

# Enhancement and Use of a Mathematical Ontology in a Tutorial Dialog System

**Dimitra Tsovaltzi**

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

**Armin Fiedler**

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

## Abstract

Despite empirical evidence that natural language dialog capabilities are necessary for the success of tutorial sessions, only few state-of-the-art tutoring systems use natural-language style interaction. Since domain knowledge, tutoring and pedagogical knowledge, and dialog management are tightly intertwined, the modeling and integration of proper natural language dialog capabilities in a tutoring system turns out to be barely manageable.

In the DIALOG project, we aim at a mathematical tutoring dialog system that employs an elaborate natural language dialog component. To tutor naive set theory, we use a formally encoded mathematical theory including definitions and theorems along with their proofs. In this paper we present how we enhance this ontology by making relations explicit and we show how these relations can be used by the socratic tutoring strategy, which we employ, in planning the next system utterance. The decisive characteristic of the socratic strategy is the use of hints in order to achieve self-explanation.

## 1 Introduction

Despite empirical evidence that natural language dialog capabilities are necessary for the success of tutorial sessions [15], only few state-of-the-art tutoring systems use natural-language style interaction that requires menu-based input or exact wording of the input [16; 1; 10]. Since domain knowledge, tutoring and pedagogical knowledge, and dialog management are tightly intertwined, the modeling and integration of proper natural language dialog capabilities in a tutoring system is by no means a trivial task.

In the DIALOG project [17], we aim at a mathematical tutoring dialog system that employs an elaborate natural language dialog component. To tutor mathematics, we need a formally encoded mathematical theory including definitions and theorems along with their proofs, means of classifying the student's input in terms of the knowledge of the domain demonstrated, and a theory of tutoring.

We propose to meet the above aims in the following way. Since it might be impossible to precompile all possible proofs

for a given theorem, we make use of the state-of-the-art theorem prover  $\Omega$ MEGA [21] with its mathematical knowledge base and start with restricting ourselves to the domain of naive set theory. We enhanced the mathematical ontology by making relations between entities explicit which can be used in the tutoring. Moreover, to classify the student's input, we developed a categorization scheme for student answers, which draws on the mathematical ontology.

We aim at a mathematical tutoring system that is able to tutor proving in a way that not only helps the student understand the current proof, but also allows for a high learning effect. What is meant by the latter is the ability of the students to better understand the problem at hand, as well as to generalize and apply the taught strategies on their own later on.

Adhering to the psychological evidence for the high educational effect of hinting [4; 19], we propose to establish those tutoring aims by making use of the *socratic* tutoring method, whose decisive characteristic is the use of hints in order to achieve self-explanation [16; 19; 24]. Our work builds on the, little, systematic research done to date in the area [11; 8; 22].

We have been developing a taxonomy of hints for the naive set theory domain, which draws on the previously mentioned mathematical ontology. This taxonomy is used by a hinting algorithm which models the socratic tutoring method by means of calling different hint categories according to an implicit student model [9]. In this paper we show how the enhanced ontology facilitates planning the next system utterance. In particular, we propose a way of using the ontology for the automatic production of specific hint determinations for a particular context. Note that in this paper we do not address the issue of natural language realization of hints, which is also an ongoing research topic in our project.

In the present paper, we shall first provide a comprehensive description of our domain ontology in Section 2. In Section 3 we shall give a brief overview of the taxonomy of hints, the hinting algorithm and the student answer categorization scheme that we have developed for the domain of tutoring mathematics. Next, in Section 5, we shall outline the use of the ontology that we suggest in this general framework of research and we shall demonstrate through a few examples how exactly the ontology is used for automatically producing hints. In Section 6, we shall discuss our approach before we

$\in$ : element	$\notin$ : not element
$\cap$ : intersection	
$\cup$ : union	
$\subseteq$ : subset	$\not\subseteq$ : not subset
$\subset$ : strict subset	$\not\subset$ : not strict subset
$\supseteq$ : superset	$\not\supseteq$ : not superset
$\supset$ : strict superset	$\not\supset$ : not strict superset
$\mathcal{P}$ : powerset	

Table 1: Mathematical concepts.

conclude the paper.

## 2 A Mathematical Ontology for Hinting

The proof planner  $\Omega$ MEGA [21] makes use of a mathematical database that is organized as a hierarchy of nested mathematical theories. Each theory includes definitions of mathematical concepts, lemmata and theorems about them, and inference rules, which can be seen as lemmata that the proof planner can directly apply. Moreover, each theory inherits all definitions, lemmata and theorems, which we will collectively call *assertions* henceforth, as well as all inference rules from nested theories. Since assertions and inference rules draw on mathematical concepts defined in the mathematical theories, the mathematical database implicitly represents many relations that can potentially be made use of in tutorial dialog. Further useful relations can be found when comparing the definitions of concepts with respect to common patterns. The aim of this paper is to elucidate these relations, to show how they can be made explicit automatically to achieve an enhanced ontology, and to exemplify how that ontology can then be used by the hint producing dialog system.

In this section, we shall first show in Section 2.1 a part of  $\Omega$ MEGA’s mathematical database, which we shall use as an example domain henceforth. Then, we shall define in Section 2.2 the relations to be used in the hinting process.

### 2.1 A Mathematical Database

In  $\Omega$ MEGA’s database, assertions are encoded in a simply-typed  $\lambda$ -calculus, where every concept has a type and well-formedness of formulae is defined by the type restrictions. In this paper, we will exemplarily concentrate on the mathematical concepts from naive set theory given in Table 1. These concepts draw on the types *sets* and *inhabitants* of sets. We give the definitions of the mathematical concepts in intuitive terms as well as in more formal terms paraphrasing but avoiding the  $\lambda$ -calculus formulae as they are represented in  $\Omega$ MEGA’s database.

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

*Element:* The *elements* of a set are its inhabitants:  $x \in U$  if and only if  $x$  is an inhabitant of  $U$ .

