenging task by itself. Among other things, it involves the problem of content underspecification and ambiguous formulation. Interestingly, underspecification also occurs in shaped-up textbook proofs [45]. To illustrate proof-step underspecification let us consider the dialogue excerpt in Fig. 3:

T1: Please show : $K((A \cup B) \cap (C \cup D)) = (K(A) \cap K(B)) \cup (K(C) \cap K(D))$
S1: by the deMorgan rule $K((A \cup B) \cap (C \cup D)) = (K(A \cup B) \cup K(C \cup D))$ holds.

Fig. 3 An excerpt from the corpus of the first experiment.

In Figure 3, the proof-step that the utterance **S1** expresses is highly underspecified from a proof construction viewpoint: it is neither mentioned how the assertion is related to the target formula, nor how and which deMorgan rule was used. **S1** can be obtained directly from the second deMorgan rule $\forall X,Y.K(X \cap Y) = K(X) \cup K(Y)$ by instantiating X with $(A \cup B)$ and Y with $(C \cup D)$. Alternatively, it could be inferred from T1 by applying the first deMorgan rule $\forall X,Y.K(X \cup Y) = K(X) \cap K(Y)$ from right to left to the subterms $K(A) \cap K(B)$ and $K(C) \cap K(D)$. Successful proof tutoring requires that the meaning of the student utterance can be sufficiently determined to allow further processing. The capability to differentiate and prioritise between proof construction alternatives as illustrated by our example is thus an important

resource of a tutoring system. And as illustrated, mathural processing may involve non-trivial domain reasoning tasks (here theorem proving tasks).

The corpora also illustrate the style and logical granularity of human-constructed proofs. The style is mainly declarative, for example, the students declaratively described the conclusions and some (or none) of the premises of their inferences. This is in contrast to the procedural style employed in many proof assistants where proof steps are invoked by calling rules, tactics, or methods, i.e., some proof refinement procedures. The hypothesis that assertion level reasoning [23] plays an essential role in this context has been confirmed. The fact that assertion level reasoning may by highly underspecified in human-constructed proofs, however, is a novel finding [3].

In this article we will not further discuss the processing and generation of mathural language and refer to the twin article [44] in this volume. In the following we assume that the meaning of a student utterance can always be successfully determined by the mathural processing resources available to the tutor system.[3]

C: Dialogue State and Proof Management

The successive dialogue moves performed by student and tutor form a dialogue state which is the context for the analysis of further moves. Part of this dialogue state is an incrementally developing partial proof object which is (hopefully) shared between the student and the tutor. It represents the status of the proof under development by the student at the given point in the dialogue. The maintenance and manipulation of such dynamically changing proof objects thus have to be realised in a proof tutoring system. Ideally the formalised proof objects in a tutor system closely match the mental proof objects as shared by students and human tutors. In particular, to support cognitively adequate proof step evaluation they should not differ significantly with respect to the underlying logical calculus and the granularity of the proof steps.

D: Proof Step Evaluation

A human tutor who has understood a proof step utterance of his student will subsequently analyse it in the given tutorial context. A main task thereby is to evaluate the *correctness*, *granularity* and the *relevance* of the student proof step in the given tutorial context. Let us neglect the tutorial context for the moment and concentrate, for better understanding, solely on the pure logical dimension of the problem. This pure logical dimension will be illustrated using the artificially simplified example in Fig. 4.

Correctness analysis requires that the domain reasoner can represent, reconstruct and validate the uttered proof step (including all the justifications used by the student) within the domain reasoner's representation of the proof state. Consider, for instance, utterance (a) in Fig. 4: Verification of the soundness of this utterance boils

[3] Note, that in practice we can support mathural processing with the help of clarification subdialogues or by appropriately restricting the flexibility of the mathural language.

Proof State

(A1) $A \wedge B$.
(A2) $A \Rightarrow C$.
(A3) $C \Rightarrow D$.
(A4) $F \Rightarrow B$.

(G) $D \vee E$.

Some Student Utterances

(a) From the assertions follows D.
(b) B holds.
(c) It is sufficient to show D.
(d) We show E.

Fig. 4 PSE example scenario: (A1)-(A4) are assertions that have been introduced in the discourse and that are available to prove the proof goal (G). (a)-(d) are examples for possible proof step directives of the student in this proof situation.

down to adding D as a new assertion to the proof state and to proving that: **(P1)** $(A \wedge B), (A \Rightarrow C), (C \Rightarrow D), (F \Rightarrow B) \vdash D$. Solving this proof task confirms the logical soundness of utterance (a). If further explicit justifications are provided in the student's utterance (e.g. a proof rule) then we have to take them into consideration and, for example, prove **(P1)** modulo these additional constraints.

