Dependent Types in Haskell

Ningning Xie
13 Dec., 2018
Hong Kong Functional Programming Meetup

1 The University of Hong Kong
Why Haskell?

Why not Coq, Idris, Agda, ...?

- Start with Functional Programming, and decide to use dependent types if necessary.
- Backward-compatibility.
- No termination or totality checking.
What is in this talk?

• How to write dependently typed programs in Haskell nowadays?
• A tour of the *singletons* library.
• Introduction of Dependent Haskell (including Coercion Quantification).
Singletons
Singletons

**Singletons**: a library introduced in *Dependently Typed Programming with Singletons*, Haskell’12 (Eisenberg and Weirich, 2012). The library uses Template Haskell to

- automatically generate singleton types
- automatically lift functions to the type level
- automatically refine functions with rich types.
Singletons library is useful for writing dependent type programs. It generates boilerplate code for you, which enables us to write programs similar in other dependent type languages, e.g. Idris.
Lessons we learned

Any question so far?
• A set of language extensions for GHC that provides the ability to program \textbf{as if} the language had dependent types.
The price we paid

- A set of language extensions for GHC that provides the ability to program **as if** the language had dependent types. We used (only) 8 language extensions...
• A set of language extensions for GHC that provides the ability to program as if the language had dependent types. We used (only) 8 language extensions...
More on the way.

{-# LANGUAGE DataKinds, TypeFamilies, PolyKinds, TypeInType, GADTs, RankNTypes, TypeOperators, FunctionalDependencies, ScopedTypeVariables, TypeApplications, Template Haskell, UndecidableInstances, InstanceSigs, TypeSynonymInstances, KindSignatures, MultiParamTypeClasses, TypeFamilyDependencies, AllowAmbiguousTypes, FlexibleContexts ... #-}
• There is no unified meta-theory for the extensions.
The price we paid

- There is no unified meta-theory for the extensions.
- Duplications for term-level and type-level functions, either written by programmer or generated by *singletons*. 
The price we paid

- There is no unified meta-theory for the extensions.
- Duplications for term-level and type-level functions, either written by programmer or generated by *singletons*.
- Restrictions:
  - All applications of a type family must be fully saturated with respect to that arity;
  - Data families are not promoted;
  - ...

Future plan for Dependent Haskell
To extend GHC with **full-spectrum dependent types** in a way that is **compatible with the current implementation**, with the goal of simplifying and unifying many of GHC’s extensions (Eisenberg, 2016; Gundry, 2013; Weirich et al., 2017).
The Plan

Adding dependent types to GHC in one patch...
The Plan

Adding dependent types to GHC in one patch... is very difficult ¹.

¹High-level Dependency Graph from https://ghc.haskell.org/trac/ghc/wiki/Commentary/ModuleStructure
• GHC incorporates several compilation phases \(^2\).

\(^2\)Compiler Pipeline from https://ghc.haskell.org/trac/ghc/wiki/Commentary/Compiler/HscPipe
The Plan

- GHC incorporates several compilation phases\(^2\).
- Dependent Core, as steps are taken towards dependent Haskell (Weirich et al., 2017).

\(^2\)Compiler Pipeline from https://ghc.haskell.org/trac/ghc/wiki/Commentary/Compiler/HscPipe
The Plan

- GHC incorporates several compilation phases \(^2\).
- Dependent Core, as steps are taken towards dependent Haskell (Weirich et al., 2017).
- Some discussions can be found in Haskell wiki\(^3\).

\(^2\)Compiler Pipeline from https://ghc.haskell.org/trac/ghc/wiki/Commentary/Compiler/HscPipe

\(^3\)https://ghc.haskell.org/trac/ghc/wiki/DependentHaskell/Phase2
The big picture
The big picture

The World of Haskell
The World of Haskell

Dependent Haskell
The big picture

The World of Haskell

Dependent Haskell

Dependent Core
The big picture

The World of Haskell

Dependent Haskell

Dependent Core

Homogeneous Equality
The big picture

The World of Haskell

Dependent Haskell

Dependent Core

Homogeneous Equality

Coercion Quantification
Coercion Quantification

- Haskell Implementors’ Workshop (HIW’18) talk (Xie and Eisenberg, 2018)
- Extended abstract, slides: https://xnning.github.io/
References


Dependent Types in Haskell

Ningning Xie ¹
13 Dec., 2018
Hong Kong Functional Programming Meetup

¹The University of Hong Kong