Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Electronic Notes in Theoretical Computer Science

ELSEVIER Electronic Notes in Theoretical Computer Science 93 (2004) 5–23
www.elsevier.com/locate/entcs

Assertion-level Proof Representation with Under-Specification

Serge Autexier[1,2], Christoph Benzmüller[1], Armin Fiedler[3], Helmut Horacek[1], and Bao Quoc Vo[1]

[1] *Department of Computer Science, Saarland University, Saarbrücken, Germany*
[2] *German Research Center for Artificial Intelligence (DFKI) Saarbrücken, Germany*
[3] *CISA, School of Informatics, University of Edinburgh, Edinburgh, Scotland*
Email: {serge|chris|afiedler|horacek|bao}@ags.uni-sb.de

Abstract

We propose a proof representation format for human-oriented proofs at the assertion level with under-specification. This work aims at providing a possible solution to challenging phenomena worked out in empirical studies in the DIALOG project at Saarland University. A particular challenge in this project is to bridge the gap between the human-oriented proof representation format with under-specification used in the proof manager of the tutorial dialogue system and the calculus- and machine-oriented representation format of the domain reasoner.

Keywords: informal proofs, proof representation, assertion level, under-specification

1 Introduction

In this paper we propose a proof representation format for human-oriented proofs at the assertion level with under-specification. This work aims at providing a possible solution to challenging phenomena worked out in empirical studies in the DIALOG project at Saarland University. In these studies 24 students with varying background were interacting with an intended tutorial natural language dialogue system in a wizard-of-oz (WOz) study, where the dialogue system was mimicked

⋆ 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.

1571-0661 © 2004 Elsevier B.V. Open access under CC BY-NC-ND license.
doi:10.1016/j.entcs.2003.12.026

by a human mathematician; the joint task was to construct proofs in naive set theory. Neither the tutor system nor the subjects were constrained in this experiments with respect to particular proof formats. The obtained corpus has now been analysed and provides us with valuable insights in the notion and nature of proofs constructed by humans in this field. Among the most prominent differences concerning specification of proof fragments and references to them are the level of granularity of these proofs, which is typically the application of theorems, axioms and lemmata, and the aspect of under-specification, for example, missing references to premise assertions, rule and instantiation specifications.

The term under-specification is borrowed from the field of computational linguistics; it means that certain aspects in the semantic representation of a natural language utterance are left uninterpreted, such that their proper treatment can be deferred to later stages of processing in which more contextual information is available. The most prominent aspect for which under-specification has been used in computational linguistics is scopus of quantifiers. Since this use differs from the aspects relevant for our purposes, we refer to under-specification only for proof representations when seen from the mathematical/logical perspective.

A particular challenge is to develop an appropriate proof representation format since these human-constructed proofs need to be represented and maintained in the tutorial dialogue system. A challenge for the interaction with the domain reasoner (a theorem prover for naive set theory) is to bridge the gap between this human-oriented proof representation format and the calculus- and machine-oriented representation format of the domain reasoner, such that feedback on several relevant aspects in the context can be produced by the domain reasoner for each proof step uttered by the subjects.

The experiments conducted and the proof representation format proposed in this paper are relevant not only for the tutorial systems community, but also for the theorem proving community, especially in the following aspects:

- *Human-Oriented Interactive Proof:* Many research papers in the field of deduction systems refer to the notion of human-oriented proofs. From the psychological perspective, there is a limited degree of evidence how humans communicate in inference-rich discourse, such as proof descriptions, and where they have difficulties. It has been shown that human capabilities in understanding deductive syllogisms may differ significantly — understanding the most frequent one, modus ponens, is much easier [12]. Therefore, it is required to present logically redundant information for more complex reasoning tasks in conversation with humans [22]. By and large, these insights favour the assertion level [11] as the most appropriate level of granularity, unless complex reasoning patterns are to be described. Moreover, humans typically exploit contextual expectations, so that several details can be left implicit in descriptions [20]. From the per-

spective of referring to proof fragments, this corresponds to under-specification. However, rather few empirical data are available in form of corpora like ours, in which proofs were constructed interactively by students and teachers. In order to precise the notion of human-oriented interactive proof we propose to analyse these corpora. In contrast, many scientist and system developers in the theorem proving community still take for granted that sequent- or natural-deduction-based proof representation formats are already a nearly optimal solution. The aspect of under-specification in this context has hardly been addressed so far in the literature. In fact, the corpus we collected indicates that neither the natural-deduction nor the sequent calculus nor a pure rewriting-based approach are sufficient here, whereas assertion-level proofs come closer. However, without being able to address under-specification none of these formats are sufficiently suited as a human-oriented proof format.

- *User-Interfaces for Theorem Provers:* The clarification of the notion of human-oriented proofs is very relevant for the design of user-interfaces for theorem provers. Our viewpoint coincides with [19], which investigates paper proofs instead of interactively constructed proofs with under-specification, in the sense that a clear separation between the optimally user-oriented proof representation in the user interface and the usually machine-oriented proof representation in the theorem prover appears increasingly important. The design and development of most current user interfaces for theorem provers often starts from the existing functionalities of the concrete system and this approach typically strongly constrains the achieved and achievable solution.

For a discussion of the notion of *human-oriented proofs* it is reasonable to differentiate between cleaned-up textbook proofs (which have been investigated in the literature) and interactive dialogues on proofs, for instance, between students and tutors, as in our setting. Thus, not only the tutoring systems community but also the theorem proving community could strongly benefit from further data about human-constructed proofs in a tutorial setting without assuming some a priori constraints on proof styles and calculi.

