Natural Language Dialog with a Tutor System for Mathematical Proofs*

Christoph Benzmüller[1], Helmut Horacek[1], Ivana Kruijff-Korbayová[2], Manfred Pinkal[2], Jörg Siekmann[1], and Magdalena Wolska[2]

[1]Computer Science, Saarland University, Saarbrücken, Germany
[2]Computational Linguistics, Saarland University, Saarbrücken, Germany

Abstract. Natural language interaction between a student and a tutoring or an assistance system for mathematics is a new multi-disciplinary challenge that requires the interaction of (i) advanced natural language processing, (ii) flexible tutorial dialog strategies including hints, and (iii) mathematical domain reasoning. This paper provides an overview on the current research in the multi-disciplinary research project DIALOG, whose goal is to build a prototype dialog-enabled system for teaching to do mathematical proofs. We present the crucial sub-systems in our architecture: the input understanding component and the domain reasoner. We present an interpretation method for mixed-language input consisting of informal and imprecise verbalization of mathematical content, and a proof manager that supports assertion-level automated theorem proving that is a crucial part of our domain reasoning module. Finally, we briefly report on an implementation of a demo system.

1 Introduction

The goal of the DIALOG project is to develop a conversational tutoring system helping students to construct proofs of mathematical theorems. Empirical evidence shows that collaborative problem solving, question answering, and error correction are among the most prominent features of naturalistic one-to-one tutoring and that efficient tutoring exhibits certain dialog patterns characteristic of these collaborative processes [16]. In our project, we aim at a flexible tutorial dialog in which students interact with the system by proposing proof steps using an unconstrained mixture of natural language and mathematical symbols, and the system responds with pedagogically plausible and effective feedback and guidance toward the solution.

Since little is known about the use of natural language in student–tutor dialogs about proofs, we conducted two data–collection experiments. Students with varying educational backgrounds and little to fair prior mathematical knowledge

* This work has been funded by the Deutsche Forschungsgemeinschaft (DFG) Collaborative Research Center on *Resource-Adaptive Cognitive Processes*, SFB 378 (`http://www.coli.uni-saarland.de/projects/sfb378/`).

R. Lu et al. (Eds.): Chinese-German Ws on Cognitive Systems 2005, LNAI 4429, pp. 1–14, 2007.

solved proofs in naive set theory[1] (e.g., $K((A \cup B) \cap (C \cup D)) = (K(A) \cap K(B)) \cup (K(C) \cap K(D))$) and binary relations (e.g., $(R \circ S)^{-1} = S^{-1} \circ R^{-1}$) with the help of a system simulated in a Wizard-of-Oz setup. Mathematics teachers were hired as tutors. Both students and tutors were allowed to formulate turns using natural language (German) typed on the keyboard and/or mathematical symbols available on a graphical interface. The students were instructed to enter proof steps, rather than complete proofs, to encourage a dialogue with the system.

More details on the setup of the first experiment can be found in [5] and the second in [8]. [32] and [9, 33] present the first and the second corpus respectively. Fig. 1 shows an excerpt from a typical session.[2]

The experience gained in the Wizard-of-Oz experiments and the analysis of the collected data let us identify three major research topics:

A. Interpretation of informal input mixing natural language and formal expressions. The mathematical content in the dialog utterances is (i) verbalized to varying degree, resulting in *a mixture of natural language and mathematical expressions*, and (ii) presented *informally* and *imprecisely*. These characteristics affect input analysis at the sentence-level, the discourse-level as well as at the level of domain interpretation. The language phenomena are by themselves not new, but the genre of an informal mathematical dialog adds new twists to them [6, 18]. The mixed language and the imprecision call for deep syntactic and semantic analysis to ensure a correct mapping of the surface input to the formal representation of the proof step.

B. Evaluation of proof steps. In order to evaluate a proof contribution, a domain reasoner that supports the tutoring process must be capable of judging contextual appropriateness of proof-steps. We identified the following appropriateness criteria pertinent to tutoring:

Soundness: Can the proof step be reconstructed by a formal inference system and logically and pedagogically verified?
Granularity: Is the 'argumentative complexity' or 'size' of the proof step logically and pedagogically acceptable?
Relevance: Is the proof step logically and pedagogically useful given the goal?

Existing Mathematical Assistant systems, such as ΩMEGA [28], are capable of interactive proof search and correctness verification. Soundness is a fairly tractable criterion for which different techniques are readily available [34]. However, proof step evaluation with respect to *granularity* and *relevance* is a novel and interesting application challenge for theorem proving systems.

C. Handling underspecification. A typical characteristic of proofs constructed by students is content underspecification and ambiguity. Interestingly, underspecificationalso occurs in shaped-up textbook proofs [34]. To illustrate

[1] K denotes the set complement.
[2] Here and further **T** and **S** denote tutor and student turns respectively.

