Setting up Set-Theoretical Foundations in ℕaproche

Marcel Schütz[2], Adrian De Lon[1], and Peter Koepke[1]

[1] University of Bonn, Germany, https://www.math.uni-bonn.de/ag/logik/
[2] TU Darmstadt, Germany

Abstract. Bringing out the potential of interactive theorem provers requires efficient mathematical foundations. The current release of the natural language proof assistant ℕaproche has become sufficiently stable to allow a broader and principled approach towards libraries of basic mathematical material. We present ℕaproche's new ontology of objects, classes, maps, etc. and two foundational libraries about basic set theory and number theory. These foundations are then used, e.g., in a formalization of the Cantor–Schröder–Bernstein theorem.

1 Introduction

The ℕaproche (Natural Proof Checking) proof assistant [12] is being developed for a high degree of naturalness of accepted proof texts and of its user interaction [3,8,10]. It is available as a component of the Isabelle prover platform [6]. ℕaproche supports prototypical formalizations of university-level material in a natural mathematical language and style that is immediately readable by mathematicians. ℕaproche comes with a number of chapter-sized example formalizations which also comprise the foundational files discussed in this paper.

Most ℕaproche formalizations thus far are self-contained texts that introduce their own axiomatic environments and lead up to individual well-known theorems. Ad-hoc axiomatic set-ups allow elegant mathematical "miniatures" with a minimal number of axioms, but the consistency and compatibility of axiomatic assumptions pose problems. Moreover, duplicate definitions cause ambiguities and reproving basic theorems is inefficient.

In this paper we describe steps towards a next level of formalizations with fixed ontological foundations and libraries of basic material. The current release of ℕaproche has a built-in language for classes, objects, and maps within a Kelley–Morse-like set theory [7,13]. Some Kelley–Morse axioms are fixed in foundational files whereas further axioms, like the axiom of infinity can be postulated by a user if needed. The current ℕaproche release includes two example libraries on sets and on integers that are then used to formalize more specialized results.

Kevin Buzzard, Temur Kutsia (Eds.): Work-in-progress papers presented at the 15th Conference on Intelligent Computer Mathematics, pp. 58–70, 2022.
https://www3.risc.jku.at/publications/download/risc_6584/proceedings-CICM2022-informal.pdf

2 Intuitive mathematical ontologies

The point of departure for *natural* formal mathematics and *natural* proof assistants is the language that is actually employed in contemporary mathematical publications, combining argumentative natural language and symbolic terms. So far, the language of mathematics has been subject to only a small number of linguistic investigations [5,1]. Like ordinary language, this language is ambiguous and incomplete, but mathematicians use it as an efficient means of specialized communication.

The language of mathematics indicates possible or intended ontologies by its choice of natural language words and phrases and by their grammatical categories: e.g.,

- definite nouns denote specific objects or individua of some universe, whether such a universe is taken for real or a convenient mental construct;
- common nouns denote collections, classes, sets, sorts, or types of (related) objects;
- these collections can be modified and specified by adjectives or relative sentences.

"Aristotle" is a distinct individuum which belongs to the collection of "humans"; the common noun "human" can be modified by adjectives like "mortal". Similarly "zero" denotes a constant which is an element of the "natural numbers"; natural numbers can, e.g., be "even" or "odd". Symbolic notation is used for brevity and exactness: "0" is used for "zero" and "$\mathbb{N}$" for the collection of natural numbers.

Collections are intuitively expressed as the "collection, class, type, set, ... of all x such that ϕ" or by *comprehension terms* like

$$\{x \mid \phi\} \text{ or } \{x : \phi\}$$

where ϕ is a mathematical statement. Often such collections are considered to be objects themselves. One can, e.g., say that $\mathbb{N}$ is (an object that is) infinite.

This brief impression suggests ontologies of mathematical objects and collections of objects, where the distinctions between various categories and their properties are somewhat fluid, similar to other natural language ontologies.

From a type-theoretic perspective (mathematical) common nouns like "number" or "rectangle" can be interpreted as types. Since common nouns do not exactly satisfy the laws of mathematical type theories, one speaks of "soft" types [18]. The properties of soft types lie between sets and types: there are set-theoretical properties like inclusions of types ("natural numbers" form a subtype or subset of all "numbers"). On the other hand natural language quantifications are usually bounded by types: we say "every integer ..." instead of "for every n, if n is an integer then ...". A computer-processable natural language for mathematics should use soft types to denote collections of mathematical objects.

