Ghost dependent types
for proof assistants

M2 internship proposed by Théo Winterhalter
theo.winterhalter@inria.fr
Location: ENS Paris Saclay within Deducteam (Inria Saclay)

Second semester 2023/2034

Note: This PDF may be updated later. For now this is the first version published on 30 October 2023.

Motivations

In the world of verification one considers ghost data which represents data that is only present at the specification level, but which is irrelevant for computation (the concrete value may not depend on the value of ghost data). This fact is all the clearer once an actual program is extracted from the tool: all ghost data is gone.

Languages with dependent types—such as the ones of the proof assistants Coq, Agda, Lean, F* or Idris—can also benefit from having ghost data. First of all, we may want to use them, the same way people use them in other verification software. With dependent types this also manifests with invariants encoded within data structures that end up polluting extracted programs. For instance, vectors vec $A\ n$ are a data structure corresponding to lists of length n containing elements are of type A, described by the two constructors vnil : vec $A\ 0$ and vcons $a\ n\ (v : \text{vec}\ A\ n) : \text{vec}\ A\ (\text{S}\ n)$. Vectors should ideally be extracted to lists, completely forgetting about the length invariant that is only used in a rich type theory and not necessary at the program level. Unfortunately, current extraction in Coq will yield a type for which the vcons constructor still stores a natural number. By marking the index as ghost we can recover the expected property that vectors extract to lists (and this should apply to the whole vector library) as well as some new gained flexibility in how to manipulate vectors: we can go from vec $A\ n$ to vec $A\ m$ (provided $n = m$) without blocking computation. I wrote more on this in a draft [Win23], but there are many things left to explore.

Beyond all that, having ghost types is interesting to design translations between various type theories, one of the main goals of Deducteam and the Dedukti project. The draft

talks a bit about that but there is also a lot to develop in that direction.

Goal(s) of the internship

The interested student would get familiar with ghost types and contribute to the design of a type theory with ghost dependent types. Asides from the methods used in the draft, we could study the meta-theory of the resulting theory with recent formalised tools such as those of Abel, Öhman and Vezzosi [AÖV17] and Adjedj et al. [Adj+23].

Challenges include (but are not limited to):

- Handling accessibility predicates as ghost data (and assessing whether it makes sense). Currently, they are—together with equality—one of the main reason why we need a universe of proposition that is not strict (in the sense of Gilbert et al. [Gil+19]).
- Properly stating what normalisation would mean in such a calculus and ideally showing it.
- Looking at local forms of computations that would be limited to ghost postions (*e.g.* having more powerful computation or automation in the length index of vectors). This would be connected to longer term ideas about interoperability between proof assistants and local extensions of type theories.

Requirements

I'm looking for someone with a good understanding and interest in dependent type theory and its implementation (experience with a proof assistant). This work would typically involve thinking about the design of dependent type systems and formalisation of meta-theoretical properties in a proof assistant. We could also think about prototype implementations.

References

[Adj+23] Arthur Adjedj, Meven Lennon-Bertrand, Kenji Maillard, Pierre-Marie Pédrot and Loïc Pujet. "Martin-Löf à la Coq". working paper or preprint. Sept. 2023. URL: https://inria.hal.science/hal-04214008.

[AÖV17] Andreas Abel, Joakim Öhman and Andrea Vezzosi. "Decidability of conversion for type theory in type theory". In: *Proceedings of the ACM on Programming Languages* 2.POPL (2017), pp. 1–29.

[BMM04] Edwin Brady, Conor McBride and James McKinna. “Inductive families need not store their indices”. In: *Types for Proofs and Programs: International Workshop, TYPES 2003, Torino, Italy, April 30-May 4, 2003, Revised Selected Papers*. Springer. 2004, pp. 115–129.

[FGP16] Jean-Christophe Filliâtre, Léon Gondelman and Andrei Paskevich. “The spirit of ghost code”. In: *Formal Methods in System Design* 48 (2016), pp. 152–174.

[Gil+19] Gaëtan Gilbert, Jesper Cockx, Matthieu Sozeau and Nicolas Tabareau. “Definitional Proof-Irrelevance without K”. In: *Proceedings of the ACM on Programming Languages*. POPL’19 (Jan. 2019), pp. 1–28. DOI: 10.1145/329031610.1145/3290316. URL: https://inria.hal.science/hal-01859964.

[Win23] Théo Winterhalter. “Extensionality of Ghost Dependent Types for Free”. working paper or preprint. July 2023. URL: https://hal.science/hal-04163836.