Verifying Textbook Proofs

Claus Zinn[*]
Lehrstuhl für Künstliche Intelligenz
Universität Erlangen-Nürnberg

In the first half of the 1960s, 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 did not succeed. 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 formalised, 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 it a trivial task to program a computer to check a rigorous proof; however, it it not a trivial task to create such a proof from a textbook proof"*.

Abrahams was right. In all later projects, proofs had to be written in a formal language in order to verify them. One well-known example is the the Automath project: van Benthem-Jutting formalised a whole textbook of Landau, the 'Grundlagen der Analysis', into a formal language, `aut-qe` [8]. A second example is the Mizar project [5]: proofs have to be written in the Mizar formalism to get them verified.
The machine verification of textbook proofs — without human interaction that translates them into a formal language — has been tackled seriously first by Simon [6]. In the same line is our work.

The automatic verification of mathematical textbook proofs is a complex task. Think about some problems a student might face when trying to understand a mathematical argument: no proof outline is given; the reasoning is incomplete; lemmas necessary for the understanding of the proof are omitted; several steps have been done at once etc. And, of course, often it is the student that has not the required knowledge about the theory behind the theorem to be proven and about common proof techniques.

Our goal is to implement a program for the machine verification of textbook proofs. This program reads textbook proofs and is able to communicate its knowledge about what it has read. This proof understander is able to recognise 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.

Of course, this is a text understanding task. The text, however, is of a very special nature. Because textbook proofs are not written in some logic formalism but in a natural

[*]IMMD-VIII, Am Weichselgarten 9, 91058 Erlangen, Germany, `zinn@informatik.uni-erlangen.de`

language, it is far more difficult to parse them. Nevertheless, the expert language used by mathematicians has several characteristics that seem to make this task feasible: its poor vocabulary, the use of standard phrases and keywords that introduce and combine simple sentences, the large use of terms and formulae for abbreviation etc. In addition, the art of writing good mathematical texts focuses at clearness and conciseness and not on an embellished style of expression. Albeit those characteristics, all kinds of linguistic phenomena which can occur in other text-sorts, show also up in textbook proofs.

Next, textbook proofs are, in general, a highly structured form of discourse. Deriving the discourse relations of a given textbook proof means reconstructing the logical structure of the proof: identifying assumptions and conclusions, the scope and quantification of variables, substructures which itself form subproofs etc.

Verifying textbook proofs faces thus 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. Recognising 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 proof plans to low-level inference steps.

Understanding textbook proofs involves the combination of techniques from both Natural Language Processing and Automated Reasoning. 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. Naturally, during semantics construction, a text proof parser could exploit this logical information employing an external automated theorem prover as a tool to verify intermediate parse results. Does the parsing result of some sentence (i) is in contradiction with some part of the formal theory under study, (ii) contributes something new to the proof, (iii) fits into the currently assumed proof plan?

We are writing a prototypical textbook proof reader to implement our ideas. A linguistic analysis of mathematical texts, focusing on the treatment of terms and formulae, is given in [9]. We propose a DRT-based semantics construction. We have implemented (in Prolog) a sufficiently large lexicon and grammar such that the first proofs of [4] can be parsed. Textbook proofs have been analysed in order to extract their proof plan by hand. For computing structure automatically, we are interested how to identify segmental boundaries in the proof. These proof segments then serve as a local context for the interpretation of referential expressions. The identification and classification of discourse relations has been started using keywords, cue phrases and proof plans. The problem of proof refinement has not yet been tackled. The intermediate results of our system are very encouraging.

References

[1] P. W. Abrahams. *Machine verification of mathematical proofs.* PhD thesis, MIT, 1963.

[2] D. G. Bobrow. *Natural language input for a computer problem solving system.* PhD thesis, MIT, 1964.

[3] A. Bundy, L. Byrd, and G. Luger. Solving mechanics problems using meta-level inference. In *6th. International Joint Conference on Artificial Intelligence*, pages 1017–1027, 1979.

[4] G. Hardy and E. Wright. *An introduction to the theory of numbers.* Oxford at the Clarendon Press, 4th. edition, 1971.

[5] P. Rudnicki. An overview of the MIZAR project. Technical report, Department of Computer Science, University of Alberta, Edmonton, 1992.

[6] D. L. Simon. Checking natural language proofs. In *9th. International Conference on Automated Deduction*, volume 310 of *Lecture Notes in Computer Science.* Springer, 1988.

[7] D. Solow. *How to read and do proofs.* John Wiley & Sons, 1990.

[8] L.S. van Benthem Jutting. *Checking Landau's "Grundlagen" in the Automath system.* PhD thesis, Technische Hogeschool Eindhoven, 1977.

[9] C. Zinn. A DRT-based approach for formula parsing in textbook proofs. In *Third International Workshop on Computational Semantics (IWCS-3)*, Tilburg, 1999. To appear.