3 The ℕaproche system

The ℕaproche (Natural Proof Checking) proof assistant [12] is available as a component in the latest release of the Isabelle prover platform [6] for Linux, macOS, and Windows. We have also built a simple web interface for a quick start without installation or high system requirements [17]. ℕaproche supports prototypical formalizations of university-level material in a natural mathematical language and a style that is immediately readable by mathematicians.

The ℕaproche system uses the controlled natural language ForTheL (Formula Theory Language) as its input language which is designed to closely approximate common constructs of the language of mathematics [16]. At the same time ForTheL is a completely formal language which allows its translation into formal logics and further processing by automated theorem provers or checkers. ForTheL has been developed over several decades and is an outgrowth of the Evidence Algorithm project [11].

ForTheL handles soft types as "notions", and it provides various methods for the introduction and processing of notions. Notions can be viewed midway between sets, classes, and types. ℕaproche translates notions into first-order predicates which can be used to form type guards. The language of a ForTheL text is fixed by signature commands and definitions. A language for the additive structure of the natural numbers can be introduced by the commands:

[synonym number/numbers]

Signature. A natural number is a notion.

Signature. 0 is a natural number.

Signature. Assume that m, n are natural numbers. $m + n$ is a natural number.

ForTheL may be embedded into LaTeX documents; for the example above one could enter the following into a `.ftl.tex` document in Isabelle/jEdit:

```
\begin{forthel}
  [synonym number/numbers]
  \begin{signature}
    A natural number is a notion.
  \end{signature}
  \begin{signature}
    $0$ is a natural number.
  \end{signature}
  \begin{signature}
    Assume that $m,n$ are natural numbers.
    $m+n$ is a natural number.
  \end{signature}
\end{forthel}
```

After fixing a bit of vocabulary ("number" and its plural "numbers" can be used synonymously in the sequel) there are three LaTeX signature environments using the usual combination of natural language and symbols. The first-order meaning of the commands can be seen by mouse-hovering over the jEdit buffer. The noun phrase "natural number" corresponds to a newly introduced unary predicate symbol `aNaturalNumber( )`. The second command introduces the internal constant symbol 0 and translates to the (tagged) first-order formula

$$\forall v_0.\ ((\texttt{HeadTerm} :: v_0 = 0) \to \texttt{aNaturalNumber}(v_0))$$

which is logically equivalent to `aNaturalNumber(0)`. Finally the introduction of the addition symbol + is translated to the following first-order formulas:

$$\frac{\texttt{aNaturalNumber}(m) \wedge \texttt{aNaturalNumber}(n)}{\forall v_0.\ ((\texttt{HeadTerm} :: v_0 = m + n) \to \texttt{aNaturalNumber}(v0))}$$

These formulas also express that addition has the type $+ : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$.

Note that the first-order approach to types yields a very flexible *dependent* type system where number systems can be cumulative ($\mathbb{N} \subseteq \mathbb{R} \subseteq \mathbb{C}$), and notions can depend on parameters ("subsets of $\mathbb{N}$", "divisors of n").

4 Sets and classes in ℕaproche

As ℕaproche is directed at common mathematical usage, comprehension terms in verbal and symbolic form are provided by ForTheL. One can, e.g., define

Definition. $\mathbb{N} = \{\, x \mid x$ is a natural number $\}$.

or, verbally,

Definition. $\mathbb{N}$ is the collection of all natural numbers.

ℕaproche translates these definitions into an internal first-order format which will eventually be passed to external ATPs. Therefore a treatment of comprehension terms in the sense of standard set or class theories is hard-coded into ℕaproche. In line with Kelley–Morse class theory with urelement (KMU) [13] comprehension terms are registered as *classes* which consist of "small" *objects*.

Thus ℕaproche translates our definition to the first-order formula

$$\forall v_0.\ (v_0 = \mathbb{N} \leftrightarrow \texttt{aClass}(v_0) \wedge$$
$$\forall v_1.\ (v_1 \in v_0 \leftrightarrow \texttt{aObject}(v_1) \wedge \texttt{aNaturalNumber}(v_1))).$$

In the ℕaproche implementation of KMU we have objects, elements, sets and classes – note that ℕaproche allows to use the terms 'class' and 'collection' synonymously – satisfying:

– every element of a class is an object;
– a set is a class that is an object.

