Textbook Mathematics in the Naproche-SAD System

Peter Koepke

Mathematical Institute, University of Bonn, Germany

Abstract

We report on the Naproche-SAD project (Natural Proof Checking - System for Automated Deduction) as part of a wider programme of *Natural Formal Mathematics*. After recalling the classical SAD system with its natural proof language ForTheL (Formula Theory Language), we describe its further development into Naproche-SAD, and present snippets from recent formalizations of textbook material.

Introduction. To faithfully formalize and automatically proof-check mathematical textbooks has long been a challenge in formal mathematics. Lambert S. van Benthem Jutting [vBJ77] formalized Landau's *Grundlagen der Analysis* [Lan30] in the Automath system, the formalization of *A Compendium of Continuous Lattices* was a collaborative project in Mizar [BR02] , and Lawrence Paulson translated parts of *Set Theory* by Kunen [Kun11] as a basis for his proof of the Relative Consistency of the Axiom of Choice in Isabelle/ZF [Pau03]. Since the input languages of these prominent proof checking systems are akin to computer languages, the formalizations do *not* resemble mathematical textbooks.

To formalize mathematics in the style and language of textbooks is part of a programme of *Natural Formal Mathematics* which may be described as:

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

This paper illustrates our current focus on discovering and developing strong natural linguistic mechanisms and proof structures for mathematics. Questions of axiomatic consistency, text correctness or prover strengths are crucial in the long run, but liberal axiomatics and high-performance ATPs are essential for the present exploratory phase.

SAD. The System for Automated Deduction by Andrei Paskevich [Pas07] 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 [Glu70]. The project involved many aspects of automated theorem proving and proof checking (see [LV10]), centered around the modelling of the formulation and checking of theorems and proofs. After a comprehensive analysis of mathematical texts, the controlled natural language ForTheL was developed as an input language for a proof assistant [GKL$^+$72].

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. The text is typeset by LaTeX, making use of an overlap with the ForTheL format for symbolic patterns: a sequence like `\Product{p}{1}{r}` can denote a ternary function symbol in ForTheL, and with an appropriate macro definition of `\Product` will be typeset as $\prod_{i=1}^{r} p_i$. We consider high-quality mathematical typesetting to be an important component of future natural language proof assistants.

In: C. Kaliszyk, E. Brady, J. Davenport, W.M. Farmer, A. Kohlhase, M. Kohlhase, D. Müller, K. Pąk, and C. Sacerdoti Coen (eds.): Joint Proceedings of the FMM and LML Workshops, Doctoral Program and Work in Progress at the Conference on Intelligent Computer Mathematics 2019 co-located with the 12th Conference on Intelligent Computer Mathematics (CICM 2019), Prague, Czech Republic, July 8–12, 2019, published at http://ceur-ws.org

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. 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. □

This text is written in the ForTheL language of SAD as available for download and through a web interface at [LVP]. ForTheL supports the Definition - Theorem - Proof structure typical of 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 enable 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. Linguistic and symbolic patterns for terms and predicates can be introduced freely, as long as the parser is able to identify the positions of variables. On the other hand there is a wide scope for further extensions and improvements of the original ForTheL language; a phrase like "p lists A in r steps", e.g., should be replaced by more familiar constructs like $A = \{p_1, \ldots, p_r\}$.

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 [Cra13] built the Naproche proof checker in which he successfully formalized some initial part of Landau's *Grundlagen*.

Andrei Paskevich and Marcos Cramer did not continue this work after about 2013. 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, and the software has been partially rewritten, extended and annotated in the Master project of Steffen Frerix (see [FK18]). The Naproche-SAD system is now able to efficiently check some chapter-sized texts at the level of beginning university mathematics.

The checking of ForTheL input texts proceeds as 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") 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. 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. Various intermediate proof results about soft types are cached for later use.

Naproche-SAD is written in Haskell. It uses E prover as its ATP, but other first-order provers could easily be attached instead. There is a LaTeX filter which accepts text written in LaTeX and 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 [Nap18].

