Development of a termination checker for Dedukti

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

Dedukti [12] 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. lists of size \( n \)) and an equivalence relation on types generated by user-defined rewrite rules (like in Agda or Haskell). 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 [8, 7, 2]. But, for Dedukti to behave well, the rewrite rules must satisfy some properties like confluence, preservation of typing and termination [5, 13].

The goal of this internship is to develop a termination checker for Dedukti. Such a termination checker could then be extended to other languages like Coq, Agda or Haskell, and participate to the international competition of termination provers [14].

As a starting point, one should implement the size-change principle (SCP) [11, 9] by simply comparing terms in the subterm ordering. An extension of SCP to dependent types and strong elimination is studied in [15].

Next, instead of comparing terms in the subterm ordering, one can compare them through a semantic-based notion of size [1, 3, 4]. This approach requires to annotate types with size expressions and extend the type system with some subtyping relation. But a most general size annotation can be computed by using well known graph algorithms or linear programming techniques.

Finally, one can also consider to extend to dependent types the computability path ordering [6, 10].

Expected abilities: basic knowledge of lambda-calculus and types.

References