In this paper we shall briefly sketch the DIALOG project and discuss the setting of the empirical studies (Section 2) and present examples of the resulting corpus data (Section 3). In the main part of the paper, relevant phenomena of the collected corpus, such as under-specification, will be discussed (Section 4) and a preliminary proof representation format addressing these phenomena will be proposed and some implications on the required capabilities and features for the theorem prover to be used as domain reasoner in the DIALOG project will be discussed (Section 5). Finally, we shall discuss related work and conclude the paper.

2 Empirical Studies in the DIALOG Project

In the DIALOG project [17], we aim at a mathematical tutoring system that employs an elaborate natural language dialogue 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, and a theory of tutoring that should make use of hints.

In this paper, we are primarily concerned with defining a data structure for the proof that the user of the system is developing. The user is supposed to prove a theorem by informing the system about every proof step he/she takes. Since the communication between user and system is to be in natural language, it is important to devise an interpretation component that is able to map the natural language descriptions of the proof, such as the steps the user utters, into a formal representation of the proof under development.

As a first step toward the design of the system, we started with empirical investigations to pinpoint linguistic phenomena and relevant aspects of the domain that need to be addressed. To this end, we followed the *Wizard-of-Oz (WOz)* approach [5], where the subjects in an experiment are told that they interact with a computer system, but where the computer system is, either as a whole or in part, simulated by a human expert, called the *wizard*. If the wizard is restricted in his/her behaviour to follow certain algorithms, the data collected in the experiment reflect the interaction between the subjects and the envisioned system. This allows for an early evaluation of algorithms before they have been implemented. Moreover, the data also allows for the analysis of the subjects' behaviour in the interaction with the envisioned system, even if the wizard has not been restricted in his behaviour. Note that it is important that the subjects think that they interact with a computer system, since it has been observed that humans interact differently with computers than with other humans [10,6].

In our project, we plan for a series of experiments where the wizard will be first increasingly restricted in his possibilities to comply with the specific algorithms of the envisioned system and then gradually replaced by the implemented algorithms. To that end, we developed a tool that allows for such a progressive refinement of the experimental setup [8].

For the first stage, we decided to restrict ourselves to the domain of naive set theory, which is a rather basic mathematical theory that directly builds upon logic without making use of other complex theories. A major advantage of this theory is that it is manageable to a reasonable extent by current automated theorem provers. The investigation in more complex mathematical theories will be done in subsequent experiments.

In the first experiment, we asked the subjects to evaluate a tutoring system

with natural language dialogue capabilities. To work with the system, the subjects had to prove three theorems. The purpose of the first theorem, which stated that $K((A \cup B) \cap (C \cup D)) = (K(A) \cap K(B)) \cup (K(C) \cap K(D))$ (where K stands for the complement of a set), was to make the subject familiar to the system's interface. The main part of the experiment consisted of two further more complex theorems stating (a) $A \cap B \in P((A \cup C) \cap (B \cup C))$ (where P stands for the power-set) and (b) If $A \subseteq K(B)$, then $B \subseteq K(A)$, which were given to the subjects in varying order. For every utterance of the subjects the wizard had to decide if the utterance corresponds to a relevant and correct step of a proof of the theorem under consideration. If so, he only had to acknowledge the step, otherwise a hinting algorithm was invoked that decided what kind of hint should be produced by the system and the wizard had to verbalise the hint. The experiment setup is presented in more detail in [4]. The details of the hinting algorithm can be found in [9].

In the next section, we shall show some example dialogues recorded during the first experiment.

3 An Example Dialogue

In the example sessions presented below the utterances of tutor (wizard) and subject are labelled with **T***n* and **S***n*, respectively, where *n* is a consecutive numbering. [1]

Example 3.1 *(Subject 16)* The proof constructed interactively by the student is a forward proof starting from a well chosen instance of the deMorgan's second law.

T1: Bitte zeigen Sie: $K((A \cup B) \cap (C \cup D)) = (K(A) \cap K(B)) \cup (K(C) \cap K(D))$
[Please show: $K((A \cup B) \cap (C \cup D)) = (K(A) \cap K(B)) \cup (K(C) \cap K(D))$*]*

S1: nach deMorgan-Regel-2 ist $K((A \cup B) \cap (C \cup D)) = (K(A \cup B) \cup K(C \cup D))$
[by deMorgan-Rule-2 $K((A \cup B) \cap (C \cup D)) = (K(A \cup B) \cup K(C \cup D))$ *holds]*

T2: Das ist richtig!
[This is correct!]

S2: $K(A \cup B)$ ist laut deMorgan-1 $K(A) \cap K(B)$
[$K(A \cup B)$ *is* $K(A) \cap K(B)$ *according to deMorgan-1]*

T3: Das stimmt auch.
[That is also right.]

S3: und $K(C \cup D)$ ist ebenfalls laut deMorgan-1 $K(C) \cap K(D)$
[and $K(C \cup D)$ *is also* $K(C) \cap K(D)$ *according to deMorgan-1]*

T4: Auch das stimmt.
[That also is right.]

[1] The complete corpus is available at http://www.ags.uni-sb.de/~dialog/.

