A Natural Formalization of the Mutilated Checkerboard Problem in Naproche

Adrian De Lon ✉
Universität Bonn, Germany

Peter Koepke ✉
Universität Bonn, Germany

Anton Lorenzen ✉
Universität Bonn, Germany

──── **Abstract** ────────────────────────────

Naproche is an emerging *natural proof assistant* that accepts input in a controlled natural language for mathematics, which we have integrated with LaTeX for ease of learning and to quickly produce high-quality typeset documents. We present a self-contained formalization of the *Mutilated Checkerboard Problem* in Naproche, following a proof sketch by John McCarthy. The formalization is embedded in detailed literate style comments. We also briefly describe the Naproche approach.

1 Introduction

We illustrate the potential of *natural interactive theorem proving* by a formalization of the *Mutilated Checkerboard Problem* in the interactive proof assistant Naproche (Natural Proof Checking). The formalization employs the (controlled) natural mathematical language ForTheL (Formula Theory Language), which is immediately readable by mathematicians and has obvious first-order semantics. ForTheL allows familiar definition-axiom-theorem-proof text structures. The language is integrated into LaTeX (see also [10]) so that the formalization document can be viewed and printed in high-quality mathematical typesetting. In the spirit of *literate programming* [8], the actual formalization, indicated with a grey background, is embedded into ample commentary for the benefit of human readers. Thus the whole article is a valid proof-checked ForTheL document.

The *Mutilated Checkerboard Problem*, which will be explained in detail in the formalization in Section 3, was proposed by John McCarthy as a challenge ("tough nut") [12] to automatic and interactive theorem proving. By now there are many formalizations of the problem (see the survey article *A Tough Nut for Mathematical Knowledge Management* by Manfred Kerber and Martin Pollet [7]).

Mathematically, our formalization follows McCarthy's sketch *The mutilated checkerboard in set theory* [13]. Our formalization is fully self-contained and includes definitions and axioms about finite sets, functions and cardinalities. Since Naproche is based on a weakly typed first-order logic, modelling the checkerboard employs first-order relations, functions and constants. Our axioms are evidently true in a "standard model" of a checkerboard and finite sets. In a stronger foundational theory like Zermelo-Fraenkel set theory, our axioms are provable when one replaces "set" by "finite set".

In Section 2 we briefly describe the Naproche system and its input language ForTheL. Section 3 contains the actual formalization of the *Mutilated Checkerboard Problem*; the text includes explanations of axiomatic and mathematical details of the argument. Further technical remarks about the formalization are elaborated in Section 4. In the final Section 5 we propose further improvements to the Naproche system.

We think that naturalness of interaction will be a crucial factor for the acceptance of proof assistants by the mathematical community.

2   The Natural Proof Assistant Naproche

While state-of-the-art interactive theorem provers have been used to prove and certify highly non-trivial research mathematics, they are still, according to Lawrence Paulson, "unsuitable for mathematics. Their formal proofs are unreadable."[17]. *Natural proof assistants* intend to bridge the wide gap between intuitive mathematical texts and the formal rigour of logical calculi. This requires in particular

- input languages close to the mathematical vernacular, including symbolic expressions;
- text structurings like the axiom-definition-theorem-proof schema;
- natural argumentative phrases for various proof tactics;
- familiar logics and mathematical ontologies;
- strong automatic theorem proving to fill in obvious proof details;
- an intuitive editor for text and theory development which interactively integrates the checking process and guides formalization.

The Naproche proof assistant stems from two long-term efforts devoted to these goals: the Evidence Algorithm (EA) / System for Automated Deduction (SAD) projects at the universities of Kiev and Paris [15, 16, 19, 20], and the Naproche project at Bonn [11, 3, 4, 9]. The ForTheL input language of SAD has been extended and embedded in LaTeX, allowing mathematical typesetting; the original proof-checking mechanisms have been made more efficient and varied. Moreover, Naproche has been integrated into the Isabelle Proof Interactive Development Environment (Isabelle/PIDE) [21] which supports interactive editing and checking of mathematical texts. Naproche, however, is not connected to the standard logics implemented in Isabelle. Some comprehensive readable formalizations at the level of undergraduate mathematics have been undertaken in Naproche and are available online [2].

