A Type-Preserving Compiler for Dependently Typed Functions with Side Effects

依存型で型付けされた副作用のある関数のための型保存コンパイラ

A type-preserving compiler is developed for a dependently typed lambda calculus that uses separation logic to check the behavior of functions that possibly have side effects. The compiler and a proof of its type preservation property are written using the Coq proof assistant. The compilation process consists of several steps such as CPS conversion and closure conversion and the type preservation property is proved for each of the steps. The proof deals with types but not with the semantics of compiled programs. This makes the proof easier and simpler than proving correctness of a certified compiler. Nevertheless, since its dependent types and separation logic can specify the runtime semantics of programs, the type-preserving compiler can be considered equivalent to a certified compiler.

So far, the CPS conversion algorithm for the simply typed lambda calculus, which does not have dependent types or separation logic, has been implemented and its type preservation property has been proved. Currently, the author is extending the implementation and the proof to support dependent types and separation logic. The compilation steps other than CPS conversion are yet to be implemented.

副作用のある関数の動作をseparation logicを用いて検査することができる依存型付きラムダ計算をコンパイルする型保存コンパイラを開発する。コンパイラ及びその型保存性の証明は定理証明支援系Coqを用いて書かれる。コンパイラ全体はCPS変換やクロージャ変換などのいくつかのステップからなっており、それぞれのステップに対して型保存性が証明される。証明では型を扱うが、プログラムの意味論を直接扱うことはないため、certified compilerの正しさの証明よりも証明が簡単になる。しかしながら、依存型とseparation logicを用いてプログラムの意味論を規定することができるため、この型保存コンパイラはcertified compilerと等価であると考えられる。

これまでに (依存型やseparation logicを含まない) 単純型付きラムダ計算に対してCPS変換を行うアルゴリズムの実装とその型保存性の証明が完了しており、現在はそれを依存型とseparation logicに対応させるための拡張を行っているところである。CPS変換以外のコンパイルステップについては現在未着手である。