

Checking textbook proofs

Claus Zinn

Lehrstuhl für Künstliche Intelligenz (IMMD VIII)
Universität Erlangen-Nürnberg
Am Weichelgarten 9, 91058 Erlangen, Germany
`zinn@informatik.uni-erlangen.de`

Abstract. Our long-range goal is to implement a program for the machine verification of textbook proofs. We study the task from both the linguistics and deduction perspective and give an in-depth analysis for a sample textbook proof. A three phase model for proof understanding is developed: parsing, structuring and refining. It shows that the combined application of techniques from both NLP and AR is quite successful. Moreover, it allows to uncover interesting insights that might initiate progress in both AI disciplines.

Keywords: automated reasoning, natural language processing, discourse analysis

1 Introduction

In [12], John McCarthy notes that *“Checking mathematical proofs is potentially one of the most interesting and useful applications of automatic computers”*. In the first half of the 1960s, one of his students, namely Paul Abrahams, implemented a Lisp program for the machine verification of mathematical proofs [1]. The program, named **Proofchecker**, *“was primarily directed towards the verification of textbook proofs, i.e., proofs resembling those that normally appear in mathematical textbooks and journals”*. Abrahams soon revised his goal. If, so Abrahams, *“a computer were to check a textbook proof verbatim, it would require far more intelligence than is possible with the current state of the programming art”*. Therefore, so Abrahams, *“the user must create a rigorous, i.e., completely formalized, proof that he believes represents the intent of the author of the textbook proof, and use the computer to check this rigorous proof”*. Abrahams points further out that *“it is a trivial task to program a computer to check a rigorous proof; however, it is not a trivial task to create such a proof from a textbook proof”*. Abrahams was right. In his implementation and in all later projects, proofs had to be written in a *formal language* using a *restricted set of proof construction commands* in order to verify them. A human user is required to fulfill the formalization task. Basically, two main approaches towards the formalization and verification of proofs were taken.

In the first approach, the Automath/Mizar approach [14, 15], the user is required to give a *full* and *explicit* construction of a proof. The proof component

then checks the proof for correctness (the compiler approach). A well-known example of a larger formalization task for *Automath* is van Benthem-Jutting's translation of Landau's 'Grundlagen der Analysis' into *aut-qe*, one of the formalisms of the *Automath*-language family [19]. The *Mizar* system offers, beside the rich formal language and the proof checker component, a large library of formalized mathematics which allows to start proving theorems without excessive preparatory work.

In the second approach, the user constructs interactively with the proof system the proof (the interpreter approach). The user has a set of proof construction commands at hand and asks the system to apply them. The system holds track of the proof obligations and guarantees that the so constructed proof is correct. Well-known systems are *Nuprl* [8] and *Coq* [9].

Goal. It is time to work out a third approach, the machine verification of textbook proofs — without human interaction that translates them into a formal language. Our goal is to implement a program for the machine verification of textbook proofs. More generally, this program reads textbook proofs and is able to communicate its knowledge about what it has read. This proof understander is able to recognize the proof structure as well as obvious definitorial and logical dependencies. It answers questions about the proof accurately and is capable to identify gaps or flaws in the argumentation line. The proof system offers a high-level analysis of the proof as well as a technical low-level access to details of the proof. We will look at this goal from two different perspectives.

The linguistic view. Verifying textbook proofs is a text understanding task. The expert language used by mathematicians has several characteristics that seem to make this task feasible: its poor vocabulary, the use of standard phrases [18] and keywords that introduce and combine simple sentences, the large use of terms and formulae for abbreviation etc. The art of writing good mathematical texts focuses at clearness and conciseness and not on an embellished style of expression [20]. In addition, textbook proofs are, in general, a highly structured form of discourse. A crucial prerequisite to understand the course of the argumentation within a proof is to identify the discourse relations between sentences of that discourse. Deriving the discourse relations of a given textbook proof means reconstructing the intentional structure (describing how sentences within a discourse segment contribute to a common discourse purpose) and the informational structure (describing how sentences within a segment are related to each other by some relation) [13] of the proof.

Albeit those characteristics, all kinds of linguistic phenomena which can occur in other text-sorts, show also up in textbook proofs [21].