Naproche uses classical first-order logic as its underlying logic (following the SAD approach), giving direct access to strong ATPs like E [18]. The input language ForTheL has been carefully designed to approximate the weakly typed natural language of mathematics whilst being efficiently translatable to the language of first-order logic. In ForTheL, standard mathematical types are called *notions*, and these are internally represented as unary predicates with first-order definitions. This leads to a flexible type system where number systems can be cumulative ($\mathbb{N} \subseteq \mathbb{R}$), and notions can depend on parameters (subsets of $\mathbb{N}$, divisors of n).

The first-order language of notions, constants, relations, and functions is introduced and extended by *signature* and *definition* commands as in this example which is unrelated to the checkerboard formalization:

Signature. A real number is a notion.

Definition. $\mathbb{R}$ is the set of real numbers.

Signature. 0 is a real number.

Definition. A nonzero number is a real number that is not equal to 0.

Signature. Let x, y be real numbers. Let y be a nonzero number. $\frac{x}{y}$ is a real number.

ForTheL requires that all variables and terms have some type in the declared system of notions. As part of proof-checking, Naproche will first check the type-correctness of all terms in a ForTheL statement before proceeding to the further processing of the statement. This type-checking phase is called *ontological checking*. Due to the first-order dependent definition of types, type-checking is more involved than in static type systems. On the other hand this approach naturally supports standard partial functions like $\frac{x}{y}$ for reals x, y. The guard $y \neq 0$ will only be checked after the checking process has gone through all the text preceding the term $\frac{x}{y}$ in question.

Our formalization of the *Mutilated Checkerboard Problem* in the next section can be read as a gentle introduction to ForTheL, which declares a language of checkerboards, dominoes and tilings, postulates some axioms, and proceeds to show simple propositions which result in the final non-tileability.

The formalization is carried out in the Isabelle 2021 PIDE which includes a Naproche component. Further mathematical and logical particulars are contained in the literate comments in the formalization; technical information on the use of Naproche and Isabelle is given in Section 4.

3 The Mutilated Checkerboard Problem Formalized in Naproche

3.1 Introduction
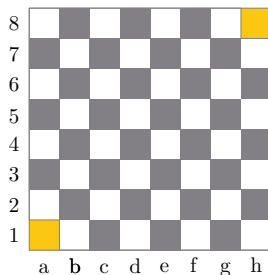
The *Mutilated Checkerboard Problem* asks the following:

Consider an 8×8 checkerboard with two diametrically opposed corners ■ removed, leaving 62 squares.

Is it possible to place 31 dominoes of size 2×1 so as to cover all 62 remaining squares?

Max Black proposed this problem in his book *Critical Thinking* (1946). It was later discussed by Martin Gardner in his *Scientific American* column, *Mathematical Games*. John McCarthy, one of the founders of Artificial Intelligence, described it as a *Tough Nut for Proof Procedures* and discussed fully automatic or interactive proofs of the solution.

3.2 Setting up the checkerboard

We introduce *types* (or *notions*) and constants to model checkerboards as a Cartesian product of *ranks* $1, 2, \ldots, 8$ and *files* $a, b, \ldots, h$, where we follow standard checkerboard notation. In future versions of Naproche these signature declarations should be grouped as a single declaration of an inductively defined set, allowing phrasings such as *1, 2, 3, 4, 5, 6, 7, 8 are ranks*. Note that the effect of signature declarations is to extend the underlying first-order language. Naproche treats $1, 2, \ldots$ as new constant symbols which have no connection to the homonymous integers and in particular do not carry assumptions about distinctness. Here our approach diverges from McCarthy's who employs integers modulo 8, but this would require us to formalize part of the theory of $\mathbb{Z}/8\mathbb{Z}$.

Naproche allows to group elements into *classes* and *sets* as long as they are *setsized* (informally also called *small*).

Signature 1. A rank is a notion.

Let r, s denote ranks.

Axiom 2. r is setsized.

Signature 3. 1 is a rank.

Signature 4. 2 is a rank.

Signature 5. 3 is a rank.

Signature 6. 4 is a rank.

Signature 7. 5 is a rank.

Signature 8. 6 is a rank.

Signature 9. 7 is a rank.

Signature 10. 8 is a rank.

Definition 11. $\mathbf{R} = \{1, 2, 3, 4, 5, 6, 7, 8\}$.

Signature 12. A file is a notion.

Let f, g denote files.

Axiom 13. f is setsized.

