Dependently Typed Programming in Agda

Ulf Norell
Chalmers University of Technology
ulfn@chalmers.se

Abstract

Dependently typed languages have for a long time been used to describe proofs about programs. Traditionally, dependent types are used mostly for stating and proving the properties of the programs and not in defining the programs themselves. An impressive example is the certified compiler by Leroy (2006) implemented and proved correct in Coq (Bertot and Castéran 2004).

Recently there has been an increased interest in *dependently typed programming*, where the aim is to write programs that use the dependent type system to a much higher degree. In this way a lot of the properties that were previously proved separately can be integrated in the type of the program, in many cases adding little or no complexity to the definition of the program. New languages, such as Epigram (McBride and McKinna 2004), are being designed, and existing languages are being extended with new features to accomodate these ideas, for instance the work on dependently typed programming in Coq by Sozeau (2007).

This talk gives an overview of the Agda programming language (Norell 2007), whose main focus is on dependently typed programming. Agda provides a rich set of inductive types with a powerful mechanism for pattern matching, allowing dependently typed programs to be written with minimal fuss. To read about programming in Agda, see the lecture notes from the Advanced Functional Programming summer school (Norell 2008) and the work by Oury and Swierstra (2008).

In the talk a number of examples of interesting dependently typed programs chosen from the domain of programming language implementation are presented as they are implemented in Agda.

Categories and Subject Descriptors D.1.1 [*Applicative (functional) programming*]

General Terms Languages, Verification

Keywords dependent types, programming

References

Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions*. Texts in Theoretical Computer Science. Springer Verlag, 2004.

Xavier Leroy. Formal certification of a compiler back-end, or: programming a compiler with a proof assistant. In *33rd symposium Principles of Programming Languages*, pages 42–54. ACM Press, 2006.

C. McBride and J. McKinna. The view from the left. *Journal of Functional Programming*, 14(1):69–111, January 2004.

Ulf Norell. Dependently typed programming in Agda. In *Lecture notes on Advanced Functional Programming*, 2008. To appear.

Ulf Norell. *Towards a practical programming language based on dependent type theory*. PhD thesis, Department of Computer Science and Engineering, Chalmers University of Technology, SE-412 96 Göteborg, Sweden, September 2007.

Nicolas Oury and Wouter Swierstra. The Power of Pi. In *ICFP'08: Proceedings of the 2008 SIGPLAN Internation Conference on Functional Programming*, 2008.

Matthieu Sozeau. Program-ing Finger Trees in Coq. In *ICFP'07: Proceedings of the 2007 ACM SIGPLAN International Conference on Functional Programming*, pages 13–24. ACM Press, 2007.

TLDI'09, January 24, 2009, Savannah, Georgia, USA.
ACM 978-1-60558-420-1/09/01.