These notions with some elementary properties are hard-coded into ℕaproche. Moreover we provide notions of functions and maps, which in close analogy with sets and classes behave as follows:

- the application of a map to an object in its domain is an object;
- a function is a map which is an object;
- these notions are kept abstract, but behave like the usual set-theoretic encoding of functions and maps as graphs.

4.1 Kelley–Morse-like foundations in ℕaproche

ℕaproche's built-in vocabulary and mechanisms for classes and maps do not form a sufficient basis for formalizations. Formalizations in earlier ℕaproche versions each defined their own additional axioms as preliminaries: for instance, one would find multiple and even slightly different definitions of "subset" across different formalizations.

We have replaced such ad-hoc preliminaries with a common shared file `preliminaries.ftl.tex`, which describes a weak fragment of KMU. It expands on ℕaproche's built-in notions with some common axioms (e.g. extensionality) and defines general notions such as "subclass". Besides some minor technical definitions and axioms we require:

Axiom (Class Extensionality). If S is a subclass of T and T is a subclass of S then $S = T$.

Axiom (Separation Axiom). Assume that X is a set. Let T be a subclass of X. Then T is a set.

Axiom (Functional Extensionality). Assume $\mathrm{dom}(f) = \mathrm{dom}(g)$ and for every $x \in \mathrm{dom}(f)$ $f(x) = g(x)$. Then $f = g$.

Axiom (Replacement Axiom). Let X be a set. Assume that X is a subset of the domain of f. Then $f[X]$ is a set.

Here $f[X]$ denotes the image of X under f. Although the axioms of separation and replacement are potentially powerful axioms, the theory described in `preliminaries.ftl.tex` constitutes a relatively weak theory since it includes neither the axiom of infinity nor the axiom of powersets.

4.2 Applications

The preliminaries file is used in some of the example formalizations that come with ℕaproche. The mutilated checkerboard problem is only concerned with finite sets of checkerboard positions and dominoes [2]. Thus we do not need the axiom of infinity and the weaker class theory of `preliminaries.ftl.tex` suffices.

Our example file `numbers.ftl.tex` on the number systems $\mathbb{N} \subseteq \mathbb{Z} \subseteq \mathbb{Q} \subseteq \mathbb{R}$ in contrast demands that the various infinite collections of numbers are sets. This is expressed axiomatically by

Axiom. $\mathbb{R}$ is a set.

In standard set theory this could be proved by the axiom of infinity which says that $\mathbb{N}$ is a set, and then one could construct the set $\mathbb{R}$ from $\mathbb{N}$ by forming the powerset of $\mathbb{N}$ together with other, weaker set construction principles.

5 Foundational libraries in ℕaproche

In addition to the `preliminaries.ftl.tex` file, ℕaproche also provides two larger libraries which cover foundational formalizations about basic set theory and arithmetic. In contrast to previous formalizations as stand-alone documents, this is a first systematic attempt to design a collection of formalizations which are primarily intended to be read in other ForTheL texts. The two libraries are organized as LaTeX documents with a book-like chapter structure which are written in a literary style with comments around the formalized core[3]. Importing their contents can be done chapter by chapter so that one is not forced to import the whole library when one is just interested in reusing some specific definitions or theorems they provide. The libraries are designed towards a high degree of human readability which aims to be a step towards mathematical texts being accessible both to computers and to mathematicians unfamiliar with texts written in synthetic formal languages. Besides being used as a foundation for more ambitious formalizations in ℕaproche, the libraries can be regarded as a presentation of formally verified undergraduate mathematics in the style of, e.g., the Stacks Project [15].

5.1 A foundational library for sets

The set theory library[4] is a self-contained introduction to a Zermelo–Fraenkel-like set theory, without the axiom of infinity, embedded into ℕaproche's built-in fragment of KMU. It states the usual ZF axioms, defines common operations on sets, like unions, intersections and complements, and provides detailed proofs of their algebraic properties. Moreover, the notion of functions between sets is introduced together with basic properties like being one-to-one or invertible. As with sets, common operations on functions are defined, e.g., composition, restriction, images and preimages. Their algebraic properties are proved in detail as well. Finally, the notion of equinumerosity is introduced.