Signature 14. a is a file.

Signature 15. b is a file.

Signature 16. c is a file.

Signature 17. d is a file.

Signature 18. e is a file.

Signature 19. f is a file.

Signature 20. g is a file.

Signature 21. h is a file.

Definition 22. $\mathbf{F} = \{a, b, c, d, e, f, g, h\}$.

Signature 23. A square is a notion.

Axiom 24. (f, r) is a square.

Let v, w, x, y, z denote squares.

Is there a set of all squares? This may not be true for an arbitrary notion, but it is true for squares, so we assume it as an axiom. Note that we can always form the class C of all inhabitants of a notion as long as $x \in C$ can only be true for setsized x. Morse and Kelley [6, 14] use the same approach in their axiomatization of set theory.

Definition 25. $\mathbf{C}$ is the class of squares x such that $x = (f, r)$ for some element f of $\mathbf{F}$ and some element r of $\mathbf{R}$.

Axiom 26. $\mathbf{C}$ is a set.

3.3   Preliminaries about sets and functions

We enrich the small built-in set theory with further properties and axioms that will be used in the course of our argument. To keep the document fully self-contained we formulate the necessary definitions and axioms ourselves. Note that there are many degrees of freedom in picking an axiomatic setting.

In Definition 27, we do not define the *relation* $A \subseteq B$, but rather the *type* of subsets of a given set B. The type is parametrized by a member of the type *set*. This is a kind of dependent type, depending on B, in the sense of dependent type theory. Therefore the wording *A [...] is a [...]* is used in Definition 27 as well as in Definition 29. More specifically, we use a form with an argument prefixed by *of*, a so-called *of*-notion. This allows certain grammatical variants and constructs like *B has a subset A* or *every subset of A satisfies [...]*. Note that an *element* similarly is an *of*-notion so that one could write phrases like *x is an element of B* or even more complicated ones like *for all nonequal elements A, B of C*. One could also have defined the *relation* $A \subseteq B$ by a statement of the form: $A \subseteq B$ *iff [...]*. Definition 30 defines disjointness in that format.

Let A, B, C denote sets.

Definition 27. A subset of B is a set A such that every element of A is an element of B.

Axiom 28 (Extensionality). If A is a subset of B and B is a subset of A then $A = B$.

Definition 29. A proper subset of B is a subset A of B such that $A \neq B$.

Definition 30. A is disjoint from B iff there is no element of A that is an element of B.

Definition 31. A family is a set F such that every element of F is a set.

Definition 32. A disjoint family is a family F such that A is disjoint from B for all nonequal elements A, B of F.

Definition 33. $B \cap C = \{x \in B \mid x \in C\}$.

Definition 34. $B \setminus C = \{x \in B \mid x \notin C\}$.

The notion of *object* is the built-in *largest notion*, containing all other notions. Also note that the proof of the lemma below really is omitted and not merely hidden: with its internal "reasoner" and in non-trivial cases with the help of automated theorem provers such as E, Naproche can accept some theorems without any additional argumentation.

Lemma 35. Every set is an object.

The built-in ordered pair notation that we already used in the first subsection does not include the universal property of ordered pairs, so we postulate it as an axiom.

Axiom 36. Let $\alpha, \beta, \gamma, \delta$ be objects. If $(\alpha, \beta) = (\gamma, \delta)$ then $\alpha = \gamma$ and $\beta = \delta$.

(Unary) functions are built into Naproche; $F(t)$ denotes the application of a function F to an argument t and $\mathrm{Dom}(F)$ stands for the domain of F. In our exposition we shall use functions to compare cardinalities of black and white squares. As with sets, we introduce some further properties of functions.

Let F, G denote functions.

Definition 37. $F : A \to B$ iff $\mathrm{Dom}(F) = A$ and $F(x)$ is an element of B for all elements x of A.

Bijective functions are the basis of the modern theory of cardinalities; sets have the same cardinality iff there is a bijection between them.

Definition 38. $F : A \leftrightarrow B$ iff $F : A \to B$ and there exists G such that $G : B \to A$ and (for all elements x of A we have $G(F(x)) = x$) and (for all elements y of B we have $F(G(y)) = y$).

3.4 Cardinalities of Finite Sets

Definition 39. A is equinumerous with B iff there is F such that $F : A \leftrightarrow B$.

