Dependent Types in GHC

John Leo

Halfaya Research

April 8, 2017

References

- Pointer to this talk on github
- Pointer to references

What are Dependent Types?

```
data ℕ : Set where
  zero : ℕ
  suc : ℕ → ℕ

data Vec (A : Set) : ℕ → Set where
  [] : Vec A zero
  _::_ : {n : ℕ} → A → Vec A n → Vec A (suc n)
```

Vector Append

```
_+_ : ℕ → ℕ → ℕ
zero + n = n
suc m + n = suc (m + n)

_++_ : {A : Set} → {m n : ℕ} →
  Vec A m → Vec A n → Vec A (m + n)
[] ++ y = y
(x ∷ xs) ++ y = x ∷ (xs ++ y)
```

Vector Lookup

```
data Fin : ℕ → Set where
  zero : {n : ℕ} → Fin (suc n)
  suc : {n : ℕ} → (m : Fin n) → Fin (suc n)

lookup : {A : Set} → {n : ℕ} → Fin n → Vec A n → A
lookup zero (x :: _ ) = x
lookup (suc n) (_ :: xs) = lookup n xs
```

Some Basic Types

```
data Bool : Set where
  true : Bool
  false : Bool

data _≡_ {A : Set} : A → A → Set where
  refl : {a : A} → a ≡ a
```

Vector Lookup 2

```
_¡_ : ℕ → ℕ → Bool
m    ¡ zero = false
zero ¡ suc n = true
suc m ¡ suc n = m ¡ n

lookup' : {A : Set} → {n : ℕ} →
  (m : ℕ) → m ¡ n ≡ true → Vec A n → A
lookup' _ () []
lookup' zero refl (x :: _ ) = x
lookup' (suc m) p (_ :: xs) = lookup' m p xs
```

Dependent Types

Advantages over other types

Applications

- Better correctness guarantees for software.
- Mechanical verification of mathematics.

Dependent Types are Not New

Timeline

The Golden Age is Now

- DeepSpec, etc.
- BigProof, etc.

Robet Harper Quote

Dependent Types in Haskell

Richard Eisenberg's PhD Thesis

1. Introduction
2. Preliminaries
3. Motivation
4. Dependent Haskell
5. PICO: The Intermediate Language
6. Type Inference and Elaboration, or how to BAKE a PICO
7. Implementation
8. Related and Future Work