The automated reasoning view. Verifying textbook proofs is a deductive task and means to identify the logical structure of the proof: identifying assumptions and conclusions, the scope and quantification of variables, substructures which itself form subproofs etc. A theorem is always proven in some mathematical theory obeying some proof plan. Often, the form of the theorem presupposes possible

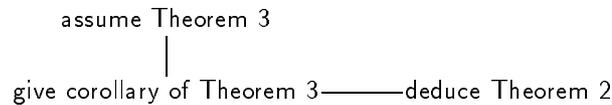
proof plans and the concepts it contains often hint to definitions being used in the proof. Identifying the proof plan of a textbook proof is a prerequisite for verification. It allows to follow as close as possible the proof authors argumentation line.

Three phase model. Therefore, verifying textbook proofs faces three major problems: parsing, structuring and verifying, *all closely interconnected*:

1. Parsing the textbook proof: proceeding incrementally, sentence by sentence. The semantics of the current phrase is determined using the context that has been established by having parsed the former sentences.
2. Recognizing the proof structure: doing high-level proof analysis, structuring the internal representation by attaching a proof plan to it. The resulting object, the *proof sketch*, does not only reveal the proof structure but also logical dependencies between parts of the structure. Proof gaps and minor flaws, common in textbook proofs, are detected and repaired.
3. Refining the proof sketch: bridging the large gap between a high-level proof to a formal proof. The proof sketch is expanded into a formal proof by refining mathematical arguments to low-level inference steps.

2 Example: Proof analysis

Fig. 1 depicts the fundamental theorem of arithmetic and its proof (taken from [10]). We have set the ‘meta-information’, which is necessary for the understanding of the proof, in **typewriter** tiny-size. The proof outline for Theorem 2 is then as follows:



In the sequel, we will analyze the proof. We will see that (i) although mathematicians try to communicate knowledge in a precise manner in order to exclude ambiguities and multiple interpretations, and (ii) although human readers might find that the mathematical argument in Fig. 1 is presented in a precise way, a machine encounters numerous problems. First, we will discuss some imprecise formulations.

The corollary should be formulated more precisely:

COROLLARY-1 OF THEOREM 3:
If p is prime, and $p \mid abc \dots l \rightarrow p \mid a$ or $p \mid b$ or $p \mid c \dots$ or $p \mid l$.

It is not clear if the phrase starting with *in particular* belongs to the corollary. It should better be stated as a corollary itself:

COROLLARY-2 OF THEOREM 3/COROLLARY-1:
If p is prime, and $a, b, c \dots, l$ are prime, and $p \mid abc \dots l$, then $p = a$ or $p = b$ or $p = c \dots$ or $p = l$.

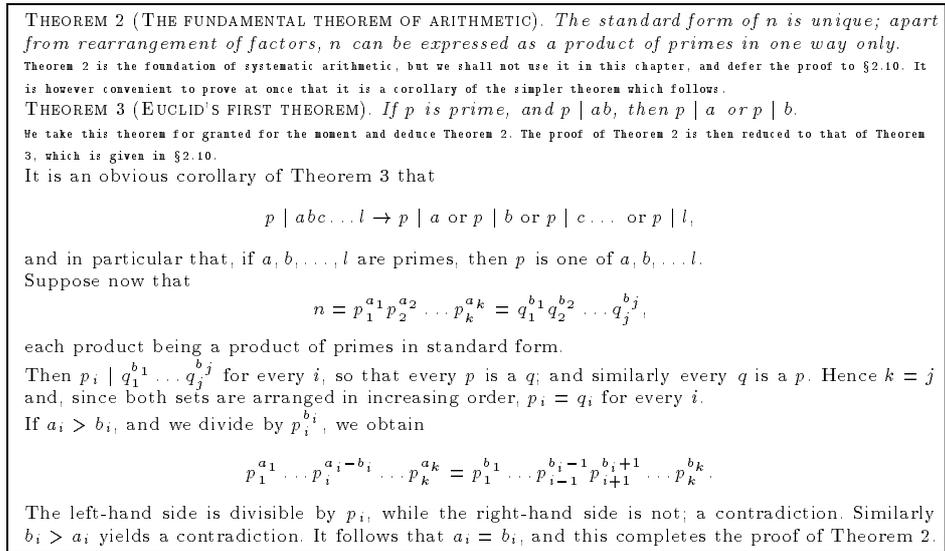


Fig. 1. The fundamental theorem of arithmetic

Note that in both corollaries the variable p as well as each of the variables a, b, c, \dots, l is universally quantified. Hence, for Corollary-2 we obtain:

$$\forall p : \forall a \dots \forall l : pr(a) \wedge pr(b) \wedge \dots \wedge pr(l) \wedge p \mid abc \dots l \rightarrow p = a \vee p = b \vee p = c \vee \dots \vee p = l.$$