*Intersection:* The *intersection* of two sets is the set of their common elements:  $U \cap V = \{x \mid x \in U \text{ and } x \in V\}$ .

*Union:* The *union* of two sets is the set of the elements of both sets:  $U \cup V = \{x \mid x \in U \text{ or } x \in V\}$ .

*Subset:* A set is a *subset* of another set if all elements of the former are also elements of the latter:  $U \subseteq V$  if and only if for all  $x \in U$  follows that  $x \in V$ .

*Strict Subset:* A set is a *strict subset* of another set if the latter has at least one element more:  $U \subset V$  if and only if  $U \subseteq V$  and there is an  $x \in V$  such that  $x \notin U$ .

*Superset:* A set is a *superset* of another set if all elements of the latter are also elements of the former:  $U \supseteq V$  if and only if for all  $x \in V$  follows that  $x \in U$ .

*Strict Superset:* A set is a *strict superset* of another set if the former has at least one element more:  $U \supset V$  if and only if  $U \supseteq V$  and there is an  $x \in U$  such that  $x \notin V$ .

*Equality:* Two sets are equal if they share the same elements:  $U = V$  if and only if for all  $x \in U$  follows that  $x \in V$  and for all  $x \in V$  follows that  $x \in U$ .

*Powerset:* The *powerset* of a set is the set of all its subsets:  $\mathcal{P}(V) = \{U \mid U \subseteq V\}$ .

The definition of the negated concepts from Table 1 is straightforward.

Furthermore, we give examples of lemmata and theorems that use some of these concepts. Let  $U, V$  and  $W$  be sets:

**Commutativity of Union:**  $U \cup V = V \cup U$ .

**Associativity of Union:**  $(U \cup V) \cup W = U \cup (V \cup W)$ .

**Distributivity:**  $U \cup (V \cap W) = (U \cup V) \cap (U \cup W)$ .

**Equality of Sets:** If  $U \subseteq V$  and  $V \subseteq U$  then  $U = V$ .

**Union of Powersets:**  $\mathcal{P}(U) \cup \mathcal{P}(V) \subseteq \mathcal{P}(U \cup V)$ .

**Union in Powerset:** If  $U \in \mathcal{P}(W)$  and  $V \in \mathcal{P}(W)$  then  $U \cup V \in \mathcal{P}(W)$ .

Finally, we give some examples for inference rules. Inference rules have the following general form:

$$\frac{\varphi_1, \dots, \varphi_n}{\psi} R$$

where  $\varphi_1, \dots, \varphi_n, n \geq 0$  are the *premises*, from which the *conclusion*  $\psi$  is derived by rule  $R$ .

Let  $U$  and  $V$  be sets. Example inference rules are:

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

$$\frac{U \in \mathcal{P}(W) \quad V \in \mathcal{P}(W)}{U \cup V \in \mathcal{P}(W)} \cup \mathcal{P}$$

Note that the upper left inference rule encodes the lemma about commutativity of union, the upper right inference rule encodes the lemma about the equality of sets, and the lower inference rule encodes the lemma about the union in powersets. Obviously, every lemma and theorem can be rewritten as an inference rule and every inference rule can be rewritten as a lemma or theorem. Therefore, we shall identify lemmata and theorems with inference rules henceforth.

## 2.2 Enhancing the Ontology

The mathematical database implicitly represents many relations that can potentially be made use of in tutorial dialog. Further useful relations can be found when comparing the definitions of concepts with respect to common patterns. We consider relations between mathematical concepts, relations between mathematical concepts and inference rules, and relations among mathematical concepts, formulae and inference rules. By making these relations explicit we convert the mathematical database into an enhanced ontology that can readily be used in tutoring mathematics.

### Mathematical Concepts

Let  $\sigma, \sigma'$  be mathematical concepts. We define the following relations between mathematical concepts:

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

Examples: antithesis( $\in, \notin$ ), antithesis( $\subseteq, \not\subseteq$ ),  
antithesis( $\subset, \not\subset$ ), antithesis( $\supseteq, \not\supseteq$ ),  
antithesis( $\supset, \not\supset$ )

*Duality:*  $\sigma$  is *dual* to  $\sigma'$  if and only if  $\sigma$  is defined in terms of  $\varphi \Rightarrow \psi$  and  $\sigma'$  is defined in terms of  $\psi \Rightarrow \varphi$  for some formulae  $\varphi, \psi$ .

Examples: dual( $\subseteq, \supseteq$ ), dual( $\subset, \supset$ )

*Junction:*  $\sigma$  is in a *junction* to  $\sigma'$  if and only if  $\sigma$  is defined in terms of  $\varphi_1 \wedge \dots \wedge \varphi_n$  and  $\sigma'$  is defined in terms of  $\varphi_1 \vee \dots \vee \varphi_n$ , or vice versa, for some formulae  $\varphi_1, \dots, \varphi_n$ , where  $n > 1$ .

Examples: junction( $\cap, \cup$ )

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

Examples: hypotaxon( $\subseteq, \subset$ ), hypotaxon( $\subseteq, \mathcal{P}$ )  
 $\in$  is a hypotaxon of  $\subseteq, \supseteq, \cup, \dots$

*Primitive:*  $\sigma$  is a *primitive* if and only if there is no hypotaxon of  $\sigma$ .

Examples: primitive( $\in$ )

Note that  $\in$  is a primitive in  $\Omega$ MEGA's database, since it is defined using *inhabitant*, which is a type, but not a defined concept.

*Specialization:*  $\sigma$  is a *specialization* of  $\sigma'$  if and only if  $\sigma$  is defined as  $\sigma' \wedge \varphi$  for some formula  $\varphi$ .

Examples: specialization( $\subset, \subseteq$ ), specialization( $\supset, \supseteq$ )

*Generalization:*  $\sigma$  is a *generalization* of  $\sigma'$  if and only if  $\sigma'$  is a specialization of  $\sigma$ .