Granularity evaluation requires analysing the 'complexity' or 'size' of proofs instead of asking for the mere existence of proofs. For instance, evaluating utterance (a) above boils down to judging the complexity of the generated proof task **(P1)**. Let us, for example, use Gentzen's natural deduction (ND) calculus as the proof system $\vdash$. As a first and naive logical granularity measure, we may determine the number of $\vdash$-steps in the smallest $\vdash$-proof of the proof task for the proof step utterance in question; this number is taken as the argumentative complexity of the uttered proof step. For example, the smallest ND proof for utterance (a) has '3' proof steps: we need one 'Conjunction-Elimination' step to extract A from $A \wedge B$, one 'Modus Ponens' step to obtain B from A and $A \Rightarrow B$, and another 'Modus Ponens' step to obtain C from B and $B \Rightarrow C$. On the other hand, the smallest ND proof for utterance (b) requires only '1' step: B follows from assertion $A \wedge B$ by 'Conjunction-Elimination'. If we now fix a threshold that tries to capture, in this sense, the 'maximally acceptable size of an argumentation' then we can distinguish between proof steps whose granularity is acceptable and those which are not. This threshold may be treated as a parameter determined by the tutorial setting. However, as we will further discuss in Section 4, using ND calculus together with an naive proof step counting is generally insufficient to solve the granularity challenge. More advanced approaches are needed.

Relevance asks questions about the usefulness and importance of a proof step with respect to the original proof task. For instance, in utterance (c) the proof goal $D \vee E$ is refined to the new proof goal D using backward reasoning, i.e., the previously open goal $D \vee E$ is closed and justified by a new goal. Answering the logical relevance question in this case requires to check whether a proof can still be generated in the new proof situation. In our case, the task is thus identical to proof task **(P1)**. A backward proof step that is not relevant according to this criterion is (d) since it reduces to the proof task: **(P2)** $(A \wedge B), (A \Rightarrow C), (C \Rightarrow D), (F \Rightarrow B) \vdash E$ for

which no proof can be generated. Thus, (d) is a sound refinement step that is not relevant.

E: Tutorial Context

Dynamic proof step evaluation enables tutoring in the spirit of didactic constructivism [26] (i.e., allowing the student to explore rather than expect him to follow a prescribed solution path). Dynamic proof step evaluation poses already a non-trivial challenge to mathematical domain reasoning if we assume a static tutorial context. In practice, however, the tutorial context dynamically changes. This tutorial context comprises the dynamically changing knowledge and experience of the student, the possibly dynamically changing teaching goal and strategy, and the dynamically changing knowledge of the teacher about the student's dynamically changing capabilities. Incorporating this dynamically changing context information poses an additional challenge to the tutor system's proof step evaluation mechanism since the system needs to adapt its analysis to both the tutorial model and the student model.

E: Failure Analysis

Context sensitive proof step evaluation supports the separation of acceptable from unacceptable student proof steps. In case of acceptable proof steps the student will be encouraged to continue his proof. More challenging is to compute and present useful feedback also in the case of unacceptable proof steps, that is, proof steps which are erroneous. Standard tutoring systems typically rely on information provided in advance by the author of teaching materials. Since we are in a setting where solutions are determined on the fly, we face the issue whether solution proofs can be dynamically annotated with information on the reason for failure. Such additional information provides important input for the generation of didactic useful feedback. In order to obtain such additional information of the reasons for failure the tutoring system needs to dynamically solve further analysis tasks in the domain reasoner.

G: Didactic Strategies, Feedback Generation and Hinting

The tutor can decide to offer a hint to the student in a number of situations, for example after repeated student errors, a long period of silence, or a direct request. Given information about the proof step such as correctness, granularity, and relevance as well as information about the student—encoded in the student model—the tutor should react in a way that optimises the progress of the student. In general, the behaviour of the tutor is encoded in a teaching strategy. *Socratic teaching* strategies that focus on posing questions to that student, not answers, have shown to be more effective than simply presenting the student with concrete parts of the solution.

H: Flexible Dialogue Modelling

Human maths tutors usually show impressive skills with respect to (at least) all of the aspects above. These tutoring skills—typically they are acquired in special training courses—are important resources limiting the tutor's capabilities for effective proof tutoring. These skills are consequently also important resources for an automated proof tutor system. It requires modules addressing these skills and these modules need to interact in a suitable way. Controlling the overall dialogue, invoking these modules, combining their results and determining appropriate dialogue moves is the task of the dialogue manager. Human tutors are generally capable of flexibly applying and interleaving their tutoring skills. This calls for flexible approaches and flexible architectures for dialogue management to convey this flexibility of human tutors to the tutor system. An example for an interleaving of skills has been hinted at before: in order to disambiguate a content-underspecified proof step utterance of the user, mathural processing may want to consult the proof manager and proof step evaluation in order to rule out incorrect readings. Details on the dialog management architecture are found in the twin article [44] in this volume.

4 Dynamic Proof Step Evaluation with ΩMEGA

A main focus in the DIALOG project has been on proof step evaluation. We have already argued that dynamic, context sensitive proof step evaluation requires support from a sophisticated and ideally cognitively adequate mathematical domain reasoner[4]. The ΩMEGA system, with its various support tools for human-oriented, abstract level proof representation and proof construction, has therefore been chosen as the domain reasoner of choice in the DIALOG project (see the paper by [44] in this volume).