T1: Bitte zeigen Sie: $A \cap B \in P((A \cup C) \cap (B \cup C))$!
Please show $A \cap B \in P((A \cup C) \cap (B \cup C))$ *!*
S1: $(A \cup C) \cap (B \cup C)) = C$
T2: Das ist falsch. Nach der Distributivität gilt: $(A \cup C) \cap (B \cup C) = (A \cap B) \cup C$. Somit gilt auch: $P((A \cup C) \cap (B \cup C)) = P((A \cap B) \cup C)$. Wie geht es nun weiter?
[This is wrong. By distributivity we have: $(A \cup C) \cap (B \cup C) = (A \cap B) \cup C$. *Therefore, we also have* $P((A \cup C) \cap (B \cup C)) = P((A \cap B) \cup C)$. *How to continue?]*
S2: $(A \cap B)$ muss in $P((A \cup C) \cap (B \cup C))$ sein, da $(A \cap B) \in (A \cap B) \cup C$
[$(A \cap B)$ *must be in* $P((A \cup C) \cap (B \cup C))$ *since* $(A \cap B) \in (A \cap B) \cup C$*]*
T3: Was meinen Sie mit: $(A \cap B)$ muss in $P((A \cup C) \cap (B \cup C))$ sein?
[What do you mean by: $(A \cap B)$ *must be in* $P((A \cup C) \cap (B \cup C))$*?]*
S3: da $P((A \cup C) \cap (B \cup C))$ nach Distributivgesetz gleich $P((A \cap B) \cup C)$ ist muss $(A \cap B)$ in $P((A \cup C) \cap (B \cup C))$ enthalten sein.
[since $P((A \cup C) \cap (B \cup C))$ *equals by distibutivity* $P((A \cap B) \cup C)$ $(A \cap B)$ *must be contained in* $P((A \cup C) \cap (B \cup C))$ *]*
T4: Was meinen Sie mit: muss $(A \cap B)$ in $P((A \cup C) \cap (B \cup C))$ enthalten sein. Meinen Sie Element oder Teilmenge?
[What do you mean by: $(A \cap B)$ *must be in* $P((A \cup C) \cap (B \cup C))$*. Do you mean as element or subset?]*
S4: $(A \cap B) \in P((A \cap B) \cup C)$
T5: Meinten Sie vorhin wirklich: $(A \cap B) \in (A \cap B) \cup C$?
[Do you really mean $(A \cap B) \in (A \cap B) \cup C$*?]*
S5: ja
[yes]
T6: Das ist falsch, denn $(A \cap B) \subseteq (A \cap B) \cup C$. $(A \cap B)$ und $(A \cap B) \cup C$ sind Mengen und keine Elemente voneinander. Was folgt als nächstes? ...
[This is wrong because $(A \cap B) \subseteq (A \cap B) \cup C$. $(A \cap B)$ *and* $(A \cap B) \cup C$ *are sets and not elements of each other. What comes next? ...]*

Fig. 1. An example dialog. P denotes the powerset.

proof-step underspecification let us consider the following excerpt from the first corpus:

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

From the point of view of linguistic analysis, **S** is unambiguous. However, the proof-step that the utterance 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. **S** 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 **T** 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 $K(C) \cap K(D)$. Proof assistant systems, typically require such detailed specification to execute the user's proof-step directive. Differentiating between proof construction alternatives can be important from the tutoring perspective.

Based on the empirical findings, we implemented a prototype system that can handle variations on the dialog in Fig. 1 and several other dialogs from our corpora. The demo system consists of a graphical user interface, an input analyzer, a proof manager, a tutorial manager, and a natural language generator. The modules are connected and controlled by an Information State Update-based dialogue manager [29]. Our research and, in particular, the implementation focus has been mainly on the *input analyzer*, whose task is to interpret and formally represent the linguistic content of the student's dialog contributions, and the *proof manager*, whose task is to evaluate the student proof step proposals with the help of a domain reasoner: the automated theorem prover ΩMEGA. For the other modules we provided baseline functionality required to carry out the dialogs. More details on the DIALOG demo system can be found in [10].

The remainder of this paper is organized as follows: In Sections 2 and 3 we describe our approach to mixed language interpretation and proof step evaluation, respectively. In Section 4, we overview the related work. In Section 5, we summarize and present the conclusions.

2 Interpreting Informal Mathematical Discourse

For student utterances that contain proof-relevant parts, such as **S2** in Fig. 1, the task of input interpretation is to identify these and represent them in a format interpretable by a domain reasoner. To ensure correct mapping to this representation, deep analysis is needed. It is further justified by the varying degrees of mathematical content verbalization and imprecision, common in the informal mathematical discourse in our corpus, as well as the need for consistency of interpretation required for proof-step evaluation by the domain reasoner.