The material in the library is presented in a naturally readable format, as demonstrated for instance by the formulation of the Knaster–Tarski fixed point theorem:

[3] See e.g., the PDF-printout `$ISABELLE_HOME/contrib/naproche-20212111/examples/set-theory/set-theory.ftl.pdf`

[4] Located in `$ISABELLE_HOME/contrib/naproche-20212111/examples/set-theory`

Theorem 11.5 (Knaster-Tarski). Let h be a function from $\mathcal{P}(x)$ to $\mathcal{P}(x)$ that preserves subsets. Then h has a fixed point.

In this example, $\mathcal{P}(x)$ denotes the powerset of x, where x has been pretyped as a set earlier in the document.

In contrast to the foundational material in the file `preliminaries.ftl.tex` presented above, which puts a rather strong emphasis on the notion of classes and maps, the primary notions of this library are sets and functions. Classes and maps only play a rather technical role in the library, for instance by using them to formulate statements which in ZF could only be formulated as axiom *schemas*. The decision to follow a more *set*-theoretical approach in the sense of Zermelo–Fraenkel in contrast to a more *class*-theoretical one according to Kelley–Morse is motivated by the lower degree of complexity of the ontology in a setting where just sets and functions play a major role compared to a setting where in addition to them also classes and maps are involved on an equal footing and not just as technical helpers.

The first chapter of the library introduces the notion of subsets and states the axioms of *set extensionality*, *separation*, *set existence*, *pairing* and *union*. Moreover, unions, intersections and complements are defined and some of their algebraic properties are proved. To state the axiom of separation, which in ZF would actually be an axiom *schema*, we make use of ℕaproche's built-in notion of classes together with the built-in axiom schema of *class comprehension*. Just as we postulated *separation* and *replacement* in the above file `preliminaries.ftl.tex`, we again formulate:

Axiom 1.9 (Separation). Let C be a collection and x be a set. Assume that every element of C is contained in x. Then C is a set.

(Note that the term *collection* is just a built-in synonym for *class*.) Using this axiom to define a set y which contains all elements of a given set x satisfying a formula φ, we can proceed as follows. First, by class comprehension, we can define y as the *class* $\{ u \in x \mid \varphi(u) \}$ and then, by the separation axiom, we can show that y is actually a set. As an example of this procedure, consider the proof of the existence of relative complements:

Lemma 1.46. Let x, y be sets. There exists a set z such that $z = \{ w \mid w \in x \text{ and } w \notin y \}$.

Proof. Define $z = \{ w \mid w \in x \text{ and } w \notin y \}$. Then every element of z is contained in x. Hence z is a set (by Separation). □

Here we use ℕaproche's built-in mechanism of defining classes via comprehension terms which serves as an implementation of the mentioned axiom schema of class comprehension. This way we define a class z which is then shown to be actually a set by referencing to the separation axiom (for more on this referencing mechanism see below).

Whereas $\mathbb{N}$aproche allows to define classes of the form $\{u \mid \psi(u)\}$ or $\{u \in x \mid \psi(u)\}$ (where the latter is just an abbreviation for $\{u \mid u \in x \wedge \psi(u)\}$) for arbitrary formulas ψ, it is currently not possible to directly refer to formulas in a ForTheL statement as if they were objects because of the first-order nature of ForTheL. Thus the common ZF-like formulation of the separation axiom as "Let ψ be a formula. Then for any set x there exists a set y such that $y = \{\, u \in x \mid \psi(u) \,\}$" would not be valid ForTheL.

The following chapters of Part I of the library introduce the *powerset axiom* (chapter 2), the *axiom of regularity* (chapter 3), the symmetric difference and its algebraic properties (chapter 4), ordered pairs via Kuratowski's definition (chapter 5) and finally Cartesian products (chapter 6). In this last chapter the Cartesian product of two sets x and y is defined as follows:

Definition 6.2. $x \times y$ is the set z such that $z = \{\, (u, v) \mid u \in x$ and $v \in y \,\}$.

To ensure that this definition is well-formed, i.e. that a unique such set z actually exists, it is preceded by the following lemma granting the existence of z – uniqueness follows immediately from $\mathbb{N}$aproche's built-in extensionality axiom for classes.

Lemma 6.1. There exists a set z such that

$$z = \{\, (u, v) \mid u \in x \text{ and } v \in y \,\} .$$

Within $\mathbb{N}$aproche's ontology, we could also use a slightly different way of defining the Cartesian product. Namely, we could define $x \times y$ as the class $\{\, (u, v) \mid u \in x$ and $v \in y \,\}$ and show afterwards that this class is actually a set. Whereas this order of first defining an object and proving afterwards that it is of a certain type is the more common approach in everyday-mathematics, we decided to do it the other way around (i.e. first proving that an object of a certain type exists and afterwards defining something to be this object). This ensures that the assertion that the Cartesian product of two sets is again a set is already part of the definition which makes the proof search in $\mathbb{N}$aproche a little bit more efficient.