Lemma 40. Assume that A is equinumerous with B. Then B is equinumerous with A.

Lemma 41. Assume that A is equinumerous with B and B is equinumerous with C. Then A is equinumerous with C.

Proof. Take a function F such that $F : A \leftrightarrow B$. Take a function G such that $G : B \to A$ and (for all elements x of A we have $G(F(x)) = x$) and (for all elements y of B we have $F(G(y)) = y$). Take a function H such that $H : B \leftrightarrow C$. Take a function I such that $I : C \to B$ and (for all elements x of B we have $I(H(x)) = x$) and (for all elements y of C we have $H(I(y)) = y$). Define $J(x) = H(F(x))$ for x in A. $J : A \leftrightarrow C$. Indeed define $K(y) = G(I(y))$ for y in C.   ◀

For the finite checkerboard problem we only need to consider finite sets. We can thus assume that all sets considered are finite, and then we have the following finiteness axiom:

Axiom 42. If A is a proper subset of B then A is not equinumerous with B.

3.5 The Mutilated Checkerboard

Defining the mutilated checkerboard is straightforward: we simply remove the two corners.

Definition 43. $\mathbf{C}' = \{(a, 1), (h, 8)\}$.

Definition 44. $\mathbf{M} = \mathbf{C} \setminus \mathbf{C}'$.

Let the mutilated checkerboard stand for $\mathbf{M}$.

3.6 Dominoes

To define dominoes, we introduce concepts of adjacency by first declaring new relations and then axiomatizing them. As usual, chaining of relation symbols indicates a conjunction.

Signature 45. r is vertically adjacent to s is a relation.
    Let $r \sim s$ stand for r is vertically adjacent to s.

Axiom 46. If $r \sim s$ then $s \sim r$.

Axiom 47. $1 \sim 2 \sim 3 \sim 4 \sim 5 \sim 6 \sim 7 \sim 8$.

Signature 48. f is horizontally adjacent to g is a relation.

Let $f \sim' g$ stand for f is horizontally adjacent to g.

Axiom 49. If $f \sim' g$ then $g \sim' f$.

Axiom 50. a $\sim'$ b $\sim'$ c $\sim'$ d $\sim'$ e $\sim'$ f $\sim'$ g $\sim'$ h.

Definition 51. x is adjacent to y iff there exist f, r, g, s such that $x = (f, r)$ and $y = (g, s)$ and $((f = g$ and r is vertically adjacent to $s)$ or $(r = s$ and f is horizontally adjacent to $g))$.

Definition 52. A domino is a set D such that $D = \{x, y\}$ for some adjacent squares x, y.

3.7 Domino Tilings

Definition 53. A domino tiling is a disjoint family T such that every element of T is a domino.

Let A denote a subset of **C**.

Definition 54. A domino tiling of A is a domino tiling T such that for every square x x is an element of A iff x is an element of some element of T.

We shall prove:

Theorem. The mutilated checkerboard has no domino tiling.

3.8 Colours

We shall solve the mutilated checkerboard problem by a cardinality argument. Squares on an actual checkerboard are coloured black and white and we can count colours on dominoes and on the mutilated checkerboard **M**.

The introduction of colours can be viewed as a creative move typical of mathematics: changing perspectives and introducing aspects that are not part of the original problem. The mutilated checkerboard was first discussed under a cognition-theoretic perspective: can one solve the problem *without* inventing new concepts and completely stay within the realm of squares, subsets of the checkerboard and dominoes.

Signature 55. x is black is a relation.

Signature 56. x is white is a relation.

Axiom 57. x is black iff x is not white.

Axiom 58. If x is adjacent to y then x is black iff y is white.

Axiom 59. $(a, 1)$ is black.

Axiom 60. $(h, 8)$ is black.

Definition 61. **B** is the class of black elements of **C**.

Definition 62. **W** is the class of white elements of **C**.

Lemma 63. **B** is a set.

Lemma 64. **W** is a set.

3.9 Counting Colours on Checkerboards

The original checkerboard has an equal number of black and white squares. Since our setup does not include numbers for counting, we rather work with equinumerosity. The following argument formalizes that we can invert the colours of a checkerboard by swapping the files a and b, c and d, and so on. We formalize swapping by a first-order function symbol Swap.

Signature 65. Let x be an element of **C**. Swap x is an element of **C**.

Let t denote an element of **R**.

