A Review of Levity

Lifted = has \bot, with kind ★
- \text{Int, Bool, Int\#} \to \text{Int\#},
- \text{forall } a. \text{Maybe } a

Unlifted = no \bot, with kind #
- \text{Int\#, Word\#,}
- (\text{# Bool, Char\# #}),
- \text{forall } a. \text{Array\# a}
An Unlifted Disturbance

Off to GHCi...
Sub-kinding

OpenKind

★ #
Sub-kinding

Simon Peyton Jones: This is a gross hack.
Sub-kinding $\rightarrow$ Polymorphism

Solution:
Use polymorphism

SPJ's idea
Welcome to the Future
Levity Polymorphism

ordinary datatype:
  
  data Levity = Lifted | Unlifted

highly magical:
  
  TYPE ::: Levity \rightarrow ★

ordinary type synonyms:
  
  type ★ = TYPE 'Lifted
  type # = TYPE 'Unlifted
Levity Polymorphism

**undefined ::**

\[ \forall (v :: \text{Levity}) (a :: \text{TYPE } v). a \]

**error ::**

\[ \forall (v :: \text{Levity}) (a :: \text{TYPE } v). [\text{Char}] \rightarrow a \]
Levity Polymorphism

And it works!
Subtlety: Kind of ∀-types

kind of (∀ a. τ) = kind of τ;
always ★ or #

Quiz: Kind of

∀ (v :: Levity)
(a :: TYPE v). a

?
Subtlety: Kind of \( \forall \)-types

Quiz: Kind of

\( \forall (v :: \text{Levity}) (a :: \text{TYPE } v). a \)

? 

Answer: \text{TYPE } v

But \( v \) is out of scope!
Subtlety: Kind of \( \forall \)-types

\[
\Pi (v :: \text{Levity}).
\forall (a :: \text{TYPE } v). \ a
\]

has kind \( \star \)
Subtlety: When to allow LP?

id :: ∀ (v :: Leivty) (a :: TYPE v).

a → a

NO!

no code to generate
Subtlety: When to allow LP?

Answer: No levity-polymorphic binders

(this includes datatype parameters)
Further reading

GHC wiki page:
https://ghc.haskell.org/trac/ghc/wiki/NoSubKinds

Draft paper on issues around (★ :: ★): available from
http://www.cis.upenn.edu/~eir/pubs.html
Demo

Fun with ★ :: ★
Levity Polymorphism in Dependent Haskell

Richard A. Eisenberg
University of Pennsylvania
eir@cis.upenn.edu

Sunday, 30 August, 2015
HIW, Vancouver, BC, Canada