A syntax for proof-relevant Observational Type Theory

Supervisor: Loïc Pujet

Background

Extensionality Principles in Type Theory

Dependent type theories, and especially those based on Martin-Löf's type theory (MLTT) [4], are a popular choice of foundations for proof assistants. For instance, Agda, Lean, and Rocq are all based on some extension of MLTT.

An interesting aspect of these theories is the coexistence of two distinct notions of equality: a *definitional* equality that records the equations which are automatically handled by the system (mostly consisting of β and η reductions), and a *propositional* equality that is a proper type and can be used to state theorems about equality. Unfortunately, the propositional equality fails to capture some principles which are commonly in mathematical reasoning, most notably

- the principle of *function extensionality*, which states that two functions are equal if and only if they send every input to equal outputs, and
- the principle of *propositional extensionality*, which states that two propositions are equal if and only if they are logically equivalent.

Of course, one can always postulate these principles as axioms, but doing so breaks the Curry-Howard correspondence between proofs and programs. This makes the theory less usable in practice, since such axioms do not have computational content.

The setoid model

There is a folklore trick to recover function and proposition extensionality without breaking the Curry-Howard correspondence: define a *setoid* to be a type A which is equipped with an equivalence relation (called the *setoid equality* on A), and define a *setoid morphism* between two setoids to be a function that preserves the setoid equality. By using setoids and systematically proving that all the relevant functions preserve setoid equalities, we can do extensional mathematics in MLTT without needing any axiom. The downside of this technique is that it makes the reasoning more bureaucratic, as most functions respect the natural setoid equalities for trivial reasons, yet we are forced to spell them explicitly.

In order to automatically take care of these boring proof obligations, Hofmann had the idea of designing a *model* of type theory which interprets types as setoids, equality types as setoid equalities, and terms as setoid morphisms [3]. Crucially, Hofmann's model is written in plain MLTT, which means that it can be used as a *compiler*: the user is free to work in a type theory that is extended with extensionality axioms, and once their proof is finished, the user can use Hofmann's model to translate it into plain, axiom-free MLTT.

Observational type theory

Hofmann's original model did not support all the rules of MLTT, but Altenkirch later discovered a way to recovers the missing rules [1]. However, his solution requires that the setoid equalities do not carry any computationally relevant information. This is not a problem for modelling all of MLTT plus extensionality, but it does means that some other desirable principles (such as *unique choice* and *large elimination of Acc*) cannot be interpreted in that model.

A few years down the line, Altenkirch, McBride and Swierstra used the setoid model to design a new type theory, called Observational Type Theory (OTT) [2]. OTT is an extension of MLTT which supports the missing extensionality principles without losing any computational content [7]. As such, OTT constitutes a reasonable choice of foundations for a proof assistant.

Goals of the internship

The goal of this internship is to design a type theory based on a new setoid model which supports stronger principles, such as the large elimination rule for Acc [6, 5]. This type theory should extend MLTT in a similar way that OTT does, but with improved expressive power.

If time permits, the intern can then try to prove some important properties of the resulting theory, such as *canonicity* and *normalisation*. Another possible direction is to try to implement the type theory, turning it into a practical system that can be used for proving theorems.

Required skills

The intern is expected to have some experience with proof assistants and with type theory. Ideally, the intern will have some familiarity with a programming such as OCaml, Haskell or Rust, but this is not mandatory.

Organization

The internship will take place in the University of Strasbourg.

References

[1] Thorsten Altenkirch. Extensional Equality in Intensional Type Theory. In *Proceedings of the 14th Annual IEEE Symposium on Logic in Computer Science*, LICS '99, page 412, USA, 1999. IEEE Computer Society.

[2] Thorsten Altenkirch, Conor McBride, and Wouter Swierstra. Observational equality, now! pages 57–68, 10 2007.

[3] Martin Hofmann. *Extensional Concepts in Intensional Type Theory*. PhD thesis, University of Edinburgh, 1995.

[4] Per Martin-Löf. An Intuitionistic Theory of Types: Predicative Part. In H.E. Rose and J.C. Shepherdson, editors, *Logic Colloquium '73*, volume 80 of *Studies in Logic and the Foundations of Mathematics*, pages 73–118. Elsevier, 1975.

[5] Loïc Pujet. A strict, proof-relevant setoid model. `https://github.com/loic-p/strict-setoids/tree/main`, 2025.

[6] Loïc Pujet. An Inductive Universe for Setoids. `https://pujet.fr/pdf/TYPES25_setoid_universe.pdf`, 2025.

[7] Loïc Pujet and Nicolas Tabareau. Observational Equality: Now For Good. *Proceedings of the ACM on Programming Languages*, 6(POPL):1–29, January 2022.