After outlining the kind of reduction used and the theorems and corollaries necessary to perform it, the main proof starts with **suppose now that**. We propose the proof structure depicted in Fig. 2. Before we explain how this structure can be obtained mechanically, we make some comments on the proof. We start analyzing from the deductive view.

The theorem, *the standard form of n is unique*, indicates that a proof with a uniqueness method can be used. The definition of *standard form* is as follows:

If we arrange them [i.e., the primes in $n = p_1 p_2 \dots p_k$] in increasing order, associate sets of equal primes into single factors, and change the notation appropriately, we obtain

$$n = p_1^{a_1} p_2^{a_2} \dots p_k^{a_k} \quad (a_1 > 0, a_2 > 0, \dots, p_1 < p_2 < \dots).$$

We then say that n is expressed in standard form.

So, in **[1]**, the idea is: assume that there are two objects being both a product of primes for some number n and show that these two objects are equal. To show that these two objects are in fact equal, domain knowledge about term equality has to be applied. at the beginning, the proof obligations are:

1. show that $k = j$.

Suppose now that	
	$n = p_1^{a_1} p_2^{a_2} \dots p_k^{a_k} = q_1^{b_1} q_2^{b_2} \dots q_j^{b_j},$
each product being a product of primes in standard form.	1
Then $p_i \mid q_1^{b_1} \dots q_j^{b_j}$ for every i	2a
so that every p is a q_i	2b
and similarly	3a
every q is a p .	3b
Hence $k = j$	4
and, since both sets are arranged in increasing order	5a
$p_i = q_i$ for every i .	5b
If $a_i > b_i$	6a
and we divide by $p_i^{b_i}$	6b
we obtain $p_1^{a_1} \dots p_i^{a_i - b_i} \dots p_k^{a_k} = p_1^{b_1} \dots p_{i-1}^{b_{i-1}} p_{i+1}^{b_{i+1}} \dots p_k^{b_k}$.	6c
The left-hand side is divisible by p_i	6d
while the right-hand side is not	6e
a contradiction.	6f
Similarly	7a
$b_i > a_i$ yields a contradiction.	7b
It follows that $a_i = b_i$	8a
and this completes the proof of Theorem 2.	8b

Fig. 2. Structuring the proof

2. show that $p_i = q_i$ for every i .
3. show that $a_i = b_i$ for every i .

In **2a** — **5b** it is shown that the number of bases p_i and q_j are equal, and that $p_i = q_i$ for every i . This is shown by the *direct uniqueness method* [17]. In **6a** — **8a** it is shown that the exponents are equal, too. This is shown by the *indirect uniqueness method*. The proposition in **2a** is obtained by forward reasoning from the assumption. The conclusion in **2b** has been made using Corollary-2: $\forall i, 1 \leq i \leq k: p_i$ is prime, $q_1 \dots q_j$ are prime, $p_i \mid q_1^{b_1} \dots q_j^{b_j}$ implies $p_i = q_1^{b_1} \vee \dots \vee p_i = q_j^{b_j}$ using the fact that $p \mid q^b \rightarrow p \mid q$. The conclusion in **4** is due to **2b** and **3b** and the definition of *standard form*. The proposition in **5a** is part of the definition of *standard form*. From **6a** for every i is lacking. A central idea of the proof is **6b** and **6c**. The conclusion **8a** follows from $(\neg(a_i > b_i)) \wedge (\neg(b_i > a_i))$. **8b** states that the list of proof obligations is empty.

