Comp 311
Functional Programming

Nick Vrvilo, Two Sigma Investments
Robert “Corky” Cartwright, Rice University

Nick – Education and Work

- Computer Science BS from Brigham Young University
  - Interned at PolyServe (startup now owned by HP)
  - Co-founded startup building full-stack web apps
- Computer Science MS & PhD from Rice University
  - Research in resilience, formal verification, and HPC languages/runtimes
  - Internships:
    - Future Technologies Berkeley Lab
    - Embedded Computing Lab, National Taiwan University (NSF East Asia and Pacific Summer Institutes Program Fellow)
    - Extreme-scale Technology Group, Intel Corp (twice)
- Software Engineer at Two Sigma Investments
  Developing open source cluster scheduling software

About the Course

Course Overview

- An Introduction to Functional Programming
- Lectures: Tuesdays and Thursdays 4pm–5:15pm
- Office hours:
  - Corky: See course website
  - Nick: Tuesdays & Thursdays, immediately after class

Course Mechanics

- Course website:
  https://wiki.rice.edu/confluence/display/FPSCALA/2018-Fall
- Syllabus, lectures and homework assignments are posted there
- Lecture topics are subject to change

Piazza

- We'll be using Piazza for questions and discussions: https://piazza.com/rice/fall2018/comp311/home
- We will make a best effort to answer questions posted on this page in a timely manner
- *Bring your questions to class and office hours*

Course Overview

- No required textbook
  - We will draw from a variety of sources
- Coursework consists primarily of biweekly homework assignments
  - Make sure you do these!
  - Missing even one assignment will significantly impact your grade

Homework Assignments

- Think of the assignments in this class as short essays
- Focus as much on style as you would for an essay
- 50% of a homework grade is based on clarity and style
- 50% on correctness

Homework Assignments

- Projects are due two weeks after being assigned.
- There will be no “slip days” or other late policy. The assignments are due when they are due.
- If you have a serious conflict with the course schedule, please contact the instructors *before* the assignment due date to make arrangements.
- Aiming for roughly 10 hours of coursework per week.
- Block this time off now. Make a priority of respecting it.

Homework Assignments

- Assignments are published on Thursdays.

- Start on assignments early so that you have time to ask questions in class, on Piazza, and at office hours.

Homework Assignments

- Assignments will be programming exercises in Scala.

- We will cover the parts of Scala needed for the assignments in class.

Homework Assignments

- You have the option of DrScala and IntelliJ IDEA for assignments. DrScala is less professional but better supported.
  - Installed on all Rice systems and available for download from the course website.
- We will use SVN (turnin on CLEAR) for all assignments.
  - Instructions on the course website: https://wiki.rice.edu/confluence/display/FPSCALA/Homework+Submission+Guide

What is Functional Programming?

