Tutorial Dialogs on Mathematical Proofs[*]

Christoph Benzmüller[1], Armin Fiedler[1], Malte Gabsdil[2], Helmut Horacek[1], Ivana Kruijff-Korbayová[2], Manfred Pinkal[2], Jörg Siekmann[1], Dimitra Tsovaltzi[2], Bao Quoc Vo[1], Magdalena Wolska[2]

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

Abstract

The representation of knowledge for a mathematical proof assistant is generally used exclusively for the purpose of proving theorems. Aiming at a broader scope, we examine the use of mathematical knowledge in a mathematical tutoring system with flexible natural language dialog. Based on an analysis of a corpus of dialogs we collected with a simulated tutoring system for teaching proofs in naive set theory, we identify several interesting problems which lead to requirements for mathematical knowledge representation. This includes resolving reference between natural language expressions and mathematical formulas, determining the semantic role of mathematical formulas in context, and determining the contribution of inference steps specified by the user.

1 Introduction

In a mathematical proof assistant (MPA), knowledge representation (if any) is used for the purpose of proving theorems. State-of-the-art MPAs such as Coq, Nuprl, Mizar, Isabelle-Hol, Pvs and ΩMEGA usually provide a combination of proof automation and facilities for user interaction and most of them are connected to a structured mathematical knowledge base. In spite of their common purpose (proving theorems), the heterogeneity of MPAs (they are based on different logics, calculi, semantics, representations of proofs, etc.) poses a challenge for the communication of mathematical knowledge between them, and most importantly, a common ontology and semantics are missing. Some of these issues are currently investigated in the Mathematical Knowledge Management research initiative [4]. However, appropriate knowledge representation in MPAs to support the search for a proof is only *one* of the issues to be addressed in the future of computer-aided mathematics, and in computer-aided mathematical education in particular.

Among the challenges involved in human-oriented automated proving is the coupling of MPAs with natural language processing. This in turn gives rise to additional requirements on knowledge representation. For example, it has been shown in [9] that the mathematical domain representation as used for proof search and proof planning is not sufficient for the purpose of proof presentation. Some methods for more natural references to rules have been demonstrated in [16]. In this paper, we present further requirements on mathematical knowledge representation for the purpose of handling flexible natural language dialog in a mathematical tutoring system. Our discussion is based on data we collected through experiments with a simulated tutoring dialog system for teaching proofs in naive set theory.

Some state-of-the-art tutorial systems allow limited dialog, where the input is either menu-based or requires exact wording [24; 2; 13]. This contrasts with Moore's empirical findings showing that flexible natural language dialog is needed to support active learning [23]. The latter approach is taken for example in the CircSim-Tutor project [22] which aims to build a natural language-based tutoring system for first-year medical students to learn about the reflex control of blood pressure.

The goal of our project is to develop a mathematical tutoring system with flexible natural language dialog to support mathematical problem solving. We employ a modular approach keeping a strict separation between the different kinds of knowledge involved in the processing. The design of the system components is informed by the analysis of a corpus of tutorial dialog data we collected in an experiment.

The outline of this paper is as follows. We first

[*]This work is supported by the SFB 378 at Saarland University, Saarbrücken, and the EU training network CALCULEMUS (HPRN-CT-2000-00102) funded in the EU 5th framework.

present the aims of our project, illustrate the current application scenario and motivate the choice of the mathematical domain. The modeling of static and dynamic knowledge within this domain is our first contribution. Next, we describe an experiment in which we collected a corpus of natural language tutorial dialogs in the chosen mathematical domain. On the basis of the analysis of our corpus we then present the key requirements and challenges for the representation of mathematical knowledge and the design of a mathematical reasoning tool.

2 The DIALOG Project

The goal of the DIALOG project[1] [25] is (i) to empirically investigate the use of flexible natural language dialog in tutoring mathematics, and (ii) to develop an experimental prototype system gradually embodying the empirical findings. The experimental system will engage in a dialog in written natural language (and later also in multimodal forms of communication based on diagrams, spoken language and animated mathematical function displays) to help a student understand and construct mathematical proofs. The overall scenario for the system is illustrated in Figure 1. We describe its components below.

Learning Environment In our scenario, the student takes an interactive course in some field of mathematics within a web-based learning environment. We use ACTIVEMATH [19; 21], a generic web-based learning system that dynamically generates interactive (mathematical) courses adapted to the student's goals, preferences, capabilities, and knowledge. It enables a student to select the material he/she wants to study and to review his/her knowledge about the subject matter. After finishing a learning unit the student may opt for an interactive exercise session to actively apply what he/she has learned. It is primarily the interactive exercises that we aim to enrich with the possibility of flexible tutoring dialog using natural language. The features of ACTIVEMATH include: user modeling and monitoring facilities; user-adapted content selection, sequencing, and presentation; support of active and exploratory learning by external tools; use of (mathematical) problem solving methods, and re-usability of the encoded content as well as interoperability between systems. ACTIVEMATH maintains a dynamically updated *student model (SM)* containing information about the axioms, definitions, theorems (hence the assertions) and the proof techniques the student has studied and mastered so far.[2] This information will be used also by the tutoring dialog system. In addition, we also assume an *idealized student model (ISM)* set up by the author of the learning unit, which specifies the mathematical material a student ideally should know after studying the unit.