Now, we make some comments from a linguistic point of view. The proof contains a couple of referential expressions to be resolved, e.g., *each product*, *both sets* and *the left/right-hand side*. Also, the indefinite NP *a contradiction* has to be treated anaphorically. Elliptic constructions can be found, too. The *and*

similarly in [3a] means: *Then $q_i \mid p_1^{a_1} \dots p_k^{a_k}$ for every i .* The similarly in [7a] indicates that a parallel construction (to [6b]–[6e]) has to be performed. The text in [6b] and [6c], describing a rewriting operation, is also elliptic and involves a kind of state change anaphora.

The proof contains also several discourse markers: in [1], the clue word **now** indicates that some preparatory work is finished; **suppose** indicates an assumption to be stated; in [6e], **while** indicates that the sentence following contrasts to some former statement (which is to be identified).

From our analysis, it can be seen that proof authors do not often explicitly mention which premises lead to which conclusion by which rule of inference. The proof method is not mentioned either. The (human/machine) proof reader has to cope with a variety of different linguistic phenomena.

3 Implementation

3.1 Parsing

We have written a DCG grammar using the λ -DRT formalism [11] (and having adapted program code from [3]) that covers the first proofs of [10]. Example DRT encodings for terms and formulae, the lexical entries for a constant (2), a variable (n), a binary function (*expt*), and a predicate ($=$), are shown in Fig. 3.

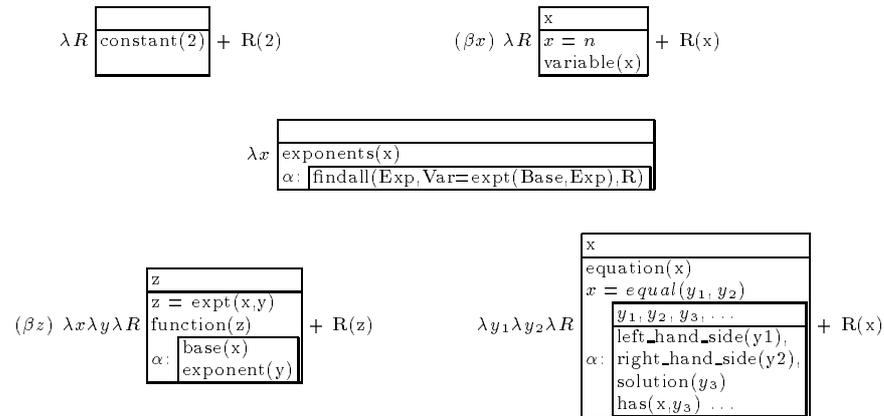


Fig. 3. Some lexical entries for terms and formulae.

Constants do not introduce discourse referents. For the first occurrence of a variable, say n , in a theorem-proof segment, we (i) introduce a discourse referent, say X , and (ii) introduce the conditions: *variable(X)* and $X = n$. For all other occurrences of n , we look for its anaphoric antecedent, that is a discourse referent,

say Y , for which the conditions $variable(Y)$, and $Y = n$ hold. The handling of functional terms is similar: for each functional term that is new to the context, we introduce a discourse referent which refers to that term. Parsing $f(a) + f(a)$ will result to

$$\lambda R \frac{\boxed{\begin{array}{l} X, Y, Z \\ X = plus(Y, Y), Y = f(Z), Z = a, \\ variable(Z), function(X) \end{array}}}{+ R(X)}.$$

In order to have proper semantics for phrases like *every p is prime* or *every p is a q*, we adapted a couple of the DRS construction rules as given in [11].

If a definite description cannot be linked to a suitable antecedent which was introduced earlier into the discourse context, we employ a strategy called *bridging*. Fig. 3 depicts the lexicon entry for $=$ which enables us to resolve anaphora like *the left-hand side* and *the right-hand side*. The lexicon entry for *exponents* is enriched with a computational component, a Prolog goal. This component is activated during anaphora resolution. It is evaluated in an environment that is defined by the accessible part of the DRS. The accessible environment is seen as a database of Horn clauses.

3.2 Structuring

It is common knowledge that a good model of discourse segmentation is a prerequisite for handling linguistic phenomena: segments serve as a local context for the interpretation of anaphora and enables us to handle elliptic expressions (like the similarly constructs in [3a](#) and [7a](#)).

