Specialising Functional Patterns

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

Functional patterns [2] are an extension to the pattern matching in functional logic programming that allows the patterns in function definitions to also include already defined symbols. They allow more declarative representations of specifications and support the high-level programming of queries and transformations of complex data structures. In contrast to standard pattern matching like in Haskell, functional patterns inherently introduce non-determinism when used in conjunction with variables in the patterns. Although this approach fits very well in the setting of functional logic languages, the non-determinism can cause a notably overhead when defining deterministic functions using functional patterns. We therefore investigate a new approach to partially evaluate [1] functional patterns at compile time to specialise the function definitions to more efficient, or ideally, deterministic ones.

References

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

[2] S. Antoy and M. Hanus, *Declarative programming with function patterns*, Proceedings of the International Symposium on Logic-based Program Synthesis and Transformation (LOPSTR'05), Springer LNCS 3901, 2005, pp. 6–22.