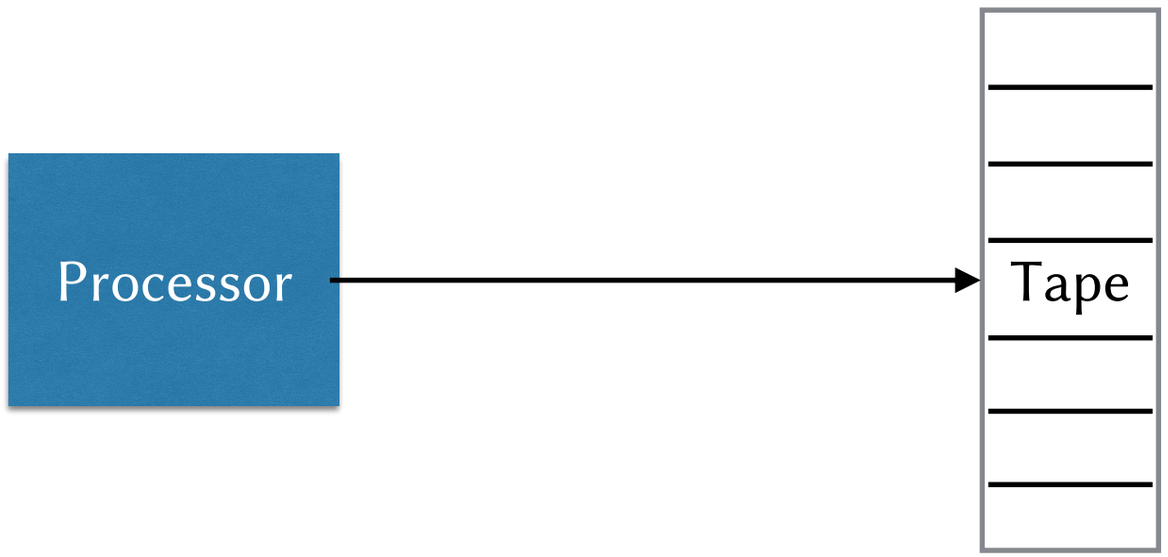
Early Models of Computation

- Turing Machines (Turing)
- Type-0 Grammars (Chomsky)
- The Lambda Calculus (Church)
- *... and many others*

Early Models of Computation

- Turing Machines (Turing)
- Type-0 Grammars (Chomsky)
- The Lambda Calculus (Church)
- *... and many others*
- To the surprise of their inventors, all of these systems turned out to be equivalent in expressive power.
- Suggests there is a deeper structure to the nature of computation.

Turing Machines



- Processor is a finite state machine that loads and stores *memory cells.*

- Turing coined the term “compute” and introduced the notion of storage.

- Many programs, languages, and computer architectures are heavily influenced by this model (and its derivates: Von Neumann, etc.) .

Early Models of Computation

- Turing Machines (Turing)
- Type-0 Grammars (Chomsky)
- **The Lambda Calculus (Church)**
- *... and many others*
- To the surprise of their inventors, all of these systems turned out to be equivalent in expressive power.
- Suggests there is a deeper structure to the nature of computation.

The Lambda Calculus

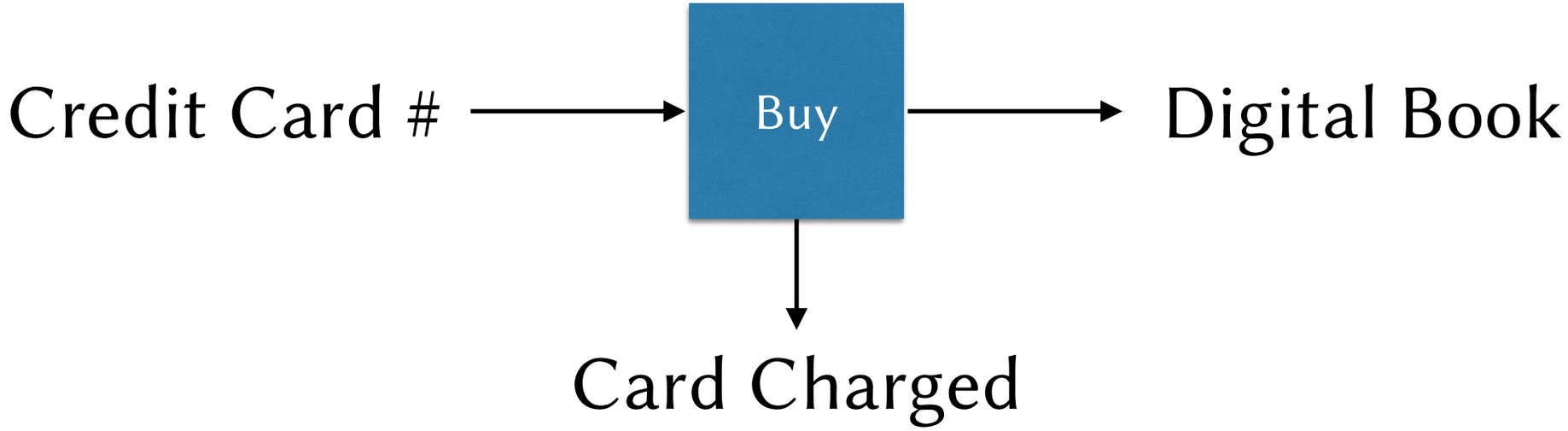
- A *calculus* consists of a set of rules for rewriting symbols
- An attempt to rebuild all of mathematics on the notion of *functions* and *applications*
- There is no mutation in the lambda calculus
- Every program consists solely of applications of functions to arguments (which are also functions)
- Applications of functions return values (which are also functions)

What is Functional Programming?

