Formally verified programming with monads in Coq

(Formalnie zweryfikowane programowanie z monadami w Coqu)

Zeimer

Praca inżynierska

Promotor: dr Wpisuyashi TODO

Uniwersytet Wrocławski
Wydział Matematyki i Informatyki
Instytut Informatyki

czerwiec 2019

Abstract

We introduce hsCoq, a Coq library for formally verified general-purpose programming with Haskell-style abstractions: functors, applicatives, monads, monad transformers and class-based monads. We discuss the design choices made and illustrate the working of the library with examples from [1].

...

Contents

Chapter 1

Introduction

This is the first section.sss

Chapter 2

About Coq

wuuut

Chapter 3

Computational effects

Chapter 4

Design

Chapter 5

Examples

Chapter 6

A case study in proof engineering

Chapter 7

Conclusion

Chapter 8

TODO

1. Introduction: functional programming, formally verified programming and proving.
2. Approaches to computational effects: chaos, ML-style, monads, algebraic effects.
3. A description of the inner workings of the library: design choices, file structure, implementation.
4. Examples: some from Just Do It, maybe some custom ones.
5. Safety: some theorems and proofs.
6. Theoretical comparison of the ease of use with Haskell and Idris.
7. Practical comparison with MERC.
8. Cite some literature: some Coq papers, Moggi, Just Do It, Experimenting with Monadic Equational Reasoning in Coq
9. Technical matters:
   (a) Mention where's the implementation and put it to Coq's repository of user libraries.
   (b) Installation guide.
   (c) Tools: why no ssreflect?
   (d) Documentation (it's in the source code).
10. More: a case study in proof engineering - how do the tactics hs, monad and (maybe) the one for reflective functor simplifcation work?
11. Deficiencies, conclusion and further work.
12. Points to make: this is a library for general purpose programming, without some deep goal.

Bibliography

[1] Jeremy Gibbons and Ralf Hinze, *Just do It: Simple Monadic Equational Reasoning*, 2011