Proof tactics 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 [2].

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, ∞-categories or models of Voevodsky's homotopy type theory.

The goal of this internship is to develop a set of basic tactics for trying to solve Dedukti subgoals automatically, or help develop Dedukti proofs interactively. Of greatest interest in practice would be:

- a tactic for trying to solve any subgoal by calling an external SAT solver or automated theorem provers for first-order logic. This requires to convert Dedukti subgoals into TPTP problems, the *de facto* standard for first-order logic problems, by abstracting away non first-order terms.
- a tactic for trying to solve arithmetic subgoals by calling any external decision procedure for (Presburger) arithmetic.
- a tactic implementing subgoal rewriting like the `rewrite` tactic of Coq [5] or Ssreflect [4].

Depending on his/her skills and preferences, many followings are possible. The candidate could for instance extend the first tactic to SMT solvers and first-order logic with equality by taking into account equational axioms. Another one is to define a general tactic language following works like Isar [7], Tinycals [3, 1], Ssreflect [4], MTac [8] or the new implementation of LTac in Coq [6].

Expected abilities: basic knowledge of logic.

References

[1] A. Asperti, W. Ricciotti, C. Sacerdoti Coen, and E. Tassi. A new type for tactics. In *Proceedings of the ACM SIGSAM International Workshop on Programming Languages for Mechanized Mathematics Systems*, 2009.

[2] 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.

[3] C. Sacerdoti Coen, E. Tassi, and S. Zacchiroli. Tinycals: step by step tacticals. In *Proceedings of the 7th*, Electronic Notes in Theoretical Computer Science 174(2), 2006.

[4] G. Gonthier, A. Mahboubi, and E. Tassi. A small scale reflection extension for the Coq system. Technical Report 6455 version 16, INRIA and Microsoft Research, 2015.

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

[6] A. Spiwack. An abstract type for constructing tactics in Coq. In *Proceedings of the Workshop on Proof Search in Type Theories*, 2010.

[7] M. Wenzel. Isar - a generic interpretative approach to readable formal proof documents. In *Proceedings of the 12th International Conference on Theorem Proving in Higher Order Logics*, Lecture Notes in Computer Science 1690, 1999.

[8] B. Ziliani, D. Dreyer, N. R. Krishnaswami, A. Nanevski, and V. Vafeiadis. Mtac: a monad for typed tactic programming in Coq. In *Proceedings of the 18th ACM International Conference on Functional Programming*, SIGPLAN Notices 48(9), 2013.