Proof Management, Correctness Analysis and Content Underspecification

In the DIALOG context ΩMEGA is used to (i) represent the mathematical theory in which the proof exercise is carried out, that is definitions, axioms, and theorems of a certain mathematical domain, (ii) to represent the ongoing proof attempts of the student, in particular the management of ambiguous proof states resulting from underspecified or ambiguous proof steps the student enters (iii) to maintain the mathematical knowledge the student is allowed to use and to react to changes of this knowledge, and (iv) to reconstruct intermediate steps necessary to verify the

[4] There is a discrepancy between the level of argumentation in mathematics and the calculus level in contemporary automated theorem proving. We argue that cognitively motivated domain reasoning, such as reasoning at the assertion level, can overcome the limitations of theorem proving with commonly used calculi such as resolution or natural deduction calculi (cf. [12]).

correctness of a step entered by the student, thereby also resolving ambiguity and underspecification.

Proofs are represented in ΩMEGA's proof data structure (PDS) which allows the shared representation of several (ongoing) proof variants [4]. The PDS is a collection of proof trees (with nodes as multi-conclusion sequents), which can be linked to one another (in order to express the dependency of one proof on another one whose proof task is treated as a lemma).

These reconstructed proofs serve as the basis for further analysis of the students' proof steps w.r.t. granularity and relevance. Thereby, our analysis components take advantage of ΩMEGA's abstract proof representation at the assertion level. We now sketch some of the project achievements.

As the basis for proof step evaluation, each proof step proposed by the user is reconstructed in ΩMEGA. As explained before for the example student utterance S1 in Fig. 3, even most ordinary human proof steps can generally include a number of tacit intermediate steps, which become apparent when modelling these proof steps in a rigorous formal system. Therefore, the reconstruction generally requires proof search in order to determine the different (correct) readings of the student proof step.

The assessment module we have realised as part of the ΩMEGA system maintains an assertion level proof object that represents the current state of the proof under construction, which can include several proof alternatives in the case of underspecified, that is, insufficiently precise, proof step utterances by the student causing ambiguities (cf. [6, 16]). For each proof step uttered by the student, the module uses a *depth-limited breadth-first search* (with pruning of superfluous branches) to expand the given proof state to all possible successor states up to that depth. From these, those successor states that match the given utterance wrt. to some filter function (analysing whether a successor state is a possible reading of the student proof step) are selected. Thus, we have combined the resolving of *content underspecification and the verification of the correctness of a proof step* as a joint task in our solution in ΩMEGA. If a student proof step matches with a step in one of the possible assertion level proofs (expanding the current proof state to a certain depth) as generated by ΩMEGA, then it is considered as correct. The matcher and intermediate steps in the ΩMEGA proof object not addressed by the student are then the formally relevant content that was left unspecified in the student utterance.

This way we obtain, modulo our filter function, assertion level counterparts to all possible interpretations of correct student proof steps. If for a given utterance, no matching successor state can be reached, the utterance is considered as incorrect.

In [12] we report on a case study in which we applied our ΩMEGA based assessment module with a depth-limit of four assertion level steps to 17 dialogues from the second DIALOG corpus; these 17 dialogues contain a total of 147 proof steps. All the steps within a dialogue were passed to the assessment module sequentially until a step that is labelled as correct cannot be verified, in which case we move on to the next dialogue. This way, we correctly classify 141 out of the 147 steps (95.9%) as correct or wrong. Among the remaining six steps are three where the verification fails, and further three remain untouched.

This experiment confirms our decision in the ΩMEGA project to replace the previous ND based logical core by assertion level reasoning: Using breadth-first proof search for automated proof step analysis in proof tutoring appears unreasonable at first sight (if we employed ND calculus it likely would be, because of the challenging, deep search space; it remains unclear whether strategically guided ND proof search as employed e.g. in AProS [39] can sufficiently reduce the search). However, we employ assertion level breadth-first search which turns out to be well suited for the given task. This is because in ΩMEGA we obtain more adequate formal counterparts of the human proofs as was possible before: Looking at the human level proof steps in our corpus from the perspective of assertion level reasoning our analysis shows that they seldom exceed size three. Interestingly, already a depth limit of just four assertion level steps enables our breadth-first search based approach to correctly classify 95.9% of the proof steps in the corpus of the experiment.

Granularity Analysis

An early study within this project on granularity [37, 36, 35] investigated the use of proof reconstructions in natural deduction calculi for obtaining a measure for granularity. We investigated the viewpoint outlined in Section 3, where the number of steps in a formal deductive system (here, a natural deduction system) is treated as an indicator for granularity. Natural deduction is a self-evident first candidate for modelling human proof steps. We studied two human-oriented calculi, the classic natural deduction calculus by Gentzen [20] and the more recent psychologically-motivated calculus PSYCOP [32]. For our investigation, we made use of the proofs in the experiment corpus of the second Wizard-of-Oz experiment, where each step from the student is annotated with a granularity judgment by the human wizard, which can take one out of three values *too detailed*, *appropriate* or *too coarse-grained*.