### Mathematical Concepts and Inference Rules

Let  $\sigma, \sigma'$  be mathematical concepts and  $R$  be an inference rule. We define the following relations:

*Relevance:*  $\sigma$  is *relevant* to  $R$  if and only if  $R$  can only be applied when  $\sigma$  is part of the formula at hand (either in the conclusion or in the premises).

Examples: relevant( $\cup, \cup\text{Com}$ ), relevant( $=, \cup\text{Com}$ ),  
relevant( $\subseteq, \text{Set}=\$ ), relevant( $\supseteq, \text{Set}=\$ ),  
relevant( $=, \text{Set}=\$ )

*Dominance:*  $\sigma$  is *dominant* over  $\sigma'$  for rule  $R$  if and only if  $\sigma$  appears in both the premises and the conclusion, but  $\sigma'$  does not.  $\sigma'$  has to appear in one of the premises or the conclusion.

Examples: dominant( $\in, \mathcal{P}, \cup\mathcal{P}$ ),

*Subordination:*  $\sigma$  is *subordinate* to  $\sigma'$  for rule  $R$  if and only if  $\sigma'$  is dominant over  $\sigma$  for rule  $R$ .

### Mathematical Concepts, Formulae and Inference Rules

Let  $\sigma$  be a mathematical concept,  $R$  be an inference rule and  $\varphi_1, \dots, \varphi_n, \psi$  formulae, where  $\varphi_1, \dots, \varphi_n$  are the premises and  $\psi$  the conclusion of  $R$ .

*Introduction:* Rule  $R$  *introduces*  $\sigma$  if and only if  $\sigma$  occurs in the conclusion  $\psi$ , but not in any of the premises  $\varphi_1, \dots, \varphi_n$ .

Examples: introduces( $\text{Set}=\, =$ ), introduces( $\cup\mathcal{P}, \cup$ )

*Elimination:* Rule  $R$  *eliminates*  $\sigma$  if and only if  $\sigma$  occurs in at least one of the premises  $\varphi_1, \dots, \varphi_n$ , but not in the conclusion  $\psi$ .

Examples: eliminates( $\text{Set}=\, \subseteq$ ), eliminates( $\text{Set}=\, \supseteq$ )

### Automation

The automatic enhancement of the mathematical database by explicitly adding the relations defined previously is straightforward. A syntactic occurrence check suffices in the implementation of the relations hypotaxon, primitive, relevant, dominant, subordinate, introduces and eliminates. Pattern matching is necessary to implement the relations antithesis, dual, junction, specialization and generalization.

The automation of the enhancement allows us to plug in any mathematical database and to convert it according to the same principles into a database that includes the relations we want to make use of in hinting.

## 3 General Framework for the Ontology

In this section we shall give a summary of our work so far in the DIALOG project in order to provide the background for a use of the naive set theory ontology, which we are proposing here. We shall briefly talk about our hint taxonomy, our socratic hinting algorithm and our student answer categorization.

## 4 A Taxonomy of Hints

In this section we explain the philosophy and the structure of our hint taxonomy. We also look into some hints that are used in the algorithm. The names of the categories are intended to be as descriptive of the content as possible, and should in some cases be self-explanatory. The taxonomy includes more than the hint categories mentioned in this section. The full taxonomy is given in Table 2. Some categories are not strictly speaking real hints (e.g., *point-to-lesson*), but have been included in the taxonomy since they are part of the general hinting process.

	active	passive
domain-relation	elicit-antithesis elicit-duality elicit-junction elicit-hypotaxis elicit-specialization elicit-generalization	give-away-antithesis give-away-duality give-away-junction give-away-hypotaxis give-away-specialization give-away-generalization
domain-object	give-away-antithesis give-away-duality give-away-junction give-away-hypotaxis give-away-specialization give-away-generalization	give-away-relevant-concept give-away-hypotactical-concept give-away-primitive-concept
inference rule	give-away-relevant-concept give-away-hypotactical-concept give-away-primitive-concept elaborate-domain-object	give-away-inference-rule
substitution	give-away-inference-rule elicit-substitution	spell-out-substitution
meta-reasoning	spell-out-substitution	explain-meta-reasoning
performable-step	explain-meta-reasoning confer-to-lesson	give-away-performable-step
pragmatic	ordered-list unordered-list elicit-discrepancy	take-for-granted point-to-lesson

Table 2: The taxonomy of hints.

#### 4.1 Philosophy and Structure

Our hint taxonomy was derived with regard to the underlying function that can be common for different surface realizations of hints. The underlying function is mainly responsible for the educational effect of hints. Although the surface structure, which undoubtedly plays its own significant role in teaching, is also being examined in the our project, we do not address this issue in this paper.

We defined the hint categories based on the needs in the domain. To estimate those needs we made use of the objects and the relations between them as defined in the mathematical ontology. An additional guide for deriving hint categories that are useful for tutoring in our domain was a previous hint taxonomy, which was derived from the BE&E corpus [22].

The structure of the hint taxonomy reflects the function of the hints with respect to the information that the hint addresses or is meant to trigger. To capture different functions of a hint we define hint categories across two dimensions.

Before we introduce the dimensions, let us clarify some terminology. In the following, we distinguish performable steps from meta-reasoning. *Performable steps* are the steps that can be found in the formal proof. These include premises, conclusion and inference methods such as lemmata, theorems, definitions of concepts, or calculus-level rules (e.g., proof by contradiction). *Meta-reasoning steps* consist of everything that leads to the performable step, but cannot be found in the formal proof. To be more specific, meta-reasoning consists of everything that could potentially be applied to any particular proof. It involves general proving tech-

niques. As soon as a general technique is instantiated for the particular proof, it belongs to the performable step level.

The two hint dimensions consist of the following classes:

- (1) active vs. passive
- (2) domain-relation vs. domain-object vs. inference-rule vs. substitution vs. meta-reasoning vs. performable-step

In the second dimension, we ordered the classes with respect to their subordination relation. We say, that a class is *subordinate* to another one if it reveals more information.

