Microsoft Research

PENN PL

Levity Polymorphism in Dependent Haskell

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

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

A Review of Levity

Lifted = has ⊥, with kind ★

```
Int, Bool, Int# → Int#,
forall a. Maybe a
```

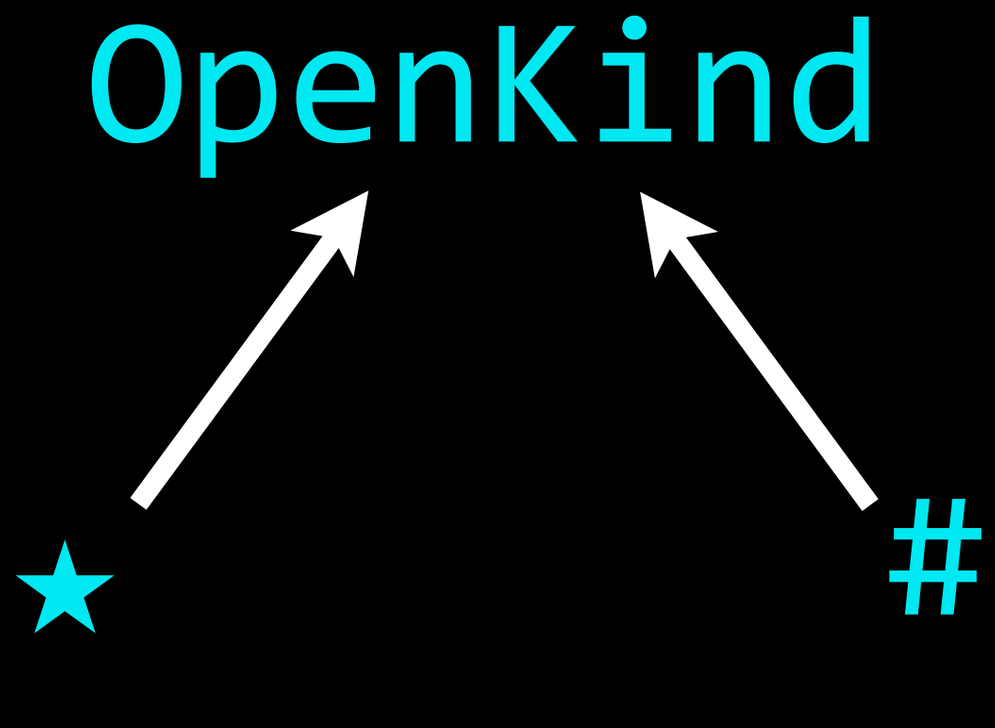
Unlifted = no ⊥, with kind #

```
Int#, Word#,
(# Bool, Char# #),
forall a. Array# a
```

An Unlifted Disturbance

Off to GHCi...

Sub-kinding



Sub-kinding

Simon Peyton Jones:
This is a gross hack.

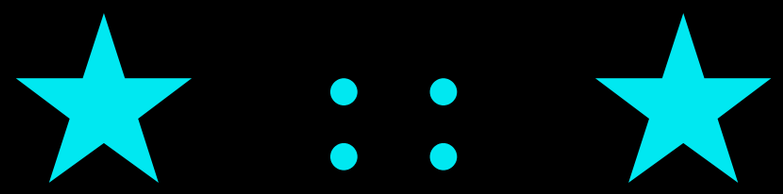
Sub-kinding → Polymorphism

SPJ's idea

Solution:

Use polymorphism

Welcome to the Future



Levity Polymorphism

ordinary datatype:

```
data Levity = Lifted
            | Unlifted
```

highly magical:

```
TYPE :: Levity → ★
```

ordinary type synonyms:

```
type ★ = TYPE ‘Lifted
type # = TYPE ‘Unlifted
```

Levity Polymorphism

```
undefined ::
  ∀ (v :: Levity)
    (a :: TYPE v).
    a
```

```
error ::
  ∀ (v :: Levity)
    (a :: TYPE v).
    [Char] → 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 ∀-types

Quiz: Kind of

```
∀ (v :: Levity)
  (a :: TYPE v). a
```

?

Answer: `TYPE v`

But `v` is out of scope!

Subtlety: Kind of ∀-types

```
∏ (v :: Levity).
∀ (a :: TYPE v). a
```

has kind

★

Subtlety: When to allow LP?

```
id :: ∀ (v :: Levity)
       (a :: TYPE v).
     a → a
```

NO!

no code to generate

Subtlety: When to allow LP?

Answer:
No levity-polymorphic binders
(this includes datatype parameters)

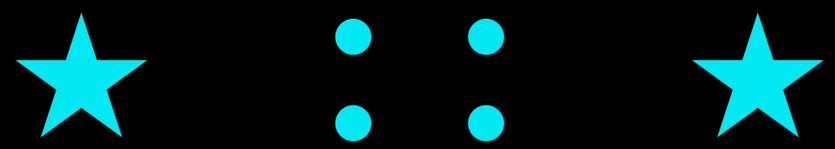
Further reading

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



Microsoft Research

PENN PL

Levity Polymorphism in Dependent Haskell

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

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