The Power of Types for Extending Compilers

Giuseppe Maggiore Michele Bugliesi
Università Ca' Foscari Venezia
Dipartimento di Informatica
{maggiore,bugliesi}@dsi.unive.it

Abstract

In this presentation we discuss a series of techniques for static analysis and program optimization. These techniques are usually applied in a cumbersome manner, by either defining new, specialized languages or by creating full-blown analysis and verification tools.

We show how all these techniques can be implemented inside a modern functional language such as Haskell or OCaML, and even more how these techniques can all be defined in terms of a relatively simple framework of type functions and monads.

1 Introduction

In this introduction, we start with a brief presentation of monads: what they are, how they are used to hide boilerplate code for repetitive operation, and we show one particular monad, the **state monad** [13].

We discuss **phantom types** and how they are used to tag terms with static information to perform various kinds of static analyses but without any performance hit whatsoever since phantom types exist only at compile time and often have only $\perp$ as their sole inhabitant [6, 5, 4].

We study a bit of **type functions** and **type classes** to show what kind of power the Haskell type system puts at our disposal [1, 8, 12].

2 Applications

We then move to a few practical applications of the techniques seen above.

3 Generalization

4 Additional Applications

References

[1] O. de Moor, J. Gibbons, and G. Jones. The fun of programming.

[2] Iavor S. Diatchki and Mark P. Jones. Strongly typed memory areas programming systems-level data structures in a functional language. In *Haskell '06: Proceedings of the 2006 ACM SIGPLAN workshop on Haskell*, pages 72–83, New York, NY, USA, 2006. ACM.

[3] Matthew Fluet and Greg Morrisett. Monadic regions. *SIGPLAN Not.*, 39(9):103–114, 2004.

[4] Matthew Fluet and Riccardo Pucella. Phantom types and subtyping.

[5] Matthew Fluet and Riccardo Pucella. Practical datatype specializations with phantom types and recursion schemes.

[6] Ralph Hinze and J. Cheney. Phantom types.

[7] Oleg Kiselyov and Chung chieh Shan. Position: Lightweight static resources: Sexy types for embedded and systems programming. In *In TFP 07, the 8 th Symposium on Trends in Functional Programming*, 2007.

[8] Oleg Kiselyov, Simon Peyton Jones, and Chung-chieh Shan. Fun with type functions. *Reflections on the Work of C. A. R. Hoare*, pages 303–333.

[9] Oleg Kiselyov and Chung-chieh Shan. Lightweight static capabilities. *Electron. Notes Theor. Comput. Sci.*, 174(7):79–104, 2007.

[10] Oleg Kiselyov and Chung-chieh Shan. Lightweight monadic regions. In *Haskell '08: Proceedings of the first ACM SIGPLAN symposium on Haskell*, pages 1–12, New York, NY, USA, 2008. ACM.

[11] Erik Meijer and Peter Drayton. Static typing where possible, dynamic typing when needed: The end of the cold war between programming languages.

[12] Tim Sheard and Simon Peyton Jones. Template meta-programming for haskell. In *Haskell '02: Proceedings of the 2002 ACM SIGPLAN workshop on Haskell*, pages 1–16, New York, NY, USA, 2002. ACM.

[13] Philip Wadler. How to declare an imperative. *ACM Comput. Surv.*, 29(3):240–263, 1997.