Textbook Mathematics in the Naproche-SAD System

Peter Koepke

University of Bonn

Abstract

1 Introduction

To faithfully formalize and automatically proof-check mathematical textbooks has long been an important challenge in formal mathematics. Benthem Jutting formalized Landau's *Grundlagen der Analysis* in the Automath system, the formalization of *A Compendium of Continuous Lattices* was an important project in Mizar, and Lawrence Paulson translated parts of *Set Theory* by Kunen as a basis for his proof of the Relative Consistency of the Axiom of Choice in Isabelle/ZF. Since the input languages of these eminent proof checking systems are akin to computer languages, the formalizations do not resembe textbooks at all.

To formalize mathematics in the style and language of textbooks belongs to the programme of *Natural Formal Mathematics* which may be described like:

> to increase the degree of formality of the mathematical language so that the language itself becomes fully formal with respect to some formal grammar, whilst staying within the common mathematical language and symbolism.

In this contribution I shall report on progress in this direction in the Naproche-SAD project. I shall recall the classical SAD system with its proof language ForTheL, present its further development in the current Naproche-SAD system, and give examples from recent formalizations of textbook material.

2 SAD

The System for Automated Deduction by Andrei Paskevich [XXX] is the culmination of a longterm research effort that goes back to the Evidence Algorithm project (EA) started by Victor Glushkov in the 1960's [XXX]. The project involved many aspects of automated theorem proving and proof checking [historical presentation], centered around the faithful modelling of the formulation and checking of theorems and proofs. After a comprehensive analysis of mathematical texts, the contolled natural language ForTheL (Formula Theory Language) was developed as an input language for a proof assistant [Literatur von Verchinine].

The following is a sample of an SAD formalization of the standard proof of the infinitude of prime numbers, based on some preliminary theory about natural numbers, divisibility, finite sets etc. that we omit for brevity:

```
. . .
Signature. A number is a notion. Let m, p stand for numbers.
. . .
Definition. p is prime iff p is nontrivial and for every divisor d of p  d = 1 or d = p.
. . .
Theorem. The set of prime numbers is infinite.
Proof. Let A be a finite set of prime numbers.
Take a function p and a number r such that p lists A in r steps.
```


In: A. Editor, B. Coeditor (eds.): Proceedings of the XYZ Workshop, Location, Country, DD-MMM-YYYY, published at http://ceur-ws.org

```
ran p \subseteq \NNplus. \Product{p}{1}{r} != 0.
Take n=\Product{p}{1}{r}+1. n is nontrivial.
Take a prime divisor q of n (by PrimDiv).
Let us show that q is not an element of A. Assume the contrary. Take i such that
(1 \leq i \leq r and q=\subfunc{p}{i}).
\subfunc{p}{i} divides \Product{p}{1}{r} (by MultProd). Then q divides 1 (by DivMin).
Contradiction. qed.
Hence A is not the set of prime numbers. qed.
```

This text is written in the ForTheL language of SAD VERSION, available for download and through a web interface at URLXXX. ForTheL supports the Definition - Theorem - Proof structure typical in modern mathematics. The formulation uses complete and grammatical English sentences interwoven with symbolic phrases. Texts are parsed into an annotated first-order presentation for further processing and checking.

The example illustrates the use of notions in ForTheL to achieve a soft typing comparable to typing by nouns in natural languages. A notion (noun) like `number` is introduced by a `Signature` command. Notions can be modified by predicates (adjectives): we define the predicate is prime and obtain a modified notion `prime number`. ForTheL provides standard constructors of complicated terms like `set of`. The combinations of such mechanisms allows elegant variable-free formulations of first-order statements like `The set of prime numbers is infinite`.