In particular, we related the step size of proofs in the experiment corpus (as indicated by the wizards) to the step size of these two calculi. As reported in [37], large (i.e., *too coarse-grained*) proof steps (as identified by the wizards) corresponded usually to longer sequences of natural deduction inference applications. However, there remained a large gap w.r.t. step size remained between human-generated proofs and natural deduction proofs. A single proof step as it typically occurs in the experiment corpus generally requires numerous deduction steps at the level of natural deduction, which are often of rather technical nature. It became apparent that the sheer number of inference steps in the natural deduction proof reconstructions was an insufficient measure for granularity.

Learning Granularity Evaluation

The previous studies [37, 35] motivated the investigation of assertion-level proof reconstructions in ΩMEGA as a basis for granularity analysis. Furthermore, the exper-

iments hinted at other criteria as indicators for steps size besides counting the number of calculus-level inference steps required to reconstruct a human-made proof step. Based on the experiment corpus, we have identified a number of potentially granularity-relevant criteria.

Homogeneity: A single human-made step that involves the application of several different mathematical facts can is distinguished w.r.t granularity from a step where only one fact is applied several times.

Verbal Explanation: A human-made step that is accompanied by verbal justification of the argumentation (e.g., the name of a theorem, definition, etc.) is distinguished w.r.t. granularity from a step where only the result (possibly only a formula) is given.

Introduction of Hypotheses or Subgoals: Proof steps which introduce a new hypothesis or new subgoal are given a special status w.r.t. granularity.

Learning Progress: A proof step that involves (one or several) concepts that are known to the student (as recorded by a student model) can be distinguished w.r.t. granularity from proof steps involving (one or several) yet unknown (i.e., too-be-learnt) concepts.

For a given proof step, these criteria can easily be determined from the proof step's assertion-level reconstruction and with the help of a student model. Since, for example, proof reconstruction with ΩMEGA delivers the mathematical assertions employed in the reconstruction process (i.e. facts such as definitions, theorems, lemmata), the question whether these assertions are already known to the user can be answered by a simple lookup in the student model.

However, this leaves the question open in how far each of the criteria contributes to the overall verdict on granularity for that step.

We have developed an approach to learning the relationship between the criteria and the final granularity judgments from an empirical corpus, using standard machine-learning techniques. This allows us to adapt our framework to a particular mathematical domain and the style of a particular human tutor. Thus, we employ two modules for granularity analysis (see Fig. 5); one serves to obtain training instances, from which the associations between granularity criteria and granularity judgements can be learned. This results in a classifier, which is used within a second judgement module to automatically perform granularity judgements.

Training instances can be constructed from annotated corpora such as in the second experiment. Consider for example the utterance **S5** in Fig. 2, which is the first step in the dialogue the tutor considers correct. This utterance by the student is sent to the proof step analysis module (see Fig. 5), and again handed over to ΩMEGA for proof reconstruction, where it advances the proof state maintained by ΩMEGA by two assertion applications: (i) the (backward) application of the definition of $=$ (such that $(R \circ S)^{-1} \subseteq S^{-1} \circ R^{-1}$ and $S^{-1} \circ R^{-1} \subseteq (R \circ S)^{-1}$ remain to be shown), and (ii) the (backward) application of the definition of $\subseteq$, i.e. in order to show $(R \circ S)^{-1} \subseteq S^{-1} \circ R^{-1}$ it is assumed that $(x,y) \in (R \circ S)^{-1}$ and $(x,y) \in S^{-1} \circ R^{-1}$ remains to be shown. Proof step reconstruction thus delivers the information that two different concepts (the definitions of $=$ and $\subseteq$) were possibly employed by the

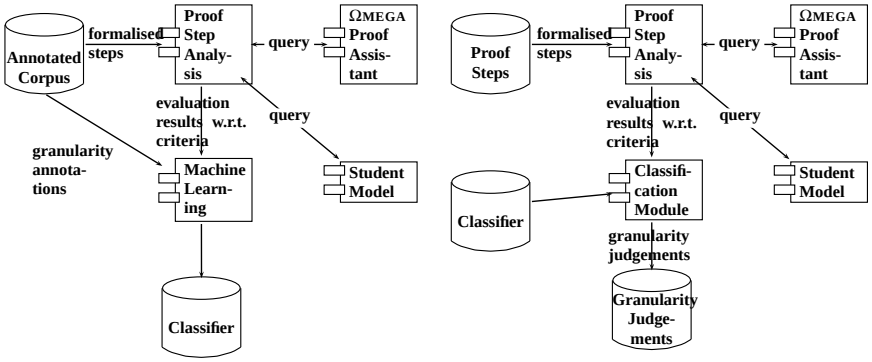

Fig. 5 Training module (left) and judgment module (right)

student in utterance **S5**. The proof step is now analysed with respect to the granularity criteria and a student model (which is updated during the course of the exercise). The results of our evaluation of the criteria for a given proof step are only numeric; they indicate how many assertion-level steps the reconstruction contains (in our running example '2'), how many different concepts the reconstruction involves (again '2'), how many inference steps are unexplained ('2', since the student does not mention the concepts verbally, this would have been for example: "By the definition of equality and the subset relation, ..."), how many times (if any) new subgoals or hypotheses are introduced (here '1' hypothesis, and '3' new subgoals), and how many concepts are new to the learner, according to the student model ('0', if we assume the student is familiar with naive set theory, and in particular equality and subset relation).