In this section, we present an overview of our input interpretation procedure; we omit obvious pre-processing such as sentence- and word-tokenization. We first describe three basic components that provide minimal functionality required to analyze simple cases of mixed language. Then we discuss extensions for some of the more complex phenomena. For a more detailed discussion of language phenomena and interpretation procedure see [30, 31, 18, 19]

2.1 Baseline Processing

A simple utterance consisting of a mixture of mathematical expressions and natural language is the utterance **S** presented in Section 1: "by the deMorgan rule $K((A \cup B) \cap (C \cup D)) = (K(A \cup B) \cup K(C \cup D))$ holds.". We shall use it to illustrate the step-wise analysis process that proceeds as follows: we first identify mathematical expressions in order to encapsulate them before syntactic parsing.

During syntactic parsing, a domain-independent semantic representation is constructed. This we then refine to obtain a domain-specific interpretation suitable as input to the domain reasoner.

Mathematical expression parsing. To recognize and parse mathematical expressions we use knowledge about operators and identifiers the domain and relevant for the given problem, e.g., $\cup, \cap, K$. For the purpose of subsequent syntactic and semantic parsing, each mathematical expression is assigned a symbolic token corresponding to its type.

In our example, the expression $K((A \cup B) \cap (C \cup D)) = (K(A \cup B) \cup K(C \cup D))$ is assigned the type `FORMULA`. The token representing the expression type is substituted for the original expression resulting in the following input to the parser: "by the deMorgan rule `FORMULA` holds."

Syntactic parsing and semantic interpretation. The parser processes sentences and syntactically well-formed fragments, and constructs a representation of their *linguistic meaning* (LM). The LM is represented as a relational dependency structure closely corresponding to the *tectogrammatical level* in [26].

To obtain the LM, we use the OpenCCG parser (`openccg.sourceforge.net`) for which we develop a lexically-based Combinatory Categorial Grammar for German [13]. Our motivation for using this grammar framework is two-fold: first, it is well known for its account of coordination phenomena, widely present in mathematical discourse. Second, mathematical expressions, represented by their types, lend themselves to a perspicuous categorial treatment as follows: In the course of parsing, we treat symbolic tokens representing mathematical expressions on a par with lexical units. The parser's lexicon encodes entries for each mathematical expression type represented by its token (*e.g.* `TERM`, `FORMULA`) together with the syntactic categories the expression may take (*e.g.* the category of a noun phrase, `np`, for `TERM`, the category of a sentence, `s`, for `FORMULA`). By designing one grammar that allows a uniform treatment of the linguistic content and the mathematical content, we aim at a consistent analysis of different degrees of mathematical content verbalization.

Domain interpretation. The LM representations built by the parser are domain-independent. To obtain domain-specific interpretations, we implemented a stepwise procedure that gradually assigns domain-specific semantics to predicates and relations in the LM.

As a first step, we use a *s*emantic lexicon to map (parts of) the LM representations to domain-independent conceptual frames. The input structures are described in terms of tectogrammatical valency frames of lexemes that evoke a given concept. The output structures are the evoked concepts with roles indexed by tectogrammatical frame elements. Where relevant, sortal information for role fillers is given. For example, the `Norm` tectogrammatical relation (TR) evokes the concept of a *Rule*. The dependent in the `Norm`-relation identifies the rule according to which the main proposition holds.

As a second step, semantic lexicon concepts are mapped to domain-specific interpretations using a *domain ontology*. The ontology is an intermediate

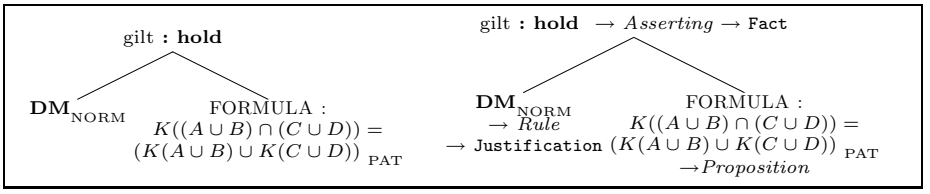

Fig. 2. Interpretation of the utterance "by the deMorgan rule $K((A \cup B) \cap (C \cup D)) = (K(A \cup B) \cup K(C \cup D))$ holds."; DM stands for DeMorgan

representation that mediates between the discrepant views of linguistic analysis and deduction systems' representation [17]; it thus has a potential of providing a direct link to logical definitions in a *mathematical knowledge base*, such as MBase [21]. The motivation for using an intermediate representation instead of directly accessing a mathematical knowledge base will become clear when we discuss ambiguity in Section 2.2.

