CS 611: Theory of Computation

Hongmin Li

Department of Computer Science
California State University, East Bay

Complexity Theory, Computability Theory and Automata Theory

- Automata Theory: Theory of computation begins with the question: What is a computer? Real computers are quite complicated, we use an idealized computer called a Computational Model.
- Computability Theory: Study whether some problems can be solved by a computer or not (solvable or not solvable).
- Complexity Theory: Classify problems as easy ones and hard ones.

Decision Problems

Decision Problems

Given input, decide "yes" or "no"

- Examples: Is x an even number? Is x prime? Is there a path from s to t in graph G?
- i.e., Compute a boolean function of input

General Computational Problem

In contrast, typically a problem requires computing some non-boolean function, or carrying out interactive/reactive computation in a distributed environment

- Examples: Find the factors of x. Find the balance in account number x.

- In this course, we will study decision problems because aspects of computability are captured by this special class of problems

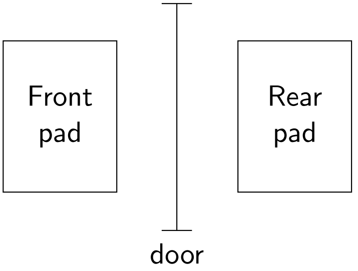
What Does a Computation Look Like?

- Some code (a.k.a. control): the same for all instances
- The input (a.k.a. problem instance): encoded as a string over a finite alphabet
- As the program starts executing, some memory (a.k.a. state)
  - Includes the values of variables (and the "program counter")
  - State evolves throughout the computation
  - Often, takes more memory for larger problem instances
- But some programs do not need larger state for larger instances!

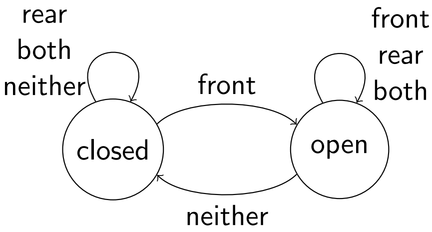
Finite State Computation

- Finite state: A fixed upper bound on the size of the state, independent of the size of the input
  - A sequential program with no dynamic allocation using variables that take boolean values (or values in a finite enumerated data type)
  - If t-bit state, at most 2^t possible states
- Not enough memory to hold the entire input
  - "Streaming input": automaton runs (i.e., changes state) on seeing each bit of input

An Automatic Door



Top view of Door



State diagram of controller

- Input: A stream of events <front>, <rear>, <both>, <neither> . . .
- Controller has a single bit of state.

Finite Automata

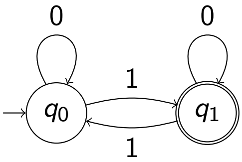
Details

Automaton

A finite automaton has: Finite set of states, with start/initial and accepting/final states; Transitions from one state to another on reading a symbol from the input.

Computation

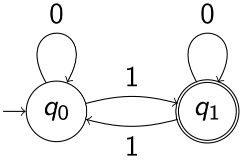
Start at the initial state; in each step, read the next symbol of the input, take the transition (edge) labeled by that symbol to a new state. Acceptance/Rejection: If after reading the input w, the machine is in a final state then w is accepted; otherwise w is rejected.



Transition Diagram of automaton

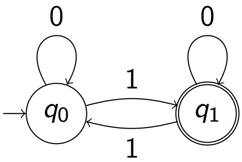
Example: Computation

- On input 1001, the computation is
  1. Start in state q_0. Read 1 and goto q_1.
  2. Read 0 and goto q_1.
  3. Read 0 and goto q_1.
  4. Read 1 and goto q_0. Since q_0 is not a final state 1001 is rejected.



Example: Computation

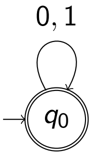
- On input 1001, the computation is
  1. Start in state q_0. Read 1 and goto q_1.
  2. Read 0 and goto q_1.
  3. Read 0 and goto q_1.
  4. Read 1 and goto q_0. Since q_0 is not a final state 1001 is rejected.
- On input 010, the computation is
  1. Start in state q_0. Read 0 and goto q_0.
  2. Read 1 and goto q_1.
  3. Read 0 and goto q_1. Since q_1 is a final state 010 is accepted.



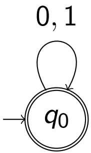
Break

Let's take a break and ask questions.
Examles next.

Example I

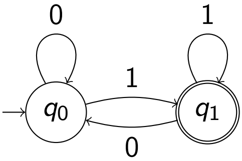


Example I

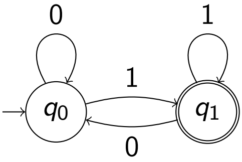


Automaton accepts all strings of 0s and 1s

Example II

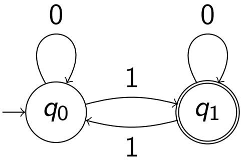


Example II

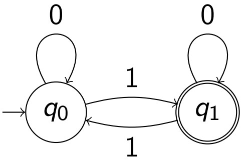


Automaton accepts strings ending in 1

Example III

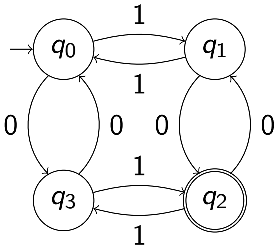


Example III

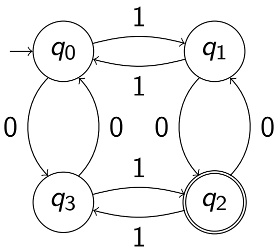


Automaton accepts strings having an odd number of 1s

Example IV



Example IV



Automaton accepts strings having an odd number of 1s and odd number of 0s

Finite Automata in Practice

- `grep`
- Thermostats
- Coke Machines
- Elevators
- Train Track Switches
- Security Properties
- Lexical Analyzers for Parsers

Break

Let's take a break and ask questions. Formal Definitions of DFA next.