Further mechanisms familiar from mathematical texts are provided. To avoid repetitive type declarations, variables can be pretyped as in `Let m, p stand for numbers`. Linguistic and symbolic patterns for terms and predicates can be introduced freely, as long as the parser is able to identify the position of variables. ForTheL allows the pattern `\Product{p}{1}{r}` to denote a finitary product of prime numbers. This pattern complies with LaTeXmacro syntax; with an appropriate macro definition it will be typeset as $\prod_{i=1}^{r} p_i$. One can mechanically transform the prime number text into proper LaTeXso that Euclid's theorem reads:

Theorem 1 *The set of prime numbers is infinite.*

Proof Let A be a finite set of prime numbers. Take a function p and a number r such that p lists A in r steps. ran $p \subseteq \mathbb{N}^+$. $\prod_{i=1}^{r} p_i \neq 0$. Take $n = \prod_{i=1}^{r} p_i + 1$. n is nontrivial. Take a prime divisor q of n.

Let us show that q is not an element of A. Assume the contrary. Take i such that ($1 \leq i \leq r$ and $q = p_i$). p_i divides $\prod_{i=1}^{r} p_i$ (by MultProd). Then q divides 1 (by DivMin). Contradiction. qed.

Hence A is not the set of prime numbers. □

We consider high-quality mathematical typesetting to be an important component of natural language proof assistants.

3 Naproche-SAD

The Naproche project (Natural Proof Checking) at the University of Bonn pursues similar aims as EA and SAD, with particular emphasis on a linguistically correct input grammar. In his PhD work Marcos Cramer built the Naproche VERSION in which he successfully formalized XXX of Landau's Grundlagen.

Andrei Paskevich and Marcos Cramer left formal mathematics after their PhDs, so that both SAD and Naproche lay dormant for a while. In 2017, in accordance and consultation with Andrei Paskevich, the Naproche project has adopted the SAD system, aiming to further its efficiency and coverage. The ForTheL language has been extended by several frequently used constructs. The software has been partially rewritten, extended and annotated. The Naproche-SAD system is now able to efficiently check some chapter-sized texts at the level of beginning university mathematics.

The structure of the checking process is like in SAD: after parsing, a reasoner module sequentially processes the statements of the text within the context of previous statements. First there is a weak type checking (ontological checking) of the statement to see whether the typing presuppositions of each application of a term or a predicate are satisfied; since terms can be substituted into other terms etc. the number of these checks can be high. Moreover, since presuppositions can in principle be arbitrary first-order formulas, they may require strong first-order proving. Thereafter the logical check is performed. All checks are first attempted by the reasoner; if this fails they are exported to an external first-order ATP (Automatic Theorem Prover); if the external prover fails there are a number of retries with successively stronger premisses. The specific organisation of these processes is decisive for the correctness and efficiency of the proving. Usually weak type checking should be of lower

complexity than the logical checking of statements. Various intermediate proof results about soft types are cached for later use.

Naproche-SAD is written in Haskell. It uses eprover as its ATP, but other first-order provers could easily be attached instead. There is a LaTeXfilter which accepts texts written in LaTeXand transforms them to ForTheL for checking. Naproche-SAD has now been integrated into the jedit PIDE (Proof Integrated Development Environment) known from Isabelle to support interactive formalizations. The code is freely available on Github URL.

4 Set Theory

Since the language and techniques of set theory are omnipresent in modern pure mathematics, basic set theory needs to be formalized within a comprehensive natural formalization project. A wide-ranging formation of abstraction terms $\{x \mid \phi\}$ is at the heart of set theory, so that mechanisms for the translation of such collections into first-order logic have to be provided. In SAD, abstraction terms can be formed with arbitrary ForTheL formulas ϕ , and are registered as sets. So just *writing* the term $\{x \mid x \notin x\}$ in SAD introduces the Russell contradiction into a text. If $\{x \mid x \notin x\}$ is registered as a set a then

$$a \in a \leftrightarrow a \notin a.$$

In SAD, it is the author's responsibility to avoid contradictory abstraction terms (as in many presentations of *naive* set theory). In Naproche-SAD, we employ some ideas from class theory: we use the notions of *class*, *object* and *set*; abstraction terms are registered as classes whose elements are objects; sets are defined to be classes that are objects.