Part II of the library begins with a chapter on functions (chapter 7). It states the *axiom of replacement*, introduces the notions of injectivity, surjectivity, bijectivity, identity function and constant functions and defines composition and restriction operations. For instance, the axiom of replacement, like the axiom of separation in ZF only expressable as an axiom schema, is formulated using the notion of maps:

Axiom 7.8 (Replacement). Let f be a map and x be a set. There exists a set y such that $y = \{\, f(u) \mid u \in \mathrm{dom}(f)$ and $u \in x \,\}$.

The remaining chapters of Part II deal with images and preimages of functions (chapter 8), invertible functions (chapter 9), the behaviour of the symmetric difference under images and preimages (chapter 10), subset preserving functions and the Knaster–Tarski fixed point theorem (chapter 11) and finally the notion of equinumerosity (chapter 12).

Similar to ℕaproche's mechanism for defining classes in a proof, there is also a built-in mechanism for defining maps. For instance consider the following proof of the existence of the preimage of a set z under some function f – note that z was pretyped as a set earlier in the formalization:

Lemma 8.9. Let f be a function. There exists a set y such that $y = \{ u \in \mathrm{dom}(f) \mid f(u) \in z \}$.

Proof.

$\vdots$

Case $f(u) \notin z$ for some $u \in \mathrm{dom}(f)$. Take $w \in \mathrm{dom}(f)$ such that $f(w) \notin z$. Define

$$g(u) = \begin{cases} u & : f(u) \in z \\ w & : f(u) \notin z \end{cases}$$

for $u \in \mathrm{dom}(f)$.

$\vdots$

Take $y = \mathrm{range}(g) \setminus \{ w \}$. Then $y = \{ u \in \mathrm{dom}(f) \mid f(u) \in z \}$. End. □

Here we define a map g on the set $\mathrm{dom}(f)$ which ℕaproche can show on its own to be a function. The range of this function g is then used to define a set y which can be expressed by the comprehension term in the assertion of the lemma.

5.2 A foundational library for natural number arithmetic

The arithmetic library[5] is a self-contained elaboration of Peano Arithmetic with addition, multiplication, exponentiation and factorial with detailed proofs of their arithmetic properties. It provides the standard ordering on the natural numbers and proves various rules about its interplay with the arithmetical operations. Based on this ordering, some variants of induction are provided and also a (partial) subtraction operation. The library defines divisibility, Euclidean division and modular arithmetic. Finally, the notion of prime numbers is given together with proofs of some basic number-theoretic facts.

A typical example of a theorem provided by the library is its formulation of the Euclidean division theorem:

Proposition 14.1. For all natural numbers n, m such that m is nonzero there exist natural numbers q, r such that $n = (m \cdot q) + r$ and $r < m$.

[5] Located in `$ISABELLE_HOME/contrib/naproche-20212111/examples/arithmetic`

Chapter 1 of the library introduces the notion of natural numbers as a new sort of objects together with a constant '0' and a unary successor operation 'succ', independent of any other foundational formalizations. In contrast to setting up Peano Arithmetic on the basis of, e.g., the set theory library presented above, this stand-alone approach was chosen to avoid potential efficiency drawbacks concerning the verification in ℕaproche which a large overhead of formalizations it would otherwise depend on might cause.

The structure of the natural numbers is characterized by the following three axioms:

Axiom 1.5 (1st Peano axiom). If $\mathrm{succ}(n) = \mathrm{succ}(m)$ then $n = m$.

Axiom 1.6 (2nd Peano axiom). 0 is not the direct successor of any natural number.

Axiom 1.7 (3rd Peano axiom). Let P be a class. Assume $0 \in P$ and for all natural numbers n we have $n \in P \implies \mathrm{succ}(n) \in P$. Then every natural number is an element of P.

(Note that in the first axiom, n and m are pretyped as natural numbers and in the second one, "the direct successor of ..." is just a synonym for "succ(...)".) As in the set theory library we used ℕaproche's built-in notion of classes to formulate the induction schema of Peano Arithmetic.