Mathematical Proof Assistant The MPA is used for the problem-solving in the mathematical domain underlying the dialogs. This involves the verification (or falsification) of user specified inference steps and checking whether the application of an inference step leads to a proof state from which a complete proof can be obtained. Mathematical tutorial dialogs thus require (i) stepwise interactive as well as (ii) automated proof construction at a human-oriented level of abstraction. Ideally, these are provided by the MPA.

In addition, it should be possible to control the proof strategy used by the MPA (depending on the target of the tutorial session), and the proof(s) constructed by the MPA should only exploit the mathematical knowledge that the student possesses, that is, it should be possible to control the mathematical knowledge used in the proof(s) in accordance to the respective SM and ISM.

The ΩMEGA system [29] with its advanced proof presentation and proof planning facilities provides an adequate starting point for integrating an MPA in our scenario.

Proof Manager In the course of the interactive tutorial session, the user may explore alternative proofs, or make various attempts at constructing a valid proof, involving both valid and invalid inference steps. In addition, tutoring may require the possibility to compare the problem-solving attempts made by the user with target "master" proofs. The student's problem-solving attempts with respect to the proof space need to be monitored for the sake of managing the dialog flow. It is the task of the proof manager in our scenario to provide this interface and additional book-keeping between the MPA and the dialog manager.

Dialog Manager When the student enters a tutorial dialog session, the interaction is handled by the dialog manager. We employ the Information-State (IS) Update approach to dialog management developed in the TRINDI and SIRIDUS projects [28; 27]. The IS is a centrally maintained data structure which contains a representation of the information accumulated as the dialog progresses, including (i)

[1]The DIALOG project is part of the Collaborative Research Center on *Resource-Adaptive Cognitive Processes* (SFB 378) at University of the Saarland [26].

[2]ACTIVEMATH keeps track of what material the student has studied and for how long [20]. It also lets the student skip material he is confident to know well already.

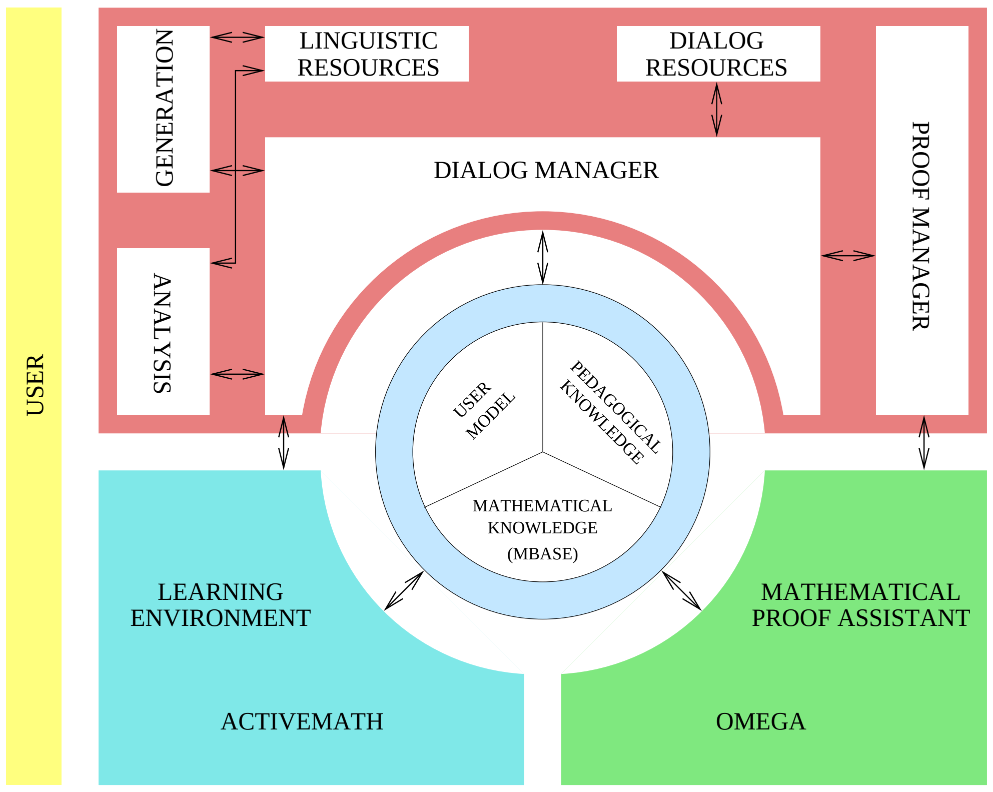


Figure 1: DIALOG project scenario.