Each of these classes consists of single hint categories that elaborate on one of the attributes of the proof step under consideration. The hint categories are grouped in classes according to the kind of information they address in relation to the domain and the proof. By and large, the hints of the passive function of a class in the second dimension constitute the hints of the active function of its immediately subordinate class, in the same dimension. In addition, the class of *pragmatic* hints belongs to the second dimension as well, but we define it such that it is not subordinate to any other class and no other class is subordinate to it.

In the following section we look at the structure of the second dimension just described through some examples of classes and the hints defined in them.

#### 4.2 First Dimension

The first dimension distinguishes between the active and passive function of hints. The difference lies in the way the information to which the tutor wants to refer is approached. The

idea behind this distinction resembles that of *backward-* vs. *forward-looking* function of dialog acts in DAMSL [5]. The *active* function of hints looks forward and seeks to help the student in accessing a further bit of information, by means of *eliciting*, that will bring him closer to the solution. The student has to think of and produce the answer that is hinted at.

The *passive* function of hints refers to the small piece of information that is provided each time in order to bring the student closer to some answer. The tutor *gives away* some information, which he has normally unsuccessfully tried to elicit previously. Due to that relation between the active and passive function of hints, the passive function of one hint class in the second dimension consists of hint categories that are included in the active function in its subordinate class.

### 4.3 Second Dimension

In this section we will give a few examples of classes and hint categories that capture the structure of the second dimension.

*Domain-relation* hints address the relations between mathematical concepts in the domain, as described in Section 2. The passive function of domain-relation hints is the active function of domain-object hints, that is, they are used to elicit domain objects.

*Domain-object* hints address an object in the domain. The hint *give-away-relevant-concept* names the most prominent concept in the proposition or formula under consideration. This might be, for instance, the concept whose definition the student needs to use in order to proceed with the proof, or the concept that will in general lead the student to understand which inference rule he has to apply. Other examples in the class are *give-away-hypotactical-concept* and *give-away-primitive-concept*. The terms *hypotactical* and *primitive concept* refer to the relation, based on the domain hierarchy, between the addressed concept and the original relevant concept, which the tutor is trying to elicit. Since this class is subordinate to *domain-relation*, the hints in it are more revealing than *domain-relation* hints. The passive function of domain-object hints is used to elicit the applicable inference rule, and, therefore, is part of the active function of the respective class.

The same structure holds for *inference-rule*, *substitution*, *meta-reasoning* and *performable-step* hints.

Finally, the class of *pragmatic* hints is somewhat different from other classes in that it makes use of minimal domain knowledge. It rather refers to pragmatic attributes of the expected answer. The active function hints are *ordered-list*, which specifically refers to the order in which the parts of the expected answer appear, *unordered-list*, which only refers to the number of the parts, and *elicit-discrepancy*, which points out that there is a discrepancy between the student's answer and the expected answer. The latter can be used in place of all other active hint categories. *Take-for-granted* asks the student to just accept something as a fact either when the student cannot understand the explanation or when the explanation would require making use of formal logic. *Point-to-lesson* points the student to the lesson in general and asks him to read it again when it appears that he cannot be helped by tutoring because he does not remember the study material. There is no one-to-one correspondence between the active and pas-

sive pragmatic hints. Some pragmatic hints can be used in combination with hints from other classes.

For more on the taxonomy of hints see [9].

### 4.4 A Hinting Algorithm

A tutoring system ideally aims at having the student find the solution to a problem by himself. Only if the student gets stuck should the system intervene. There is pedagogical evidence [4; 19] that students learn better if the tutor does not give away the answer but instead gives hints that prompt the student for the correct answer. Accordingly, based on the work by Tsovaltzi [22] we have derived an algorithm that implements an eliciting strategy that is user-adaptive by choosing hints tailored to the students. Only if hints appear not to help does the algorithm switch to an explaining strategy, where it gives away the answer and explains it. We shall follow Person and colleagues [16] and Rosé and colleagues [19] in calling the eliciting strategy *socratic* and the explaining strategy *didactic*.

The algorithm makes use of an implicit student model, which is additional to the external student model, that is, the representation of the knowledge the system already possesses, commonly used in intelligent tutoring systems<sup>1</sup>. The latter would be relevant for choosing, for instance the degree of difficulty of the task that the student is asked to perform but not for the constant evaluation that we need for the application of the hinting algorithm. It takes into account the current and previous student answers. The particular input to the algorithm is the category that the student answer has been assigned, based on our student answer categorization scheme, and the domain knowledge employed in the answer. Moreover, the algorithm computes whether to produce a hint and which category of hint to produce, based on the number of wrong answers, as well as the number and kind of hints already produced.

We will now have a closer look at the algorithm and the way the student's level is taken into account.

**The Algorithm** We will present in this paper the main function of the algorithm which implements hinting. The function *socratic* calls several other functions, which we do not look into in this paper. For a more detailed description of the algorithm see [9].

**The Function *socratic*** The bulk of the work is done by the function *socratic*, which we only outline here. The function takes as an argument the category  $C$  of the student's current answer. If the origin of the student's mistake is not clear, a clarification dialog is initiated. Note, however, that the function stops if the student gives the correct answer during that clarification dialog, as that means that the student corrected himself. Otherwise, the function produces a hint in a user-adaptive manner.

