A Partial Evaluator for FlatCurry

Björn Peemöller
Institut für Informatik, CAU Kiel, D-24098 Kiel, Germany
bjp@informatik.uni-kiel.de

Partial evaluation of programs is a technique to anticipate the evaluation of costly computations once at compile-time instead of performing them (possibly several times) at run-time. Due to the absence of side-effects in purely functional languages like Haskell or Curry, the results of partial evaluation can be shared among different calls to the same functions which can considerably decrease the run-time of such functions.

Generally, when applied to expressions, partial evaluation behaves like (head) normal form reduction executed at run-time. But in addition, when applied to expressions with unbound variables, it is capable of performing those parts of the reduction independent of the values of the unbound variables and, thus, can be used to partially evaluate function bodies as well as the branches of unevaluated case-expressions.

Partial evaluation has already been studied for both functional languages such as Haskell as well as for logic languages like Prolog. Furthermore, Albert, Hanus and Vidal [2] presented a partial evaluation scheme for FlatCurry, an intermediate representation of the functional-logic language Curry [3]. Their evaluation scheme is based on term rewriting and restricted to confluent, i.e., deterministic programs without recursive let-expressions. However, to be applicable to real-world Curry programs, it is crucial for a partial evaluator to cover the full source language, including both logic features such as non-determinism and functional features such as recursive let-expressions.

In this work, we extend the aforementioned partial evaluator of Albert, Hanus and Vidal by integrating a new evaluation scheme based on an operational semantics of FlatCurry [1] to cover the full language of FlatCurry.

References

[1] E. Albert, M. Hanus, F. Huch, J. Oliver, and G. Vidal, *Operational semantics for declarative multi-paradigm languages*, Journal of Symbolic Computation **40** (2005), no. 1, 795–829.

[2] E. Albert, M. Hanus, and G. Vidal, *A practical partial evaluation scheme for multi-paradigm declarative languages*, vol. 2002, EAPLS, 2002.

[3] M. Hanus (ed.), *Curry: An integrated functional logic language (vers. 0.8.3)*, Available at http://www.curry-language.org, 2012.