"private" information of the system, and (ii) the information considered to be "shared" between the system and the user. A dialog is modeled as a sequence of dialog moves each of which is a transition from one information state to the next one. The system interprets each user's utterance with respect to the current IS, and then computes a transition to a new IS. When it is the system's turn, the next move is selected according to the IS at that point, the corresponding utterance is produced, and again the IS is updated. The dialog manager relies on the input analysis and output generation modules to exchange data between the user and the system; it further relies on the proof-manager to monitor the mathematical problem-solving and to access the MPA.

Knowledge Resources The static knowledge in our scenario comprises *linguistic resources*, *dialog resources*, *pedagogical knowledge*, and *mathematical knowledge*. The dynamic knowledge includes the SM and ISM mentioned above, as well as the information state maintained by the dialog manager.

The linguistic resources include the grammar and the lexicon used for analyzing the natural language input and generating the output. We combine the use of generic, domain independent resources with resources specific to the particular area of mathematics being taught.

The static dialog resources include (i) dialog move selection rules (i.e., rules that determine what dialog move the system will make next, given the current information state and a communicative goal), and (ii) dialog information-state update rules (i.e., rules that dynamically change the information state depending on the dialog moves the user or the system have successfully made). We distinguish between domain independent, generic dialog moves, such as meta-communication moves (used for, e.g., clarification and correction), and domain-specific ones, such as various kinds of hinting moves [11; 31], which may be further specialized for tutoring in the matematics domain.

The pedagogical knowledge specifies generic and domain-specific teaching strategies. This includes the specification of the didactic versus socratic teaching methods. Also the hinting dialog moves mentioned above are derived from the pedagogical knowledge.

Finally, the static mathematical knowledge consists of assertions (i.e., axioms, lemmata, theorems), domain dependent proof rules and methods, corresponding diagrammatic illustrations as well as selected completed master proofs. This mathematical knowledge is typically highly structured into mathematical sub-domains and it usually forms a dependency/inheritance graph. Examples of systems maintaining structured corpora of formalized mathematics are MIZAR with its mathematical library [5], NuPrl's knowledge base [3] and the MBase system [18], which is the system of choice in our project. An essential requirement in our scenario is that the mathematical knowledge is shared between the learning environment, the DIALOG system, and the mathematical assistant. One problem in many current proof systems is to guarantee consistent handling and data flow between the declarative and the procedural view of assertions. In [32], we suggest a solution that uses declarative entries in the mathematical knowledge base to automatically generate all potential procedural views from these declarative entries for each given proof context.

We already mentioned that there may be a limited number of fixed master proofs for the proof exercises to be employed in guiding the tutorial session. These can be statically maintained in the mathematical knowledge base. Generally, however, there are infinitely many variants of proofs for a mathematical theorem and a significant number of these proofs is acceptable for being tutored relative to the knowledge and capabilities of the student. We therefore couple the static modeling of a well chosen set of master proofs with the dynamical verification of single inference steps and the dynamic generation of proofs by the MPA.

The SM (and the ISM) refer to the mathematical knowledge base in the sense that they maintain, for each student, a view on this knowledge base, separating the known from the unknown content. An additional *teacher model* could provide information such as a specification of the dominant and the subdominant mathematical concepts of a learning unit. Note that the structure imposed by the latter information is likely to differ from the hierarchical structure of the knowledge base itself [30].

Our Current Domain of Choice: Naive Set Theory For the first phase of the project we chose naive set theory as the mathematical domain of interest. We integrated a course on naive set theory into ACTIVEMATH. Basic notions (e.g., *set*) and definitions (e.g., *subset*), or set operations, (e.g., *union*, *intersection*, *set complement*, *power set*) are structurally represented in this course. Typical examples are presented after each definition, for the student to get a good intuition about the more abstract concepts. Students are also exposed to Venn diagrams which provide an intuitive understanding of set operations. Throughout the course, the student is continuously introduced to the more important properties of this domain, for example, laws of commutativity, associativity, distributivity, or de Morgan laws.

The Naive Set Theory domain has several advantages: (i) The problems in this domain are almost always automatically provable [6; 7]. (ii) The domain is not too complex for the intended users (i.e., first year students). (iii) Simple problems are typically even decidable, so that wrong proof steps can be detected by the generation of counterexamples with a model generator [6]. (iv) The domain provides interesting opportunities for multi-modal interaction using the Venn and Spider diagrams.[3] (Sound and complete inference systems exist for the representation layer of Spider diagrams; cf. [14] and the references therein.)