In the following,  $H$  denotes the number of hints produced so far and  $C_{-1}$  the category of the student's previous answer. Furthermore, the student answer category inaccurate

<sup>1</sup>In DIALOG this is provided by ACTIVE MATH [14]

is a shorthand for one of the categories complete-partially-accurate or complete-inaccurate or incomplete-partially-accurate. A hint is then produced as follows:

```

Case  $H = 0$ 
  if  $C$  is wrong or inaccurate then call elicit
  if  $C$  is incomplete-accurate
  then produce an active pragmatic hint
    {that is, ordered-list, or unordered-list}
Case  $H = 1$ 
  if  $C$  is wrong
  then if  $C_{-1}$  is wrong or incomplete-accurate
    then call up-to-inference-rule
    if  $C_{-1}$  is inaccurate then call elicit-give-away
    if  $C_{-1}$  is correct then call elicit
  else call elicit
Case  $H = 2$ 
  if  $C$  is wrong
  then if this is the third wrong answer
    then produce explain-meta-reasoning
    else if previous hint was an active substitution hint
    then produce spell-out-substitution
    else if previous hint was spell-out-substitution
    then produce give-away-performable-step
    else call up-to-inference-rule
  else call elicit-give-away
Case  $H = 3$ 
  if  $C$  is wrong and it is at least the third wrong answer
  then produce point-to-lesson and stop
    {The student is asked to read the lesson again. Afterwards,
    the algorithm starts anew.}
  else produce explain-meta-reasoning
Case  $H \geq 4$ 
  give away the answer and switch to didactic strategy;
  switch back after three consecutive correct answers
  with all counters reset
  {After four hints, the algorithm starts to guide the student more
  closely to avoid frustration. If the student is able to follow again
  the tutor's plan for addressing the task, the algorithm switches
  back to the socratic strategy and lets the student take over. If
  the student carries on giving correct answers the main algorithm
  guarantees that the tutor just accepts the answer and does not
  intervene further. Only if the student makes a mistake again will
  the hinting start anew with all counters reset.}

```

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

For a more detailed discussion of the hinting algorithm see [9].

In the next section we shall briefly present the student answer categories we use in the algorithm.

## 4.5 Student Answer Categories

As we have seen, the algorithm takes as input the categories of the student's answers. We categorize the student's answer in terms of its completeness and accuracy with respect to the expected answer. The *expected answer* is the proof step which is expected next from the student, according to the proof which has been produced by the system for the problem at hand.

### Proof Step Matching

We want to allow the student to follow his own line of reasoning. Therefore, we try to make use of that reasoning in

helping him with the task. We model that by trying to match the student's answer to an expected answer in one proof of a set of proofs. To this end we use the state-of-the-art theorem prover  $\Omega$ MEGA [21]. This makes it possible for the system to give guidance according to the proof that the student is attempting without super-imposing one of the alternatives. The student can thus benefit in three ways. First, he can reflect on his own reasoning by getting feedback on it. Second, it becomes cognitively easier to learn since learning presumably takes place based on the structures that gave rise to the particular line of reasoning. Third, the student has the feeling of achievement, which it is pedagogically encouraging.

We are currently investigating using a domain hierarchy in order to define the distance of the expected object from the object in the student answer. We can manipulate the idea of distance to choose between different possible expected answers. That will allow us to always match an answer to the possible proof step closer to the student's attempt. In case of ambiguity, when there is no difference in domain hierarchy distance, we can use varied heuristics, for instance, string matching.

We also consider (accurate or inaccurate) over-answering as several distinct answers. That is, if the student's answer has more proof steps than one, we consider the steps as multiple answers. The categorization is applied to them separately.

### Completeness and Accuracy

Our definitions of completeness and accuracy make use of the concept of a part. We now define the relevant units for the categorization of the student answer. A *part* is a premise, the conclusion and the inference rule of a proof step. A *formula* is a higher-order predicate logic formula. Every symbol defined in the logic is a function. Formulae can constitute of subformulae to an infinite degree of embedding. *Constants* are 0-ary functions that constitute the lowest level entities considered.

We say that an answer is *complete* if and only if all desired parts of the answer are mentioned. We say that a part of an answer is *accurate* if and only if the propositional content of the part is the true and desired one. Based on these notions, we define the following student answer categories:

### The Categories

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

**Complete-Partially-Accurate:** An answer which is complete, but some parts in it are inaccurate.

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

**Incomplete-Accurate:** An answer which is incomplete, but all parts that are present in it are accurate.

**Incomplete-Partially-Accurate:** An answer which is incomplete and some of the parts in it are inaccurate.

**Wrong:** An answer which is both incomplete and inaccurate.

Since we did not expect that all of the above categories would be useful for the algorithm, we collapse the categories complete-partially-accurate, complete-inaccurate and incomplete-partially-accurate to one category, namely, inaccurate.

## 5 Making Use of the Ontology

In this section we shall further explain the use of our ontology by pointing out the exact relevance of it with regard to the general working framework we just presented. We shall also give a few examples of the application of the ontology in automating hint production.

The ontology we are presenting in this paper is evoked primarily in the determination of the content of hint categories chosen by the socratic algorithm. Due to the adaptive nature of our algorithm, and our goal to dynamically produce hints that fit the needs of the student with regard to the particular proof, we cannot restrict ourselves to the use of a gamed of static hints. That is, we cannot resort to associating a student answer with a particular response by the system every time that answer is inputted by the student. Given the input described in Section 4.4, the algorithm computes the appropriate hint category from the taxonomy of hints to be produced. The hint category is chosen with respect to the implicit student model. This means that for every student and for his current performance on the proof being attempted, the hint category chosen must be realized in a different way.

Each hint category is defined based on generic descriptions of domain objects or relations. The role of the ontology is to map the generic descriptions on the actual objects or relations that are used in the particular context, that is, in the particular proof and the proof step in it at hand.

Another equally significant use of the domain ontology is in analyzing the student's answers. This use is, of course, a side-effect of the involvement of the ontology in automatically realizing hints. That is, the algorithm takes as input the analyzed student answer. In analyzing the latter, the system compares it to the expected answer (see Section 4.5) and then looks for the employment of necessary entities.

The necessary entities are defined in terms of the ontology. The algorithm checks for the student's level of understanding by trying to track the use of these concepts in the student's answer to be addressed next. The hint to be produced is then picked according to the knowledge demonstrated by the student. Note that this knowledge might as well have already been provided by the system itself, in a previous turn when dealing with the same performable step. Since the algorithm checks only for generic descriptions of those concepts, we suggest the use of the present ontology in order to map the descriptions onto the actual concepts relevant to the particular context.