Let us return to the example utterance: In Fig. 2 on the left, we show the representation of its linguistic meaning built by the parser. The structure consists of the German verb, "gilt", with the symbolic meaning **hold**, as the head, and two dependents in the TRs: Norm and Patient. The right part of Fig. 2 shows the assignment of domain-specific meaning: First, based on the semantic lexicon, the concept *Assertion* is assigned to **hold**, with Patient and Norm dependents as the *Proposition* and *Rule* respectively. Next, *Assertion* is interpreted as the **Fact** and the *Rule* as **Justification** in a proof-step. Applying re-writing transformations, we obtain the following underspecified representation used by the domain reasoner [4]: "**Fact** $K((A \cup B) \cap (C \cup D)) = (K(A \cup B) \cup K(C \cup D))$ **By** DeMorgan-1 **From .**".

The baseline processing described so far covers simple cases of the mixed language: it suffices to interpret utterances where terms or complete formulas are embedded within natural language parts. However, our corpus contains more complex cases of interleaved mathematical expressions and natural language. We turn to their processing in the next section.

2.2 Domain- and Context-Specific Processing

Our corpus contains a range of more complex phenomena typical for informal mathematical discourse, which the baseline processing described above cannot handle. In this section, we describe several extensions that we have implemented so far: parsing extensions to handle tightly interleaving mathematical expressions and natural language, and domain model extensions to handle ambiguous and imprecise formulations.

Parsing extensions. Input utterances often contain incomplete mathematical formulas interleaved with natural language expressions, where these two modes interact, *e.g.* (1) and (2) below. To handle these cases we made the following extensions: (i) the mathematical expression parser recovers information about

incomplete formulas using domain-knowledge of syntax and semantics of formal expressions, and (ii) the syntactic parser's lexicon contains the corresponding categories and their semantics.

For example, in (1), the parser recognizes the operator $\in$ as requiring two arguments: one of type *inhabitant* and the other *set*.

(1) $A \cap B$ ist $\in$ von $C \cup (A \cap B)$
$A \cap B$ *is* $\in$ *of* $C \cup (A \cap B)$

Accordingly, $\in$ is assigned a symbolic type `0_FORMULA_0`, where `0` indicates the arguments missing in the left and the right context. We have included a lexical entry `0_FORMULA_0` of syntactic category $\texttt{s/pp}_{lex:von}\texttt{\textbackslash np}$ in the lexicon of the syntactic parser.

Example (2) illustrates a case of tight interaction between mathematical expressions and the surrounding natural language:

(2) B enthaelt kein $x \in A$
B *contains no* $x \in A$

Here, the negation word "kein" has x, i.e. part of the expression $x \in A$, in its scope. The intended meaning of (2) can be paraphrased as B *contains no* x *such that* $x \in A$.

To account for this interaction we identify substructures of mathematical expressions that may lie in the scope of a natural language expression. In (2), the expression $x \in A$ is of type `FORMULA`. The relevant substructures are obtained by splitting the formula at the top node. As a result, we obtain two readings of the expression: $\texttt{TERM}_1$ $\texttt{0\_FORMULA}_1$ and $\texttt{FORMULA\_0}_2$ $\texttt{TERM}_2$ and parse the utterance with them. The lexical entry for `0_FORMULA` (formula with a missing left argument) is of syntactic category `s\np` (and semantics corresponding to `such that TERM has property FORMULA`), while the entry for `FORMULA_0` (formula with a missing right argument) is of category `s/np`. This allows us to obtain the intended reading of (2).

Domain modeling extensions. Another common phenomenon in informal mathematical discourse is ambiguity and/or imprecision. For example, the verb "enthalten" in the previously mentioned example (2), can in principle mean a subset, a membership, or a substring relation. To handle such cases, we extended the domain model. We added *semantic relations* that represent general concepts that subsume the specific mathematical relations. For example, the verb "enthalten" (*contain*) evokes either the concept of CONTAINMENT or that of STRUCTURAL COMPOSITION. CONTAINMENT in its most common interpretation specializes to the domain relations of (strict) SUBSET or ELEMENT with two roles: CONTAINER and CONTENTS. These are filled by the fillers of the TRs *Actor* (*act*) and *Patient* (*pat*) in the tectogrammatical valency frame of "enthalten", respectively. The STRUCTURAL COMPOSITION relation holds between a STRUCTURED OBJECT and its structural sub-component in the SUBSTRUCTURE role. Similarly, these roles are filled by the *Actor* and the *Patient* dependent, respectively. The semantic

$(\textbf{contain}_{pred}, \mathrm{x}_{act}, \mathrm{y}_{pat}) \rightarrow (\textsc{containment}_{pred}, \mathrm{container}_{act}, \mathrm{contents}_{pat})$ (a)
$(\textbf{contain}_{pred}, \mathrm{x}_{act:formula}, \mathrm{y}_{pat:formula}) \rightarrow$
$(\textsc{structural composition}_{pred}, \mathrm{structured\ object}_{act}, \mathrm{substructure}_{pat})$ (b)

Fig. 3. Example entries from the semantic lexicon

lexicon entry of the verb "enthalten" (*contain*) with the mappings of the concept roles described above is shown in Fig. 3. The domain ontology and semantic lexicon are presented in more detail in [31, 19].