S4: also folgt letztendlich: $K((A \cup B) \cap (C \cup D)) = (K(A) \cap K(B)) \cup (K(C) \cap K(D))$.
[hence follows finally: $K((A \cup B) \cap (C \cup D)) = (K(A) \cap K(B)) \cup (K(C) \cap K(D))$.*]*

T5: Das stimmt genau. Ich wiederhole noch einmal: Auf die linke Seite der Gleichung kann ich zuerst die zweite und danach die erste deMorgan-Regel anwenden, so daß sich folgende Argumentationskette ergibt : $K((A \cup B) \cap (C \cup D)) = K(A \cup B) \cup K(C \cup D) = (K(A) \cap K(B)) \cup (K(C) \cap K(D))$.
[This is absolutely correct. I repeat once more: On the left side of the equation I can apply first the second and then the first deMorgan rule, such that we the the following chain of argument: $K((A \cup B) \cap (C \cup D)) = K(A \cup B) \cup K(C \cup D) = (K(A) \cap K(B)) \cup (K(C) \cap K(D))$.*]*

□

4 Under-Specification and the Nature of Interactive Proofs

The example from the previous section poses interesting challenges for the representation and analysis of the consecutive proof steps. One particular challenge is the aspect of under-specification, that is, the fact that many pieces of information that are typically expected when constructing a proof interactively with a proof assistant are omitted. This includes information such as

- whether a formula given by the student is a derived fact or a subgoal for some previous goal. Sometimes this information can be derived or at least guessed by linguistic analysis from the textual parts in the student answer. However, in the experiments very often the student only inputted a formula without further description and then nothing is known about that formula.
- missing references to assertions (e.g., premises) employed in a step (e.g., in S4)
- missing information about instantiations of universally quantified formulae (e.g., in S4)
- missing information about positions where manipulations take place (e.g., in S4)

In addition to under-specification, that is, incompleteness of proof step information, further challenges can be identified. These comprise correctness of the proof step, granularity of the proposed proof step, and relevance of the proof step. In a tutorial setting all of these aspects need to be addressed when analysing whether a proof step proposed by the student should be accepted by the system or not. In particular, we propose that the uttered proof step is first represented as a hypothetical step in an appropriate proof representation format (supporting hypothetical and probably under-specified steps) such that the different analysis tasks can be performed on this representation.

We now discuss the four challenges in more detail.

Information Completeness: Information completeness refers to the question whether an information provided in a students utterance is acceptable from (a) a tutorial point of view and (b) from the perspective of the domain reasoners capabilities to potentially substitute information parts that are required for constructing a formal proof but which are not given (i.e., whether the degree of under-specification still allows the domain reasoner to potentially follow the argument). The former requires interaction with tutorial components of the systems while the latter requires mechanisms to address and resolve ambiguities.

Accuracy: A proof step (or a whole subproof) uttered by a student is accurate if it can be verified as a correct inference. This is usually done with the domain reasoner. In the context of our corpus this mainly boils down to verifying (a) whether a fact is indeed inferable from the assertions in the context or (b) whether a proof goal can indeed be refined to a subgoal using some assertions in the context. In both cases the set of assertions may be under-specified and further information as already discusses above (position, intantiations, etc.) may be taken into account.

We call a proof step *correct* if it is both accurate and information complete (cf. [21] for details of a student answer categorisation).

Granularity: The level of granularity reflects the number of inference steps required to establish the asserted proof step. The assessment whether the level of granularity is acceptable depends on expertise of the user: a proof step considered to be obvious by an expert mathematician can be a very challenging problem for a novice and possibly requires dozens of intermediate steps to elaborate the justification.

We call a proof step *communicatively adequate* if it is information complete with respect to the tutorial model, accurate, and if its level of granularity size is acceptable.

Relevance: It should be clear that the criterion of correctness as introduced can only tell us whether the proposed proof step is logically valid but gives no indication whatsoever on whether it is a right step to pursue to achieve a proof. This criterion is about whether the proposed proof step is relevant to the current proving context. There are several degrees to which the relevance of a proof step can be judged:

- On a rough scale, a proof step is completely irrelevant if no proofs have to make use of this proof step. At the other extreme, a proof step must be used if every (known) proof of the theorem employs it. And, in between, there are also proof steps that are used by some proofs but not by all proofs.

- Another aspect addresses a preferred proof (e.g., if teaching a specific proving strategy is part of the tutorial goal) and the current proof which the user and the system are pursuing (i.e., the current proof step must fit into the context of the proof under development).
- Of more advanced criteria, a proof step may also reveal the intention of the proposer to try to follow some particular proving strategy, or to explore some unknown parts of the problem, or just a creative move when trying to tackle the problem. This is certainly beyond the capabilities of current theorem provers and requires additional knowledge and heuristics. But it is certainly relevant in the DIALOG project where the system could be equipped with some pedagogical knowledge to allow students to discover problem solving techniques.

Relevance is an interesting and very challenging issue and more research is needed here. We expect that domain specific reasoning techniques such as proof planning may play an important role for the automation of relevance investigations. Traditional resolution or saturation based first-order reasoning systems, in contrast, appear less useful in this context because of the gap between their machine-oriented representations and search strategies to those used by the students.

5 A Proof Representation Format with Under-Specification

The role of the proof representation format is to serve as an interface between the linguistic analysis component in a tutor system and the so-called proof manager. The first component analyses the student's utterances and filters out, among other things, those utterances that represent proof steps. The proof manager is concerned with the reconstruction of the possible proof structure the student is performing and checks the validity of the proof steps, which should ideally also provide information that is missing in the student's utterance, such as which facts have been used and how they have been used.

The design of the intermediate proof representation format occurs in the field of tension between how precisely the proof step utterances can be categorised by linguistic means and what is minimally required to perform sensible proof reconstruction and checking.

