Exception Handling in Haskell

William S. Fisher

August 10, 2016

Abstract

Implementing exception handling in Haskell. Unlike other libraries, use named exception handlers. Use the λ^{try}-calculus to formalize and explore a series of translations between multiple calculi to arrive at a translation into Haskell. Explore properties of this translation including soundness and completeness. Publish useable Haskell library.

Acknowledgements

Thanks me

Contents

Chapter 1

Introduction

Hel

Chapter 2

Background

Formal Systems

λ-Calculus

Haskell

Logic, Types, and their Computation Interpretation

Continuations

Delimited-Continuations

$\lambda\mu$-Calculus

λ^{try}-Calculus

Delimited-Continuation Calculus

Simon Peyton-Jones *et al.* extended the λ-calculus with additional operators in order create a framework for implementing delimited continuations [1]. This calculus will be referred to as the delimited-continuation calculus or DCC. Many calculi have been devised with control mechanisms. Like the $\lambda\mu$-calculus, these control mechanisms are all specific instances of delimited and undelimited continuations. DCC provides a set of operations that are capable of expressing many of these other common control mechanisms.

The grammar of DCC is an extension of the standard λ-calculus:
The additional terms behave as follows:

- *newPrompt* returns a new and distinct prompt.
- *pushPrompt*'s first argument is a prompt which is pushed onto the continuation stack before evaluating its second argument.

Definition 2.0.1 (Grammar rules for DCC)

$$
\begin{array}{lrcl}
\text{(Variables)} & x, y, \ldots & & \\
\text{(Expressions)} & e & ::= & x \mid \lambda x.e \mid e\ e' \\
 & & \mid & \mathit{newPrompt} \mid \mathit{pushPrompt}\ e\ e \\
 & & \mid & \mathit{withSubCont}\ e\ e \mid \mathit{pushSubCont}\ e\ e
\end{array}
$$

- *withSubCont* captures the subcontinuation from the most recent occurrence of the first argument (a prompt) on the excution stack to the current point of execution. Aborts this continuation and applies the second argument (a λ-abstraction) to the captured continuation.
- *pushSubCont* pushes the current continuation and then its first argument (a subcontinuation) onto the continuation stack before evaluating its second argument.

Chapter 3

DCC Interpreter

Interpreter

Chapter 4

Translations

λ^{try}**-to-**$\lambda\mu$

$\lambda\mu$**-to-DCC**

λ^{try}**-to-DCC**

Chapter 5

Conclusion

Evaluation

Conclusion

Future Work

Bibliography

[1] R. Kent Dybvig, Simon L. Peyton Jones, and Amr Sabry. A monadic framework for delimited continuations. *J. Funct. Program.*, 17(6):687–730, 2007.