Contents

• What are dependent types?
• Organization
• Agda and Idris
A dependent type is a type that depends on a value.

–Wikipedia
Matrix : Set → ℕ → ℕ → Set
Matrix A m n = Vec (Vec A n) m

i2x2 : Matrix ℕ 2 2
i2x2 = (1 :: 0 :: []) ::
     (0 :: 1 :: []) :: []

Types that depends on values
m3x2 : Matrix \( \mathbb{N} \times 3 \times 2 \)
m3x2 = \textcolor{red}{i2x2}

Types that depends on values
Types that depend on values

\[
\text{lookup} : \forall \{A : \text{Set}\} \{n : \mathbb{N}\} \rightarrow \\
(i : \mathbb{N}) \rightarrow \\
\{\{\text{safe : } i \text{ is-smaller-than } n\}\} \rightarrow \\
\text{Vec } A \ n \rightarrow A
\]
nats : Vec \( \mathbb{N} \) 5
nats = 0 :: 1 :: 2 :: 3 :: 4 :: []

two : \( \mathbb{N} \)
two = lookup 2 nats

Types that depends on values
nats : Vec \( \mathbb{N} \) 5
nats = 0 :: 1 :: 2 :: 3 :: 4 :: []

six : \( \mathbb{N} \)
six = lookup 6 nats

Types that depends on values
Types are values
Matrix : Set → ℕ → ℕ → Set
Matrix A m n = Vec (Vec A n) m

Types are values
Id : $\text{Set} \rightarrow \text{Set}$

Id $A = A$

Types are values
-- Haskell

type Id a = a

instance Monad Id where
    return = \ x -> x
    (>>>=) = \ x f -> f x

Types are values
Types are values
-- Haskell

newtype Id a = Id { runId :: a }

instance Monad Id where
  return = \ x -> Id x
  (>>=) = \ x f -> f (runId x)

Types are values
I don't think we really expected newtypes to be quite so ubiquitous in this kind of way. The introduction of newtypes in typeclasses was much more fruitful than I’d at all expected when I designed the language.

– Simon Peyton Jones
-- Agda
Id : Set → Set
Id A = A

IdMonad : RawMonad Id
IdMonad = record
{ return = λ x → x ; _>>=_ = λ x f → f x }

Types are values
The Damas-Milner approach to type inference is alive and well and working harder than ever, even though we have dispensed with the shackles on programming which allow it to be complete.

–Altenkirch et al.
Organization

- Seminar website: ps.informatik.uni-tuebingen.de/teaching/ws15/pdt/
- Every week, somebody teaches everybody else something and leaves some homework exercises.
- First 4 topics are fixed, the rest are up to the presenter.
- There is a small individual programming project at the end.
Agda and Idris

• Very similar languages

• Choose one, both, or neither
Agda and Idris

- Agda 2.4.2.3 has a robust front-end and a questionable back-end.
- Idris 0.9.18.1 has a buggy front-end and a reasonable back-end.
Agda

- Fun interactive development environment (emacs)
- Reliable type inference and pattern matching
- Faster than computing on paper (usually)
- Little to no support for impure effects like IO or random number generation
Idris

• Shouldn’t be slower than conventional languages by much more than a constant factor

• Designed with systems programming in mind, comes with an effect system

• Gets confused by complicated pattern-matching

• Type inference is shaky, don’t rely on it
If we wanted, we could shut these machines down.

–Neo
Dirty tricks
Dirty tricks

--- Agda
open import Relation.Binary.PropositionalEquality
badcast : Bool → String
badcast x = subst (λ A → A) trustMe x

--- Idris
badcast : Bool → String
badcast x = believe_me x
Homework

1. Install Idris
2. Install Agda
3. Install Agda standard library
4. Write hello-world programs
5. Email me:
   - Your experience with dependent types
   - The topics you are most interested in
   - Your hello-world program code