Linguistic clues. We now give some heuristics to identify segmental boundaries:

- An assumption, indicated by clue words like *assume* and *suppose* defines a segment.
- The discourse marker *similarly* introduces an elliptic phrase indicating that a parallel construction to a former segment has to be performed. Therefore, this linguistic clue marks the beginning of a new segment (closing the old one).
- *hence* often indicates both a logical relation and a summary construction. In the latter case, it defines a new segment.
- Fulfilling a main proof obligation means terminating a segment (explaining the horizontal lines between [4](#) and [5a](#), [5b](#) and [6a](#)).
- Cue phrases like *a contradiction*, *we are through*, and *this completes the proof* define the end of the current segment.

In general, linguistic clues do not suffice to determine the logical structure of the proof. A simple example is the mathematical argument being depicted below (it proves Theorem 3 which is necessary to complete the proof for Theorem 2):

<p>We are now in a position to Euclid's theorem 3, and so Theorem 2. Suppose that p is prime and $p \mid ab$. If $p \nmid a$ then $(a, p) = 1$, and therefore, by Theorem 24, there are an x and a y for which $xa + yp = 1$ or</p> $xab + ypb = b.$ <p>But $p \mid ab$ and $p \mid ypb$, and therefore $p \mid b$.</p>
--

Proof plans. Often, the form of the theorem presupposes possible proof plans. In the case of Theorem 3, two possible proof plans are applicable (Fig. 4). To



Fig. 4. Two applicable plans for proof per elimination

show that a implies b_1 or b_2 , a proof by elimination can be tried. For example:

1. Assume a and not b_1 being true.
2. Work forward from a and not b_1 to establish the truth of b_2 .
3. Work backward from b_2 .

If the form of the theorem is of little help, clue words like *unique*, presuppose possible proof plans. This has been demonstrated with our analysis of the proof in Fig. 1. Also, domain knowledge of term equality has been necessary to generate a possible argumentation line.

3.3 Representational issues

Discourse representation structures (DRS) play a central role in our proof system. First, the semantics construction (for a large part) done in the parser component returns DRSs. However, to resolve anaphora and ellipsis, as we argued before, we make use of discourse/proof plans. In addition, proof plans itself can introduce discourse referents!

For example, if a textbook proof contains the definite noun phrase *the induction hypothesis implies that ...*, we link *the induction hypothesis* to an *abstract discourse object* [2] which refers to a segment in a corresponding proof plan for induction proofs. Consequently, proof plans are represented as *augmented DRS*. That is, we postulate a new level, the propositional level, on top of DRSs. We name the resulting structure *proof representation structure (PRS)*.

In Fig. 5, we depicted two PRSs, each of which represents a plan for a proof per elimination. Informally, analogous to a discourse representation structure,

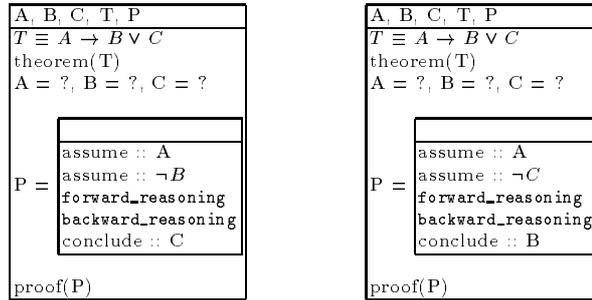


Fig. 5. Two applicable PRS (proofs per elimination)

a PRS consists of a set of referents and a set of conditions. The referents refer either to a DRS, another referent (anaphoric linkage) or to a PRS substructure.

Fig. 6 depicts the intended mapping of proof plan and mathematical argument. The mapping can be described as follows: We break the complex sentences

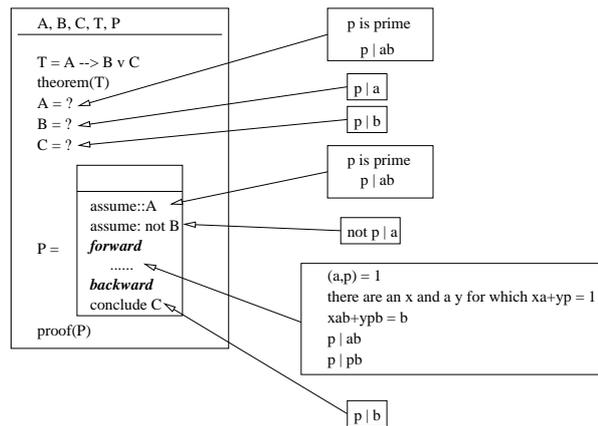


Fig. 6. Mapping proof text to proof plan

of the input proof into elementary assertions, for each of which an abstract discourse referent is introduced:

$$D_1 = \begin{array}{|l} \hline X \\ \hline X=p \\ \text{variable}(p) \\ \text{prime}(p) \\ \hline \end{array}$$

