

Theory of Computability

Spring 2006
Lecture 1—April 4, 2006
Finite State Automata

Instructor: Neil Rhodes

What we study

Computability

- What can be computed?

Complexity

- How difficult is it to compute certain things?
- How much time does it take?
- How much space does it need?

Models of computation

- Various models of how to do computation
 - Finite automata
 - Context-free grammar
 - Turing Machines
- Yield various classes of what can be done

Why we study

Interesting in its own right

- One question $P=NP?$ is one of the major open mathematical questions

Useful byproducts

- Finite automata useful for:
 - Writing scanners for compilers
 - Regular expressions (Perl/grep/etc.)
- Context-free grammars useful for:
 - Writing parsers for languages
- Turing machines useful for:
 - exploring limits on computation

Organization of course

Part 1: Finite Automata

Part 2: Context-free grammars

(Midterm)

Part 3: Turing Machines

Part 4: Complexity Theory

(Comprehensive Final)

Needed background

Discrete math

- Sets
 - Powerset
 - Cross-product
- Ordered tuple
- Functions
 - Domain
 - codomain
 - injective (one-to-one)
 - surjective (onto)
 - bijective (one-to-one and onto)
- Graphs
- Proofs
- Boolean logic
- Strings
 - Concatenation

Simplifying the problem

We'll look at computation as a problem of taking input and

- accepting the input
- rejecting the input

What is the input?

- Could be any finite alphabet
 - Often, just 0's and 1's; could always encode any larger alphabet

A computing machine M *recognizes* a language L

- $L = \{w \mid M \text{ accepts } w\}$

Finite automata

Computation where the computation can be in only 1 of a finite number of states:

- Formal definition:
 - To be covered in discussion section: a 5-tuple
- Informal: picture
 - Represent each state with a circle
 - Each state has a labeled arrow for each symbol in the alphabet pointing to another state
 - Some state has an incoming arrow (the start state)
 - Some states are double-circled (the final states)
 - Computation:
 - Start with start state
 - foreach symbol in string (from left to right)
 - Move to new state based on symbol
 - If end in a final state, accept, otherwise, reject

Example: a Finite automata recognizing strings of 0s and 1s with an even number of 0s

Example

Recognize *all* strings over $\{0, 1\}$

Example

Finite automata recognizing strings over $\{0, 1\}$ with an even number of 0s *and* even number of 1s

Example

Binary numbers with an even number of 0s *or* an even number of 1s

Example

Strings over $\{0, 1\}$ where the second symbol is not a 1

Example

Binary numbers divisible by 4

Example

Every 0 immediately followed by at least two 1s

Example

Divisible by 5

Example

Binary string contains 101101