Automatic Proof-Checking of Ordinary Mathematical Texts

Steffen Frerix and Peter Koepke

University of Bonn

Abstract

The System for Automated Deduction (SAD) by Andrei Paskevich et. al. is an automatic proof-checker that can process fairly natural mathematical input statements and short texts. We have recently made significant improvements to the system: speed-ups of the checking algorithms allow handling of chapter-sized texts with an argumentative granularity comparable to textbook mathematics; extensions of the input language provide native support of basic notions like sets and functions; SAD input can now be written in a LaTeX style which typesets like ordinary mathematical text. Based on experiences and examples so far we expect to be able to write natural textbook-style mathematics which can be automatically checked for proof correctness. Actually this paper is a proof-checked SAD document itself.

1 Introduction

The ordinary language of mathematics (OLM) has evolved to record and communicate mathematical statements and arguments succinctly and unambiguously. OLM combines natural language with symbolic terms and statements. Software assisting mathematicians should ideally communicate in the common language of mathematics: automatic provers should obtain their tasks as OLM statements; proof checkers should check the correctness of proof texts written in OLM. Current proof assistants, however, still employ input languages that rather resemble computer code.

Linguistic studies of OLM have shown that the techniques of formal, computer-supported linguistics are applicable and that parse results are better than for arbitrary natural language because of a strong tendency in mathematics towards text structuring, simplicity, and unambiguity [**?**, **?**, **?**]. These investigations suggest that *approximations* to OLM may be parsed faithfully on the sentence and text level. A formal grammar can define a *controlled natural language* (CNL) of acceptable sentences which are simultaneously man- and machine-readable.

2 Improving the System for Automated Deduction

An early linguistically informed programme for mathematical proof checking called Evidence Algorithm (EA) was initiated by Victor Glushkov [**?**]. Proofs are viewed as arrangements of evident facts stated in ordinary mathematical language and logically connected by evident proof rules. These ideas were first implemented in Kiev, including the development of a semi-natural *Formula Theory Language* (ForTheL). The programme culminated in the System for Automatic Deduction (SAD) which was the doctoral project of Andrei Paskevich [**?**], see also http://nevidal.org/sad.en.html.

The SAD proof checker is an impressive proof of concept checking some elegant mathematical "miniatures" but leaving much room for further improvements. Revising the checking algorithms we were able to cut down

In: O. Hasan, C. Kaliszyk, A. Naumowicz (eds.): Proceedings of the Workshop Formal Mathematics for Mathematicians (FMM), Hagenberg, Austria, 13-Aug-2018, published at http://ceur-ws.org

some proof-checking times from minutes to seconds. This allows the handling of chapter-sized texts and libraries of interlinked texts. The rudimentary support of sets was replaced by stronger mechanisms for sets and functions. The new SAD accepts LaTeX as an input format. More information on technical aspects can be found in the Aussois??/Oxford??-abstracts.

3 An SAD example text

Actually this paper is itself a LaTeX-document which is accepted by SAD. The ForTheL content of this paper is included in a forthel environment, marked by a vertical line in the margin. Only text embedded within such environments is parsed and proof-checked by SAD. We present an actual working example from complex analysis which proves the familiar maximum principle for holomorphic functions from other basic theorems.

4 *Ad hoc* Preliminaries

[number/-s][ontored on][checkontored on]
Let the *domain* of f stand for $\mathrm{Dom}(f)$. Let z is *in* M stand for z is an element of M.
Let M *contains* z stand for z is in M. Let $z \in M$ stand for z is in M.
Let f denote a function. Let M denote a set.

Definition 1. *A* subset *of* M *is a set* N *such that every element of* N *is an element of* M.

Definition 2. *Assume* M *is a subset of the domain of* f. $f[M] = \{f[x] \mid x \in M\}$.

Signature 1. *A* complex number *is a notion. Let* z, w *denote complex numbers.*

Axiom 1. *Every element of* $\mathrm{Dom}(f)$ *is a complex number and for every element* z *of* $\mathrm{Dom}(f)$ $f[z]$ *is a complex number.*

Axiom 2. *Every element of* M *is a complex number.*

Signature 2. *A* real number *is a notion. Let* x, y *denote real numbers.*

Signature 3. $|z|$ *is a real number.*

Signature 4. x *is* positive *is an atom. Let* ϵ, δ *denote positive real numbers.*

Signature 5. $x < y$ *is an atom. Let* $x \leq y$ *stand for* $x = y$ *or* $x < y$.