### 5.1 Examples of Use

We shall now give examples of the use of our enhanced domain ontology<sup>2</sup>. All the relations mentioned here, which are defined in the ontology have been explained in Section 2. The examples we look at are from our recently collected corpus on mathematics tutorial dialogs in German [3]. Translations in English are included, where necessary. The turn labeling

<sup>2</sup>The existing  $\Omega$ MEGA ontology is used for the identification of inference rules, premises etc.

refers to the actual one in the session from which the example is taken each time.

We shall look into examples of the realization of some hint categories. Through that, we shall show the relevance of the enhanced ontology in analyzing the student's answer.

The hint *give-away-relevant-concept* points out the right mathematical concept that the student has to bring into play in order to carry the proof out. In order to produce it we use the domain ontology in the following way:

- (1) If the inference rule to be applied involves the elimination or the introduction of a concept, then we identify the relevant concept with the concept eliminated or introduced respectively.
- (2) Otherwise, we look for a relevance relation as defined in this paper, and identify the relevant concept with that in the relation.

If the system realizes that the student shows the intention to try out a particular proof but fails to identify the relevant concept for it, then the above process applies for the proof attempted. Alternatively, if the student does not have any particular proof in mind, or the system does not recognize it, then the proof is picked by the system<sup>3</sup>.

Subject 5 was asked in Figure 1 to prove that  $A \cap B \in P((A \cup C) \cap (B \cup C))$  holds. The student's answer was wrong and there is no concept in it that can be identified with a relevant concept for one of the possible proofs. There is not even a concept that bears to a relevant concept one of the relations defined in our enhanced ontology. If the latter case held, an active domain-object hint would be provided in order to elicit the relevant concept itself (cf. Figure 4).

- S2:** Ich setze voraus dass  $A \in P((A \cup C) \cap (B \cup C))$ , und dass  $B \in P((A \cup C) \cap (B \cup C))$ , Dann gilt dass  $A \cap B \in P((A \cup C) \cap (B \cup C))$   
*I assume that  $A \in P((A \cup C) \cap (B \cup C))$ , and that  $B \in P((A \cup C) \cap (B \cup C))$ , Then  $A \cap B \in P((A \cup C) \cap (B \cup C))$  holds*
- T3:** Sie können nicht einfach voraussetzen, daß  $C$  die leere Menge ist. Die Behauptung muß für beliebige Mengen  $A$ ,  $B$  und  $C$  gelten. Für den Beweis müssen Sie die Potenzmenge benutzen.  
*You cannot just assume that  $C$  is the empty set. The proposition has to hold for any sets  $A$ ,  $B$  and  $C$ . For the proof you have to use the powerset.*

Figure 1: Give-away-relevant-concept

The first two sentences in the tutor's response address the mistake that the student made in his preceding answer<sup>4</sup>. The last sentence constitutes the actual hint *give-away-relevant-concept*. Since the student's answer does not include any useful concept, the system gives one possible relevant concept, namely powerset. No information is provided as to how powerset is to be used. However, there is some important implicit information in the hint, which is that a way to reason

<sup>3</sup>The choice of the proof will be made based on the student model.

<sup>4</sup>We give here the whole turn for the sake of precision.

about a proof is by picking a concept which is central to the expression to be proven and applying rules that are related to it. This information addresses the very abstract level of proof technique.

For the hint *elaborate-domain-object* we have to find in the domain the exact way an inference rule needs to be applied, for instance, whether it involves an elimination or an introduction. The student is informed accordingly.

In the example in Figure 2, the student (Subject 12) has to prove that  $K((A \cup B) \cap (C \cup D)) = (K(A) \cap K(B)) \cup (K(C) \cap K(D))$  holds. The student gives a wrong answer. The system analyzes the answer and searches for a relevant concept in it. It identifies complement as the relevant concept for the proof attempted. Therefore, the system chooses to give an *elaborate-domain-object* hint.

- S1:**  $K(A \cap C) \cup (B \cap D) = K((A \cap B) \cup K((C \cap D)))?$   
**T2:** Wissen Sie, wie Sie das Komplement auflösen können?  
*Do you know how to break down the complement?*

Figure 2: Elaborate-domain-object

In the particular realization of *elaborate-domain-object*, the tutor informs the student that he has to eliminate the complement. It is important to notice that instead of saying “eliminate” the tutor uses the phrase “break down” in order to avoid using a domain term that the student might not be familiar with. The information provided by this hint still does not give away the inference rule to be applied. In this case the DeMorgan rules. The tutor attempts to point the student to the fact that he needs to use these rules by telling him that he needs to eliminate complement. This realization also gives implicit feedback for the correct direction of the overall reasoning, that is that the student has used the right relevant concept. At the abstract level, the hint refers to the fact that the aim is to simplify the given expression, so that it can be further manipulated to the desired aim, which is to prove its correctness..

The hint category *give-away-hypotactical-concept*, requires finding in the domain ontology a concept employed by the inference rule to be applied which stands in a hypotaxon relation to the concept most relevant to the inference rule. The hypotaxon is given away to the student.

The example in Figure 3 is from the same session as the previous one. After the hint *elaborate-domain-object*, the student can still not follow. The system gives a *give-away-hypotactical-concept* hint.

- S2:** Nein  
*No*  
**T3:** Sie müssen eine Regel verwenden, mit der Sie den Durchschnitt und die Vereinigung verbinden können.  
*You have to use a rule by which you can relate intersection and union.*

Figure 3: Give-away-hypotactical-concept

Since the student still does not know which inference rule to use, the current hint gives more information with regard to the inference rule. This information is not only useful as

a means of reference, but it points to a way of choosing the relevant inference rule in general.