The disadvantages of this domain are: (i) Its modeling is built directly on predicate logic without higher-level concepts and fields of mathematics on many intermediate layers between the base logic and the domain itself. Hence, there are no hierarchical dependencies on other mathematical subdomains, such as real numbers, continuous functions, Abelian groups, etc. (ii) Consequently, the hierarchical expansion depth of proof plans and proofs is also relatively low. Although this raised some initial doubts about the suitability the naive set theory domain, the experiment described in the next section revealed that even such a relatively simple mathematical domain has sufficient complexity to allow meaningful tutorial dialog sessions. We shall, however, also consider more complex mathematical domains in future experiments.

3 Empirical Study

We conducted a *Wizard-of-Oz (WOz)* experiment in order to collect a corpus of tutorial dialogs in the naive set theory domain. We implemented a tool to support the experiment and collect the dialog data on-line [10].

In a WOz experiment, the subject interacts through an interface with a human "wizard" simulating the behavior of a system [8]. The WOz methodology is commonly used to investigate human-computer interaction in systems under development. One of the reasons for using a WOz setting rather than a human tutor is that it has been observed that humans interact differently with computers than with other humans. Another reason is that the tutor should follow the specific algorithm(s), which we are implementing in our system. In this way the dialog data we collect (i) represents the users' behavior in interactions following these algorithms and (ii) provides early feedback on the algorithms. In subsequent experiments

[3]These aspects are, however, not subject of this paper and will be considered in later experiments.

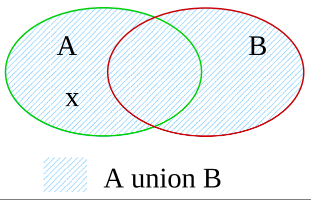
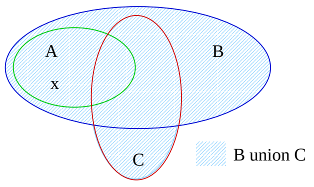
Declarative View	Procedural View	Diagrammatic View
$\forall e, A, B. e \in A \Rightarrow (e \in A \cup B)$	$\dfrac{e \in A}{e \in A \cup B}$ ∈-∪-IL	
$\forall A, B, C. A \subseteq B \Rightarrow (A \subseteq B \cup C)$	$\dfrac{A \subseteq B}{A \subseteq B \cup C}$ ⊆-∪-IL	
$\forall A, B. (A \subseteq B) \Rightarrow (A \in \wp(B))$	$\dfrac{A \subseteq B}{A \in \wp(B)}$ ℘-I	?

Figure 2: Declarative, procedural, and diagrammatic knowledge in the domain of naive set theory.

in the project, implemented components can substitute for some of the tasks now carried out by the wizard, while preserving the overall experimental setup.

We invited 24 subjects to participate in the experiment. They were students with educational background in humanities (e.g., law, economy, various languages, psychology) or sciences (e.g., biology, chemistry, computer science, computational linguistics). Their prior mathematical knowledge ranged from little to fair.

For each subject, the experiment consisted of the following phases (each of which had a fixed maximum duration): (1) *Preparation and pre-test:* First, the subject filled in a background questionnaire. Then he/she studied written lesson material, explaining basic concepts and providing a collection of six lemmata about properties of sets and eleven lemmata about properties of powersets.[4]. Finally he/she was asked to prove (on paper) the theorem $K(A) \in \mathcal{P}(K(A \cap B))$. (2) *Tutoring session:* The subject was asked to evaluate a tutoring system with natural language dialog capabilities. He/She was given three theorems to prove: The theorem K((A ∪ B) ∩ (C ∪ D)) = (K(A) ∩ K(B)) ∪ (K(C) ∩ K(D)) was used first to let the subject familiarize himself/herself with the system's interface. Then two more complex theorems were presented (in different order to different subjects): (a) A ∩ B ∈ P((A ∪ C) ∩ (B ∪ C)) (b) Wenn A ⊆ K(B) , dann B ⊆ K(A). The interface enabled the subject to type text or insert mathematical symbols by clicking on buttons; it also displayed the complete dialog with both the tutor's and the subject's utterances. The subject was instructed to enter partial steps of a proof rather than the complete proof as a whole, in order to enable a dialog with the system. (3) *Post-test and evaluation questionnaire:* The subject was asked to write down (on paper) a proof for one more theorem.[5] To conclude the experiment, he/she was asked to fill in a questionnaire addressing various aspects of the system and its usability.