Axiom 3. $x < y \to \neg y < x$.

Signature 6. f *is* holomorphic *is an atom.*

Signature 7. $B_\epsilon(z)$ *is a set that contains* z.

Axiom 4. $|z| < |w|$ *for some element* w *of* $B_\epsilon(z)$.

Definition 3. M *is* open *iff for every element* z *of* M *there exists* ϵ *such that* $B_\epsilon(z)$ *is a subset of* M.

Axiom 5. $B_\epsilon(z)$ *is open.*

Definition 4. *A* local maximal point *of* f *is an element* z *of the domain of* f *such that there exists* ϵ *such that* $B_\epsilon(z)$ *is a subset of the domain of* f *and* $|f[w]| \leq |f[z]|$ *for every element* w *of* $B_\epsilon(z)$.

Definition 5. *Let* U *be a subset of the domain of* f. f *is* constant on U *iff there exists* z *such that* $f[w] = z$ *for every element* w *of* U. *Let* f *is* constant *stand for* f *is constant on the domain of* f.

5 Basic Theorems about Holomorphic Functions

We axiomatically assume some standard theorems of complex analysis.

Axiom 6 (OpenMappingTheorem)**.** *Assume f is holomorphic and $B_\epsilon(z)$ is a subset of the domain of f. If f is not constant on $B_\epsilon(z)$ then $f[B_\epsilon(z)]$ is open.*

Signature 8. *A* region *is an open set.*

Axiom 7 (IdentityTheorem)**.** *Assume f is holomorphic and the domain of f is a region. Assume that $B_\epsilon(z)$ is a subset of the domain of f. If f is constant on $B_\epsilon(z)$ then f is constant.*

6 The Maximum Principle

The principle can be easily derived from the basic theorems.

Theorem 1. *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. The complete file starts out with basic definitions and axioms for the argument and is about three times the size of the excerpt.

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 the example and would have to be amended if we want to capture the situation in more generality: In the text, e.g., ε and δ are ranging over *positive real numbers*. This notion has only been introduced in the ad hoc preliminaries, and positivity has not been connected with the $<$-relation since that is not required for the proof of the maximum principle. A more comprehensive text would of course have to fix that liberal approach.

6. Notions provide soft-typing of 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. 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 Soft typings

Notions play a central role in the ForTheL language. They provide the basic "types" that an object may have in a text. In the example above we find notions such as "set", "complex number", "real number" and "positive real number". A variable must be declared to belong to some notion before it can be used. Moreover, we can not quantify unboundedly but only over notions. The translation to first-order logic is done using type guards. Depending on the number of variables, the formulas obtained can therefore become quite large, which burdens the ATP used to discharge proof obligations. A smart processing of notions has been an essential component for increasing the power of SAD.

Mathematians may use notions to guide their attention to a "small world". A theorem about graphs will usually not be considered when dealing with a problem of number theory. Being integer is then an ontological property rather than a logical one. Within the world of integers it cannot substantially contribute to an argument. We have developed a method to detect such notions typings and subsequently reduce the proof task for the ATP.

A key property that a ForTheL text may have is *ontological correctness*. Roughly speaking this means that every application of a predicate or function symbol is well-defined. When introducing a symbol to the signature, an author makes certain assumptions on its arguments, which we call domain conditions. These assumptions may be hidden in a text through the use of pretyped variables. For example, in Definition 2 in the above text we introduce the symbol $\cdot[\cdot](f, M)$ with the domain conditions $\mathrm{aSet}(M), \mathrm{aFunction}(f), \mathrm{aSubsetOf}(M, \mathrm{Dom}(f))$.

If φ is a formula occuring in the first-order image of a ForTheL text, we can determine for every variable x occuring in φ the most general domain that x is assumed to be in. These domain assumptions are then deleted from φ. Let us demonstrate the reduction on a short example.