The following chapters of Part I of the library provide some operations on the natural numbers, namely addition (chapter 2), multiplication (chapter 3), exponentiation (chapter 4) and factorial (chapter 5), together with detailed proofs of their computation laws. These operations are all introduced axiomatically, e.g.:

Signature 4.1. n^m is a natural number.

Axiom 4.2 (1st exponentiation axiom). $n^0 = 1$.

Axiom 4.3 (2nd exponentiation axiom). $n^{m+1} = n^m \cdot n$.

Of course, every time a new operation is introduced this way, the axiom system of the whole formalization is extended which increases the potential of accidentally getting some inconsistency the larger this system grows. Thus in the further development of this library, more robust foundations for Peano Arithmetic will be evaluated, for instance by merging it into the set theory library, which would allow to define all such operations on the basis on Dedekind's recursion theorem.

Part II of the library introduces the standard ordering on the natural numbers (chapter 6) and presents some facts about the interplay between this ordering and the addition, multiplication and exponentiation operations (chapters 7, 8, 9, resp.). Moreover, some equivalent formulations of the induction axiom are given (chapter 10), followed by a chapter on standard exercises in natural number arithmetic (chapter 11). Finally, chapter 12 introduces a (partial) subtraction operation on the natural numbers.

Part III deals with the divisibility of natural numbers. This notion is defined in chapter 13, together with proofs of its basic algebraic properties. Chapter 14 presents a proof of the possibility of Euclidean division within the natural numbers, on the basis of which modular arithmetic is introduced in chapter 15. Finally, in chapter 16 the notion of prime numbers is introduced together with some basic facts from number theory.

5.3 Usage in other formalizations

The two described libraries are used to serve as a foundation for a formalization of a proof of the Cantor–Schröder–Bernstein theorem, i.e. the assertion that two sets are equipollent if they can be embedded into each other, and of Furstenberg's proof of the infinitude of primes [4].

The former [6] is based on a version of Knaster's proof of the Cantor–Schröder–Bernstein theorem as stated in [14]. It uses ℕaproche's `readtex` instruction to import some chapter (and its dependencies) of the set theory library:

```
[readtex set-theory/sections/02_functions/06_equipollency.ftl.
tex]
```

(To avoid confusion: ℕaproche provides two instructions for importing files, `read` and `readtex`. The former is used to import files in the `.ftl` format, whereas the latter is used for files in the `.ftl.tex` format.) Since ℕaproche's import functionality is very limited, there is no option for only importing those definitions and theorems we need to state and prove the Cantor–Schröder–Bernstein theorem. Instead we have to import a full chapter of the set theory library. Having imported a chapter of a library, we can use LaTeX's `\ref` command to cite any definition, theorem, etc. it provides for the proof of the Cantor–Schröder–Bernstein theorem. For instance after a certain function h is defined which is shown to be subset-preserving, the proof states the existence of a fixed point c of h by referring to theorem 11.5 of the set theory library:

> Hence we can take a fixed point c of h (by 11.5).

The LaTeX source of this line looks as follows:

```
Hence we can take a fixed point $c$ of $h$ (by \ref{
    SetTheory_02_05_636019}).
```

The LaTeX command `\ref{SetTheory_02_05_636019}` creates a clickable link which points to the theorem with label `SetTheory_02_05_636019` (which is theorem 11.5) in the PDF file of the set theory library.

[6] see `$ISABELLE_HOME/contrib/naproche-20212111/examples/cantor-schroeder-bernstein.ftl.tex`

6 Further plans

Our approach to libraries is exploratory and so far limited by ℕaproche's rudimentary `read` instruction. We are working on an import mechanism that makes it possible to build up a graph of theories and provides more control over the visibility of theorems. We are also experimenting with options for natural language syntax and the reuse of common LaTeX commands (e.g. `\cite`) for theory imports.

Presently importing mathematical structures and associated material from other formalizations is rather inflexible. Although ℕaproche can deal with structures (see e.g. [9]), they can only be imported "verbatim". For example, changing a group operation from $\times$ to $\star$ requires cumbersome notation and incurs a considerable amount of overhead (making proof automation less effective). A logical next step is a ForTheL/ℕaproche implementation of mathematical structures similar to Isabelle's locales which shall address these issues and significantly improve reusability of future formalizations.