The tutor-wizard's task was to respond to the student's utterances following a given algorithm. The wizard first classified the completeness, accuracy, and relevance of the subject's utterance with respect to a valid proof of the theorem at hand. Then, the wizard decided what dialog moves to make next and verbalized them. Depending on the tutoring strategy employed by the wizard for a given subject, the dialog move options included informing the subject about completeness, accuracy, and relevance of the utterance, giving hints on how to proceed further, explaining a step under consideration, prompting for the next step, or entering into a clarification dialog. The wizard was free to mix text with formulas [11].

4 A Preliminary Analysis of the Test Dialogs

In this section, we examine the issues involved in the natural language analysis of the dialog utterances containing mathematical expressions and the role of mathematical domain knowledge. Examples of dialog utterances that illustrate the phenom-

[4]In the first experiment the lesson material was still presented on paper, not through the ACTIVEMATH system.

[5]The comparison of the student's performance of the pre-test and post- test proofs serve to evaluate their learning gain from the tutoring session.

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

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

ena addressed by the analysis below are shown in Figure 3 (the original German versions of utterances are presented together with their English translation).

We have identified the following three major aspects of the domain-specific knowledge analysis: (1) Identification and resolution of natural language references to terms and concepts of the domain; (2) Semantic role of formal expressions (mostly formulas) in the current dialogue; (3) Evaluation of the inference step(s) proposed by the student with respect to the considered master proof(s).

Unlike the third issue, which is specific to our domain of application: mathematical proofs, the first two issues are more generally applicable to analysis of natural language in the context of mathematics. We consider these issues in turn.

4.1 References in Natural Language Expressions

Interacting with the system students should be able to input both natural language text and formal mathematical expressions. In order to build appropriate semantics, it is essential to establish references between the formal domain concepts in the mathematical data base, the symbols in the formulas, and the corresponding natural language expressions. In particular, natural language descriptions may refer to several kinds of domain concepts (in the following, numbers refer to the example utterances in Figure 3):

- *Domain objects* have known denominations, which are frequently referred to within text chunks, e.g., "power set" as a natural language expression has the same denotation as the predicate P in a formal expression about sets. An expression may be ambiguous between *generic reference* to a domain object as a type vs. *specific reference* to a particular instance (or: token) of that type. Generic and specific references may even appear within the same utterance (cf. (1), where "Potenzmenge" (powerset) is used as a generic reference, whereas $A \cap B$ is a specific reference to a subset of a specific instance of the power set).

  Sometimes, the scope of reference is determined jointly by a natural language expression and a mathematical formula. For example, in (2), the right-hand side of the equation clarifies the scope for anchoring the referring expression "both complements".

- *Domain relations* comprise propositional logic junctors, logical derivation, and justifications. The interpretation of descriptions in which these relations appear is problematic because of, inter alia, ambiguities concerning scope. For instance, "A and B implies C" has two structurally different interpretations; (7) is an aggregated (or: abbreviated) description of two subset relations where the subsets share a common superset. Similarly, in (6), the comma is meant to be interpreted as logical "and" followed by the expression that constitutes the logical consequence. The natural language connective "and" itself poses ambiguities in that it may mean a "logical and" as well as a consequent of a "logical derivation".[6]

  Another issue concerns reference to mathematical relations which may be imprecise in the sense that the natural language formulation fits several relations (e.g., within the domain of mathematical sets, "must be in" in (4) can be interpreted as "element" or "subset"; and "both sets *together*" in (5) as union or intersection).

- *Variables* appear, ideally, in an identical form within the text and the formulas. However, ambiguities arise here as well, since an identifier in the text may refer to variables in different formulas (e.g., A may refer to a variable used in an axiom or to a variable in an expression instantiating that axiom, as in (3)). From an interpretation point of view, this means that an identifier of a variable should be treated as a referring expression, similar to a nominal group. In cases where there are two occurrences of the same identifier (form), whether they co-refer or not should be resolved (e.g., in (3), the two occurrences of A in "$K(A) = \textit{the term } K(A \cup B)$" do not co-refer).

- *Domain rules (axioms)* are associated with known denominations, similarly to domain objects as discussed above. Problems of ambiguity arise here due to duality (e.g., distributivity) or due to imprecision in formulation (e.g., "de Morgan rules" in (5)).

- *Descriptions of domain operations*, such as application of an inference rule, are often described informally (e.g., "to split" an expression as in (9)). A challenge for the natural language analysis lies in the large number of unexpected synonyms, especially uncommon in mathematical usage, where some of them have a metaphoric flavor.

As expected, the most prominent problems handling references to domain concepts are the many

[6] Students should be advised to avoid such ambiguities in tutoring contexts as far as possible. If they nevertheless occur and are in the focus of the tutoring session, the system may ask for clarification. Providing automated support for disambiguation is an issue in particular for those ambiguities outside the tutoring focus.

forms of ambiguities. We need to investigate how the proof state contributes to resolving these ambiguities. In unclear cases, a clarification sub-dialog can be initiated by the system to prompt the student to formulate more precise descriptions.

