Lecture 3:

• Deterministic Finite Automata
• Non-deterministic FAs

Reading:
Sipser Ch 1.1-1.2

Mark Bun
September 9, 2021


Last Time

• Parts of a theory of computation: Model for machines, model for problems, theorems relating machines and problems

• Strings: Finite concatenations of symbols

• Languages: Sets $L$ of strings

• Computational (decision) problem: Given a string $x$, is it in the language $L$?
Deterministic Finite Automata
A (Real-Life?) Example

• Example: Kitchen scale

• \( P = \) Power button (ON / OFF)

• \( U = \) Units button (cycles through g / oz / lb)
  
  Only works when scale is ON, but units remembered when scale is OFF

• Starts OFF in g mode

• A computational problem: Does a sequence of button presses in \( \{P, U\}^* \) leave the scale ON in oz mode?
Machine Models

- **Finite Automata (FAs):** Machine with a finite amount of unstructured memory

  Input: 
  \[
  P \quad U \quad P \quad U \quad \ldots
  \]

  Control scans left-to-right

  Finite control
A DFA for the Kitchen Scale Problem
A DFA Recognizing Parity

The language recognized by a DFA is the set of inputs on which it ends in an “accept” state

Parity: Given a string consisting of a’s and b’s, does it contain an even number of a’s?

\[ \Sigma = \{a, b\} \quad L = \{w \mid w \text{ contains an even number of } a's\} \]

Which state is reached by the parity DFA on input aabab?

a) “even”
b) “odd”
Anatomy of a DFA

$q_0$ 0 1 $q_1$ 0,1 1 $q_2$

$q_3$
Some Tips for Thinking about DFAs

Given a DFA, what language does it recognize?
- Try experimenting with it on short strings. Do you notice any patterns?
- What kinds of inputs cause the DFA to get trapped in a state?

Given a language, construct a DFA recognizing it
- Imagine you are a machine, reading one symbol at a time, always prepared with an answer
- What is the essential information that you need to remember? Determines set of states.
What language does this DFA recognize?
Practice!

• Lots of worked out examples in Sipser

• Automata Tutor: https://automata-tutor.model.in.tum.de/
Formal Definition of a DFA

A finite automaton is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$

- $Q$ is the set of states
- $\Sigma$ is the alphabet
- $\delta: Q \times \Sigma \to Q$ is the transition function
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is the set of accept states
A DFA for Parity

Parity: Given a string consisting of $a$’s and $b$’s, does it contain an even number of $a$’s?

$\Sigma = \{a, b\} \quad L = \{w \mid w \text{ contains an even number of } a\text{'s}\}$

State set $Q = \{q_0, q_1\}$

Alphabet $\Sigma = \{a, b\}$

Transition function $\delta$

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>$a$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_1$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Start state $q_0$

Set of accept states $F = \{q_1\}$
Formal Definition of DFA Computation

A DFA $M = (Q, \Sigma, \delta, q_0, F)$ accepts a string $w = w_1w_2 \cdots w_n \in \Sigma^*$ (where each $w_i \in \Sigma$) if there exist $r_0, \ldots, r_n \in Q$ such that

1. $r_0 = q_0$
2. $\delta(r_i, w_{i+1}) = r_{i+1}$ for each $i = 0, \ldots, n - 1$, and
3. $r_n \in F$

$L(M) =$ the language of machine $M$

= set of all strings machine $M$ accepts

$M$ recognizes the language $L(M)$
A DFA $M = (Q, \Sigma, \delta, q_0, F)$ accepts a string $w = w_1w_2 \cdots w_n \in \Sigma^*$ (where each $w_i \in \Sigma$) if there exist $r_0, \ldots, r_n \in Q$ such that

1. $r_0 = q_0$
2. $\delta(r_i, w_{i+1}) = r_{i+1}$ for each $i = 0, \ldots, n - 1$, and
3. $r_n \in F$

Example: Computing with the Parity DFA

Let $w = abba$

Does $M$ accept $w$?

What is $\delta(r_2, w_3)$?

a) $q_0$

b) $q_1$
Regular Languages

**Definition:** A language is regular if it is recognized by a DFA

\[ L = \{ w \in \{a, b\}^* \mid w \text{ has an even number of } a's \} \text{ is regular} \]

\[ L = \{ w \in \{0, 1\}^* \mid w \text{ contains } 001 \} \text{ is regular} \]

Many interesting problems are captured by regular languages

NETWORK PROTOCOLS

COMPILERS

GENETIC TESTING

ARITHMETIC
Let $\text{TCPS} = \{ w \mid w \text{ is a complete TCP Session} \}$

**Theorem.** TCPS is regular
Comments:

- Are delimited by /* */
- Cannot have nested /* */
- Must be closed by */
- */ is illegal outside a comment

**COMMENTS** = \{strings over \{0,1, /, *\} with legal comments\}

Theorem. **COMMENTS** is regular.
Genetic Testing

DNA sequences are strings over the alphabet \( \{A, C, G, T\} \).

A gene \( g \) is a special substring over this alphabet.

A genetic test searches a DNA sequence for a gene.

\[
\text{GENETICTEST}_g = \{ \text{strings over} \ \{A, C, G, T\} \ \text{containing} \ g \ \text{as a substring} \}
\]

Theorem. \( \text{GENETICTEST}_g \) is regular for every gene \( g \).
Arithmetic

Let $\Sigma = \{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \}$

• A string over $\Sigma$ has three ROWS (ROW₁, ROW₂, ROW₃)
• Each ROW $b₀b₁b₂ \ldots bₙ$ represents the integer
  $$b₀ + 2b₁ + \ldots + 2ⁿbₙ.$$ 
• Let $ADD = \{ S \in \Sigma^* \mid ROW₁ + ROW₂ = ROW₃ \}$

**Theorem.** ADD is regular.
Non-deterministic Finite Automata
Nondeterminism

In a DFA, the machine is always in exactly one state upon reading each input symbol.

In a nondeterministic FA, the machine can try out many different ways of reading the same string:
- Next symbol may cause an NFA to “branch” into multiple possible computations.
- Next symbol may cause NFA’s computation to fail to enter any state at all.
A Nondeterministic Finite Automaton (NFA) accepts if there exists a way to make it reach an accept state.
Non-determinism

Example: Does this NFA accept the string 1100?
Non-determinism

Example: Does this NFA accept the string 11?
Some special transitions

0

0, 1

ε

1
Example

\[ L(M) = \]
Example

$L(N) = \begin{align*}
\text{a) } & \{w \mid w \text{ ends with 101}\} \\
\text{b) } & \{w \mid w \text{ ends with 11 or 101}\} \\
\text{c) } & \{w \mid w \text{ contains 101}\} \\
\text{d) } & \{w \mid w \text{ contains 11 or 101}\}
\end{align*}$