Simple Types

CS4450/7450

Fall 2018

William Harrison

Today: Looking Closer at Types

- Types are themselves a language
- Wide variety of type languages of varying **expressiveness**
  - E.g., In Haskell's type system, types have
    - variables
    - variety of constructors: →, products, sums, base types
    - Polymorphism
    - "higher kinds"
    - Type class constraints
    - `(>>=) :: Monad m => m a → (a → m b) → m b`

"Simple" Types

- Today we'll start out with so-called "simple" types
- "Simple" means "no variables"
- E.g., `Int → Int` is a simple type because it contains no type variables
  - `[a] → Int` is not simple
- Today: a language of simple types
  - Its abstract syntax
  - Parsing simple types with Parsing.lhs

Abstract Syntax of Simple Types

- Simple types are combinations of base types and type constructors
  - Base types are like `Int` and `Bool` in Haskell
  - Type constructors are, well, anything that constructs a type (incl. base types), but also constructors like →
    - There are even type systems for type languages called "kind systems"
      - (→) :: Type → Type → Type

Simple Types Syntax

- BNF: Ty ::= Number | Boolean | Ty → Ty
- `data Ty = Number | Boolean | Arrow Ty Ty`
- Other possibilities include
  - products `(t1,…,tn)`
  - sums `Either t1 t2`