4.2 The Role of Formal Expressions

The mathematical formulas within an utterance may vary according to their contextual embedding. In most cases, however, the intended meaning is explicitly indicated in the text. In general, there are at least the following possibilities (again, numbers refer to the example utterances in Figure 3):

- *References* were typically introduced as variables to refer to subexpressions, in order to ease multiple references in the text. (e.g., in (3), $K(A)$ is used as a variable in the de Morgan rule, and is subsequently substituted by $K(A \cup B)$ when the rule is applied. The opposite case is also possible, i.e. a simpler expression can be introduced as a variable to substitute a complex expression).
- *Assumptions* occured as explicit statements that the truth of a logical formula was hypothetically assumed.
- *Assertions* , in the form of mathematical expressions, may appear without a textual embedding. These expressions may be theorems or simple corollaries of theorems, or they may be justified by the context of the current state of the proof (provided the assertions are correct). In both cases, their relevance and truth depends on the current state of the proof.
- *Logical consequences* are similar to assertions in that they (implicitly) constitute derivations and are usually expressed by natural language phrases such as“follows from”, “implies”, or simply by “=”.
- *Descriptions of goal* occurred in natural language as an abstract form of a formal expression. They typically served to indicate the purpose of building a new expression from expressions introduced earlier; see (8).

In all cases, ambiguities may arise when parentheses are required in a mathematical formula, but are omitted or misplaced by the student. Potentially, interpretation may be impossible as a result. A clarification sub-dialog would be initiated in such cases to address the syntactically invalid formula.

4.3 Inference Steps

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

Correctness of expressions: The first task in checking a proposed inference step is to verify its correctness. Students may make mistakes or produce expressions that are otherwise confused and thus need to be classified as partially incorrect or wrong. A particular important issue in dealing with mistakes is the identification of near misses (as in the legendary “bridge” cases of Patrick Winston [33]). They are treated differently in the tutorial context than severe errors that show a misconception or a complete lack of understanding. We interpret an inference as a “near miss” if the expression inferred differs from the correct one in a single element only. The incorrect element may be a variable or a constant; for example, a typing mistake. It may also be an operator, but the correct operator and the incorrect one must be conceptually related. Examples of commonly confused conceptually related operations in our specific domain include the use of “$=$” instead of “$\subseteq$” (too specific relation) or “$\in$” instead of “$\subset$” (sort incompatibility on similar relations).

Relevance of expressions: For a correct inference step, its potential contribution to the proof at hand needs to be determined. In order for an inference to be considered relevant, that is, beneficial for making progress toward solving the given problem, it needs to be an inference step in the proof to be taught. Depending on the previous proof steps, the student may have committed himself to one or another proof out of the set of potential solution paths known to the system. In case an inference step is incompatible with the prior followed proof path, the intention of the student to revise his proof strategy is checked (see utterance (9) in Figure 3).

The *granularity* of the inference steps suggested by the students has been at the assertion level (cases where the partial assertion level [15] becomes relevant never occurred in our experiment). However, certain “easy” assertion level steps were often combined with other inference rules (e.g., a step involving a relatively easy concept of commutativity was often performed together with some other operation.) On the other hand, students also produced more compound inferences which would normally require a few assertion level steps as in (6) in Figure 3). In such a cases, the student may be asked to explain his reasoning in more detail in order to confirm his understanding.

5 Consequences for logical systems

To adequately support tutorial dialog, the mathematical assistant must be based on a flexible logical system. The non-standard requirements for such system comprise: (1) dedicated static knowledge sources, (2) dynamic handling of domain-specific information, and (3) non-standard proof handling capabilities. We consider them in turn.

Static knowledge sources

- *Representation of domain entities* Knowledge about domain entities needs to be organized in such a way that not only proof-relevant details, but also domain-relevant commonalities are captured. This can be achieved by adding additional hierarchical and association links in the representation of mathematical theories. Hierarchical relations are needed to capture generalizations, such as "de Morgan rule" which specializes into the well-known two variants. Associations can be used to capture conceptual similarities among relations, such as "$\in$" and "$\subseteq$", which students tend to confuse, as the experiments have demonstrated.
- *Representation of master proofs* There are typically several ways of proving a theorem—even for the simple tasks investigated in our experiments. Conceptually distinct variants of the master proofs relevant for the tutorial goal need to be determined a priori.[7] Moreover, the master proofs need to be represented in a way that allows minor variations, for example, a different order of steps and symmetrical variants within a single proof representation.

