Implementing a library for scoped algebraic effects in Agda

HTWK Leipzig

Jonas Höfer
Informatik
69555

Juli 2020

Abstract

Contents

Chapter 1

Introduction

1.1 Goals

Chapter 2

Preliminaries

2.1 Agda

Agda is a functional language with dependent types. The current version, originally known as Agda2, was originally developed by Ulf Norell [Nor07]

```
data ℕ : Set where
  zero : ℕ
  suc  : ℕ → ℕ
```

2.1.1 Dependent Types

```
data Vec (A : Set) : ℕ → Set where
  _::_ : {n : ℕ} → A → Vec A n → Vec A (suc n)
  []   : Vec A 0
```

2.1.2 Propositions as Types

2.1.3 Strict Positivity

Container

2.1.4 Termination Checking

Well Founded Induction / Sized Types

```
open import Agda.Builtin.Size public
  renaming ( SizeU to SizeUniv ) -- sort SizeUniv
  using ( Size                   -- Size : SizeUniv
        ; Size<_                 -- Size<_ : Size → SizeUniv
        ; ↑_                     -- ↑_ : Size → Size
        ; _⊔ˢ_                   -- _⊔ˢ_ : Size → Size → Size
        ; ∞ )                    -- ∞ : Size

data List (A : Set) : Set where
  _::_ : A → List A → List A
  []   : List A

map-list : {A B : Set} → (A → B) → (List A → List B)
map-list f (x :: xs) = f x :: map-list f xs
map-list f []        = []

data Rose (A : Set) : Size → Set where
  rose : ∀ {i} → A → List (Rose A i) → Rose A (↑ i)

map-rose : {A B : Set} {i : Size} → (A → B) → (Rose A i → Rose B i)
map-rose f (rose x xs) = rose (f x) (map-list (map-rose f) xs)
```

2.2 Curry and Call-Time-Choice

let x = coin in x + x

Chapter 3

Algebraic Effects

3.1 Free Monads

3.2 Handler

3.2.1 Nondet

3.2.2 State

3.3 Scoped Effects

3.3.1 Cut

3.4 Call-Time-Choice as Effect

Chapter 4

Higher Order

Chapter 5

Conclusion

5.1 Summary

Bibliography

[Nor07] Ulf Norell. “Towards a practical programming language based on dependent type theory”. PhD thesis. SE-412 96 Göteborg, Sweden: Department of Computer Science and Engineering, Chalmers University of Technology, Sept. 2007.