In Figure 4 we give an imaginary realisation of the hint *give-away-duality* since the hint does not occur in the corpus, but we believe that it can prove a very useful one. The hint captures in intuitive terms the relation in the domain of the concept used by the student to the right one. The relation is duality. The respective hint should, thus, be produced to point that out to the student in order to lead him to the right concept.

- T:** Superset is the opposite of the concept you need to use.

Figure 4: Give-away-duality

This particular realization of the hint would be appropriate if the student chose the concept superset whereas the right concept would be subset. Note that the students are not familiar with the relation duality as such and he does not need to be, but “opposite” should be an appropriate term to express duality.

Note that, in general, the procedure used for finding the relevant concept needs to be followed before we can produce any of the hint categories in the taxonomy which aim at eliciting the relevant concept or the ones used for elaborating on it, as these hints are defined in relation to the latter. That is, we cannot know, for example, what the hypotactical (cf. Figure 3) or the dual concept (cf. Figure 4) is unless we have first computed the relevant concept itself.

The examples we have used here are to clarify the use of the domain ontology that we propose in this paper. The actual way the hints will be realized at the sentence and discourse structure levels each time is an ongoing research in DIALOG. Moreover, the reader should keep in mind that not all of these hints will be provided to one student when tutoring one proof. The choice of hints is down to the hinting algorithm as described in Section 4.4 as well as [9].

## 6 Related Work and Discussion

In this section we shall discuss only work related to building and using domain ontologies in the context of intelligent tutoring systems.

CIRCSIM-Tutor [13; 7] uses an ontology at three levels: The knowledge of domain concepts, the computer context of tutoring and the meta-language on attacking the problem. The ontology we present in this paper is not concerned with the second level. The first level corresponds to our existing knowledge base. The third level can be viewed as a simplified attempt to model tutoring, which we do via hinting. They do, however, use their domain ontology in categorizing the student answer and fixing mistakes.

AIMS [6] is a tool that could be used in combination with our tutorial dialog system. It provides a structured database and retrieval strategies that are based on user modeling. The aim is to both retrieve and present information relevant to a task, in a way that is both economical and potentially cognitively useful. It is targeted to on-line courses. Therefore, it is suitable for presenting the study material that our system presupposes. However, it is not built to provide guidance through or solutions to tasks. Their use of domain ontologies

is done in terms of building a user model and facilitating the presentation of the information for more efficient use.

Within the framework of STEVE [18], Diligent [2] is a tool for learning domain procedural knowledge. Knowledge is acquired by observing an expert's performance of a task, as a first step, subsequently conducting self-experimentation, and finally by human corrections on what Diligent has taught itself. The most relevant part of the knowledge representation is the representation of procedures in terms of steps in a task, ordering constraints, causal links and end goals. Although this is an elaborate learning tool it is not equally elaborate in its use for interacting with students. It is currently limited to making use of the procedure representation to learning text in order to provide explanations.

Finally, the aims of SHIECC [12] are quite different from ours, and the two approaches converge mostly in the use of the domain ontology in evaluating the student level. SHIECC is a tutoring system for the classroom. An expert collaborates with a knowledge acquisition module through questionnaires in building and structuring a domain knowledge base, which includes basic concepts and their characteristics. This knowledge base, extended with expressions between concepts, constitutes their domain ontology. The ontology is used to evaluate the student answers and build a student model, as well as to create lesson plans. Both the ontology and the lesson plans are validated by the expert.

In this paper we are concerned with building an ontology for facilitating tutoring and we suggest an application for automating hinting. We envisage a flexible modular system that deals with tutoring outside of and in collaboration with the understanding and generation modules. In effect, our current ontology can be seen as a resource for planning the tutoring and for generation. The results of the use of the ontology that we have looked at in Section 5 can be the input to the generation module, which might as well be making use of yet another ontology for the actual surface realization of hints. The same is true for the analysis, on the other end.

For discussion on hinting and student answer categorization see [9; 23].

## 7 Conclusion

We have presented a domain ontology for the naive set theory in mathematics. We propose to use this ontology in the general framework of investigating tutorial dialogs for the domain. More specifically in the framework presented in this paper, we have been developing a taxonomy of hints and a socratic algorithm for hint production, in order to take advantage of psychological evidence for the fact that hinting in tutorial dialogs has a positive effect on the student's learning.

Our ontology defines relations between mathematical concepts, formulae and inference rules in the domain of naive set theory. We propose to make use of it in mapping descriptions of objects used in the hint categories onto to actual objects in the domain, which are each time different according to the proof under consideration. This facilitates the automatic production of hints tailored to the needs of a particular student for a particular attempt with a proof.

We are currently analyzing the empirical data collected

through the Wizard-of-Oz experiments that we performed. The analysis has already yielded, as hoped, possible improvements for the automation of hinting. More specifically, we have data that will help us improve the hint taxonomy and make use of more categories already in there. The same is true of the student answer categories, since we now have the data which we needed in order to make use of them in the algorithm. The algorithm, itself will be improved in order to account for the above findings.

Moreover the data will be precious for our research in the actual sentence level realization of hints. We intend to incorporate the study of these empirical data into the smaller research of automatically generating the hints that we can produce. The ontology presented in this paper is particularly relevant to this aim, as well as the analysis of the student's input. It is already obvious that without the domain knowledge that the ontology captures we cannot attempt either of these tasks.