Dynamic handling of information

- *Proof status* In the course of a tutorial session, evidence needs to be maintained about the relevance of the set of master proofs in view of the proof strategy adopted by the student. This means focusing on a reduced set when the student has committed himself to a strategy, as well as keeping track of which steps the student has already accomplished and which ones still need to be addressed.
- *Discourse memory* A major task for tutorial dialogs in natural language is maintaining evidence about instances of domain entities which are introduced and later referred to in the course of a tutorial session. Within the linguistic modules, this information is captured in a discourse memory (as part of the information state maintained by the dialog manager). There exist various models for the management of entities in the discourse memory (using a simple ordered list, a stack, or a cache model). The active part of the discourse memory is accessed in the process of reference resolution. For the tutorial dialogs, specificities of mathematical formulas need to be incorporated to complement linguistically motivated criteria for introducing entities into the discourse memory. In particular, the information conveyed in the form of mathematical formulas is cognitively present, however, not linguistically expressed. As previously pointed out, accurate identification of composite subexpressions provides the scope for interpreting the referring expression "both complements" in utterance (2).

[7]By conceptually distinct, we mean application of different assertions, that is, assertions which are not just symmetrical variants of one another.

Proof handling Another central task for a tutorial session is the interpretation of student actions in view of the tutorial goal. In our domain, this means the interpretation of suggestions for inference steps in view of the focused set of master proofs in the given state. Since the inference step suggested may be ambiguous and inaccurate, the emphasis is on a preference to resolve ambiguities. On the other hand, as part of ensuring that the student is aware of the correct proof steps, the tutor must strike a balance between performing extensive reasoning based on assumed student's knowledge, versus initiating a clarification subdialog with the student [34].

6 Conclusions for Future Research

We presented the overall framework of the DIALOG project which aims at the development of a mathematical tutoring system with flexible dialog. We concentrated in particular on the requirements such application poses on the representation of different kinds of knowledge. We reported on a WOz experiment in which we collected a corpus of tutorial dialogs with 24 subjects on several problems in naive set theory. As part of the experiment preparations, we developed an initial algorithm for selecting the system's dialog move based on an evaluation of the student's turn and the adopted tutoring strategy [11; 31]. The collected corpus let us evaluati these preliminary specifications and further develop them. The analysis of the dialogs also revealed that even a relatively simple mathematical domain has sufficient complexity for meaningful tutorial dialog sessions. In future experiments, we will extend this methodology to more complex mathematical domains which can be expected to lead to more complex dialogs and linguistic phenomena.

The corpus also enabled us to establish the relevant and significant features of tutorial dialogs in the mathematical domain and to investigate the relationships between different knowledge sources in our scenario. In particular, we concentrated on the use of the mathematical domain knowledge in dialog interpretation and management, and in the underlying mathematical proof assistant system. We identified cases of the use of natural language which pose interesting challenges for the processing of domain-specific knowledge and for the interface between the dialog manager and the MPA. These include the natural language references to logical concepts, the embedding of formal expressions within natural language ones, and the way inference steps are described in natural language, all of these are now being implemented in our representation language.

In the next stages of the project, we will continue to investigate the interaction between various components in our system architecture and the demands on user modeling that emerge in this setting and gradually implement the still missing system components.

References

[1] *Papers from the 2000 AVAIL Fall Symposium on Building Dialogue Systems for Tutorial Applications*. AAAI Press, 2000.

[2] V. Aleven and K. Koedinger. The need for tutorial dialog to support self-explanation. In AAAI [1], pages 14–19.

[3] S. Allen, R. Constable, R. Eaton, C. Kreitz and L. Lorigo. The Nuprl Open Logical Environment. In Proc. of CADE-17, LNAI 1831, Springer, 2000.

[4] A. Asperti, B. Buchberger, and J. H. Davenport, editors. *Mathematical Knowledge Management, Second International Conference, MKM 2003*, Bertinoro, Italy, February 16-18 2003. LNCS 2594, Springer.

[5] G. Bancerek and P. Rudnicki. A Compendium of Continuous Lattices in MIZAR: Formalizing recent mathematics. *Journal of Automated Reasoning*, 29(**3**):189–224, 2002. See also MIZAR homepage: `www.mizar.org`

[6] C. Benzmüller, M. Jamnik, M. Kerber, and V. Sorge. Experiments with an agent-oriented reasoning system. In Proc. of *KI 2001*, LNAI 2174, Springer, 2001.

[7] C. Benzmüller and M. Kohlhase. Leo – a higher-order theorem prover. In Proceedings of CADE-15, LNAI 1421, Springer, 1998.

[8] N. O. Bernsen, H. Dybkjær, and L. Dybkjær. *Designing Interactive Speech Systems — From First Ideas to User Testing*. Springer, 1998.

