Type Driven Development with Idris
Lecture 2: Proofs, Predicates and Totality

Edwin Brady (ecb10@st-andrews.ac.uk)
University of St Andrews, Scotland, UK
@edwinbrady

MGS, April 10th 2018
Type-driven Development
Type-driven Development
A total function is a function which, for all well-typed inputs, either

- **Terminates** with a well-typed result
- ** Produces a finite, non-empty** prefix of a well-typed **infinite** result in finite time
Why do we care?

If we care about types, we should care about totality

Given \( f : T \)

- If \( f \) is total, we know that it will always give a result of type \( T \)
- If \( f \) is partial, we know that if it gives a result, it will be of type \( T \)
Why do we care?

If we care about *types*, we should care about *totality*

Given $f : \text{Theorem}$

- If $f$ is *total*, we know that it will *always* give a result of type *Theorem*
- If $f$ is *partial*, we know that *if* it gives a result, it will be of type *Theorem*
Idris checks:

- **Coverage**: patterns for all *well-typed inputs*
- **Termination**: there is a *decreasing* argument
- **Productivity**: recursive call is *guarded* by a constructor
Demonstration: Proofs, Predicates and Totality