# Refinement in Dedukti

- Lab: LSV, ENS Cachan, France
- Team: Deducteam
- Advisor: Frédéric Blanqui (INRIA)

Dedukti is a formal proof checker based on a logical framework called the $\lambda\Pi$-calculus modulo, which is an extension of the simply-typed lambda-calculus with dependent types (e.g. matrices) and an equivalence relation on types generated by user-defined rewrite rules. Proofs generated by some automated theorem provers (e.g. Zenon, iProver) or proof assistants (e.g. HOL, Coq, Matita) can be checked in Dedukti by encoding function definitions and axioms by rewrite rules [3].

But, currently, no proof assistant fully uses all the features of Dedukti, which allows arbitrary user-defined rewrite rules (e.g. $(x+y)+z \to x+(y+z)$, where $+$ is itself defined by other rules). Such rules are indeed necessary if one wants to ease the use of dependent types and, for instance, be able to define types for representing simplicial sets of arbitrary dimensions, $\infty$-categories or models of Voevodsky's homotopy type theory.

We therefore would like to develop a tool for interactively building Dedukti proofs using all the features of Dedukti. But, to scale up and be able to handle large developments, especially with dependent and polymorphic types, we need to allow the users to write down incomplete terms (e.g. the type of the elements of a list), and provide an inference engine for finding out the missing subterms.

The goal of this internship is then to develop such a refinement engine for Dedukti. To start with, the student could for instance adapt to Dedukti a simplified version of the refinement engine of Matita [2]. This engine is based on a unification algorithm. At the beginning, the student could implement a simple first-order unification algorithm, and extend it later to take into account user-defined rewrite rules.

Finally, the candidate could take into account unification hints like in Matita and Coq for handling type classes and canonical structures [1, 5, 4], which are powerful techniques for handling overloaded symbols.

Expected abilities: basic knowledge of type inference.

## References


[1] A. Asperti, W. Ricciotti, C. Sacerdoti Coen, and E. Tassi. Hints in unification. In *Proceedings of the 22nd International Conference on Theorem Proving in Higher Order Logics*, Lecture Notes in Computer Science 5674, 2009.

[2] A. Asperti, W. Ricciotti, C. Sacerdoti Coen, and E. Tassi. A bi-directional refinement algorithm for the calculus of (co)inductive constructions. *Logical Methods in Computer Science*, 8:1–49, 2012.

[3] A. Assaf, G. Burel, R. Cauderlier, D. Delahaye, G. Dowek, C. Dubois, F. Gilbert, P. Halmagrand, O. Hermant, and R. Saillard. Expressing Theories in the lambda-Pi-Calculus Modulo Theory and in the Dedukti System, 2016. Draft.

[4] A. Mahboubi and E. Tassi. Canonical Structures for the working Coq user. In *Proceedings of the 4th International Conference on Interactive Theorem Proving*, Lecture Notes in Computer Science 7998, 2013.

[5] M. Sozeau. A new look at generalized rewriting in type theory. *Journal of Formalized Reasoning*, 2(1):41–62, 2009.