Although the proofs conducted in the WOz experiments have a poor proof structure, the proof representation format we introduce provides a richer structure including goal refinement to multiple subgoals and case distinction. These richer structure elements will be required for more complex theorems which will be beneficial for the design of further experiments.

5.1 Proof Language

We now define formally a language to represent both formal and informal proofs.

References

As pointed out previously, references in the students' utterances to facts or subgoals are often under-specified or even absent. Often, the students use the name or the formula of some fact or goal, or information on how they are used, for instance, by specifying the subformula or subterm. Suppose we have grammars to define non-terminals name, formula and occurrence. Then, we define

$$\begin{aligned} \text{Names} \quad & \mathsf{N} ::= \mathsf{name} \mid . \\ \text{Formulae} \quad & \mathsf{F} ::= \mathsf{formula} \mid . \\ \text{Occurrences} \quad & \mathsf{O} ::= \mathsf{occurrence} \mid . \end{aligned}$$

where '.' denotes under-specification. Then we define references as triples

$$\text{References} \quad \mathsf{R} ::= (\mathsf{N}, \mathsf{F}, \mathsf{O})$$

Proof Language

The analysis of the proofs from the experiments was guided by the question "Which are the typical, linguistically categorisable proof steps the students performed?" This leads to the following categories:

(i) Derivation of new facts with potentially under-specified references to the used facts.

(ii) Introduction of a new subgoal by referring to the replaced goal.

(iii) Decomposition of complex goal formulae by introduction of new hypotheses.

(iv) Assignment of values to instantiable variables.

(v) Introduction of abbreviations for complex formulae or subterms.

(vi) Statement that the proof is finished.

Furthermore, since linguistic analysis is not always able to uniquely categorise a given utterance, we introduce a non-deterministic branching over possible proofs (`Or`) to represent the different alternative interpretations. Finally, in order to obtain a richer proof representation format suitable for more complex proofs, we extend these identified steps in two ways: firstly, for the goal reduction we allow for more than one subgoal. Secondly, we add case distinction.[2] This results in the proof

[2] While these aspects are not not occurring in our corpus, it is clear that they generally play a role in mathematics. The investigation whether our language designed here is still appropriate for further corpora where these aspects do occur is further work. Generally we will have to face the question

Step	S	$::= .$
		$\mid$ `Trivial`
		$\mid$ `Fact` $N : F$ `from` $R^*; S$
		$\mid$ `Subgoals` $(N : F)^+$ `for` R `by` R^* `in` S^+ `End`
		$\mid$ `Assume` H^* `prove` $N : F$ `(from` R`)` `in` S `End`
		$\mid$ `Assign` $(SUBST \mid ABBRV); S$
		$\mid$ `Or`$(S_1 \parallel \ldots \parallel S_n)$
		$\mid$ `Cases` $F^+ : (\texttt{Case}\ N : F : S\ \texttt{End})^+$ `End`
Hypotheses	H	$::= N : F \mid CONST : TYPE? \mid VAR : TYPE?$
Substitutions	$SUBST$	$::=$ `Let` $VAR := TERM$
Abbreviations	$ABBRV$	$::=$ `Let` $CONST := TERM$
Constants	$CONST$	$::=$ const N
Variables	VAR	$::=$ var N
Types	$TYPE$	$::= \ldots$

Fig. 1. Proof representation language

language presented in Fig. 1.

In our language the proof step "." indicates an unfinished (sub-)proof while `Trivial` indicates that the (sub-)proof should be completed now, for instance there is a formula that occurs both as a goal and a hypothesis. The step `Fact` $N : F$ `from` R^* indicates that a fact F has been derived from the current facts and has been assigned the name N. A proof step `Subgoals` $(N : F)^+$ `for` R `by` R^* `in` S^+ `End` represents the fact that we introduced a list of subgoals $(N : F)^+$ for some previous goal R and the proofs in S^+ are the subproofs for these subgoals. Note that the facts used to perform that goal reduction may be indicated in R^*. The proof steps `Assume` H^* `prove` $N : F$ `(from` R`)` `in` S `End` are used to decompose a goal R into the new hypotheses H^* and the new goal F of name N. The hypotheses can be either named formulae $N : F$, or new constants and variables, possibly with some type. The `Or`$(S_1 \parallel \ldots \parallel S_n)$ describes a situation, where from the linguistic analysis there are several possible interpretations resulting in different possible proofs. Resolving this non-determinism by proof checking and eliminate impossible interpretation

with each new corpus whether our language needs to be adapted to new phenomena.

S1: `Fact .` : $K((A \cup B) \cap (C \cup D)) = (K(A) \cap K(B)) \cup (K(C) \cap K(D))$ `from` (deMorgan-Rule-2, ., .);

S2: `Fact .` : $K(A \cup B) = K(A) \cap K(B)$ `from` (deMorgan-Rule-1, ., .);

S3: `Fact .` : $K(C \cup D) = K(C) \cap K(D)$ `from` (deMorgan-Rule-1, ., .);

S4: `Fact .` : $K((A \cup B) \cap (C \cup D)) = (K(A) \cap K(B)) \cup (K(C) \cap K(D))$ `from` (., ., .);

S4: `Trivial`

Fig. 2. The example proof in the representation language.

is one example of important feedback information to be returned to the linguistic analyser. Finally, case distinctions can be introduced by the `Cases` construct, where for each formula φ in F^+ there is exactly one case $n : \varphi$.