Since ℕaproche formalizations omit many details and heavily rely on proof automation, checking speeds have always been slower than established proof assistants. Running ℕaproche can be compared to continuously invoking Sledgehammer for every explicit and implicit claim in a document. We are working on more robust local caching and efficient usage of external provers so that ℕaproche remains usable on mid-range hardware.

Performance problems have also been exacerbated by basing formalizations on general libraries and moving towards richer ontologies. While Kelley–Morse class theory with urelements is a strong and expressive foundation for mathematics, we have also noticed some trade-offs compared to ZF. Presently first-order formulas exported to ATPs are burdened with type guards for classes and elements, making proof search more difficult. In practice this meant that ℕaproche proofs had to become more detailed and some previously working proofs had to be refactored. We also observed that proof obligations stemming from the class-set-distinction (e.g. having to prove that a certain comprehensions forms a set) can be a major stumbling block for students working with ℕaproche.

References

1. Avigad, J.: Mathematics and language (2015)
2. De Lon, A., Koepke, P., Lorenzen, A.: A natural formalization of the mutilated checkerboard problem in Naproche. In: Cohen, L., Kaliszyk, C. (eds.) 12th International Conference on Interactive Theorem Proving (ITP 2021). Leibniz International Proceedings in Informatics (LIPIcs), vol. 193, pp. 16:1–16:11. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2021). https://doi.org/10.4230/LIPIcs.ITP.2021.16, https://drops.dagstuhl.de/opus/volltexte/2021/13911
3. Frerix, S., Koepke, P.: Automatic proof-checking of ordinary mathematical texts. Proceedings of the Workshop Formal Mathematics for Mathematicians (2018)
4. Furstenberg, H.: On the infinitude of primes. The American Mathematical Monthly **62**(5), 353–353 (1955)

5. Ganesalingam, M.: The Language of Mathematics. Springer (2013)
6. Isabelle contributors: The Isabelle2021-1 release (December 2021), https://isabelle.in.tum.de/website-Isabelle2021-1/index.html
7. Kelley, J.L.: General Topology. Springer New York (1955)
8. Koepke, P.: Textbook mathematics in the Naproche-SAD system. Joint Proceedings of the FMM and LML Workshops (2019)
9. Koepke, P., Penquitt, J., Schütz, M., Sturzenhecker, E.: Formalizing foundational notions in Naproche-SAD. In: NFM 2020 - Workshop on Natural Formal Mathematics (at CICM 2020) (2020)
10. Lon, A.D., Koepke, P., Lorenzen, A., Marti, A., Schütz, M., Wenzel, M.: The Isabelle/Naproche Natural Language Proof Assistant. In: Platzer, A., Sutcliffe, G. (eds.) Automated Deduction - CADE 28 - 28th International Conference on Automated Deduction, Virtual Event, July 12-15, 2021, Proceedings. Lecture Notes in Computer Science, vol. 12699, pp. 614–624. Springer (2021). https://doi.org/10.1007/978-3-030-79876-5_36, https://doi.org/10.1007/978-3-030-79876-5_36
11. Lyaletski, A.V., Verchinine, K.: Evidence Algorithm and System for Automated Deduction: A retrospective view. In: Autexier, S., Calmet, J., Delahaye, D., Ion, P.D.F., Rideau, L., Rioboo, R., Sexton, A.P. (eds.) Intelligent Computer Mathematics, 10th International Conference, AISC 2010, 17th Symposium, Calculemus 2010, and 9th International Conference, MKM 2010, Paris, France, July 5-10, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6167, pp. 411–426. Springer (2010). https://doi.org/10.1007/978-3-642-14128-7_35, https://doi.org/10.1007/978-3-642-14128-7_35
12. Naproche on GitHub, https://github.com/naproche/naproche
13. Rubin, J.E.: Set Theory for the Mathematician. Holden-Day, 1st edn. (1967)
14. Schröder, B.S.W.: The fixed point property for ordered sets. Arabian Journal of Mathematics **1**, 530 (2012). https://doi.org/10.1007/s40065-012-0049-7
15. The Stacks project, https://stacks.math.columbia.edu/
16. Verchinine, K., Paskevich, A.: ForTheL — The Language of Formal Theories. International Journal of Information Theories and Applications **7**(3), 120–126 (2000)
17. Naproche-Webinterface, https://naproche.github.io/#/
18. Wiedijk, F.: Mizar's soft type system. In: Theorem Proving in Higher Order Logics. Springer Lecture Notes in Computer Science, vol. 4732, pp. 383–399 (2007)