The notion of a *function* is as basic as that of a set or class. So it is also predefined in Naproche-SAD, and a flexible mechanism for the definition of functions by cases and recursion is provided. We are in the process of formalizing a standard course in set theory and have so far covered relations, functions, ordinals and cardinals. The formalization of the crucial Zermelo Wellordering Theorem is close to the given text.

Theorem 2 *For every set x there exist an ordinal α and function f such that $f : \alpha \leftrightarrow x$*

Proof Let x be a set. Define

$$F(\alpha) = \begin{cases} x, \text{ if } x \setminus F[\alpha] = \emptyset, \\ \text{some } v \in x \setminus F[\alpha], \text{ if } x \setminus F[\alpha] \text{ has an element.} \end{cases}$$

for α in Ord.
(1) There exists α such that $F(\alpha) = x$. *Proof.* Assume the contrary. . . . □

In the definition of F, the `\cases` construct from LaTeX is also accepted by Naproche-SAD. This construct spawns further proof obligations, e.g., that the two cases do not overlap and cover all possibilities. Also the axiom of choice enters the definition by choice of v in the second case, indicated by the word some v. Naproche-SAD supports several forms of recursions; in this case, the definition of $F(\alpha)$ recurs to the set $F[\alpha] = \{F(\beta) \mid \beta < \alpha\}$ of previous values of the function which is justified by the strong wellfoundedness of the $\in$-relation.

5 *Principles of Mathematical Analysis* by Walter Rudin

The Principles of Mathematical Analysis [XXX] have been a standard textbook for decades. In a practical Bachelor module we have formalized parts of some initial chapters. Sometimes the formalization turns out to be very close to the original, sometimes our term-rewriting facilities are weak and require several explicit extra steps. On the positive side we present Theorems 1.20(a) and (b) of [XXX] with our formalizations on the left.

6 The Maximum Principle

The principle can be easily derived from the basic theorems.

Theorem 3 *Assume f is holomorphic and the domain of f is a region. If f has a local maximal point then f is constant.*

Proof Let z be a local maximal point of f. Take ϵ such that $B_\epsilon(z)$ is a subset of $\mathrm{Dom}(f)$ and $|f[w]| \leq |f[z]|$ for every element w of $B_\epsilon(z)$.
Let us show that f is constant on $B_\epsilon(z)$. Assume the contrary. Then $f[B_\epsilon(z)]$ is open. We can take δ such that $B_\delta(f[z])$ is a subset of $f[B_\epsilon(z)]$. Therefore there exists an element w of $B_\epsilon(z)$ such that $|f[z]| < |f[w]|$. Contradiction. end.
Hence f is constant. □

We make some comments which also describe important aspects of the ForTheL-language:

1. This text is typeset from a LaTeX-file which is also accepted and proof-checked by the improved SAD within a few seconds. SAD strips away LaTeX-commands and replaces some of them by ForTheL keywords.

2. The text is formulated in the restricted natural language ForTheL, which is immediately understandable by mathematicians. The language is apparently translatable to first-order logic, but it is more flexible and natural than just using logical connectives. There is, e.g., some typing and use of anonymous variables like in the definition: "a *subset* of M is a set N such that every element of N is an element of M".

3. The language constructs of ForTheL have been carefully modelled after OLM to allow elegant formulations of logical dependencies without (nested) brackets or other formal devices.

4. ForTheL allows to freely introduce new undefined *notions* by signature commands and specify their properties by *axioms*, without worrying about grounding everything in some foundational system like set theory. Other notions may be based on previous notions by definitions.

5. The attentive reader will have noticed that some notions and axioms are formulated just for our example and would have to be amended if we want to capture the situation in more generality. The notion of a *region*, e.g., is introduced to allow standard formulations of the Identity Theorem and the Maximum Principle. We only require regions to be open, whereas in a comprehensive foundation regions also have to be connected and non-empty. Similar comments apply to other notion in the text. A more comprehensive text would have to fix that liberal approach.