Set Theory. Since the language and techniques of set theory are omnipresent in modern pure mathematics, a comprehensive natural formalization project must include basic set theory. A wide-ranging formation of abstraction terms $\{x \mid \varphi\}$ 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*. Thus simply 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 *function* is as basic as that of a set or class. So it is 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 lecture notes.

Theorem. *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 don't overlap and cover all possibilities. Also the axiom of choice enters the definition by the 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.

***Principles of Mathematical Analysis* by Walter Rudin.** The *Principles of Mathematical Analysis* [Rud86] have been a standard textbook for decades. In a practical Bachelor module we have formalized parts of some initial chapters. To demonstrate the similarity between [Rud86] and our formalization let us first quote the original Theorem 1.20(a) of [Rud86].

Theorem (120a)**.** *(a) If $x \in \mathbb{R}$, $y \in \mathbb{R}$, and $x > 0$, then there is a positive integer n such that*

$$nx > y.$$

Proof. Let A be the set of all nx, where n runs through the positive integers. If (a) were false, then y would be an upper bound of A. But then A has a *least* upper bound in $\mathbb{R}$. Put $\alpha = \sup A$. Since $x > 0$, $\alpha - x < \alpha$, and $\alpha - x$ is not an upper bound of A. Hence $\alpha - x < mx$ for some positive integer m. But then $\alpha < (m+1)x \in A$, which is impossible, since α is an upper bound of A. □

Here the de Bruijn factor is close to 1:

Theorem (Naproche-SAD version)**.** *If $x \in \mathbb{R}$ and $y \in \mathbb{R}$ and $x > 0$ then there is a positive integer n such that*

$$n \cdot x > y.$$

Proof. Define $X = \{n \cdot x \mid n \text{ is a positive integer}\}$. Assume the contrary. Then y is an upper bound of X. Take a least upper bound α of X. $\alpha - x < \alpha$ and $\alpha - x$ is not an upper bound of X. Take an element z of X such that not $z \leq \alpha - x$. Take a positive integer m such that $z = m \cdot x$. Then $\alpha - x < m \cdot x$ (by 15b).

$$\alpha = (\alpha - x) + x < (m \cdot x) + x = (m + 1) \cdot x.$$

$(m + 1) \cdot x$ is an element of X. Contradiction. Indeed α is an upper bound of X. □

The Appendix of *General Topology* by John L. Kelley. The well-known *General Topology* by Kelley [Kel75] is written in a formalistic style which lends itself to formalizations. We are currently formalizing the Appendix which introduces the Kelley-Morse theory of classes which is related to the foundational class-object-set theory that we have implemented in Naproche-SAD. The Appendix is structured as a sequence of about 200 numbered axioms, definitions and theorems, most of them without proofs. Naproche-SAD together with E prover is often able to find proofs automatically, so that we achieve the same proof granularity as the original text. One could even consider extending ForTheL so that some parts of Kelley are formally correct ForTheL texts.

We present our formalization of Kelley's version of the Kuratowski ordered pair. Kelley uses the universe $\mathcal{U}$ of all sets as an indicator for "undefined". Later in the text the ordered pair will be treated as an undefined basic notion with the axiomatic property stated in Theorem 55. Thereafter ordered pairs will be used to define relations and functions.

Ordered Pairs

Definition (48). $(x, y) = \{\{x\}, \{x, y\}\}$. *Let the* ordered pair *of x and y stand for (x, y).*

Theorem (49a). *(x, y) is a set iff x is a set and y is a set.*

Theorem (49b). *If (x, y) is not a set then $(x, y) = \mathcal{U}$.*

Theorem (50). *If x and y are sets then $\bigcup(x, y) = \{x, y\}$ and $\bigcap(x, y) = \{x\}$ and $\bigcup\bigcap(x, y) = x$ and $\bigcap\bigcap(x, y) = x$ and $\bigcup\bigcup(x, y) = x \cup y$ and $\bigcap\bigcup(x, y) = x \cap y$.*

Theorem (Unnumbered statement). *If x is not a set or y is not a set then $\bigcup\bigcap(x, y) = 0$ and $\bigcap\bigcap(x, y) = \mathcal{U}$ and $\bigcup\bigcup(x, y) = \mathcal{U}$ and $\bigcap\bigcup(x, y) = 0$.*

Definition (51). $1^{st}z = \bigcap\bigcap z$. *Let the* first coordinate *of z stand for $1^{st}z$.*

Definition (52). $2^{nd}z = (\bigcap\bigcup z) \cup ((\bigcup\bigcup z) \sim \bigcup\bigcap z)$. *Let the* second coordinate *of z stand for $2^{nd}z$.*

Theorem (53). $2^{nd}\mathcal{U} = \mathcal{U}$.