For each student step, these results of the analysis are combined with the judgement from the tutor, which is stored in the corpus, and the resulting instance is added to the set of training instances for machine learning. In our running example, the values of the analysis are associated to the verdict "appropriate", thus they become an example of an *appropriate* step for the machine learning algorithm. However, the evaluation values for the next step **S6** become an example for a *too coarse-grained* student step. The task performed by machine learning is to build a model of these examples (on a given training sample) that allows us to classify further new, yet unseen instances according to their granularity. Like the training module, the judgment module receives student proof steps, and analyses them with the help of ΩMEGA and the student model. However, the classifier learnt with the help of the training module now permits automatic granularity judgments.

Currently, we use C5 decision tree learning (see [34] and also [31]) as the learning algorithm. We have also compared this to the performance of other machine learning algorithms on our data, as reported in [38]. One result is that the classifier

SMO [30], which implements a support vector machine, achieved a better classification on our sample from the experiment corpus than C5 (see Fig.6).

	Naive classification	C5	SMO
Mean error	27.5	13.0	6.4
Kappa	0.0	0.65	0.84

Fig. 6 Performance of C5 and SMO on a sample of 47 proof steps from our corpus using 10-fold cross validation, compared to naively assigning all proof steps to the majority class *appropriate*.

However, using decision tree learning (as with C5) has the additional value that the resulting decision trees can easily be interpreted, and thus reveal which of the criteria are relevant to the granularity decisions (w.r.t. the particular corpus and a particular tutor). For example, a case study on a small test set produced the following decision tree depicted in Fig. 7.

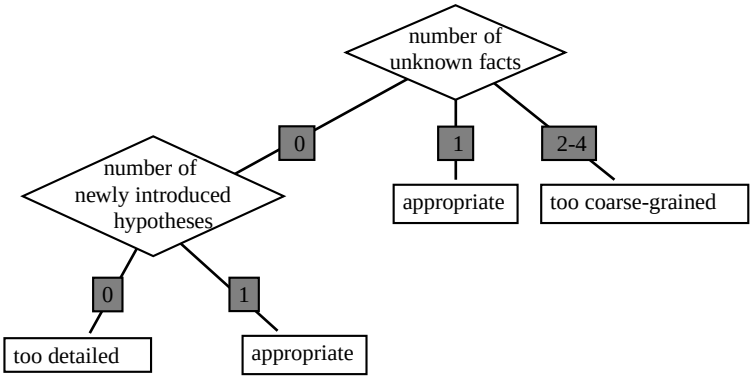

Fig. 7 Example decision tree produced during a case study

In this example, the algorithm has learnt that for the particular "judge" who has trained the system in the case study, the number of previously unknown facts (i.e., facts that the student has not used before), and the number of newly introduced hypotheses are those criteria that his explain the judge's behaviour w.r.t. granularity best. However, the criterion of verbosity has been pruned from the tree, which indicates that this criterion is not relevant for the particular training sample. Note that the judge only provides examples, and does not need to reflect about granularity criteria or the working of the training module at all.

Student Modelling

Student modelling is an indispensable ingredient for the analysis of the student's proof attempts and a prerequisite for the generation of tailored feedback. We draw on the experience of the ActiveMath project with various techniques of student modelling, and employ a student model based on [29], which is currently used for the LeActiveMath system [21]. For each concept, eight competencies are modelled, namely to *think*, *argue*, *model*, *solve*, *represent*, *language*, *communicate* and *use tools* w.r.t. the concept. The degree of mastery w.r.t each of these competencies is expressed in four levels: *elementary*, *simple-conceptual*, *complex* and *multi-step*. Using this well-established approach to student modelling allows us to embed the techniques of the DIALOG project into the LeActiveMath (and possible other) teaching environments.

The student model is seen as a dynamic component that further contributes to the adaptivity of the system. Competency levels in the student model associated with mathematical facts are updated whenever the mathematical fact is successfully applied, i.e. when it is employed in a student proof step verified by ΩMEGA as correct and categorised as appropriate w.r.t. granularity. Furthermore, each confirmed proof step is turned into a lemma (in case it is not already part of the assertions for the given mathematical domain) and added to the set of available assertions in ΩMEGA, together with a new student model entry. This allows to model common mathematical practice, where previously solved problems become building blocks for subsequent proof construction.

Further Work

As mentioned, a start has been made to incorporate a student model into granularity evaluation. However, more work is needed to incorporate further tutorial context information into proof step evaluation, in particular, into correctness and relevance evaluation. Relevance has generally only been preliminarily addressed in this project so far. This also applies to fine-grained failure analysis.

5 Didactic Strategies and Dialogue Modelling

The socratic teaching challenge has not been a main research focus of the DIALOG project. However, in close collaboration with our project, Tsovaltzi and Fiedler have studied hint taxonomies [41, 42] and dialogue-adaptive hinting [18].