6. Notions provide soft-typing all variables and constants. In natural language, soft-typing serves to direct the readers attention to a "small world" delineated by the types in the statements under immediate consideration. They are useful in automatic theorem proving for selecting premises from the context which contain common types with the statement to be proved.

7. The proof-checking employs a reasoner which generates proof obligations along the text. This involves also ontological checks that terms belong to certain types. Ontological checking has similarities with strong type checking for programming languages, and it helps to find formalization errors.

8. Some type information is redundant *within* a small world and correspondingly need not be given to the external automatic prover. We are systematically exploiting that observation and considerably cutting down on proving times.

9. The logical context of a SAD text is that the conjunction of all premises implies the conjunction of all theorems. Such implications can be pieced together to build up mathematical theories.

7 Further plans

We are pursuing a comprehensive project for transforming the original SAD system into a productive formalization workbench. We shall use the Isabelle editor as an IDE for formalizing mathematics and for giving more feedback to the user. The compact Haskell source code of SAD is being systematized and documented to ensure the sustainability of the project. We shall try to better separate the parsing from the proof-checking process so that the language module could be used with other formal mathematics systems.

We are collaborating with linguists for the definition and implementation of a proper natural language grammar. Natural language words should not be arbitrary letter combinations, as is possible now, but taken from an English dictionary. We shall increase linguistic flexibility without risking unique readability.

The ForTheL language will be enriched by further constructs for proof structuring and for algebraic structures and inductive data types. Standard domains like number systems will be formalized in a basic library of texts useful for many purposes. Some aspects like the handling of natural numbers could be taken over into the software, to provide some computational power. Also our term rewriting system will be improved.

We shall undertake the formalization of comprehensive texts at the level of undergraduate mathematics. We shall also examine research articles whether a partial formulation in ForTheL is profitable. To evaluate usability, students of mathematics will be asked to prepare homework solutions in the system.

Texts about various domains can be arranged in interlinked libraries. The above example could be linked to an introductory text about holomorphic functions, which again could be linked to some development of complex numbers etc. Texts could be linked by a simple reading-in of other texts, or by more sophisticated, truth-preserving operations like unifications of notations, or they might require some logical bridging between conclusions of one text and premisses of the other. A systematic study of such relations is required for building larger libraries.

8 Discussion

We have come to a peculiar situation where a formal language is able to cover broad areas of a subject area and becomes nearly indistinguishable from the natural language of a domain. The convergence of the formal and the natural leads to a host of serious questions, ranging from practical to philosophical issues. In any case our research demonstrates that the formal approach in mathematics is not restricted to foundations but that it can be used all the way up to sophisticated theories provided that the formalism is set up prudently in a hierarchical fashion.

We conjecture that in a few years time it will be routinely possible to formulate substantial textbook mathematics and some advanced mathematics in a ForTheL-like controlled and proof-checked natural language. Other proof assistants could similarly be equipped with natural language input. Formal mathematics could be carried out naturally in a text-orientated way, using collections of interlinked texts which are readable and understandable by men and machines.

References

[Gan10] Mohan Ganesalingam. *The language of mathematics.* PhD thesis, Springer, 2010.

[Glu70] Victor M. Glushkov. Some problems in the theories of automata and artificial intelligence. *Cybernetics*, 6(2):17–27, 1970.

[KCKS09] Daniel Kühlwein, Marcos Cramer, Peter Koepke, and Bernhard Schröder. The naproche system. *Intelligent Computer Mathematics, Springer LNCS, ISBN*, 978:3–642, 2009.

[Pas07] Andriy Paskevych. *Méthodes de formalisation des connaissances et des raisonnements mathématiques: aspects appliqués et théoriques.* PhD thesis, Université Paris 12, 2007. In French.

[Ran93] Aarne Ranta. Type theory and the informal language of mathematics. In *International Workshop on Types for Proofs and Programs*, pages 352–365. Springer, 1993.