Theorem (54a). *If x and y are sets then $1^{st}(x, y) = x$.*

Theorem (54b). *If x and y are sets then $2^{nd}(x, y) = y$.*

Proof. Let x and y be sets. $2^{nd}(x, y) = (\bigcap\bigcup(x, y)) \cup ((\bigcup\bigcup(x, y)) \sim \bigcup\bigcap(x, y)) = (x \cap y) \cup ((x \cup y) \sim x) = y$. □

Theorem (54c). *If x is not a set or y is not a set then $1^{st}(x, y) = \mathcal{U}$ and $2^{nd}(x, y) = \mathcal{U}$.*

Theorem (55). *If x and y are sets and $(x, y) = (r, s)$ then $x = r$ and $y = s$.*

Discussion and Further Plans. Our examples and experiences demonstrate that natural formalizations of undergraduate mathematics in the familiar textbook style are realistically possible. The main components for a successful system are a sophisticated model of the language of mathematics as implemented in ForTheL and its parser, the efficient generation and organization of a large number of first-order proof obligations, and powerful first-order ATPs. Much work is needed to expand the coverage of the language and to create interlinked libraries of basic mathematical theories. The possibility of combining Naproche-SAD with big systems like Isabelle/ZF has to be investigated. Currently we are continuing the development of Naproche-SAD and textbook formalizations as above as Bachelor- and Mastertheses and practicals. Two Bachelor projects deal with the formalization of standard set theory. A practical module is concerned with integrating the ForTheL language into LaTeX and translating from ForTheL to Lean.

References

[BR02] Grzegorz Bancerek and Piotr Rudnicki. A compendium of continuous lattices in MIZAR. *J. Autom. Reasoning*, 29(3-4):189–224, 2002.

[Cra13] Marcos Cramer. *Proof-checking mathematical texts in controlled natural language*. PhD thesis, University of Bonn, 2013.

[FK18] Steffen Frerix and Peter Koepke. SAD Revisited. In *AITP 2018, Abstracts of the talks*, pages 15–16, 2018.

[GKL+72] V. M. Glushkov, Yu. V. Kapitonova, A. A. Letichevskii, K. P. Vershinin, and N. P. Malevanyi. Construction of a practical formal language for mathematical theories. *Cybernetics*, 8:730–739, 1972.

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

[Kel75] J.L. Kelley. *General Topology*. Graduate Texts in Mathematics. Springer New York, 1975.

[Kun11] K. Kunen. *Set Theory*. Studies in logic. College Publications, 2011.

[Lan30] Edmund Landau. *Grundlagen der Analysis*. Akademische Verlagsgesellschaft, Leipzig, 1930.

[LV10] Alexander Lyaletski and Konstantin Verchinine. Evidence algorithm and system for automated deduction: A retrospective view. In *Proceedings of 17th Calculemus Conference on Intelligent Computer Mathematics*, pages 411–426, Berlin, Heidelberg, 2010. Springer-Verlag.

[LVP] Alexander Lyaletski, Konstantin Verchinine, and Andriy Paskevych. Sad web page. Accessed: 2018-04-08.

[Nap18] Naproche-SAD. `https://github.com/Naproche/Naproche-SAD`, 2018.

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

[Pau03] L. C. Paulson. The relative consistency of the axiom of choice - mechanized using Isabelle/ZF. *LMS Journal of Computation and Mathematics*, 6:198–248, 2003.

[Rud86] W. Rudin. *Principles of Mathematical Analysis.* McGraw - Hill Book C., 1986.

[vBJ77] Lambert S. van Benthem Jutting. *Checking Landau's "Grundlagen" in the Automath system.* PhD thesis, Eindhoven, 1977.