We have so far concentrated on a proof-of-concept implementation of the input interpretation components. Our interpretation module is capable of producing analyzes of utterances similar to the ones in Fig. 1 as well as several variants thereof. We have implemented an OpenCCG grammar that covers variants of the syntactic structures of the utterances. We have also manually encoded the relevant part of the domain ontology required to account for the domain-specific interpretation. Our focus so far has not been on robust coverage, but rather on a systematic consistent representation of the most frequent constructions in the format readable by the domain reasoner.

3 Mathematical Domain Reasoning

Determining whether a proof step is appropriate requires that a mathematical domain reasoner can represent, reconstruct, and validate the proof step (including all the justifications used by the student) within its representation of the current proof.

Consider, for instance, utterance (a) in Fig. 4: Verification of the soundness of this utterance boils 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.

Soundness evaluation can be supported by different methods — including some that avoid dynamic theorem proving system. On one extreme "gold-standard" proofs could be selected and the proposed partial proofs could be matched against them. The other extreme would be to interpret the problem as a challenge to proof theory and try to develop and implement a proper proof theoretic approach to differentiate between 'pedagogically good' proofs and proof steps and pedagogically 'less acceptable' proofs and proof steps in the space of all proofs for a given problem. A notion of a 'good proof' is, for instance, presented in [11].

Soundness is, however, only one of the criteria along which a proof step should be evaluated in a tutorial context. For instance, a proof step may be formally relevant in purely logical terms, but considered irrelevant when additional tutorial aspects are taken into account. This is, for instance, the case when the goal

Proof State	**Example Student Utterances**
(A1) $A \wedge B$.	(a) From the assertions follows D.
(A2) $A \Rightarrow C$.	(b) B holds.
(A3) $C \Rightarrow D$.	(c) It is sufficient to show D.
(A4) $F \Rightarrow B$.	(d) We show E.
(G) $D \vee E$.	

Fig. 4. Proof step evaluation scenario: (A1)-(A4) are assertions that have been introduced in the discourse and that are available to prove the goal (G). (a)-(d) are typical examples of proof step proposed by students.

of the session is to teach a particular proof technique. Well known examples are proof by induction on the naturals, proof by structural induction, and proof by diagonalization. Often different proof techniques are applicable to one and the same problem and this causes pedagogically different, but formally correct and relevant, proof directions.

On the other hand, a step that is sufficiently close to a valid proof step may be considered pedagogically relevant while being logically irrelevant. Proof step evaluation should therefore support dynamic step-by-step analysis of the proof constructed by the student using the criteria of soundness, granularity and relevance not only with respect to a purely logical dimension, but also a tutorial dimension.

So far we have mainly focused on the logical dimension; the hypothesis is that the solution in the logical dimension is a prerequisite for solving the proof step evaluation problem involving also the tutorial dimension. Much further research in this direction is clearly needed.

In the following sections, we discuss some of the issues related to evaluating *granularity* and *relevance*. We illustrate the challenges using a constructed example in Fig. 4. See also [7].

3.1 Granularity

Granularity judgment refers to analysis of the 'complexity' or 'size' of proofs instead of the mere existence of proofs. For instance, evaluation of (a) boils down to judging the complexity of the generated proof task **(P1)**.

Let us, for example, use Gentzen's natural deduction (ND) calculus [12] 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) requires three proof steps: we need one 'Conjunction-Elimination' step to extract A from $A \wedge B$, one 'Modus Ponens' step to obtain C from A and $A \Rightarrow C$, and another 'Modus Ponens' step to obtain D from C and $C \Rightarrow D$. 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 a single argument, 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, the ND calculus together with naive proof step counting does not always provide a cognitively adequate basis for granularity analysis. The reason is that two intuitively very similar student proof steps (such as **(i)** from $A = B$ and $B = C$ infer $A = C$ and **(ii)** from $A \Leftrightarrow B$ and $B \Leftrightarrow C$ infer $A \Leftrightarrow C$) may actually expand into base-level ND proofs of completely different size. Related research has shown that the standard ND calculus does not adequately reflect human-reasoning in this respect [24]. Two important and cognitively interesting questions thus concern the appropriate choice of a proof system $\vdash$ and ways to measure the 'argumentative complexity' of an admissible proof step.