Alphabet

Definition

An alphabet is any finite, non-empty set of symbols. We will usually denote it by Σ.

Example

Examples of alphabets include $\{0, 1\}$ (binary alphabet); $\{a, b, \ldots, z\}$ (English alphabet); the set of all ASCII characters; $\{\text{moveforward, moveback, rotate90}\}$.

Strings

Definition

A string or word over alphabet Σ is a (finite) sequence of symbols in Σ. Examples are '0101001', 'string', '$\langle\text{moveback}\rangle\langle\text{rotate90}\rangle$'

- ϵ is the empty string.
- The length of string u (denoted by $|u|$) is the number of symbols in u. Example, $|\epsilon| = 0$, $|011010| = 6$.
- Concatenation: uv is the string that has a copy of u followed by a copy of v. Example, if $u = \text{`}cat'$ and $v = \text{`}nap'$ then $uv = \text{`}catnap'$. If $v = \epsilon$ the $uv = vu = u$.
- u is a prefix of v if there is a string w such that $v = uw$. Example 'cat' is a prefix of '$catnap$'.

Languages

Definition

- For alphabet Σ, Σ^* is the set of all strings over Σ. Σ^n is the set of all strings of length n.
- A language over Σ is a set $L \subseteq \Sigma^*$. For example $L = \{1, 01, 11, 001\}$ is a language over $\{0, 1\}$.
    - A language L defines a decision problem: Inputs (strings) whose answer is 'yes' are exactly those belonging to L

Set Notation

We will often define languages using the set builder notation. Thus, $L = \{w \in \Sigma^* \mid p(w)\}$ is the collection of all strings w over Σ that satisfy the property p.

Example

- $L = \{w \in \{0,1\}^* \mid |w| \text{ is even}\}$ is the set of all even length strings over $\{0,1\}$.

Set Notation

We will often define languages using the set builder notation. Thus, $L = \{w \in \Sigma^* \mid p(w)\}$ is the collection of all strings w over Σ that satisfy the property p.

Example

- $L = \{w \in \{0,1\}^* \mid |w| \text{ is even}\}$ is the set of all even length strings over $\{0,1\}$.

Set Notation

We will often define languages using the set builder notation. Thus, $L = \{w \in \Sigma^* \mid p(w)\}$ is the collection of all strings w over Σ that satisfy the property p.

Example

- $L = \{w \in \{0,1\}^* \mid |w| \text{ is even}\}$ is the set of all even length strings over $\{0,1\}$.

Set Notation

We will often define languages using the set builder notation. Thus, $L = \{w \in \Sigma^* \mid p(w)\}$ is the collection of all strings w over Σ that satisfy the property p.

Example

- $L = \{w \in \{0,1\}^* \mid |w| \text{ is even}\}$ is the set of all even length strings over $\{0,1\}$.
- $L = \{w \in \{0,1\}^* \mid \text{there is a } u \text{ such that } wu = 10001\}$ is

Set Notation

We will often define languages using the set builder notation. Thus, $L = \{w \in \Sigma^* \mid p(w)\}$ is the collection of all strings w over Σ that satisfy the property p.

Example

- $L = \{w \in \{0,1\}^* \mid |w| \text{ is even}\}$ is the set of all even length strings over $\{0,1\}$.
- $L = \{w \in \{0,1\}^* \mid \text{there is a } u \text{ such that } wu = 10001\}$ is the set of all prefixes of 10001.

Defining an Automaton

To describe an automaton, we to need to specify

- What the alphabet is,
- What the states are,
- What the initial state is,
- What states are accepting/final, and
- What the transition from each state and input symbol is.

Thus, the above 5 things are part of the formal definition.

Finite Automata

Formal Definition

Definition

A finite automaton is $M = (Q, \Sigma, \delta, q_0, F)$, where

- Q is the finite set of states
- Σ is the finite alphabet
- $\delta : Q \times \Sigma \to Q$ "Next-state" transition function
- $q_0 \in Q$ initial state
- $F \subseteq Q$ final/accepting states

Deterministic Finite Automata

Formal Definition

Definition

A deterministic finite automaton (DFA) is $M = (Q, \Sigma, \delta, q_0, F)$, where

- Q is the finite set of states
- Σ is the finite alphabet
- $\delta : Q \times \Sigma \to Q$ "Next-state" transition function
- $q_0 \in Q$ initial state
- $F \subseteq Q$ final/accepting states

Given a state and a symbol, the next state is "determined".