Didactic Strategies and Hinting

The approach described in [18] is to dynamically produce hints that fit the needs of the student with regard to the particular proof. Thus, the proof tutor system cannot restrict itself to a repertoire of static hints, associating a student answer with a particular response by the system. [42] defines a multi-dimensional hint taxonomy where each dimension defines a decision point for the associated cognitive function. The domain knowledge can be structured and manipulated for tutoring decision purposes and generation considerations within a tutorial manager. Hint categories abstract from the strict specific domain information and the way it is used in the tutoring, so that it can be replaced for other domains. Thus, the teaching strategy and pedagogical strategy of the proof tutor system can be retained for different domains. More importantly, the discourse management aspects of the dialogue manager can be independently manipulated.

The hint taxonomy [42] was derived with regard to the underlying function of a hint that can be common for different NL realisations. This function is mainly responsible for the educational effect of hints.

Dialog Modelling

Dialogue systems act as conversational agents, that is, they look at an analysis of an incoming utterance and call on conversational expertise, encoded for instance as a dialogue grammar, to determine an appropriate response in the current dialogue state. A model of dialogue state, containing for example a record of utterances and their analyses, is continually updated to reflect the effect of both system and user utterances. In order to successfully manage and model tutorial dialogue, the dialogue state must be centrally stored and the results of computations by system modules, such as natural language analysis, must be made available. To satisfy these requirements, we have been working within a model characterised by a centrally-placed dialogue manager. The dialogue manager maintains the model of dialogue state, facilitates the collaboration of single system submodules and controls top-level execution in the system. By using the current dialogue state and accessing its model of conversational expertise, the dialogue manager is in the position to choose the most appropriate system response.

In developing a suitable model for managing tutorial dialogue we have met a number of challenges: How should we facilitate interleaving the processing carried out by system modules in the analysis of students' utterances? How can we combine the results of each module's analysis into a representation that forms the context of the choice of system response? And what information needs to be modelled at the dialogue level as opposed to the task or tutoring level?

In keeping with our project goal of flexible tutorial dialogues on mathematical proofs, we have been continually developing a demonstrator dialogue system which implements this type of model. It serves as a framework in which single modules can be tested, and this work is presented in the twin article [44] in this volume.

6 Related Work and Conclusion

All existing systems for proof tutoring employ automated theorem proving techniques for checking, analysing or generating complete proofs. Examples are the proof tutoring systems EPGY [40], ETPS [2], TUTCH [1], INTELLIGENT BOOK [14], and WINKE [15]. If proof checking fails the only feedback these systems return is that the step could not be verified. If the verification succeeds, there is no further analysis whether that proof step is actually relevant to complete the proof or whether it is of appropriate granularity. The ETPS-system is the only system which provides hints about how to continue the proof in case the student gets stuck. An approach that uses information about completed proofs is realised in WHY2-ATLAS [27]. This system checks a student's proof in two stages: First, it uses domain specific rules to generate different possible proofs for the given conjecture using abductive logic programming [24] and then it compares the found proofs (including those using some didactically motivated buggy rules) to the student's proof and selects the most similar to provide feedback to the student. Finally, the APROS project [39] uses automated proof search for natural deduction as the basis for the dynamic *Proof Tutor* system. It has been successfully used as a part of the course "Logic & Proofs" at Carnegie Mellon University, which at current time has been attended by more than 2000 students.

Main contributions of our work include empirical experiments that served to pinpoint the particular research challenges evoked by our ambitious dialogue scenario for dynamic proof tutoring. As a result, we determined that proof step evaluation does not only include the aspect of correctness, but also the analysis of granularity and relevance. Besides an elaborate discussion of the various functions relevant for proof tutoring and their overarching architecture, we have examined the analysis of granularity in detail, and developed an adaptive architecture based on the analysis of granularity-related features and machine learning techniques. Our work has resulted in a demonstrator system and prototype implementations, which allowed partial evaluation. Ultimately, such a system should be evaluated regarding its benefits for students' learning performance. However, this is still future work, since it already presupposes a high degree of maturity of the investigated system.

Acknowledgements

We are grateful to the reviewers of this paper who provided many useful comments and suggestions. The second author gratefully acknowledges the support from the German National Merit Foundation (Studienstiftung des deutschen Volkes e.V.).

References

[1] Abel A, Chang BYE, Pfenning F (2001) Human-readable machine-verifiable proofs for teaching constructive logic. In: Egly U, Fiedler A, Horacek H, Schmitt S (eds) Proceedings of the Workshop on Proof Transformations, Proof Presentations and Complexity of Proofs (PTP'01), Universitá degli studi di Siena

[2] Andrews P, Bishop M, Brown C, Issar S, Pfenning F, Xi H (2004) Etps: A system to help students write formal proofs. Journal of Automated Reasoning pp 75–92

[3] Autexier S, Benzmüller CE, Fiedler A, Horacek H, Vo BQ (2004) Assertion-level proof representation with under-specification. Electronic Notes in Theoretical Computer Science 93:5–23