Example 5.1 We illustrate the proof language by representing the complete proof from Example 3.1 in Fig. 2. In the proof we indicate to which student utterance the individual proof steps belong. □

5.2 Proof Checking

In this section, we propose rules that allow us to check the obtained proofs by checking each individual proof step. For each individual proof step we need to know all visible hypotheses, denoted by Γ, and all previous goals, which is a list of named formulae and is denoted by Δ. The proof checking rules are given in Fig. 3. Note that we explicitly refrain from fixing a specific logic in these rules, as we envision the possibility to use the same abstract proof format for proofs in different domains. Thus, the proof checking system is parameterised over the calculus for the specific logic. The connection to these calculi is established via the local lemmata arising during proof checking.

Proof checking individual proof steps may give rise to a local lemma that needs to be verified to establish the validity of this step. Most proof step kinds give rise to a specific kind of lemma, such as $\Gamma \Longrightarrow_{\mathsf{Triv}} \Delta$ for `Trivial` proof steps, where Γ and Δ are as above. For each kind K of lemma $\Gamma \Longrightarrow_{\mathsf{K}} \Delta$ we allow to have a specific *strategy* to establish this kind of lemma. [3] Note that the strategies need to

[3] Our notion of *strategy* here is rather general and probably we should rather speak of a strategic reasoner that is parameterised over some tutorial knowledge. This tutorial knowledge controls and restricts the search space of the strategic reasoner. In the DIALOG project we employ the proof planner of the ΩMEGA system which can be controlled by domain specific strategic information (e.g., control rules). Generally, however, any kind of reasoner that is controllable by some domain specific information generally qualifies as a candidate; however, see the discussion in Section 5.4.

$$\frac{P:\Gamma \Longrightarrow_{\mathsf{Triv}} \Delta}{\Gamma\langle \texttt{Trivial}\rangle\Delta}\ \mathsf{Trivial} \qquad \frac{\Gamma[S_i]\Delta}{\Gamma\langle \texttt{Or}(S_1 \parallel \ldots \parallel S_n)\rangle\Delta}\ \mathsf{Or}$$

$$\frac{P(R^*):\Gamma \Longrightarrow_{\mathsf{Fact}} F,\Delta \quad \Gamma,N:F[S]\Delta}{\Gamma\langle \texttt{Fact}\ N:F\ \texttt{from}\ R^*;S\rangle\Delta}\ \mathsf{Fact}$$

$$\frac{P(R,R^*):\Gamma,(F_1\wedge\ldots\wedge F_k) \Longrightarrow_{\mathsf{Subgoal}} \Delta \quad \Gamma[S_1]N_1:F_1,\Delta \quad \ldots \quad \Gamma[S_k]N_k:F_k,\Delta}{\Gamma\langle \texttt{Subgoals}\ N_1:F_1,\ldots,N_k:F_k\ \texttt{for}\ R\ \texttt{by}\ R^*\ \texttt{in}\ S_1 \mid \ldots \mid S_k\ \texttt{End}\rangle\Delta}\ \mathsf{Subgoals}$$

$$\frac{P(R):\Gamma,F \Longrightarrow_{\mathsf{Focus}} \Delta \quad P'(R):\Gamma \Longrightarrow_{\mathsf{Hyp}} (H_1\wedge\ldots\wedge H_n),\Delta, \quad \Gamma,H_1\wedge\ldots\wedge H_n[S]N:F,\Delta}{\Gamma\langle \texttt{Assume}\ H_1,\ldots,H_n\ \texttt{prove}\ N:F\ \texttt{(from}\ R\texttt{)}\ \texttt{in}\ S\ \texttt{End}\rangle\Delta}\ \mathsf{Assume}$$

$$\frac{\texttt{var}\ x:\tau\in\Gamma \quad \Gamma \Longrightarrow_{\mathsf{Type}} t:\tau \quad P:\Gamma \Longrightarrow_{\mathsf{Subst}} x=t,\Delta \quad \Gamma,.:x=t[S]\Delta}{\Gamma\langle \texttt{Assign var}\ x:=t;S\rangle\Delta}\ \mathsf{Assign\text{-}Subst}$$

$$\frac{\Gamma \Longrightarrow_{\mathsf{Type}} t:\tau \quad c\notin\Gamma\cup\Delta \quad \Gamma,.:\texttt{const}\ c:\tau,.:c=t[S]\Delta}{\Gamma\langle \texttt{Assign const}\ c:=t;S\rangle\Delta}\ \mathsf{Assign\text{-}Abbrv}$$

$$\frac{P:\Gamma \Longrightarrow_{\mathsf{Case}} F_1\vee\ldots\vee F_n,\Delta \quad \Gamma,N_1:F_1[S_1]\Delta \quad \ldots \quad \Gamma,N_n:F_n[S_n]\Delta}{\Gamma\langle \texttt{Cases}\ F_1,\ldots,F_n:\texttt{Case}\ N_1:F_1:S_1\ \texttt{End}\ldots\texttt{Case}\ N_n:F_n:S_n\ \texttt{End End}\rangle\Delta}\ \mathsf{Case}$$

Fig. 3. Proof Checking Rules

implement a decision procedure that checks if the conditions for communicative adequacy are fulfilled. The individual kinds of lemma are:

Triv: These lemmata arise when the student states that the (sub-)proof is trivially proved.

Fact: These lemmata ensure the validity of derived facts.

Subgoal: These lemmata ensure the validity of goal reductions.

Focus and Hyp: These two kinds of lemma arise when the student decomposes a complex goal formula and states the additional assumptions and the new subgoal. The *Focus*-lemmata ensure that indeed the formula F is a subgoal contained in the complex goal, while the *Hyp*-lemmata ensure that the indicated hypotheses are indeed valid hypotheses for that subgoal. Note that encoding these as lemmata gives the student more flexibility, since the indicated hypotheses and subgoals need not be exact subformulae of the goal formula, but can already be consequences of these, which is checked by the corresponding strategies.

