

# Dependent Types in Haskell

---

Ningning Xie <sup>1</sup>

13 Dec., 2018

Hong Kong Functional Programming Meetup

<sup>1</sup>The University of Hong Kong

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**: 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 **as if** the language had dependent types.

- 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...

## 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...

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*.

- 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).

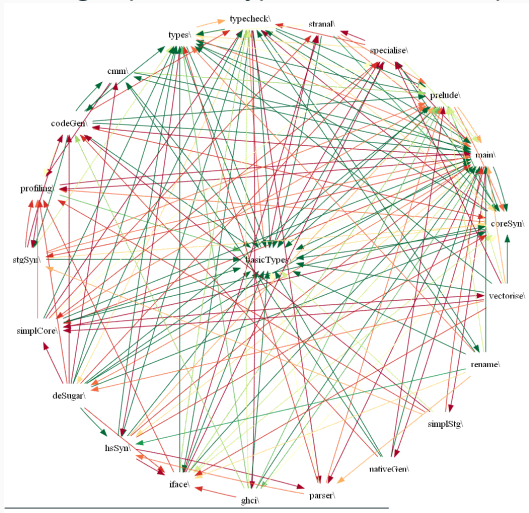
# The Plan

---

Adding dependent types to GHC in one patch...

# The Plan

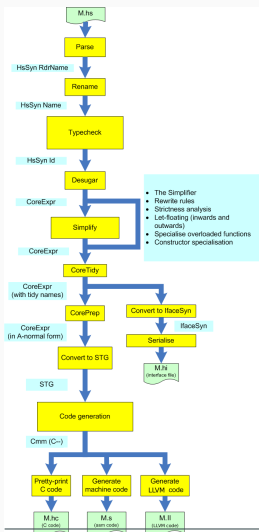
Adding dependent types to GHC in one patch... **is very difficult** <sup>1</sup>.



<sup>1</sup>High-level Dependency Graph from

<https://ghc.haskell.org/trac/ghc/wiki/Commentary/ModuleStructure>

# The Plan

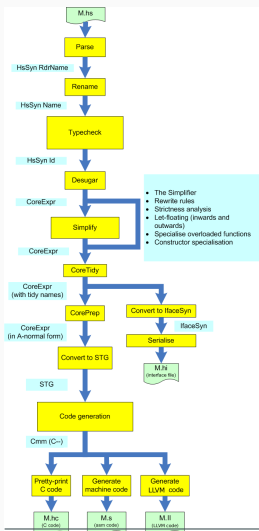


- GHC incorporates several compilation phases <sup>2</sup>.

<sup>2</sup>Compiler Pipeline from

<https://ghc.haskell.org/trac/ghc/wiki/Commentary/Compiler/HscPipe>

# The Plan

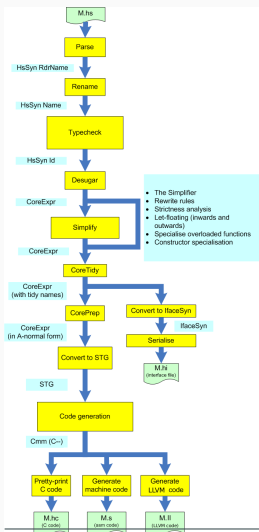


- GHC incorporates several compilation phases <sup>2</sup>.
- Dependent Core, as steps are taken towards dependent Haskell (Weirich et al., 2017).

<sup>2</sup>Compiler Pipeline from

<https://ghc.haskell.org/trac/ghc/wiki/Commentary/Compiler/HscPipe>

# The Plan



<sup>2</sup>Compiler Pipeline from

<https://ghc.haskell.org/trac/ghc/wiki/Commentary/Compiler/HscPipe>

<sup>3</sup><https://ghc.haskell.org/trac/ghc/wiki/DependentHaskell/Phase2>

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

# The big picture

The World of Haskell



# The big picture

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

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

# References

---

- Richard A. Eisenberg. 2016. *Dependent Types in Haskell: Theory and Practice*. PhD Dissertation. University of Pennsylvania.
- Richard A. Eisenberg and Stephanie Weirich. 2012. Dependently Typed Programming with Singletons. In *Proceedings of the 2012 Haskell Symposium (Haskell '12)*. ACM, New York, NY, USA, 117–130. <https://doi.org/10.1145/2364506.2364522>
- Adam Michael Gundry. 2013. *Type Inference, Haskell and Dependent Types*. PhD Dissertation. University of Strathclyde.
- Stephanie Weirich, Antoine Voizard, Pedro Henrique Avezedo de Amorim, and Richard A Eisenberg. 2017. A Specification for Dependent Types in Haskell. *Proceedings of the ACM on Programming Languages* 1, ICFP (2017), 31.
- Ningning Xie and Richard A Eisenberg. 2018. Coercion Quantification. (2018).

# Dependent Types in Haskell

---

Ningning Xie <sup>1</sup>

13 Dec., 2018

Hong Kong Functional Programming Meetup

<sup>1</sup>The University of Hong Kong