3.2 Relevance

Relevance is about usefulness and importance of a proof step with respect to the given proof task. For instance, in utterance (c) the proof goal $D \vee E$ is refined to a new goal D using backward reasoning, i.e., the previously open goal $D \vee E$ is closed and justified by a new goal.

Solving logical relevance problem requires in this case checking whether a proof can still be generated in the new proof situation. In this case, the task is thus identical to **(P1)**. An irrelevant backward proof step, 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 however irrelevant. This simple approach appears plausible, but needs to be refined. The challenge is to exclude detours and to take tutorial aspects into account (in a tutorial setting we are often interested in teaching particular styles of proofs, particular proof methods, etc.). This also applies to the more challenging forward reasoning case where for instance, utterance (b) should be identified as an irrelevant step.

4 Related Work

Input analysis in dialog systems is commonly done with shallow syntactic analysis combined with keyword spotting where short answers may be elicited by asking closed-questions [14]. Slot-filling templates, however, are not suitable representations of the content in our domain. Moreover, the interleaving of natural and symbolic language makes keyword spotting difficult because of the variety of possible verbalizations.

Statistical methods are often employed in tutorial systems to compare student responses with pre-constructed gold-standard answers [15]. In our context, such a static modeling solution is impossible because of the wide quantitative and qualitative range of acceptable proofs, i.e., generally, our set of gold-standard answers is infinite. In this respect our approach is closely related to the Why2-Atlas tutoring system [22]. This system presents students with qualitative physics

questions and encourages them to explain their answers with natural language. Different to our approach is that the students first present a complete essay of their answer to the system. The system then employs propositional logic representations and propositional abductive reasoning to analyze the answer with respect to a set of anticipated solutions. The analysis results are then used to create a dialog in which misconceptions in the students essay are addressed. In our scenario propositional logic appears insufficient and we employ first-order and higher-order representations and reasoning techniques. Similar to our scenario, however, is the problem of multiple proof alternatives that have to considered in the analysis tasks.

To analyze human-oriented mathematical proofs, shaped-up textbook proofs have been investigated in the deduction systems community (see [34, 27]). Claus Zinn [34], for instance, addresses complete, carefully structured textbook proofs, and relies on given text-structure, typesetting and additional information that identifies mathematical symbols, formulas, and proof steps. The DIALOG corpus provides an important alternative view, since textbook proofs neither reveal the dynamics of proof construction nor do they show the actual student's utterances, i.e., the student's proof step directives. Our corpus also illustrates 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. By contrast, many proof assistants employ a procedural style in which proof steps are invoked by calling rules, tactics, or methods, i.e., some proof refinement procedures.

Recent research into dialog modeling has delivered a variety of approaches more or less suitable for the tutorial dialog setting. For instance, scripting is employed in AutoTutor [23] and a knowledge-based approach similar to ours is implemented in the Geometry Tutor [1, 2]. Outside the tutorial domain, the framework of Information State Update (ISU) has been developed in the EU projects TRINDI (`http://www.ling.gu.se/research/projects/trindi/`) and SIRIDUS (`http://www.ling.gu.se/projekt/siridus/`) [29], and applied in various projects targeting flexible dialog. An ISU-based approach with several layers of planning is used in the tutorial dialog system BEETLE [35].

5 Conclusion

We presented our approach to interpreting informal mathematical discourse in the context of tutorial dialogue and to evaluating the proof steps proposed by the student by a back-end domain reasoning component. We employ a stratified approach to interpreting the mixed natural- and mathematical language; we first developed methods for basic cases, and enhanced them by techniques for handling additional levels of complexity. Our interpretation method has the potential of putting a tutorial system in a good position to apply strategies for enhancing higher-level problem-solving skills of the student.

We have identified two previously unconsidered aspects of proof-step evaluation: relevance and granularity of proof-steps, that are important from the

tutoring point of view. To address these criteria, it is not sufficient to merely establish the existence of proofs. The system has to construct proofs with particular properties. It may be the case that evaluating different criteria requires different theorem provers. Moreover, the system also needs to closely mirror and reflect reasoning steps as they are typically preferred by humans. Generally, the system will need to adapt to the capabilities of individual students and the requirements of varying tutorial settings.

We have implemented an input interpretation component capable of representing student utterances consisting of a mixture of natural language and mathematical expressions, in a format that is interpretable by an automated reasoning engine. The application of our approach to mathematical domains that are more challenging than naive set theory, and its evaluation therein is ongoing work. The hypothesis that assertion level reasoning [20] plays an essential role in evaluating appropriateness of underspecified partial proofs has been confirmed. The fact that assertion level reasoning may be highly underspecified in human-constructed proofs is a novel finding of our project (cf. [4]).