Type: Types for terms need to be derived and depending on the underlying type system, this can be a non-trivial task, especially when instantiations for polymorphic types have to be found. Thus, we add type-inference as extra *Type*-lemmata.

Case: Case distinction requires to establish the validity of the disjunction locally with respect to the actual context. These proof obligations are represented by *Case*-lemmata.

Subst: These lemmata ensure the admissibility of a substitution.

To prove the lemmata we not only require that *some* proof can be found, but rather we must be provided with a proof object. Such a proof must use the references possibly given by the student, indicated for instance by $P(R^*)$ in the Fact rule, since we are not only interested in the new fact being a valid derivation, but also that the student indicated the *right* references. If we fail to find such a proof but find another proof then we can give the tutor component the feedback that the fact is actually valid, but the student provided some inaccurate details. Any further fact or goal from Γ and Δ used in the proof and not indicated by the student are hints that enable the completion of under-specified information.

Example 5.2 We illustrate the proof checking rules by applying them to the proof presented in Example 5.1. Assume DM_1 and DM_2 are the formulae of the deMorgan laws 1 and 2 respectively. Thus, our initial hypotheses list is

$$\begin{aligned}\Gamma_I := \ & \mathsf{deMorgan\text{-}Rule\text{-}1} : DM_1, \mathsf{deMorgan\text{-}Rule\text{-}2} : DM_2, \\ & \texttt{const}\ A,B,C,D : \mathsf{Set}, \texttt{const}\ \cup,\cap : \mathsf{Set}\times\mathsf{Set}\to\mathsf{Set}, \\ & \texttt{const}\ K : \mathsf{Set}\to\mathsf{Set}\end{aligned}$$

and the initial goal list is

$$\begin{aligned}\Delta_I := \mathsf{theorem} : \ & K((A\cup B)\cap(C\cup D)) = \\ & (K(A)\cap K(B))\cup(K(C)\cap K(D)).\end{aligned}$$

For sake of readability we do not present the complete derivation, but only the lemmata arising during proof checking. For the first `Fact`-statement we obtain the *Fact*-lemma

$$\begin{aligned}(1)\qquad & P_1((\mathsf{deMorgan\text{-}Rule\text{-}2},.,.)) : \Gamma_I \\ & \Longrightarrow_{\mathsf{Fact}}\ . : K((A\cup B)\cap(C\cup D)) = \\ & \quad (K(A)\cap K(B))\cup(K(C)\cap K(D)), \Delta_I\end{aligned}$$

Let F_1 be $K((A\cup B)\cap(C\cup D)) = (K(A)\cap K(B))\cup(K(C)\cap K(D))$. Then for the second `Fact`-step we obtain the *Fact*-lemma

$$\begin{aligned}(2)\qquad & P_2((\mathsf{deMorgan\text{-}Rule\text{-}1},.,.)) : \Gamma_I, . : F_1 \\ & \Longrightarrow_{\mathsf{Fact}}\ . : K(A\cup B) = K(A)\cap K(B), \Delta_I\end{aligned}$$

Analogously we obtain for the third `Fact`-proof step the lemma

$$
\begin{aligned}
(3) \qquad & P_3((\mathsf{deMorgan\text{-}Rule\text{-}1}, ., .)) : \Gamma_I, . : F_1, \\
& . : K(A \cup B) = K(A) \cap K(B) \\
& \quad \Longrightarrow_{\mathsf{Fact}} . : K(C \cup D) = K(C) \cap K(D), \Delta_I
\end{aligned}
$$

For the last `Fact`-proof step we then obtain

$$
\begin{aligned}
(4) \qquad & P_4 : \Gamma_I, . : F_1, . : K(A \cup B) = K(A) \cap K(B), \\
& . : K(C \cup D) = K(C) \cap K(D) \\
& \quad \Longrightarrow_{\mathsf{Fact}} . : K((A \cup B) \cap (C \cup D)) = \\
& \qquad (K(A) \cap K(B)) \cup (K(C) \cap K(D)), \Delta_I
\end{aligned}
$$

And finally for the `Trivial`-proof step we obtain the *Triv*-lemma

$$
\begin{aligned}
(5) \qquad & P_5 : \Gamma_I, . : F_1, . : K(A \cup B) = K(A) \cap K(B), \\
& . : K(C \cup D) = K(C) \cap K(D), \\
& . : K((A \cup B) \cap (C \cup D)) = (K(A) \cap K(B)) \cup (K(C) \cap K(D)) \\
& \quad \Longrightarrow_{\mathsf{Triv}} \Delta_I
\end{aligned}
$$

□

Soundness and Completeness

The proof checking rules defined for the proof representation language are mostly based on a cut rule for the underlying logic. More specifically, the rules Fact, Subgoals, Assume, Assign-Subst and Cases are all based on the cut rule. Note that doing so for Assign-Subst the necessary admissibility check for that substitution is also deferred to the underlying calculus. The rule Triv is simply a call to the underlying calculus, while the rule Assign-Abbrv is a straightforward context checking rule. Thus, the proof checking rules are sound relative to the soundness of the underlying calculus. Due to the presence of the rule for `Trivial` statements, the calculus is complete in theory if the underlying logic has a complete calculus. However, this theoretical observation is only of limited interest in our context, since we envision the use of specific strategies to establish the respective lemmata. Thereby it certainly is not the case that these strategies are complete in the logical sense. A more worthwhile question is whether the set of strategies used for each individual kind of lemma implements a complete strategy. Or, formulating the question slightly differently, what are sufficient conditions to be fulfilled by each individual strategy in order to obtain a complete strategy? The answer to this question remains a challenging research problem.

