Lecture 3:

- Deterministic Finite Automata
- Non-deterministic FAs

Reading:
Sipser Ch 1.1-1.2

Mark Bun
September 13, 2022
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 $L$?
Languages

Languages = computational (decision) problems

Input: String $x \in \Sigma^*$

Output: Is $x \in L$? (YES or NO?)

The language **recognized** by a program is the set of strings $x \in \Sigma^*$ that it **accepts**

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

On input $x = x_1x_2 \ldots x_n$:
- flag = FALSE
  - For $i = 1, \ldots, n$:
    - If $x_i = b$:
      - flag = !flag
    - Else:
      - If flag: accept
      - Else: reject
Deterministic Finite Automata
A (Real-Life?) Example

• Example: Kitchen scale

• $P = \text{Power button (ON / OFF)}$

• $U = \text{Units button (cycles through g / oz / lb)}$
  
  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 U P U ...  
```

- **Finite control**

Control scans left-to-right

1) How does the control start?
2) What are the different “states” that the control can be in?
3) When the control reads a new input character, how does it transition to a new state?
4) How do I know if I’m in the desired state at the end?
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?

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

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

a) “even”  
b) “odd”
Anatomy of a DFA

States
- start state
- accept states ("final states")

Transitions

$q_0$ 0 $q_1$
$q_1$ 1 $q_2$
$q_2$ 1 $q_3$
$q_3$ 0 $q_0$

$q_0$ 0 $q_1$
$q_1$ 0 $q_2$
$q_2$ 0 $q_3$
$q_3$ 0 $q_0$
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?

\[ L = \{ \sum^n \mid 001 \text{ is a substring of } \sum^3 \} \]
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

\[ Q = \{ q_0, q_1, q_2 \} \]
\[ \Sigma = \{ a, b \} \]

\( \delta \) is given by:

- For example:
  \[ F = \{ q_1, q_2 \} \]
  \[ F = \emptyset \]
A DFA for Parity

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

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

State set $Q = \{q_0, q_1\}$
Alphabet $\Sigma = \{a, b\}$
Transition function $\delta : Q \times \Sigma \to Q$

Start state $q_0$
Set of accept states $F = \{q_0\}$
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 $M$, “the language recognized by $M$”

= set of all strings machine $M$ accepts

$M$ recognizes the language $L(M)$
Example: Computing with the Parity DFA

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. $r_n \in F$

Let $w = abba$ Does $M$ accept $w$?

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

a) $q_0$

b) $q_1$

Let $w = w_1w_2w_3w_4$. What is $\delta(r_3, w_4)$?

$\delta(r_3, w_4) = \delta(q_1, a) = q_0$
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
Compilers

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.

**GENETICTEST**$_g$ = \{strings over \{A, C, G, T\} containing $g$ as a substring\}

**Theorem:** GENETICTEST$_g$ is regular for every gene $g$. 
LET $\Sigma = \{ [0], [01], [10], [11], [1], [10], [11], [11] \}$

- A string over $\Sigma$ has three ROWS (ROW$_1$, ROW$_2$, ROW$_3$)
- Each ROW $b_0 b_1 b_2 \ldots b_N$ represents the integer $b_0 + 2b_1 + \ldots + 2^N b_N$.
- Let ADD = $\{ S \in \Sigma^* \mid$ ROW$_1 +$ ROW$_2 =$ ROW$_3 \}$

**Theorem.** ADD is regular.