[9] A. Fiedler, A. Franke, H. Horacek, M. Moschner, M. Pollet, and V. Sorge. Ontological Issues in the Representation and Presentation of Mathematical Concepts. Workshop on Ontologies and Semantic Interoperability at ECAI-2002, 2002.

[10] A. Fiedler and M. Gabsdil. Supporting Progressive Refinement of Wizard-of-Oz Experiments. In Proceedings of the Sixth International Conference on Intelligent Tutoring Systems—Workshop W6: Empirical Methods for Tutorial Dialogue Systems, 2002.

[11] A. Fiedler and D. Tsovaltzi. Automating Hinting in Mathematical Tutorial Dialogue. In press: Proceedings of the EACL-03 Workshop on Dialogue Systems: interaction, adaptation and styles of management, Budapest, 2003.

[12] B. Grosz and C. Sidner. Attention, intention, and the structure of discourse. *Computational Linguistics*, 12:175–206, 1986.

[13] N. Heffernan and K. Koedinger. Intelligent tutoring systems are missing the tutor: Building a more strategic dialog-based tutor. In AAAI [1], pages 14–19.

[14] J. Howse, F. Molina, J. Taylor, S. Kent, and J. Gil. Spider diagrams: A diagrammatic reasoning system. *Journal of Visual Languages and Computing*, 12(3):299–324, 2001.

[15] H. Horacek Presenting Proofs in a Human-Oriented Way. In Proc. OF CADE-16, LNAI 1632, Springer, 1999.

[16] H. Horacek Expressing References to Rules in Proof Presentations. *Proc. IJCAR-2001 - Short Papers*, pages 76–85, 2001.

[17] X. Huang. Reconstructing proofs at the assertion level. In *Proc. of CADE-12*, LNAI, Springer, 1994.

[18] M. Kohlhase and A. Franke. Mbase: Representing knowledge and context for the integration of mathematical software systems. *Journal of Symbolic Computation*, 32(4):365–402, 2000.

[19] E. Melis et al. ACTIVEMATH: A generic and adaptive web-based learning environment. *Artifical Intelligence in Education*, 12(4), 2001.

[20] E. Melis and C. Ullrich. The Poor Man's Eyetracker in ActiveMath. International World Conference on E-Learning in Corporate, Gevernment, Healthcare, and Higher Education 2002.

[21] E. Melis, E. Andres, A. Franke, G. Goguadse, M. Kohlhase, P. Libbrecht, M. Pollet, and C. Ullrich. ACTIVEMATH system description. In *Artificial Intelligence and Education*, 2001.

[22] CIRCSIM-Tutor Project: `http://www.csam.iit.edu/~circsim/` (Illinois Institute of Technology).

[23] J. Moore. What makes human explanations effective? In *In Proc. of the Fifteenth Annual Conference of the Cognitive Science Society*, Hillsdale, NJ. Earlbaum, 2000.

[24] N. Person, A. Graesser, D. Harter, and E. Mathews. Dialog move generation and conversation management in AutoTutor. In AAAI [1], pages 45–51.

[25] M. Pinkal, J. Siekmann, and C. Benzmüller. Projektantrag Teilprojekt MI3 — DIALOG: Tutorieller Dialog mit einem mathematischen Assistenzsystem. In *Fortsetzungsantrag SFB 378 — Ressourcenadaptive kognitve Prozesse*, Saarbrücken, Germany, 2001. Universität des Saarlandes.

[26] SFB 378 web-site: `http://http://www.ling.gu.se/projekt/siridus/`.

[27] TRINDI project: `http://www.ling.gu.se/research/projects/trindi/`.

[28] TRINDI project: `http://www.ling.gu.se/research/projects/trindi/`

[29] J. Siekmann et al. Proof development with ΩMEGA. In *Proceedings of the 18th Conference on Automated Deduction*, LNAI 2392, Copenhagen, Denmark, 2002. Springer Verlag.

[30] D. Tsovaltzi and A. Fiedler. Construction and Use of an Mathematical Ontology in a Tutorial Dialogue System. IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems, Acapulco, Mexico, 2003.

[31] D. Tsovaltzi and C. Matheson. Formalising Hinting in Tutorial Dialogues. EDILOG: 6th workshop on the semantics and pragmatics of dialogue, Edinburgh, Scotland, 2002.

[32] Q. B. Vo, C. Benzmüller, and S. Autexier. An approach to assertion application via generalized resolution. SEKI Report SR-03-01, Fachrichtung Informatik, Universität des Saarlandes, Saarbrücken, Germany, 2003.

[33] Winston. P. H. Learning structured descriptions from examples. Technical Report MAC-TR-76, 1970.

[34] D. Tsovaltzi, A. Fiedler. An Approach to Facilitating Reflection in a Mathematics Tutoring System. AIED Workshop on Learner Modelling for Reflection, Sydney, 2003.