The implemented proof manager demonstrator helps to resolve underspecification and to evaluate proof steps based on heuristically guided abstract-level domain reasoning realized of the ΩMEGA-CORE framework [3]. The PM has been integrated also into the overall DIALOG system to communicate with the other components of the system.

The evaluation of the system so far concentrates mainly on individual analysis of specific aspects of single modules. One example presented in [25] is the evaluation of mechanized granularity judgments of proof steps, using deductive techniques based on natural deduction calculus and the PSYCOP approach [24].

References

1. V. Aleven, O. Popescu, and K. Koedinger. Towards tutorial dialog to support self-explanation: Adding natural language understanding to a cognitive tutor. In *Proceedings of the 10th International Conference on Artificial Intelligence and Education (AIED-01)*, pages 246–255, 2001.
2. V. Aleven, O. Popescu, and K. Koedinger. Pilot-testing a tutorial dialogue system that supports self-explanation. In *Proceedings of 6th International Conference on Intelligent Tutoring Systems (ITS-02)*, pages 344–354. Springer Verlag, 2002.
3. S. Autexier. *Hierarchical Contextual Reasoning.* PhD thesis, Saarland University, Germany, 2003.
4. S. Autexier, C. Benzmüller, A. Fiedler, H. Horacek, and B.Q. Vo. Assertion-level proof representation with under-specification. *Electronic Notes in Theoretical Computer Science*, 93:5–23, 2003.
5. C. Benzmüller, A. Fiedler, M. Gabsdil, H. Horacek, I. Kruijff-Korbayová, M. Pinkal, J. Siekmann, D. Tsovaltzi, B.Q. Vo, and M. Wolska. A Wizard-of-Oz experiment for tutorial dialogues in mathematics. In *AIED2003 — Supplementary Proceedings of the 11th International Conference on Artificial Intelligence in Education*, pages 471–481, Sydney, Australia, 2003.

6. C. Benzmüller, A. Fiedler, M. Gabsdil, H. Horacek, I. Kruijff-Korbayová, D. Tsovaltzi, B.Q. Vo, and M. Wolska. Language phenomena in tutorial dialogs on mathematical proofs. In *Proceedings of DiaBruck, the 7th Workshop on the Semantics and Pragmatics of Dialogue*, pages 165–166, Saarbrücken, Germany, 2003.
7. C.E. Benzmüller and Q.B. Vo. Mathematical domain reasoning tasks in natural language tutorial dialog on proofs. In M. Veloso and S. Kambhampati, editors, *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-05)*, pages 516–522, Pittsburgh, PA, 2005.
8. C. Benzmüller, H. Horacek, I. Kruijff-Korbayová, H. Lesourd, M. Schiller, and M. Wolska. DiaWozII – A Tool for Wizard-of-Oz Experiments in Mathematics. In *Proceedings of the 29th Annual German Conference on Artificial Intelligence (KI-06), Lecture Notes in Computer Science*, Bremen, Germany, 2006. Springer-Verlag. To Appear.
9. C. Benzmüller, H. Horacek, H. Lesourd, I. Kruijff-Korbayová, M. Schiller, and M. Wolska. A corpus of tutorial dialogs on theorem proving; the influence of the presentation of the study-material. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC-06)*, pages 1766–1769, Genoa, Italy, 2006. ELDA.
10. M. Buckley and C. Benzmüller. System description: A dialog manager supporting tutorial natural language dialogue on proofs. In *Proceedings of the ETAPS Satellite Workshop on User Interfaces for Theorem Provers*, Edinburgh, Scotland, 2005.
11. N. Dershowitz and C. Kirchner. Abstract saturation-based inference. In Phokion Kolaitis, editor, *Proceedings of the 18th Annual IEEE Symposium on Logic in Computer Science (LICS-03)*, Ottawa, Ontario, June 2003. IEEE.
12. G. Gentzen. Untersuchungen über das logische Schließen I & II. *Mathematische Zeitschrift*, 39:176–210, 572–595, 1935.
13. C. Gerstenberger and M. Wolska. Introducing Topological Field Information into CCG. In *Proceedings of the 17th European Summer School in Logic, Language and Information (ESSLLI-05) Student Session*, pages 62–74, Edinburgh, Scotland, 2005.
14. M. Glass. Processing language input in the CIRCSIM-tutor intelligent tutoring system. In *Proceedings of the 10th International Conference on Artificial Intelligence and Education in Education (AIED-01), San Antonio, Texas*, pages 210–221, 2001.
15. A. Graesser, P. Wiemer-Hastings, K. Wiemer-Hastings, D. Harter, and N. Person. Using Latent Semantic Analysis to evaluate the contributions of students in AutoTutor. *Interactive Learning Environments*, 8, 2000.
16. A. C. Graesser, N. K. Person, and J. P. Magliano. Collaborative dialogue patterns in naturalistic one-to-one tutoring. *Applied Cognitive Psychology*, 9:1–28, 1995.
17. H. Horacek, A. Fiedler, A. Franke, M. Moschner, M. Pollet, and V. Sorge. Representation of mathematical objects for inferencing and for presentation purposes. In *Proceedings of the 17th European Meetings on Cybernetics and Systems Research (EMCSR-04), Vienna, Austria*, pages 683–688, 2004.
18. H. Horacek and M. Wolska. Interpretation of mixed language input in a mathematics tutoring system. In *Proceedings of the AIED-05 Workshop on Mixed Language Explanations in Learning Environments*, pages 27–34, Amsterdam, the Netherlands, 2005.
19. H. Horacek and M. Wolska. Interpreting semi-formal utterances in dialogs about mathematical proofs. *Data and Knowledge Engineering Journal*, 58(1):90–106, 2006.

