Definedness Reasoning in Formal Mathematics and Theorem Proving

Jonas Betzendahl
Dept. of Computer Science
Erlangen
jonas.betzendahl@fau.de

FAU Erlangen-Nürnberg

Motivation

Formal systems are on the rise! From the design of safety-critical hardware chips to operating system kernels, more and more fields and researchers in industry and academia are relying on formal methods to detect errors and verify results.

This trend is also present in mathematics, including both the development of software systems to formalise mathematics in (such as Coq, Isabelle/HOL and numerous others), and the construction and verification of proofs. Successful ventures in the latter category include the proof of the Kepler conjecture and the classification theorem for finite simple groups.

One part of mathematics that has so far not been widely incorporated into formal systems is undefinedness, the notion of terms or expressions that are well-formed but cannot be assigned a value. This typically arises when applying a *partial* function (or predicate, or operator, ...) to a value that lies outside of its domain.

Many formal systems circumnavigate the issue entirely by defaulting back to option types for all or most results of function application or by allowing only functions that are provably total and so do not give rise to undefinedness.

There are many reasons to take undefinedness seriously, however. The problem has been discussed academically for more than a century and it is clear that the assumption that all expressions can be treated as defined (even though it may be terribly convenient) is ultimately indefensible both for mathematical terms and natural language. Equation 1 is especially interesting because it clearly demonstrates the split opinions about how to deal with undefinedness.

$$\forall x, y, z : \mathbb{R} \,.\, x = \frac{z}{y} \Rightarrow x \cdot y = z \tag{1}$$

This equation is brought up in more than one context in the available literature, with opposite connotations. One source claims that Equation 1 should be considered true as is, without qualification as to $z \neq 0$. Another paper by different authors claims that without such a qualification, it would simply be false. Interestingly, *both* sources refer back to "mathematical consensus" in one form or another to justify their particular position. It becomes obvious from this, that there is more than one perspective on what undefinedness "ought" to be and how to get there.

During my PhD, I want to contribute to the development of a general and foundation-independent approach to handling undefinedness in formal systems for mathematics. Undefined terms are common in mathematics as

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

practised on paper or blackboard by mathematicians, yet few of the many systems for formalising mathematics that exist today deal with it in a principled way or allow explicit reasoning with undefined terms.

The other topic I hope to tackle that of a soft type system for mathematics. Like undefinedness, soft type systems have the potential to capture the realities of mathematics as performed by mathematicians. A structure is defined one way but can later be proven to also meet the criteria (axioms) of another structure. Both of these perspectives (and any that follow) can be used in proofs and for further definitions. This has been one of the reasons behind the success of the remarkable `Mizar` library.

Yet, in that system, the user is committed to one particular meta-logic (although the choice of axiom system is technically free, even if heavily influenced by the fact that the library uses Tarski-Grothendiek set theory axioms). It would be beneficial if they could switch between logics easily without having to give up the benefits of a soft type system. As with undefinedness, I want to make it possible for soft types to be one of many building blocks a user (of MMT) can (easily and without much hassle) chose from to build exactly the right logic for any given effort.

We expect both of these endeavours to generate a lot of relatively minor proof obligations (terms being defined, complying with certain type predicates, ...) that are unavoidable in a rigorous environment, yet do not contribute substantially to the creative effort in a mathematical proof. These "busywork" obligations need to be dealt with, but ideally not manually by human users of the system.

So, furthermore, I hope to extend the MMT framework with additional capabilities for automated and interactive theorem proving, both for reasoning in and outside the domain of undefinedness, with a special focus on the automated reasoning necessary in the context of type-checking.

Related Work

As mentioned above, there are a lot of systems for formalising mathematics, both in an out of development, such as Coq, Lean, Agda, HOL-Light, Isabelle/HOL, and many others. They all have their strengths and weaknesses and they all have lessons to teach that will hopefully be informative and of influence to my work. Let me single out two of them that are especially relevant to the topics at hand.

The `Mizar` system (invented by Andrzej Trybulec) is one of the most prominent and successful instances of a soft type system in the realm of formal mathematics with an astonishing amount of formalised knowledge readily available.

First-class undefinedness reasoning as well as the ability to switch easily between meta-logics for any given effort are, however, unlikely to be introduced.

The `IMPS` (developed by Farmer, Guttman and Thayer) stands out as one of the only proof assistants to make reasoning about (un)definedness a first-class concern. It was also one of the first systems to bring the *small theories* approach to mathematical formalisation.

However, the system has been out of development for roughly thirty years and so it is not useful as a target for future formalisation efforts as it can not adapt to moving targets of requirements.

Effort & Evaluation

Catching up to the state of the art (i.e. developing a full reasoning system that has all the features we discussed so far) in the timespan of one PhD-thesis is not feasible, so I will focus particularly on the two major building blocks of foundation-independent notions of undefinedness and soft type systems, as well as automated theorem proving support for proof obligations that pop up during the process of type-checking and the best ways to discard them automatically.

MMT as of today has support for annotating declarations so that the simplifier is able to use them as additional rules. However, MMT does not support such annotations for rules that would have premises, only simple equalities. So one task would be to implement support for premises, definedness judgements and other such constructs as well.

One way of evaluating the success of the foundation-independent "building blocks" for undefinedness and soft types that we discussed above, is to re-implement already existing set theories with soft types or logics that introduce undefinedness themselves (such as the logic underlying `IMPS`) in terms of that foundation-independent approach and see whether it can be easily achieved and yields the expected results.