Termination checking 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. vectors, matrices) and an equivalence relation on types generated by the 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, for Dedukti to behave well, the rewrite rules must satisfy some properties like confluence, preservation of typing and termination [4, 10].

The goal of this internship is to develop a termination checker for Dedukti. Such a termination checker could later be extended to other languages like Coq, Agda or Haskell, and participate to the international competition of termination provers.

A possible starting point is to implement the size-change principle (SCP) [9, 8] by simply comparing terms in the subterm ordering. An extension of SCP to dependent types and strong elimination is studied in [11].

Another one is to extend to dependent types the computability path ordering [6], by encoding ordering constraints into a SAT problem [7].

Next, instead of comparing terms in the subterm ordering, one can compare them through a semantic-based notion of size [1, 3, 5]. This approach requires to annotate types with size expressions and extend the type system with some subtyping relation. But, in some cases, a most general size annotation can be computed by using well known graph algorithms or linear programming techniques.

Expected abilities: basic knowledge of λ-calculus or functional programming.

References

[1] A. Abel. Termination checking with types. *Theoretical Informatics and Applications*, 38(4):277–319, 2004.

[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] G. Barthe, M. J. Frade, E. Giménez, L. Pinto, and T. Uustalu. Type-based termination of recursive definitions. *Mathematical Structures in Computer Science*, 14(1):97–141, 2004.

[4] F. Blanqui. Definitions by rewriting in the calculus of constructions. *Mathematical Structures in Computer Science*, 15(1):37–92, 2005.

[5] F. Blanqui. Size-based termination of higher-order rewrite systems. `http://rewriting.gforge.inria.fr/`, 2015. Submitted. 65 pages.

[6] F. Blanqui, J.-P. Jouannaud, and A. Rubio. The computability path ordering. *Logical Methods in Computer Science*, 11(4):1–45, 2015.

[7] M. Codish, J. Giesl, P. Schneider-Kamp, and R. Thiemann. SAT solving for termination proofs with recursive path orders and dependency pairs. *Journal of Automated Reasoning*, 49(1):53–93, 2011.

[8] P. Hyvernat. The Size-Change Termination Principle for Constructor Based Languages. *Logical Methods in Computer Science*, 10(1):1–30, 2014.

[9] C. S. Lee, N. D. Jones, and A. M. Ben-Amram. The size-change principle for program termination. In *Proceedings of the 28th ACM Symposium on Principles of Programming Languages*, 2001.

[10] R. Saillard. *Type Checking in the Lambda-Pi-Calculus Modulo: Theory and Practice*. PhD thesis, Mines ParisTech, France, 2015.

[11] D. Wahlstedt. *Dependent type theory with first-order parameterized data types and well-founded recursion*. PhD thesis, Chalmers University of Technology, Sweden, 2007.