Project Description

Andreas Salhus Bakseter

January 18, 2023

1 Formalization of mathematical problems

When solving mathematical problems, one often uses proofs to assert some claim. We can group proofs into two types; *informal* and *formal* proofs. An informal proof is often written in a natural language, where the truth of the proof is determined by if the reader is convinced by the proof or not.[4]

As a proof grows larger and more complex, it becomes harder to follow, which can ultimately lead to errors in the proofs reasoning. This might cause the whole proof to be incorrect.[3]

2 Proof assistants

A formal proof can be written like a computer program, where all the arguments can be checked mechanically; usually done with a *proof assistant*.
Coq is a proof assistant that enables us to write fomal proofs and verify them. Coq uses type theory to verify proofs, but can also be used as a functional programming language.[2] Other examples of proof assistants include Agda, Isabelle, Lean and HOH.

3 Type theory & propositions as types

Type theory is used to create formal systems that group mathematical objects with similar properties together by assigning them a "type".
Similarily to types in computer programming, we can use types to represent mathematical objects. For example, we can use the type `nat` to represent natural nubers.

The concept of propositions as types sees the proving of a mathematical proposition as the same process as constructing a value of that type. For example, to prove a proposition P whichs states "all integers are divisible by 2", we must construct a value of the type P that shows that this is true for all integers. Since proofs are constructed using logical propositions, we can use this correspondance to model a proof as a typed computer program. The power of this concept comes from the fact that we can use a type checker to verify that our program is typed correctly, and thus that the corresponding proof is valid.

4 Our case

We will use the Coq proof assistant to formalize parts of the proofs of the following paper, *Bezem and Coquand*. This paper actually solves two problems that occur in dependant type systems where typings depend on universe-level constraints.

...more about the paper/theorems

References

[1] Marc Bezem and Thierry Coquand. "Loop-checking and the uniform word problem for join-semilattices with an inflationary endomorphism". In: *Theoretical Computer Science* (2022). ISSN: 0304-3975. DOI: `https://doi.org/10.1016/j.tcs.2022.01.017`. URL: `https://www.sciencedirect.com/science/article/pii/S0304397522000317`.

[2] coq.inria.fr. *A short introduction to Coq*. URL: `https://coq.inria.fr/a-short-introduction-to-coq` (visited on 01/18/2022).

[3] Roxanne Khamsi. *Mathematical proofs are getting harder to verify*. 2006. URL: `https://www.newscientist.com/article/dn8743-mathematical-proofs-getting-harder-to-verify` (visited on 01/18/2022).

[4] Benjamin C. Pierce. *Software Foundations: Volume 1: Logical Foundations*. 2022. URL: `https://softwarefoundations.cis.upenn.edu/current/lf-current/index.html` (visited on 01/17/2022).