5.3 Interaction between Proof Manager and Dialogue Manager

From Dialogue Manager to Proof Manager

The dialogue manager has different information to communicate to the proof manager: First, it communicates the actual proof or individual proof steps by indicating possibly to which branch they belong. Secondly, exploiting some user model containing information about the strengths of the student for individual tasks, it can select the appropriate strategies to prove the different kinds of lemmata. For instance, if we have a weak student, then to prove *Triv*-lemmata $\Gamma \Longrightarrow_{\mathsf{Triv}} \Delta$ it may select a strategy that only checks whether some of the goal formulae in Δ occurs in Γ. If the student is stronger, then it may use a slightly stronger strategy, such as for instance the *finishing* strategy from [1].

From Proof Manager to Dialogue Manager

The proof manager works on an actual, possibly partial proof and is configured with a set of strategies to use for the individual arising lemmata. The attempts to prove a lemma may have different results:

- If the respective strategy fails to prove the lemma, then the proof manager returns the information about which step could not be proved to the dialogue manager.
- If the strategy proves the lemma and finds a proof object containing all references indicated by the student, then it informs the tutor about the validity of the student's proof step, and possibly returns facts also required in the proof and not indicated by the student.
- If the strategy proves the lemma, but fails to find a proof object containing all the references indicated by the student, for instance only the references $R_1, \ldots, R_k$, it returns the information that although the proof step is valid, it is unclear why one should use $R_1, \ldots, R_k$. Furthermore, it also informs the tutor about other facts used in the proof and not stated by the student.

Finally, the proof checking algorithm may be able to rule out some impossible alternatives given in an `Or`-proof step. This information can be returned to the dialogue manager in order to sharpen the linguistic analysis.

5.4 Consequences for the Strategies (Strategic Reasoners)

In our context the search behaviour of strategic reasoners should be externally controllable by tutorial knowledge. Moreover, strategic reasoners need to be able to return proof objects, or at least information about used facts in some subproof. Furthermore, we are not interested in finding *some* proof, but rather in finding different possible proofs in the (restricted) search space. Strategies that create proofs coming close to the level of the arguments of the students are more useful in our

context than those creating low-level machine-oriented proofs. This is particularly important for investigating relevance of a proof step.

6 Related Work

The proof representation language presented in this paper is inspired by the work of Abel and colleagues [1]; the aim of their assertion-level proof language is to support the teaching of constructive first-order proofs. However, both proof representation languages differ in various aspects due to differences in their application objectives: While our objective is to reconstruct a proof description from unrestricted natural language tutorial dialogues, Abel and colleagues were concerned with providing a proof representation language directly used by the students to write the proofs. Furthermore, our aim is to define a proof representation that is as far as possible independent of some specific logic while Abel and colleagues' proof representation was designed for first-order constructive logic only. These differences are reflected in two major aspects between both proof representations and both proof checking systems:

Under-Specification: In [1] any used fact that is required for the derivation of new facts needs to be indicated by the student, while this information is typically missing or incomplete in our application scenario. Hence, our rule for the derivation of facts does not necessarily require this information, and we have to encode this step as a subgoal, where in the worst case any information about how to prove it is missing. In Abel and colleagues' approach respective information on used facts is mandatory and exploited to define a fixed specific strategy for checking the application step.

Adaptation: Our proof checking system is explicitly parameterised over the different strategies in order to support the adaptation of the proof manager to some specific logic on the one hand and to the student's skills on the other hand. Abel and colleagues' proof checking algorithm is fixed for some logic and cannot be adapted to, for example, the student's skills. In a tutorial setting like ours, however, the ability of a framework to adapt to the student's skills and the chosen tutorial goals is crucial.

As further related work, Weak Type Theory as employed in the MathLang project [14] is to be mentioned. The MathLang framework is based on de Bruijn's Mathematical Vernacular (MV) [7] but imposes fewer logical constraints. Several other proof languages aiming for better human-oriented support in interactive proof construction have been developed and we mention some of them: the declarative proof language of MIZAR [4], the island proof sketches in OMEGA [18], the Math-

[4] www.mizar.org

ematical Proof Language of Barendregt [3], and Isabelle's ISAR language [16]. However, so far none of these approaches explicitly addresses the phenomena of under-specification and user adaptation as discussed above.

7 Conclusion

In the DIALOG project we investigate tutorial natural language dialogue on mathematical proofs. In this setting under-specification, respectively information (in)completeness, which is a well known problem in natural language analysis, is passed on to the domain reasoning component. In addition, the problem is interwoven with tutorial aspects. Further challenges in the project that are interrelated with tutorial aspects and domain reasoning concern the acceptable step size, the accuracy, and the relevance of uttered proof steps. These challenging problems seem not to be addressed in the literature and are not sufficiently solved yet.

In this paper we presented a first important step toward their solution by introducing a proof representation language and proof checking system for under-specified interactive proofs motivated by the phenomena in our corpus. In this framework local lemmata are generated during the checking of the content of an utterance (a proof step in our scenario). These lemmata reflect the implicit reasoning part underlying the content of the utterance and the proof situation is transformed in a successor state. The lemmata are tackled by domain-specific strategies guided by tutorial aspects in order to clarify whether a proof state transformation is acceptable or not.