20. X. Huang. Reconstructing Proofs at the Assertion Level. In A. Bundy, editor, *Proceedings of the 12th Conference on Automated Deduction*, number 814 in LNAI, pages 738–752. Springer, 1994.
21. M. Kohlhase and A. Franke. MBase: Representing knowledge and context for the integration of mathematical software systems. *Journal of Symbolic Computation*, 32(4), 2001.
22. M. Makatchev, P. W. Jordan, and K. VanLehn. Abductive theorem proving for analyzing student explanations to guide feedback in intelligent tutoring systems. *Journal of Automated Reasoning*, 32(3):187–226, 2004.
23. N. K. Person, A. C. Graesser, D. Harter, E. Mathews, and the Tutoring Research Group. Dialog move generation and conversation management in AutoTutor. In Carolyn Penstein Rosé and Reva Freedman, editors, *Building Dialog Systems for Tutorial Applications—Papers from the AAAI Fall Symposium*, pages 45–51, North Falmouth, MA, 2000. AAAI press.
24. L. J. Rips. *The psychology of proof*. MIT Press, Cambridge, Mass., 1994.
25. M. Schiller. Mechanizing proof step evaluation for mathematics tutoring – the case of granularity. Master's thesis, Saarland University, 2005.
26. P. Sgall, E. Hajičová, and J. Panevová. *The meaning of the sentence in its semantic and pragmatic aspects*. Reidel Publishing Company, Dordrecht, The Netherlands, 1986.
27. J. Siekmann. Proof presentation. In *Proof Presentation*. Elsevier, To Appear.
28. J. Siekmann, C. Benzmüller, A. Fiedler, A. Meier, I. Normann, and M. Pollet. *Proof Development in OMEGA: The Irrationality of Square Root of 2*, pages 271–314. Kluwer Applied Logic series (28). Kluwer Academic Publishers, 2003.
29. D. R. Traum and S. Larsson. The information state approach to dialogue management. In Jan van Kuppevelt and Ronnie Smith, editors, *Current and New Directions in Discourse and Dialogue*. Kluwer, 2003.
30. M. Wolska and I. Kruijff-Korbayová. Analysis of mixed natural and symbolic language input in mathematical dialogs. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, Barcelona, Spain, 2004.
31. M. Wolska, I. Kruijff-Korbayová, and H. Horacek. Lexical-semantic interpretation of language input in mathematical dialogs. In *Proceedings of the ACL 2nd Workshop on Text Meaning and Interpretation*, pages 81–88, Barcelona, Spain, 2004.
32. M. Wolska, B.Q. Vo, D. Tsovaltzi, I. Kruijff-Korbayová, E. Karagjosova, H. Horacek, M. Gabsdil, A. Fiedler, and C. Benzmüller. An annotated corpus of tutorial dialogs on mathematical theorem proving. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC-04)*, 2004.
33. M. Wolska and I. Kruijff-Korbayová. Factors influencing input styles in tutoring systems: the case of the study-material presentation format. In *Proceedings of the ECAI-06 Workshop on Language-enabled Educational Technology*, To Appear, Riva del Garda, Italy, 2006.
34. C. Zinn. *Understanding Informal Mathematical Discourse*. PhD thesis, University of Erlangen-Nuremberg, 2004.
35. C. Zinn, J.D. Moore, M.G. Core, S. Varges, and K. Porayska-Pomsta. The BE&E tutorial learning environment (BEETLE). In *Proceedings of DiaBruck, the 7th Workshop on the Semantics and Pragmatics of Dialogue*, pages 209-210, Saarbrücken, Germany, 2003.