Note that in Fig. 6, we omitted these propositional discourse referents. Discourse markers that connected these elementary assertions (e.g., and, therefore, assume) are maintained (attached to them) for further use.

Now, we view each of the elementary discourse constituents as anaphoric referring to its place in the proof plan. The discourse markers, we kept them, define constraints for possible sites of constituent attachment. For example, the cue word **and therefore** indicates that the constituent it introduces, should be attached to the currently “active” structure.

Discourse markers like **either** or **otherwise** indicate that a new subproof has to be opened and attached. The constituent **we are through** marks the end of a sub-proof so that consecutive phrases can no longer be attached to this part of the proof which become marked as “closed”.

3.4 Proof refinement

Having identified the proof structure, it is now possible to better treat anaphora and ellipsis. Referential expressions should now be replaced by the objects they refer to and elliptic constructs (like to similarly in our example proof) should be reconstructed. This is to make the textbook proof more rigorous because it eliminates ambiguities. The result of these refinements for our running example is depicted in Fig. 7. To get a formally checkable proof, proof refinement involves

Suppose $n = p_1^{a_1} p_2^{a_2} \dots p_k^{a_k}$ where $p_1^{a_1} p_2^{a_2} \dots p_k^{a_k}$ is a product of primes in standard form	1a
Suppose $n = q_1^{b_1} q_2^{b_2} \dots q_j^{b_j}$ where $q_1^{b_1} q_2^{b_2} \dots q_j^{b_j}$ is a product of primes in standard form	1b
Then $p_i \mid q_1^{b_1} \dots q_j^{b_j}$ for every $i, 1 < i < k$	2a
so that for every $p \in \{p_1, p_2, \dots, p_k\} \exists q \in \{q_1, \dots, q_j\} : p = q$	2b
Then $q_i \mid p_1^{a_1} p_2^{a_2} \dots p_k^{a_k}$ for every $i, 1 < i < j$	3a
so that for every $q \in \{q_1, q_2, \dots, q_j\} \exists p \in \{p_1, \dots, p_k\} : q = p$	3b
Hence $k = j$	4
and, since $p_1 < p_2 < \dots < p_k, q_1 < \dots < q_j$	5a
$p_i = q_i$ for every $i, 1 < i < k$	5b
If $a_i > b_i$ (free occurrences of i)	6a
and we divide $p_1^{a_1} p_2^{a_2} \dots p_k^{a_k} = p_1^{b_1} p_2^{b_2} \dots p_k^{b_k}$ by $p_i^{b_1}$	6b
we obtain $p_1^{a_1} \dots p_i^{a_i - b_i} \dots p_k^{a_k} = p_1^{b_1} \dots p_{i-1}^{b_{i-1}} p_{i+1}^{b_{i+1}} \dots p_k^{b_k}$	6c
$p_1^{a_1} \dots p_i^{a_i - b_i} \dots p_k^{a_k}$ is divisible by p_i	6d
while $p_1^{b_1} \dots p_{i-1}^{b_{i-1}} p_{i+1}^{b_{i+1}} \dots p_k^{b_k}$ is not divisible by p_i	6e
a contradiction	6f
If $b_i > a_i$	7a
and we divide $p_1^{a_1} p_2^{a_2} \dots p_k^{a_k} = p_1^{b_1} p_2^{b_2} \dots p_k^{b_k}$ by $p_i^{b_i}$	7b
we obtain $p_1^{a_1} \dots p_{i-1}^{a_{i-1}} p_{i+1}^{a_{i+1}} \dots p_k^{a_k} = p_1^{b_1} \dots p_i^{b_i - a_i} \dots p_k^{b_k}$	7c
$p_1^{a_1} \dots p_{i-1}^{a_{i-1}} p_{i+1}^{a_{i+1}} \dots p_k^{a_k}$ is not divisible by p_i	7d
while $p_1^{b_1} \dots p_i^{b_i - a_i} \dots p_k^{b_k}$ is divisible by p_i	7e
a contradiction	7f
it follows that $a_i = b_i$ for every $i, 1 < i < k$	8a
QED	8b

Fig. 7. Refining the proof

to refine the high-level arguments encountered in the proof to low-level inferences of some logical calculi. Again, we use the discourse markers.