> [number/-s]
>
> **Signature.** *A* real number *is a notion.*
>
> Let x, y denote real numbers.
>
> **Signature.** *$x * y$ is a real number.*
>
> **Signature.** *x is* nonzero *is an atom.*
>
> **Signature.** *Assume x is nonzero. x^{-1} is a real number.*
>
> **Axiom.** *Assume x and y are nonzero. $x * y$ is nonzero.*
>
> **Axiom.** *Assume x is nonzero. x^{-1} is nonzero.*

The first order images of the two axioms are

$$(\mathrm{aRealNumber}(x) \wedge \mathrm{aRealNumber}(y)) \to (\mathrm{isNonzero}(x) \wedge \mathrm{isNonzero}(y)) \to \mathrm{isNonzero}(x * y)$$
$$\mathrm{aRealNumber}(x) \to \mathrm{isNonzero}(x) \to \mathrm{isNonzero}(x^{-1}).$$

Being a real number is a domain condition for all the other symbols involved while being nonzero is only a condition for $()^{-1}$. Therefore for the first formula, the most general domain condition is $(\mathrm{aRealNumber}(x) \wedge \mathrm{aRealNumber}(y)$ and for the second formula it is $\mathrm{aRealNumber}(x) \wedge \mathrm{isNonzero}(x)$. We can thus reduce these formulas to

$$\mathrm{isNonzero}(x) \wedge \mathrm{isNonzero}(y) \to \mathrm{isNonzero}(x * y)$$
$$\mathrm{isNonzero}(x^{-1}).$$

This reduction corresponds to usual mathematical thinking. The term x^{-1} will always be nonzero when it is well-defined. We call the procedure ontological reduction.

Such a deletion of disjuncts from a first-order problem is clearly complete. One can moreover show the following theorem.

Theorem 2. *Let T be an ontologically correct ForTheL text. Assume that the first-order image of T has a model in which all domain conditions are non-empty. Then its ontological reduction also has a model.*

In the proof of the theorem, a model for the ontological reduction is constructed by suitably changing the interpretation of symbols outside of their domain, so that domain guards become superfluous. Ontological correctness ensures that after those changes we still have a model of the original text.

The assumption on the non-emptyness of all domain conditions corresponds to non-emptyness of sorts in many-sorted first-order logic. It will in general not be a problem, since usual mathematical symbols are defined for non-empty domains. In case where a text uses hypotheticals to finally show emptyness of a certain class, possibly problematic domain conditions can be excluded from the reduction process to ensure soundness.

Dealing with typings in a first-order setting is not a new problem. In [**?**], multiple encodings of (mono- and polymorphic) many-sorted FOL to pure FOL are developed for Sledgehammer in order to encode Isabelle's type system. Encoding by type guards, the approach formerly used by SAD, was among the worst performing. If one applies ontological reduction to a ForTheL text that describes some problem in (monomorphic) many-sorted FOL, then the result is very close to that obtained by the "featherweigth guards" encoding described in [**?**]. However, our approach is more flexible and therefore better suited for our soft type system. For example we can possibly delete predicates of arity higher than one and negated predicates. Furthermore, we can more adequately deal with the complex relations between ForTheL notions.

8 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 by new grammatical constructs without compromising 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. In our example text, the notion of a *region* 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. To connect our example to such a foundational text requires the implication that a region is open. A systematic study of such relations is required for building larger libraries.

9 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

[Eki15] Burak Ekici. *Certifications of programs with computational effects (Certification de programmes avec des effets calculatoires)*. PhD thesis, Grenoble Alpes University, France, December 2015.

[GCS14] Jason Gross, Adam Chlipala, and David I. Spivak. Experience implementing a performant category-theory library in Coq. In *Proceedings of the 5th ITP, Vienna, Austria*, July 2014.

[Hen08] Christopher Henderson. Generalized abstract nonsense: Category theory and adjunctions. Technical report, University of Chicago, 2008.

[ML71] Saunders Mac Lane. *Categories for the Working Mathematician*. Number 5 in Graduate Texts in Mathematics. Springer-Verlag, 1971.

[TJ16] Amin Timany and Bart Jacobs. Category theory in Coq 8.5. In *Proceedings of the 1st FSCD, Porto, Portugal*, pages 30:1–30:18, June 2016.