Types & Semantics - Lecture 2

Wouter Swierstra

Universiteit Utrecht

[Faculty of Science
Information and Computing Sciences]

Agda

- A functional language, a bit similar to Haskell
- Implicit arguments
- Flexible syntax: mixfix operators and unicode
- But we can only define total functions

Limits of totality

As a result of this last point, we need to be very careful defining functions such as `_!!_`:

```
_!!_ : {a : Set} -> Nat -> List a -> Maybe a
n !! xs = ...
```

Our types can never make promises that we can't keep!

Revisiting lookup

Adding the additional Maybe solves our problem.

But now we need to check every call to lookup to see if it is succesful or not.

But what if we know that the call must succeed.

How do we convince the type checker that lookup can be made total?

Demo

Retrospective

- Dependent types allow us to freely mix values and types.
- Totality is important: type checker performs evaluation and we want to trust our proofs!
- Pattern matching on dependent types is very subtle.