Universiteit Utrecht, Department of Information and Computing Sciences

Theory of Programming and Types

Mathijs Baaijens
Nico Naus, 3472353

8 juni 2014

1 abstract

In this paper we will demonstrate how to build a compiler and proof it's correctness using dependent types. This work is based on the paper "A type-correct, stack-safe, provably correct expression compiler in Epigram". We will implement a compiler for an extended language an show that the statements made in this paper still hold.

2 Introduction

3 The First Semantics : eval

3.1 Type preservation is the type of the interpreter

4 The Second Semantics : compile & exec

4.1 Typing stacks

4.2 Compiling and executing typed intermediate code

4.3 Specifying intermediate code

4.4 Implementing an Interpreter for Intermediate Code

4.5 Implementing the compiler to intermediate code

5 Compiler Correctness

6 Conclusion

7 Related work

A Certified Type-Preserving Compiler from Lambda Calculus to Assembly Language [1]. Here the author presents a certified compiler for a language similar to ours, with a machine-checked correctness proof written in Coq.

Referenties

[1] Adam Chlipala, *A Certified Type-Preserving Compiler from Lambda Calculus to Assembly Language*. Proceedings PLDI ’07, p54-65, New York, 2007.