[4] Autexier S, Benzmüller C, Dietrich D, Meier A, Wirth CP (2006) A generic modular data structure for proof attempts alternating on ideas and granularity. In: Kohlhase M (ed) Proceedings of the 5th International Conference on Mathematical Knowledge Management (MKM'05), Springer, LNAI, vol 3863, pp 126–142

[5] Autexier S, Benzmüller C, Dietrich D, Siekmann J (2008) Omega: Resource adaptive processes in an automated reasoning system. In: Crocker M, Siekmann J (eds) *Resource Adaptive Cognitive Processes*, chapter 4.5 of this volume, Springer

[6] Benzmüller C, Vo Q (2005) Mathematical domain reasoning tasks in natural language tutorial dialog on proofs. In: Veloso M, Kambhampati S (eds) Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05), AAAI Press / The MIT Press, Pittsburgh, Pennsylvania, USA, pp 516–522

[7] Benzmüller C, Fiedler A, Gabsdil M, Horacek H, Kruijff-Korbayová I, Pinkal M, Siekmann J, Tsovaltzi D, Vo BQ, Wolska M (2003) Tutorial dialogs on mathematical proofs. In: Proceedings of IJCAI-03 Workshop on Knowledge Representation and Automated Reasoning for E-Learning Systems, Acapulco, Mexico, pp 12–22

[8] Benzmüller C, Fiedler A, Gabsdil M, Horacek H, Kruijff-Korbayová I, Pinkal M, Siekmann J, Tsovaltzi D, Vo BQ, Wolska M (2003) A wizard of oz experiment for tutorial dialogues in mathematics. In: Proceedings of AI in Education (AIED 2003) Workshop on Advanced Technologies for Mathematics Education, Sydney, Australia

[9] Benzmüller C, Fiedler A, Gabsdil M, Horacek H, Kruijff-Korbayová I, Tsovaltzi D, Vo BQ, Wolska M (2003) Towards a principled approach to tutoring mathematical proofs. In: Proceedings of the Workshop on Expressive Media and Intelligent Tools for Learning, German Conference on AI (KI 2003), Hamburg, Germany

[10] Benzmüller C, Horacek H, Lesourd H, Kruijff-Korbajova I, Schiller M, Wolska M (2006) A corpus of tutorial dialogs on theorem proving; the influence of the presentation of the study-material. In: Proceedings of International Con-

ference on Language Resources and Evaluation (LREC 2006), ELDA, Genoa, Italy

[11] Benzmüller C, Horacek H, Lesourd H, Kruijff-Korbayová I, Schiller M, Wolska M (2006) Diawoz-II - a tool for wizard-of-oz experiments in mathematics. In: Freksa C, Kohlhase M, Schill K (eds) KI 2006: Advances in Artificial Intelligence. 29th Annual German Conference on AI, Springer, LNAI, vol 4314

[12] Benzmüller C, Dietrich D, Schiller M, Autexier S (2007) Deep inference for automated proof tutoring? In: Hertzberg J, Beetz M, Englert R (eds) KI, Springer, Lecture Notes in Computer Science, vol 4667, pp 435–439

[13] Benzmüller C, Horacek H, Kruijff-Korbayová I, Pinkal M, Siekmann J, Wolska M (2007) Natural language dialog with a tutor system for mathematical proofs. In: Lu R, Siekmann J, Ullrich C (eds) Cognitive Systems, Springer, LNAI, vol 4429

[14] Billingsley W, Robinson P (2007) Student proof exercises using mathtiles and isabelle/hol in an intelligent book. Journal of Automated Reasoning 39:181–218

[15] D'Agostino M, Endriss U (1998) Winke: A proof assistant for teaching logic. In: Proceedings of the First International Workshop on Labelled Deduction

[16] Dietrich D, Buckley M (2007) Verification of proof steps for tutoring mathematical proofs. In: Luckin R, Koedinger KR, Greer J (eds) Proceedings of the 13th International Conference on Artificial Intelligence in Education, IOS Press, Los Angeles, USA, vol 158, pp 560–562

[17] E Melis JS (2004) Activemath: An intelligent tutoring system for mathematics. In: Rutkowski L, Siekmann J, Tadeusiewicz R, Zadeh L (eds) Seventh International Conference 'Artificial Intelligence and Soft Computing' (ICAISC), Springer-Verlag, LNAI, vol 3070, pp 91–101

[18] Fiedler A, Tsovaltzi D (2005) Domain-knowledge manipulation for dialogue-adaptive hinting. In: Looi CK, McCalla G, Bredeweg B, Breuker J (eds) Artificial Intelligence in Education — Supporting Learning through Intelligent and Socially Informed Technology, IOS Press, no. 125 in Frontiers in Artificial Intelligence and Applications, pp 801–803

[19] Fiedler A, Gabsdil M, Horacek H (2004) A tool for supporting progressive refinement of wizard-of-oz experiments in natural language. In: Lester JC, Vicari RM, Paraguaçu F (eds) Intelligent Tutoring Systems — 7th International Conference (ITS 2004), Springer, no. 3220 in LNCS, pp 325–335

