Why Study Automata Theory and Formal Languages?

- A survey of Stanford grads 5 years out asked which of their courses did they use in their job.
- Basics like Programming took the top spots, of course.
- But among optional courses, Automata Theory stood remarkably high.
  - 3X the score for AI, for example.

Why Finite Automata and Regular Expressions?

- Regular expressions (REs) are used in many systems.
  - E.g., UNIX, Linux, OS X,… a.*b.
  - E.g., Document Type Definitions describe XML tags with a RE format like person (name, addr, child*).
- Finite automata model protocols, electronic circuits.
  - Theory is used in *model-checking*.

Why Context-Free Grammars?

- Context-free grammars (CFGs) are used to describe the syntax of essentially **every** modern programming language.
- Every modern complier uses CFG concepts to parse programs
  - Not to forget their important role in describing natural languages.
- And Document Type Definitions are really CFG's.

Why Turing Machines?

- When developing solutions to real problems, we often confront the limitations of what software can do.
  - *Undecidable* things – no program can do it 100% of the time with 100% accuracy.
  - *Intractable* things – there are programs, but no fast programs.
- A course on Automata Theory and Formal Languages gives you the tools.

Other Good Stuff

- We'll learn how to deal formally with discrete systems.
  - Proofs: You never really prove a program correct, but you need to be thinking of why a tricky technique really works.
- You'll gain experience with abstract models and constructions.
  - Models layered software architectures.

Course Outline

- Regular Languages and their descriptors:
  - Finite automata, nondeterministic finite automata, regular expressions.
  - Algorithms to decide questions about regular languages, e.g., is it empty?
  - Closure properties of regular languages.

Course Outline – (2)

- Context-free languages and their descriptors:
  - Context-free grammars, pushdown automata.
  - Decision and closure properties.

Course Outline – (3)

- Recursive and recursively enumerable languages.
  - Turing machines, decidability of problems.
  - The limit of what can be computed.
- Intractable problems.
  - Problems that (appear to) require exponential time.
  - NP-completeness and beyond.