A discourse marker like **hence** or **therefore** does not only define attachment constraints. It also defines a discourse relation and we view it as an anaphoric entity that refers to all necessary premises which are logically necessary for drawing the conclusion it introduces. It is tried to identify these premises. Note that, due to a wrong attachment, we might not be able to identify these premises (e.g, an attachment to a wrong structure might alter the set of accessible premises considerably).

For some steps in the proof, identifying the premises is an easy task. Obviously, this is the case when mathematical reasoning is close to formal reasoning. In general, however, external tools, a theorem prover and a computer algebra system, have to be employed.

To verify that a sentence, say $S1$, is a logical consequence of some context, say $C1$, we have to first translate the DRSs for $S1$ and $C1$ into first order logic formulae yielding $S1^{fol}$ and $C1^{fol}$ (cf. [11] for a translation algorithm).

For example, take the proof of Theorem 3 at the point where the forward reasoning from the assumptions start: ... then (a,p)=1. In order to formally verify this step, only two premises are accessible (the only ones stated in the proof). In this simple case, asking the external tools if $C1^{fol}$ logically entails $S1^{fol}$ will fail. The situation of missing premises will often arise, and a main task in proof refinement will therefore be to identify the proof steps which the proof author omitted, and to enrich the context $C1$ with them.

4 Related work

Similar attempts to parse texts in the mathematics and physics domain are [4] and [6]. Similar to the argument in [6], our purpose is to study natural language understanding in conjunction with (automated) mathematical reasoning. What formal representation can be obtained from textbook proofs and what is needed to formally verify textbook proofs?

The automated verification of textbook proofs has long been ignored. Except the work of Simon [16], the author knows of no project that focuses on natural language understanding of textbook proofs. The main difference between our approach and Simon's approach is that he integrated the modules for parsing, proof construction and proof checking into one system. It is therefore difficult for the user to understand which proofs are really checked because it is somehow hidden in Simon's system. We propose an approach which clearly separates different phases, and consider the result of each phase as first class objects in their own right, worth studying. The main critic is, however, that it remains unclear how Simon handle linguistic phenomena – an adequate theory for doing so is not proposed. From our textbook analysis it should be clear that an adequate theory for semantics construction is necessary to cope with the linguistic phenomena one encounters.

Abstract discourse entities have been studied extensively by Asher [2]. Asher differs between fact anaphora, event anaphora, concept anaphora and proposition anaphora. For our domain, only the latter is of interest. The DRS construction process we described begs some similarities to the one described in [2]. A major difference is that, due to our domain, (i) we can assume the existence of discourse plans that establish frames in which discourses must take place; (ii) we have only one major discourse relation, that of logical consequence, and we can establish the relation by using automated theorem provers.

The inverse of our task, translating formal proofs into natural language proofs, is described in [7]. Facing the problem that proofs generated by theorem provers are unstructured, tediously long and therefore unreadable, a readable, structured and short proof (omitting all the low-level details) has to be generated.

Please note that being able to ‘understand’ textbook proofs does not mean to understand the way how mathematicians *reason* when searching for proofs. It allows only an understanding of how mathematicians *communicate* their proofs and how they build one argument on another in order to have a convincing complex argument structure supporting the truth for some assertion. Similar to [5], we argue that Logic is not enough to understand textbook proofs. To verify mathematical proofs, the argumentation line of the proof author has to be followed, which requires a high-level strategic proof understanding. The formal proof does only provide a low-level inference-based view and the informal proof to be verified is far away from that level of detail. Proof plans allow to capture mathematical argumentation techniques.

