Making Set Theory Great Again: The Naproche-SAD Project

Steffen Frerix and Peter Koepke

University of Bonn, Germany
Email: s6stfrer@uni-bonn.de and koepke@math.uni-bonn.de

John Harrison, at AITP 2018, gave a programmatic talk "Let's make set theory great again!" [Har18] in which he proposes to take standard set theory (and classical first-order logic) as a basis for automated theorem proving, enriched conservatively and along current mathematical practice by a soft type system à la Freek Wiedijk [Wie07] and by syntactic sugar. This should lead to a standard hierarchy of number systems, a principled treatment of undefinedness, and generally to a "closer correspondence with informal texts".

SAD. A small system which embraces the Harrison approach is the *System for Automated Deduction* (SAD) by Andrei Paskevich et.al. [Pas07]. SAD combines natural language input with first-order proof checking. Mathematical texts are expressed in the controlled mathematical language ForTheL, and checked for logical correctness by a "reasoner" together with a standard automated theorem prover like E.

Naproche-SAD. In 2017 we began our work with the SAD system, based on experiences with our earlier Naproche system [KCKS09]. We have made the code more efficient and added set theoretical mechanisms [FKW18]. At AITP 2018, we reported on our progress. Meanwhile we are able to deal with chapter-sized texts at the level of first-year undergraduate mathematics. We are working on a LaTeXinterface, and, together with Makarius Wenzel, on a jedit-PIDE for Naproche-SAD (see [Wen18]).

Naturally Enriched First-Order Logic. ForTheL signature definitions like

Signature 1. *A* real number *is a notion. Let* x, y, z *denote real numbers.*

Signature 2. $\mathbb{R}$ *is the set of* real numbers.

Definition 1. x *is* positive *iff* $x > 0$.

Signature 3. *An* integer *is a real number. Let* a, b *denote integers. Let* m, n *denote positive integers.*

serve to set up convenient first-order and set-theoretical environments. ForTheL allows many natural language constructs of ordinary mathematics. Proof methods like case splits, contradiction and induction are supported by automatically generating and checking their implicit proof obligations. The example also indicates that hierarchical number systems can easily be set up in ForTheL.

Soft Typing and Undefinedness. SAD allows soft dependent types via notions and adjectives, as in the above example. As in [Wie07], types are internally translated into obvious first-order predicates. Type checking is turned into an *ontological check* at "runtime" during proving to ensure that all presuppositions are fulfilled. Usually these obligations are considerably simpler than the main proof task. This approach also encompasses a correct treatment of undefinedness: the ontological check of the notorious fraction $\frac{1}{x}$ and hence the checking of the entire text fails if the system cannot prove $x \neq 0$ in the proof context of the term.

Set Theory. The well-known difficulties of set theory in automated theorem proving stem from its vast infinite axiom system and the deep iterations of the $\in$-relation in set-theoretically defined notions like numbers. It is essential to keep the proof search away from arbitrary axioms and expansions of notions. In Naproche-SAD, instances of the problematic infinite

axiom schemes have to be explicitly invoked, e.g., by the use of abstraction terms or function definitions. The reasoner of Naproche-SAD has a restrained strategy for definition expansions which benefits set theory in particular.

Correspondence with Informal Texts. Several of our point are exemplified in the following fragment of our formalization of Rudin's [Rud76] which we compare to the original statement of Theorem 120 *a*) in [Rud76]. One could obtain even stronger natural language resemblances by adding more argumentative phrases to ForTheL.

Theorem [Naproche-SAD] 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 $A = \{n \cdot x \mid n \text{ is a positive integer}\}$. Assume the contrary. Then y is an upper bound of A. Take a least upper bound α of A. $\alpha - x < \alpha$ and $\alpha - x$ is not an upper bound of A. Take an element z of A 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 A. Contradiction. Indeed α is an upper bound of A. □

Theorem [Rudin's original text] (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. □

Comparisons with Mizar. Whilst a majority of proof assistants employ some strong type theory for fundamental reasons or to narrow down proof search, systems like Isabelle/ZF [Isa], Metamath [Met] or Mizar [Miz] are based on first-order set theory. As we share the aim of modeling ordinary mathematical practice with Mizar in particular, there are similarities concerning language and text structuring. Mizar is commited to Tarski-Grothendieck set theory, whereas in Naproche-SAD the specific foundations are variable and depend on which abstraction terms are declared to be sets. Mizar and Naproche-SAD reflect the general mathematical practice of soft typing by "types" and "notions", respectively, which are interpreted as set-theoretic predicates. Both systems have mechanisms to deal with proof obligations spawned by soft-typing. The type systems are however different, as Mizar, e.g., requires types to be non-empty.

A decisive difference between Mizar and Naproche-SAD lies in the degree of proof automation. Mizar texts are required to specify detailed proof steps which leads to a legible, yet computer orientated input language. Naproche-SAD uses strong automated theorem proving to find implicit proof steps. This allows proof granularities similar to textbook proofs, and supports the use of a (restricted) natural language as proof language. The prospect of formal mathematical texts written in natural language is a main driving force of the Naproche-SAD project.

Kelley Morse Class Theory. If one allows *class* quantifiers in the defining properties φ of abstraction terms $\{x \mid \varphi\}$ one is working in Kelley Morse class theory (KM) which is somewhat stronger than Zermelo-Fraenkel set theory. In KM, sets are those classes which are elements of some class. The abstraction term mechanism of Naproche-SAD corresponds to Kelley Morse terms. This has motivated our current formalization of the Appendix of [Kel55] in which the theory KM was introduced. Working with the Appendix has shown the necessity of splitting larger texts into chapters and using ideas of small theories and theory morphisms [Koh14] to control ontological manueuvers like turning the formation of Kuratowski ordered pairs into a basic function.

In our **Talk** we shall illustrate the above set theory orientated principles by excerpts and demonstrations of the mentioned formalisations.

References

[FKW18] Steffen Frerix, Peter Koepke, and Makarius Wenzel. Naproche-SAD. https://github.com/Naproche/Naproche-SAD, 2018.

[Har18] John Harrison. Let's make set theory great again! http://aitp-conference.org/2018/slides/JH.pdf, 2018.

[Isa] Isabelle. https://www.cl.cam.ac.uk/research/hvg/Isabelle/.

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

[Kel55] John L Kelley. General topology. *Graduate Texts in Mathematics*, 27, 1955.

[Koh14] Michael Kohlhase. Mathematical knowledge management: Transcending the one-brainbarrier with theory graphs. *EMS Newsletter*, 92:22–27, 2014.

[Met] Metamath. http://us.metamath.org/index.html.

[Miz] Mizar. http://mizar.org/.

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

[Rud76] Walter Rudin. Principles of mathematical analysis. *McGraw-Hill, Inc.*, 1976.

[Wen18] Makarius Wenzel. Isabelle/jEdit as IDE for domain-specific formal languages and informal text documents. *arXiv preprint arXiv:1811.10819*, 2018.

[Wie07] Freek Wiedijk. Mizar's soft type system. In *Theorem Proving in Higher Order Logics*, pages 383–399. Springer, 2007.