A style of programming inspired by the Lambda Calculus as a foundational model of computation.

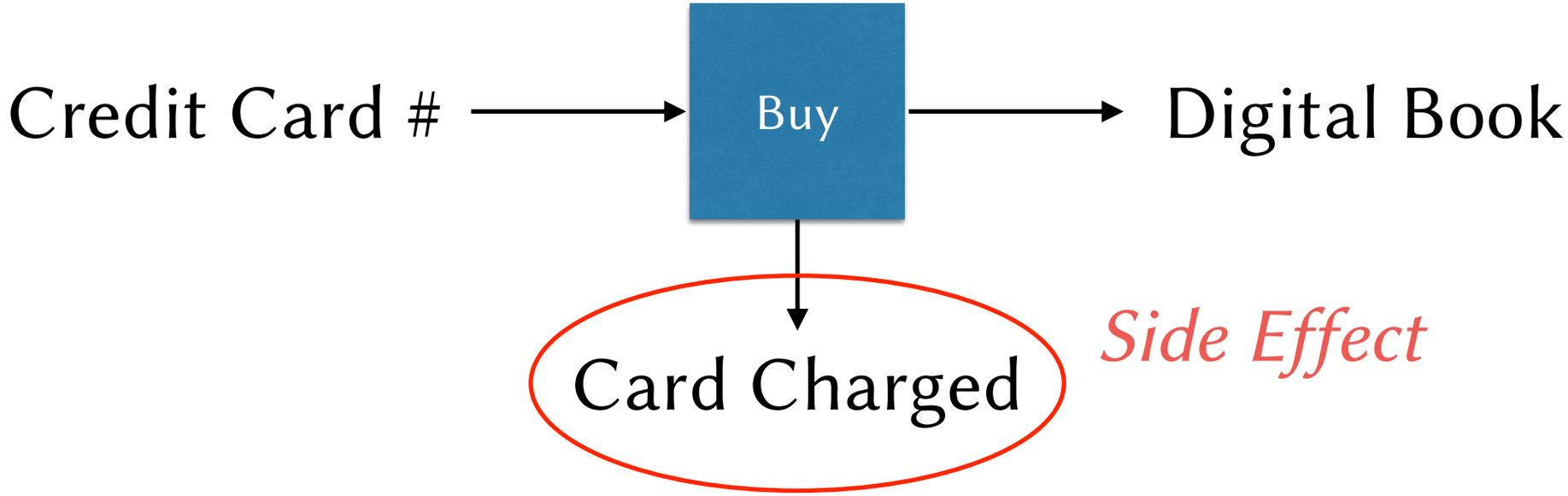
What is Functional Programming?

- A style of programming that avoids side effects



What is Functional Programming?

- A style of programming that avoids side effects



What is Functional Programming?

- A style of programming that avoids side effects

- All results of a computation are sent as output

Why Avoid Side Effects?

- **Programs are easier to write:** There are fewer interactions between program components, enabling multiple programmers (or a single programmer on multiple days) to work together more easily

- **Programs are easier to read:** Pieces of a program can be read and understood in isolation

- **Programs are easier to test:** Less context needs to be built up before calling a function to test it

- **Programs are easier to debug:** Problems can be isolated more easily, and behavior is inherently deterministic

- **Programs are easier to reason about:** The model of computation needed to understand a program without mutation is much simpler

Why Avoid Side Effects?

- **Programs are easier to execute in parallel:** Because separate pieces of a computation do not interact, it is easy to compute them on separate processors
- This is an increasingly important consideration in the era of multicore chips, big data, and distributing computing
    - *This advantage undermines an often cited argument for mutation (efficiency)*