5 Conclusion

Textbook proofs represent a structured discourse of (the important and interesting) argumentation steps supporting the truth of some assertion, and are therefore an ideal domain for discourse understanding. The domain has a rich set of well-defined mathematical concepts and discourse plans. Central to the issue of automatically checking textbook proofs is to describe the formalization process as translation from informal proofs to formal ones. In the Automath and Mizar projects, the author of the formal proof is considered to perform this translation. In our approach, the computer should perform this task.

We developed a three phase model to textbook proof understanding and verification: parsing, structuring and refining. For representing textbook proofs and proof plans, we proposed to use augmented discourse representation structures. We think that an extended DRT formalism is better suited for the process of formalizing mathematics than earlier languages proposed for this purpose [14, 15, 9] because it allows to handle phenomena that occur in natural language proofs.

Being able to process textbook proofs allows us to close the gap between the language of mathematicians and the language of proofs systems. Reaching

this goal will enable us to build AI systems that really assist mathematicians. The research topic we identified might boost progress in both Natural Language Processing and Automated Reasoning.

References

1. P. W. Abrahams. *Machine verification of mathematical proofs*. PhD thesis, MIT, 1963.
2. N. Asher. *Reference to abstract objects in discourse*. Kluwer Academic Publishers, 1993.
3. P. Blackburn and J. Bos. Representation and inference for natural language. Draft, 1997.
4. D. G. Bobrow. *Natural language input for a computer problem solving system*. PhD thesis, MIT, 1964.
5. A. Bundy. A Science of Reasoning. In H. de Swart, editor, *Automated reasoning with analytic tableaux and related methods, Int'l Conference*, volume 1397. Springer, 1998.
6. A. Bundy, L. Byrd, and G. Luger. Solving mechanics problems using meta-level inference. In *6th. Int'l Joint Conference on Artificial Intelligence*, pages 1017–1027, 1979.
7. D. Chester. The Translation of Formal Proofs into English. *Artificial Intelligence*, 7:261–278, 1976.
8. R. L. Constable. *Implementing Mathematics with the Nuprl Proof Development System*. Prentice Hall, 1986.
9. G. Dowek, A. Felty, H. Herbelin, G. Huet, C. Paulin Mohring, and B. Werner. The Coq proof assistant, version 5.6 user's guide. Technical report, INRIA – Rocquencourt, 1991.
10. G. Hardy and E. Wright. *An introduction to the theory of numbers*. Oxford at the Clarendon Press, 4th. edition, 1971.
11. H. Kamp and U. Reyle. *From Discourse to Logic*, volume 1 and 2. Kluwer, 1993.
12. J. McCarthy. Computer programs for checking mathematical proofs. In *Recursive Function Theory, Proceedings of Symposia in Pure Mathematics*, volume 5. American Mathematical Society, 1962.
13. J.D. Moore and M.E. Pollack. A Problem for RST: The Need for Multi-Level Discourse Analysis. *Computational Linguistics*, 18(4):537–544, 1992.
14. R. P. Nederpelt, J.H. Geuvers, and R.C. de Vrijer, editors. *Selected papers on Automath*, volume 133 of *Studies in Logic and the foundations of Mathematics*. North-Holland, 1994.
15. P. Rudnicki. An overview of the MIZAR project. Technical report, Dept. of Computer Science, University of Alberta, Edmonton, 1992.
16. D. L. Simon. *Checking Number Theory Proofs in Natural Language*. PhD thesis, UT Austin, 1990.
17. D. Solow. *How to read and do proofs*. John Wiley & Sons, 1990.
18. J. Trzeciak. Writing mathematical papers in english. Gdańsk Teacher's Press, Institute of Mathematics, Polish Academy of Science, 1993.
19. L.S. van Benthem Jutting. *Checking Landau's "Grundlagen" in the Automath system*. PhD thesis, TH Eindhoven, 1977.
20. A.J.M. van Gasteren. *On the shape of mathematical arguments*, volume 445 of *Lecture Notes in Computer Science*. Springer, 1990.

21. C. Zinn. Parsing formulae in textbook proofs. In H.C. Bunt and E.G.C. Thijsse, editors, *Proceedings of the Third Int'l Workshop on Computational Semantics (IWCS-3)*, pages 422–424. Tilburg University, 1999.
22. C. Zinn. Understanding Mathematical Discourse. 1999. Amsterdam workshop on the semantics and pragmatics of dialogue. To appear.