Axiom 66. $\mathrm{Swap}(\mathrm{a}, t) = (\mathrm{b}, t)$ and $\mathrm{Swap}(\mathrm{b}, t) = (\mathrm{a}, t)$.

Axiom 67. $\mathrm{Swap}(\mathrm{c}, t) = (\mathrm{d}, t)$ and $\mathrm{Swap}(\mathrm{d}, t) = (\mathrm{c}, t)$.

Axiom 68. $\mathrm{Swap}(\mathrm{e}, t) = (\mathrm{f}, t)$ and $\mathrm{Swap}(\mathrm{f}, t) = (\mathrm{e}, t)$.

Axiom 69. $\mathrm{Swap}(\mathrm{g}, t) = (\mathrm{h}, t)$ and $\mathrm{Swap}(\mathrm{h}, t) = (\mathrm{g}, t)$.

The somewhat unsightly case-splits in the following lemmas are necessary to guide the prover, since Naproche as yet has no concept of finite data types and only features well-ordered induction. As a consolation price, we can omit the last case.

Lemma 70. Let x be an element of $\mathbf{C}$. $\mathrm{Swap}\,x$ is adjacent to x.

Proof. Take f, r such that $x = (f, r)$. r is an element of $\mathbf{R}$. Case $f = \mathrm{a}$. End. Case $f = \mathrm{b}$. End. Case $f = \mathrm{c}$. End. Case $f = \mathrm{d}$. End. Case $f = \mathrm{e}$. End. Case $f = \mathrm{f}$. End. Case $f = \mathrm{g}$. End. ◀

Swap is an involution.

Lemma 71. Let x be an element of $\mathbf{C}$. $\mathrm{Swap}(\mathrm{Swap}\,x) = x$.

Proof. Take f, r such that $x = (f, r)$. r is an element of $\mathbf{R}$. Case $f = \mathrm{a}$. End. Case $f = \mathrm{b}$. End. Case $f = \mathrm{c}$. End. Case $f = \mathrm{d}$. End. Case $f = \mathrm{e}$. End. Case $f = \mathrm{f}$. End. Case $f = \mathrm{g}$. End. ◀

Lemma 72. Let x be an element of $\mathbf{C}$. x is black iff $\mathrm{Swap}\,x$ is white.

Using Swap we can define a witness of $\mathbf{B} \leftrightarrow \mathbf{W}$.

Lemma 73. $\mathbf{B}$ is equinumerous with $\mathbf{W}$.

Proof. Define $F(x) = \mathrm{Swap}\,x$ for x in $\mathbf{B}$. Define $G(x) = \mathrm{Swap}\,x$ for x in $\mathbf{W}$. Then $F : \mathbf{B} \to \mathbf{W}$ and $G : \mathbf{W} \to \mathbf{B}$. For all elements x of $\mathbf{B}$ we have $G(F(x)) = x$. For all elements x of $\mathbf{W}$ we have $F(G(x)) = x$. $F : \mathbf{B} \leftrightarrow \mathbf{W}$. ◀

Given a domino tiling one can also swap the squares of each domino, leading to similar properties.

Signature 74. Assume that T is a domino tiling of A. Let x be an element of A. $\mathrm{Swap}_T^A(x)$ is a square y such that there is an element D of T such that $D = \{x, y\}$.

Lemma 75. Assume that T is a domino tiling of A. Let x be an element of A. Then $\mathrm{Swap}_T^A(x)$ is an element of A.

Proof. Let $y = \mathrm{Swap}_T^A(x)$. Take an element D of T such that $D = \{x, y\}$. ◀

Swapping dominoes is also an involution.

Lemma 76. Assume that T is a domino tiling of A. Let x be an element of A. Then $\mathrm{Swap}_T^A(\mathrm{Swap}_T^A(x)) = x$.

Proof. Let $y = \mathrm{Swap}_T^A(x)$. Take an element Y of T such that $Y = \{x, y\}$. Let $z = \mathrm{Swap}_T^A(y)$. Take an element Z of T such that $Z = \{y, z\}$. Then $x = z$. ◀

Lemma 77. Assume that T is a domino tiling of A. Let x be a black element of A. Then $\mathrm{Swap}_T^A(x)$ is white.

