Dependent Types in the Idris Programming Language
Part 3: Streams and Interaction

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

OPLSS, July 3rd 2017
What limitations does totality impose for practical programming?
What limitations does totality impose for practical programming?

Is it possible to write partial functions and prove totality later?
Questions from Lecture 2 (1)

- What limitations does totality impose for practical programming?
- Is it possible to write partial functions and prove totality later?
- How is impossible defined? Can I use it in a case expression?
Questions from Lecture 2 (1)

- What limitations does totality impose for practical programming?
- Is it possible to write partial functions and prove totality later?
- How is \textbf{impossible} defined? Can I use it in a \textbf{case} expression?
- How does Idris deal with Universes? (i.e. what is the type of \textbf{Type}?)
Can you say more about how erasure works?
Questions from Lecture 2 (2)

- Can you say more about how erasure works?
- Where can I get a t-shirt that says “It type checks! Ship it!”
Can you say more about how erasure works?
Where can I get a t-shirt that says “It type checks! Ship it!”
Are you Scottish, because you don’t sound Scottish?
Demonstration: Streams and Interaction
Example State Machine: Opening a Door

*State Machine Diagram:

- **DoorClosed**
  - Transition to **DoorOpen** labeled "Open"
  - Transition to **DoorClosed** labeled "Close"

- **DoorOpen**
  - Transition to **DoorClosed** labeled "Close"

- *Event Flow:
  - Ring Doorbell
States in a Bank’s ATM

- **Ready**
  - EjectCard
  - InsertCard

- **CardInserted**
  - GetPIN, CheckPIN (Incorrect)
  - CheckPIN (Correct)

- **Session**
  - EjectCard
  - Dispense
Demonstration: State Machines