1 Introducing Finite Automata

1.1 Problems and Computation

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 an 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!

1.2 Finite Automata: Informal Overview

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

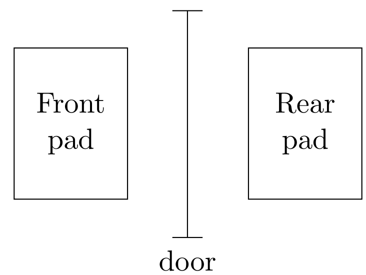


Figure 1: Top view of Door

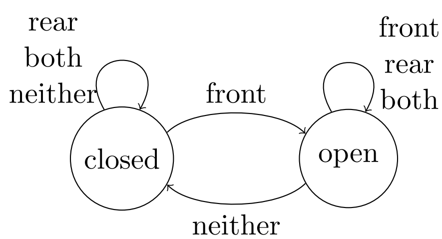


Figure 2: State diagram of controller

- *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*.

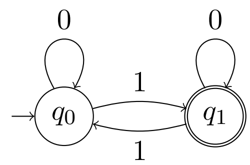


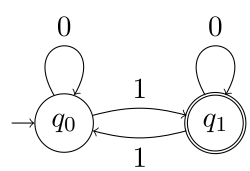
Figure 3: Transition Diagram of automaton

Conventions

- The initial state is shown by drawing an incoming arrow into the state, with no source.
- Final/accept states are indicated by drawing them with a double circle.

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*.



1.3 Applications

Finite Automata in Practice

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

---

2 Formal Definitions

2.1 Alphabets, Strings and Languages

Alphabet

Definition 1. An *alphabet* is any finite, non-empty set of symbols. We will usually denote it by Σ.

Example 2. Examples of alphabets include $\{0, 1\}$ (binary alphabet); $\{a, b, \ldots, z\}$ (English alphabet); the set of all ASCII characters; {moveforward, moveback, rotate90}.

Strings

Definition 3. A *string* or *word* over alphabet Σ is a (finite) sequence of symbols in Σ. Examples are '0101001', 'string', '⟨moveback⟩⟨rotate90⟩'

- 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 4.

- 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 5.

- $L = \{w \in \{0,1\}^* \mid |w| \text{ is even}\}$ 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.

2.2 Deterministic Finite Automaton

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.

Deterministic Finite Automata
Formal Definition

Definition 6. 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

	0	1
q_0	q_0	q_1
q_1	q_1	q_0

Figure 5: Transition Table representation

Given a state and a symbol, the next state is “determined”.

Formal Example of DFA

Example 7.

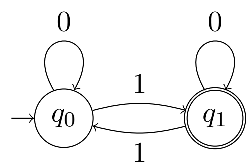


Figure 4: Transition Diagram of DFA

Formally the automaton is $M = (\{q_0, q_1\}, \{0,1\}, \delta, q_0, \{q_1\})$ where

$$\begin{array}{ll} \delta(q_0, 0) = q_0 & \quad \delta(q_0, 1) = q_1 \\ \delta(q_1, 0) = q_1 & \quad \delta(q_1, 1) = q_0 \end{array}$$

Computation

Definition 8. For a DFA $M = (Q, \Sigma, \delta, q_0, F)$, string $w = w_1 w_2 \cdots w_k$, where for each i $w_i \in \Sigma$, and states $q_1, q_2 \in Q$, we say $q_1 \xrightarrow{w}_M q_2$ if there is a sequence of states $r_0, r_1, \ldots r_k$ such that

- $r_0 = q_1$,
- for each i, $\delta(r_i, w_{i+1}) = r_{i+1}$, and
- $r_k = q_2$.

Definition 9. For a DFA $M = (Q, \Sigma, \delta, q_0, F)$ and string $w \in \Sigma^*$, we say M *accepts* w iff $q_0 \xrightarrow{w}_M q$ for some $q \in F$.

Useful Notation

Definition 10. For a DFA $M = (Q, \Sigma, \delta, q_0, F)$, let us define a function $\hat{\delta}_M : Q \times \Sigma^* \to \mathcal{P}(Q)$ such that $\hat{\delta}_M(q, w) = \{q' \in Q \mid q \xrightarrow{w}_M q'\}$.

We could say M accepts w iff $\hat{\delta}_M(q_0, w) \cap F \neq \emptyset$.

Proposition 11. *For a DFA $M = (Q, \Sigma, \delta, q_0, F)$, and any $q \in Q$, and $w \in \Sigma^*$, $|\hat{\delta}_M(q, w)| = 1$.*

Acceptance/Recognition

Definition 12. The *language accepted or recognized* by a DFA M over alphabet Σ is $\mathbf{L}(M) = \{w \in \Sigma^* \mid M \text{ accepts } w\}$. A language L is said to be *accepted/recognized* by M if $L = \mathbf{L}(M)$.

2.3 Examples

Example I

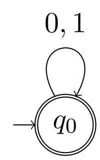


Figure 6: Automaton accepts all strings of 0s and 1s

Example II

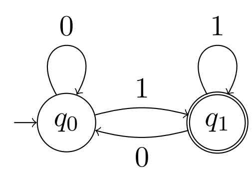


Figure 7: Automaton accepts strings ending in 1

Example III

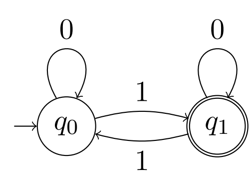


Figure 8: Automaton accepts strings having an odd number of 1s

Example IV

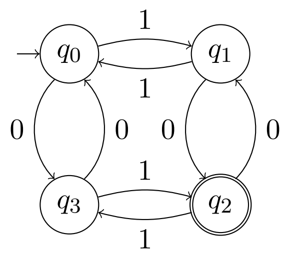


Figure 9: Automaton accepts strings having an odd number of 1s and odd number of 0s