Proof. Let $y = \mathrm{Swap}_T^A(x)$. Take an element Y of T such that $Y = \{x, y\}$.   ◀

3.10   The Theorem

We can easily show that a domino tiling involves as many black as white squares.

Lemma 78. Let T be a domino tiling of A. Then $A \cap \mathbf{B}$ is equinumerous with $A \cap \mathbf{W}$.

Proof. Define $F(x) = \mathrm{Swap}_T^A(x)$ for x in $A \cap \mathbf{B}$. Define $G(x) = \mathrm{Swap}_T^A(x)$ for x in $A \cap \mathbf{W}$. Then $F : A \cap \mathbf{B} \to A \cap \mathbf{W}$ and $G : A \cap \mathbf{W} \to A \cap \mathbf{B}$. For all elements x of $A \cap \mathbf{B}$ we have $G(F(x)) = x$. For all elements x of $A \cap \mathbf{W}$ we have $F(G(x)) = x$. $F : A \cap \mathbf{B} \leftrightarrow A \cap \mathbf{W}$.   ◀

In mutilating the checkerboard, one only removes black squares

Lemma 79. $\mathbf{M} \cap \mathbf{W} = \mathbf{W}$.

Proof. $\mathbf{M} \cap \mathbf{W}$ is a subset of $\mathbf{W}$. $\mathbf{W}$ is a subset of $\mathbf{M}$. Proof. Let x be an element of $\mathbf{W}$. $x \neq (\mathrm{a}, 1)$ and $x \neq (\mathrm{h}, 8)$. Indeed $(\mathrm{h}, 8)$ is black. End.   ◀

Now the theorem follows by putting together the previous cardinality properties. Note that the phrasing *[...] has no domino tiling* in the theorem is automatically derived from the definition of *a domino tiling of [...]*.

Theorem 80. The mutilated checkerboard has no domino tiling.

Proof. Proof by contradiction. Assume T is a domino tiling of $\mathbf{M}$. $\mathbf{M} \cap \mathbf{B}$ is equinumerous with $\mathbf{M} \cap \mathbf{W}$. Indeed $\mathbf{M}$ is a subset of $\mathbf{C}$. $\mathbf{M} \cap \mathbf{B}$ is equinumerous with $\mathbf{W}$. $\mathbf{M} \cap \mathbf{B}$ is equinumerous with $\mathbf{B}$. Contradiction. Indeed $\mathbf{M} \cap \mathbf{B}$ is a proper subset of $\mathbf{B}$.   ◀

4   Comments on the Formalization

We use ℕaproche within the current release of the Proof Interactive Development Environment (PIDE) Isabelle 2021 [5]. Isabelle 2021 is available for the operating systems Linux, Windows, and macOS. The distribution can be unpacked somewhere in one's home folder and started by clicking on the Isabelle executable in the Isabelle folder. The *Documentation* panel contains a tutorial on ℕaproche, which links to a standalone version of our formalization called `checkerboard.ftl.tex`. Opening a `.ftl` or `.ftl.tex` file in Isabelle/PIDE will automatically activate its parsing and checking. Files in `.ftl.tex` format can be typeset by LaTeX provided that text like the above **Signatures** or **Definitions** are entered in a simple LaTeX format. Only text in a `\begin{forthel} ...  \end{forthel}` environment is let through to the checking process. Everything else is treated as a comment by ℕaproche, but may be relevant for LaTeX typesetting.

```
\section{Example of a Signature Command}
\begin{forthel}
    \begin{signature}
        Let $x,y$ be real numbers. Let $y$ be a nonzero number.
        $\frac{x}{y}$ is a real number.
    \end{signature}
\end{forthel}
```

Isabelle/PIDE can show pop-up first-order translations of statements while hovering above them and indicates checking progress with coloured backgrounds. Results of checking and error messages are shown in an output window.

Checking the formalization takes roughly one to two minutes, assuming somewhat up-to-date hardware. Omitting proofs (as in Lemma 40) is convenient and concise, but significantly increases the checking time, since E has to perform a considerable number of sledgehammer-like proof searches. Lemma 40 also shows that automated theorem provers and humans have different strengths. The result is immediate from the definitions to human readers. Conversely, sometimes it is be better to continue spelling out the details of a proof even after the computer accepts it, to help the human reader understand the rest of the proof.

