Typed Tagless Final

an approach to embedded DSLs

Zixiu Su

2023-11-05

Contents

Intro

I was trying to implement a 311-style interpreter, this time in Haskell intead of Racket (which is untyped, or unityped). Very soon I found out there was a lot of hoops to jump through when it comes down to a typed language. I was not satisfied with my naive implementation and trying to improve my AST representation. Then I found Oleg's tutorial on so called "Typed Tagless Final" and learned some interesting techniques. And today I want to share with you what I learned.

What do I mean by a "311-style interpreter"?

Some object/source language :

- Simply typed lambda calculus (STLC)
- deBruijin indices
- extended with Bool/Int/...

Implemented in:

Haskell (or other **TYPED** functional languages)

Something like this?

(would be perfectly fine in Racket)

```
data Exp = ...

interp (B b)     _   = b
interp (V var)   env = lookp var env
interp (L body)  env = \arg -> interp body (arg, env)
interp (A f a)   env = (interp f env) (interp a env)
```

My Naive Approach

Unimportant differences: Int vs Bool named vs debruijned Important: explain GADTs syntax (data Exp where..) use Data.Map as env union type for values

```
interp :: Exp -> Env -> Maybe Val

type Env = M.Map String Val

data Exp where
  Int :: Integer -> Exp
  Var :: String -> Exp
  Lam :: String -> Exp -> Exp
  App :: Exp -> Exp -> Exp

data Val where
  VInt  :: Integer -> Val
  VClos :: String -> Exp -> Env -> Val
```

Choosing types

```
interp :: Exp -> Env -> Val
```

1. Env <- Data.Map
2. Define ADT

   ```
   data Exp
     = B Bool
   ```

```
  | V Var
  | L Exp
  | A Exp Exp

data Var = VS Var | V0
```

3. Val needs a union of different types (tags)

```
data Val
  = UB Bool
  | UC (Val -> Val)
```

Show result code here

Problems:

1. Boilerplate
2. AST allowing ill-typed programs (example?) => write a type checker?
3. Packing/unpacking union type values

Cause: DSL embedding is "loose"

Q: How would you improve the AST?

GADTs to the Rescue

```
{-# LANGUAGE GADTs #-}
```

Tagless "Final"

Discussion

Appendix

References

Categorical Semantics

Comparison with free monads