## References

- [1] Vincent Aleven and Kenneth Koedinger. The need for tutorial dialog to support self-explanation. In Rosé and Freedman [20], pages 65–73.
- [2] Richard Angros, Jr., W. Lewis Johnson, Jeff Rickel, and Schorel Andrew. Learning domain knowledge for teaching procedural skills. In *Proceedings of AAMAS'02*, Bologna, Italy, 2002.
- [3] Chris Benz Müller, Armin Fiedler, Malte Gabsdil, Helmut Horacek, Ivana Kruijff-Korbayová, Manfred Pinkal, Jörg Siekmann, Dimitra Tsovaltzi, Bao Quoc Vo, and Magdalena Wolska. A Wizard-of-Oz experiment for tutorial dialogues in mathematics. In *Proceedings of the AIED Workshop on Advanced Technologies for Mathematics Education*, Sidney, Australia, 2003. Submitted.
- [4] Michelene T. H. Chi, Nicholas de Leeuw, Mei-Hung Chiu, and Christian Lavancher. Eliciting self-explanation improves understanding. *Cognitive Science*, 18:439–477, 1994.
- [5] Mark G. Core and James F. Allen. Coding dialogues with DAMSL annotation scheme. In *AAAI Fall Symposium on Communicative Action in Humans and Machines*, pages 28–35, Boston, MA, 1993.
- [6] Darina Dicheva and Lora Aroyo. An approach to intelligent information handling in web-based learning environments. In *Proceedings of IC-AI*, pages 1327–1333, Antonio, Texas, 2000.
- [7] Martha W. Evens, Stefan Brandle, Ru-Charn Chang, Reva Freedman, Michael Glass, Yoon Hee Lee, Leem Seop Shim, Chong Woo Woo and Yuemei Zhang, Yujian Zhou, Joel A. Michael, and Allen A. Rovick. CIRCSIM-Tutor: An intelligent tutoring system using natural language dialogue. In *Proceedings of 12th Midwest AI and Cognitive Science Conference, MAICS-2001*, pages 16–23, Oxford OH, 2001.
- [8] Armin Fiedler and Helmut Horacek. Towards understanding the role of hints in tutorial dialogues. In

- BI-DIALOG: 5th Workshop on Formal Semantics and Pragmatics in Dialogue*, pages 40–44, Bielefeld, Germany, 2001.
- [9] Armin Fiedler and Dimitra Tsovaltzi. Automating hinting in mathematical tutorial dialogue. In *Proceedings of the EACL-03 Workshop on Dialogue Systems: interaction, adaptation and styles of management*, pages 45–52, Budapest, 2003.
- [10] Neil Heffernan and Kenneth Koedinger. Intelligent tutoring systems are missing the tutor: Building a more strategic dialog-based tutor. In Rosé and Freedman [20], pages 14–19.
- [11] Gregory D. Hume, Joel A. Michael, Rovick A. Allen, and Martha W. Evens. Hinting as a tactic in one-on-one tutoring. *Journal of the Learning Sciences*, 5(1):23–47, 1996.
- [12] Sofiane Labidi and Nilo Sérgio. Student modeling and semi-automatic domain ontology construction for SHIECC. In *Proceedings of the 30th ASEE/IEEE Frontiers in Education Conference*, Kansas City, MO, 2000.
- [13] Chung Hee Lee, Jai Hyun Seu, and Martha W. Evens. Building an ontology for CIRCSIM-Tutor. In *Proceedings of the 13th Midwest AI and Cognitive Science Conference, MAICS-2002*, pages 161–168, Chicago, 2002.
- [14] E. Melis, E. Andres, A. Franke, G. Gogvadse, M. Kohlhasse, P. Libbrecht, M. Pollet, and C. Ullrich. A generic and adaptive web-based learning environment. In *Artificial Intelligence and Education*, pages 385–407, 2001.
- [15] Johanna Moore. What makes human explanations effective? In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, pages 131–136. Hillsdale, NJ. Earlbaum, 2000.
- [16] Natalie K. Person, Arthur C. Graesser, Derek Harter, Eric Mathews, and the Tutoring Research Group. Dialog move generation and conversation management in AutoTutor. In Rosé and Freedman [20], pages 45–51.
- [17] Manfred Pinkal, Jörg Siekmann, and Christoph Benz Müller. Projektantrag Teilprojekt MI3 — DIALOG: Tutorieller Dialog mit einem mathematischen Assistenzsystem. In *Fortsetzungsantrag SFB 378 — Ressourcenadaptive kognitive Prozesse*, Universität des Saarlandes, Saarbrücken, Germany, 2001.
- [18] Jeff Rickel, Rajaram Ganeshan, Charles Rich, Candance L. Sidner, and Neal Lesh. Task-oriented tutorial dialogue: Issues and agents. In Rosé and Freedman [20], pages 52–57.
- [19] Carolyn P. Rosé, Johanna D. Moore, Kurt VanLehn, and David Allbritton. A comparative evaluation of socratic versus didactic tutoring. In Johanna Moore and Keith Stenning, editors, *Proceedings 23rd Annual Conference of the Cognitive Science Society*, University of Edinburgh, Scotland, UK, 2001.
- [20] Carolyn Penstein Rosé and Reva Freedman, editors. *Building Dialog Systems for Tutorial Applications—Papers from the AAAI Fall Symposium*, North Falmouth, MA, 2000. AAAI press.
- [21] Jörg Siekmann, Christoph Benz Müller, Vladimir Brezhnev, Lassaad Cheikhrouhou, Armin Fiedler, Andreas Franke, Helmut Horacek, Michael Kohlhasse, Andreas Meier, Erica Melis, Markus Moschner, Immanuel Normann, Martin Pollet, Volker Sorge, Carsten Ullrich, Claus-Peter Wirth, and Jürgen Zimmer. Proof development with  $\Omega$ MEGA. In Andrei Voronkov, editor, *Automated Deduction — CADE-18*, number 2392 in LNAI, pages 144–149. Springer Verlag, 2002.
- [22] Dimitra Tsovaltzi. Formalising hinting in tutorial dialogues. Master’s thesis, The University of Edinburgh, Scotland, UK, 2001.
- [23] Dimitra Tsovaltzi and Armin Fiedler. An approach to facilitating reflection in a mathematics tutoring system. In *Proceedings of AIED Workshop on Learner Modelling for Reflection*, Sydney, Australia, 2003. Submitted.
- [24] Dimitra Tsovaltzi and Colin Matheson. Formalising hinting in tutorial dialogues. In *EDILOG: 6th workshop on the semantics and pragmatics of dialogue*, pages 185–192, Edinburgh, Scotland, UK, 2002.