5   Perspectives

The readability and naturalness of non-trivial texts which proof-check in the current, still modest Naproche system call for a significant extension of this project. For Naproche to become a true assistant in mathematical research and teaching, ad hoc methods have to be replaced by professional approaches and tools:

- the input language ForTheL has to be extended for wide mathematical coverage, informed by typical mathematical texts; ForTheL needs a formal grammar and vocabulary to be processed by strong linguistic methods; the vocabulary may also encompass standard (LaTeX) symbols and possibly contain semantic information;
- logical processing has to be geared to the strengths of current automated theorem provers; *Sledgehammer*-like methods should provide efficient premise selection in large texts and theories (see also [1] for a discussion of hammers);
- the creation of libraries of ForTheL documents requires import and export mechanisms corresponding to quoting and referencing in the mathematical literature;
- proof-checking of documents should be organized as an enrichment of ForTheL texts by the generated translations and derivations; these should be stored as auxiliary files to minimize re-checking or to assemble derivations into a correctness certificate for the text;
- the natural text processing of Naproche should be interfaced with other proof assistants to leverage their strengths and libraries; we have begun work on a Naproche → Lean translation;
- the use and user experience of natural proof checking in teaching and research have to be studied and taken care of in the further development.

References

1   Jasmin C. Blanchette, Cezary Kaliszyk, Lawrence C. Paulson, and Josef Urban. Hammering towards QED. *Journal of Formalized Reasoning*, 9(1):101–148, January 2016. `doi:10.6092/issn.1972-5787/4593`.

2   Naproche contributors. FLib. URL: `https://github.com/naproche-community/FLib`.

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

4   Steffen Frerix and Peter Koepke. Automatic proof-checking of ordinary mathematical texts. *Proceedings of the Workshop Formal Mathematics for Mathematicians*, 2018.

5   Isabelle contributors. The Isabelle2021 release, February 2021. URL: `https://isabelle.in.tum.de`.

6   John L. Kelley. *General Topology*. Springer-Verlag New York, 1975.

7   Manfred Kerber and Martin Pollet. A tough nut for mathematical knowledge management. In Michael Kohlhase, editor, *Mathematical Knowledge Management*, pages 81–95, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. `doi:10.1007/11618027_6`.

8   Donald E. Knuth. *Literate Programming*. Center for the Study of Language and Information, 1992.

9   Peter Koepke. Textbook mathematics in the Naproche-SAD system. *Joint Proceedings of the FMM and LML Workshops*, 2019.

10   Peter Koepke, Anton Lorenzen, and Adrian De Lon. Interpreting mathematical texts in Naproche-SAD. In *Intelligent Computer Mathematics: 13th International Conference, CICM 2020*, pages 284–289. Springer, 2020.

11   Daniel Kühlwein, Marcos Cramer, Peter Koepke, and Bernhard Schröder. The Naproche system, January 2009.

12   John McCarthy. Tough nut for proof procedures, 1964. Stanford AI Memo.

13   John McCarthy. The mutilated checkerboard in set theory, 2001.

14   Anthony Perry Morse; Trevor J McMinn. *A theory of sets*. New York ; London : Academic press, 1965.

15   Andrei Paskevich. *Méthodes de formalisation des connaissances et des raisonnements mathématiques: aspects appliqués et théoriques*. PhD thesis, Université Paris 12, 2007.

16   Andrei Paskevich. The syntax and semantics of the ForTheL language, 2007.

17   Lawrence C. Paulson. ALEXANDRIA: Large-scale formal proof for the working mathematician. URL: `https://www.cl.cam.ac.uk/~lp15/Grants/Alexandria/`.

18   Stephan Schulz. The E theorem prover. URL: `https://eprover.org`.

19   Konstantin Verchinine, Alexander Lyaletski, and Andrei Paskevich. System for automated deduction (SAD): a tool for proof verification. *Automated Deduction–CADE-21*, pages 398–403, 2007. `doi:10.1007/978-3-540-73595-3_29`.

20   Konstantin Verchinine, Alexander Lyaletski, Andrei Paskevich, and Anatoly Anisimov. On correctness of mathematical texts from a logical and practical point of view. In *International Conference on Intelligent Computer Mathematics*, pages 583–598. Springer, 2008. `doi:10.1007/978-3-540-85110-3_47`.

21   Makarius Wenzel. Interaction with formal mathematical documents in Isabelle/PIDE, 2019. `arXiv:1905.01735`.