We are currently implementing the described approach in a demonstrator system in order to show its feasibility.

We plan to investigate and realise specific strategies based on the proof planning paradigm and also to further investigate the 'proof step relevance' challenge. We also plan to investigate the coverage and appropriateness of our language with respect to further corpora we plan to collect by experiments in other, more complex, mathematical domains.

When we are interested in the design of a tutoring system, we not only have to efficiently prove valid proof steps but also to efficiently disprove invalid proof steps. Therefore we consider it worthwhile to integrate techniques to disprove false conjectures, such as those presented in [2], into the various strategies.

References

[1] Andreas Abel, Bor-Yuh Evan Chang, and Frank Pfenning. Human-readable machine-verifiable proofs for teaching constructive logic. In Uwe Egly, Armin Fiedler, Helmut Horacek, and Stephan Schmitt, editors, *Proceedings of the Workshop on Proof Transformations, Proof Presentations and Complexity of Proofs (PTP'01)*. Universitá degli studi di Siena, June 2001.

[2] Serge Autexier and Carsten Schürmann. Disproving false conjectures. In Moshe Y. Vardi and Andrei Voronkov, editors, *Proceedings of the 10th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, volume 2850 of *LNAI*, pages 33–48. Springer, September 2003.

[3] Henk Barendregt. *Towards an Interactive Mathematical Proof Mode*, pages 27–36. In Kamareddine [13], 2003.

[4] Christoph Benzmüller, 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 Vincent Aleven, Ulrich Hoppe, Judy Kay, Riichiro Mizoguchi, Helen Pain, Felisa Verdejo, and Kalina Yacef, editors, *AIED2003 Supplementary Proceedings*, volume VIII: Advanced Technologies for Mathematics Education, pages 471–481, Sydney, Australia, 2003. School of Information Technologies, University of Sydney.

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

[6] N. Dahlbäck, A. Jönsson, and L. Ahrenberg. Wizard of Oz Studies — Why and How. *Knowledge-Based Systems*, 6(4):258–266, 1993.

[7] N.G. de Bruijn. *The mathematical vernacular, a language for mathematics and typed sets*, pages 865–936. In Nederpelt et al. [15], 1994.

[8] Armin Fiedler and Malte Gabsdil. Supporting pogressive refinement of Wizard-of-Oz experiments. In Carolyn Penstein Rosé and Vincent Aleven, editors, *Proceedings of the ITS 2002 — Workshop on Empirical Methods for Tutorial Dialogue Systems*, pages 62–69, San Sebastián, Spain, 2002.

[9] Armin Fiedler and Dimitra Tsovaltzi. An approach to automating hinting in an intelligent tutorial system. In *Proceedings of the IJCAI Workshop on Knowledge Representation and Automated Reasoning for E-Learning Systems*, Acapulco, 2003. In press.

[10] N. M. Fraser and G. N. Gilbert. Simulating speech systems. *Computer Speech and Language*, 5:81–99, 1991.

[11] Xiaorong Huang. Reconstructing proofs at the assertion level. In Alan Bundy, editor, *Proceedings of the 12th Conference on Automated Deduction*, number 814 in LNAI, pages 738–752. Springer Verlag, 1994.

[12] Philip Johnson-Laird and Ruth Byrne. *Deduction*. Ablex Publishing, 1990.

[13] Fairouz Kamareddine, editor. *Thirty Five Years of Automating Mathematics*. Kluwer Applied Logic series. Kluwer Academic Publishers, 2003.

[14] Fairouz Kamareddine and Rob Nederpelt. A refinement of de Bruijn's formal language of mathematics. *Under revision for the Journal of Logic, Language and Information*, 2004.

[15] R.P. Nederpelt, J.H. Geuvers, and R. de Vrier, editors. *Selected papers on Automath*. North-Holland, Elsevier, 1994.

[16] Tobias Nipkow. Structured Proofs in Isar/HOL. In H. Geuvers and F. Wiedijk, editors, *Types for Proofs and Programs (TYPES 2002)*, volume 2646 of *LNCS*, pages 259–278. Springer, 2003.

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

[18] Jörg Siekmann, Christoph Benzmüller, Armin Fiedler, Andreas Meier, Immanuel Normann, and Martin Pollet. *Proof Development in OMEGA: The Irrationality of Square Root of 2*, pages 271–314. In Kamareddine [13], 2003.

[19] Laurant Théry. Formal proof authoring: An experiment. In *Proceedings of the Workshop User Interfaces for Theorem Provers (UITP 2003)*, pages 143–160, Rome, Italy, 2003. MMIII ARACNE EDITRICE S.R.L. (ISBN 88-7999-545-6). Also available as: Technical Report No. 189, Institut für Informatik, Albert-Ludwig-Universität, Freiburg.

[20] Manfred Thüring and Kurt Wender. Über kausale Inferenzen beim Lesen. *Sprache und Kognition*, 2:76–86, 1985.

[21] Dimitra Tsovaltzi and Armin Fiedler. An approach to facilitating reflection in a mathematics tutoring system. In Vincent Aleven, Ulrich Hoppe, Judy Kay, Riichiro Mizoguchi, Helen Pain, Felisa Verdejo, and Kalina Yacef, editors, *AIED2003 Supplementary Proceedings*, volume V: Learner Modelling for Reflection, pages 278–287, Sydney, Australia, 2003. School of Information Technologies, University of Sydney.

[22] Marylin Walker. The effect of resource limits and task complexity on collaborative planning in dialogue. *Artificial Intelligence*, 85:181–243, 1996.