[20] Gentzen G (1934) Untersuchungen über das logische schließen. Math Zeitschrift 39:176–210, 405–431

[21] Goguadze G, Ullrich C, Melis E, Siekmann J, Gross C, Morales R (2004) Leactivemath structure and metadata model. Tech. rep., Saarland University

[22] Hoeven Jvd (2001) GNU texmacs: A free, structured, wysiwyg and technical text editor. In: Flipo D (ed) Le document au XXI-ième siècle, Metz, vol 39-40, pp 39–50, actes du congrès GUTenberg

[23] Huang X (1994) Human oriented proof presentation: A reconstructive approach. Phd thesis, Universität des Saarlandes, Saarbrücken, Germany, pub-

lished by *infix*, St. Augustin, Germany, Dissertationen zur Künstlichen Intelligenz, Volume 112, 1996
[24] Kakas A, Kowalski R, Toni F (1995) The role of abduction in logic programming. In: Handbook of Logic in Artificial Intelligence and Logic Programming, Oxford University, Oxford
[25] Kelley JF (1984) An iterative design methodology for user-friendly natural language office information applications. ACM Trans Inf Syst 2(1):26–41
[26] Liu CH, Matthews R (2005) Vygotsky's philosophy: Constructivism and its criticisms examined. Int Education J 6(3):386–399
[27] Makatchev M, Jordan P, VanLehn K (2004) Abductive theorem proving for analyzing student explanations to guide feedback in intelligent tutoring systems. Journal of Automated Reasoning 32:187–226
[28] Normand-Assadi S, Coulange L, Élisabeth Delozanne, Grugeon B (2004) Intelligent Tutoring Systems, Springer, chap Linguistic markers to improve the assessment of students in mathematics: An exploratory study, pp 380–389
[29] OECD (2004) Learning for tomorrow's world – first results from PISA 2003. OECD Publishing
[30] Platt J (1998) Machines using sequential minimal optimization. In: Schoelkopf B, Burges C, Smola A (eds) Advances in Kernel Methods - Support Vector Learning, MIT Press
[31] Quinlan JR (1993) C4.5: Programs for machine learning. Morgan Kaufmann
[32] Rips LJ (1994) The psychology of proof : deductive reasoning in human thinking. MIT Press, Cambridge, MA
[33] Rosé CP, Moore JD, VanLehn K, Allbritton D (2001) A comparative evaluation of socratic versus didactic tutoring. In: Proceedings of Cognitive Sciences Society
[34] RuleQuest Research (2007) Data mining tools see5 and c5.0. http://www.rulequest.com/see5-info.html
[35] Schiller M (2005) Mechanizing proof step evaluation for mathematics tutoring - the case of granularity. Master's thesis, Universität des Saarlandes, Saarbrücken, Germany
[36] Schiller M, Benzmüller C (2006) Granularity judgments in proof tutoring. In: Poster papers at KI 2006: Advances in Artificial Intelligence: 29th Annual German Conference on AI, Bremen, Germany
[37] Schiller M, Benzmüller C, de Veire AV (2006) Judging granularity for automated mathematics teaching. In: LPAR 2006 Short Papers Proceedings, Phnom Pehn, Cambodia, URL http://www.lix.polytechnique.fr/~hermann/LPAR2006/short/submission_147.pdf
[38] Schiller M, Dietrich D, Benzmüller C (2007) Proof step analysis for proof tutoring – a learning approach to granularity. Teaching Mathematics and Computer Science Submitted.
[39] Sieg W (2007) The apros project: Strategic thinking & computational logic. Logic Journal of IGPL 15(4)
[40] Sommer R, Nickols G (2004) A proof environment for teaching mathematics. Journal of Automated Reasoning 32:227–258

[41] Tsovaltzi D, Fiedler A (2005) Human-adaptive determination of natural language hints. In: Reed C (ed) Proceedings of IJCAI-05 Workshop on Computational Models of Natural Argument (CMNA), Edinburgh, UK, pp 84–88

[42] Tsovaltzi D, Fiedler A, Horacek H (2004) A multi-dimensional taxonomy for automating hinting. In: Lester JC, Vicari RM, Paraguaçu F (eds) Intelligent Tutoring Systems — 7th International Conference (ITS 2004), Springer, no. 3220 in LNCS, pp 772–781

[43] Wolska M, Vo BQ, Tsovaltzi D, Kruijff-Korbayová I, Karagjosova E, Horacek H, Gabsdil M, Fiedler A, Benzmüller C (2004) An annotated corpus of tutorial dialogs on mathematical theorem proving. In: Proceedings of International Conference on Language Resources and Evaluation (LREC 2004), ELDA, Lisbon, Portugal

[44] Wolska M, Buckley M, Horacek H, Kruijff-Korbayová I, Pinkal M (2008) Linguistic processing in a mathematics tutoring system: cooperative input interpretation and dialogue modelling. In: Crocker M, Siekmann J (eds) *Resource Adaptive Cognitive Processes*, chapter 3.2 of this volume, Springer

[45] Zinn C (2006) Supporting the formal verification of mathematical texts. J Applied Logic 4(4):592–621