Lecture 2:

• Deterministic Finite Automata
• Regular Operations
• Non-deterministic FAs

Reading:
Sipser Ch 1.1-1.2

Mark Bun
January 27, 2020
Deterministic Finite Automata
A (Real-Life?) Example

• **Example:** Car stereo

• $P =$ Power button (ON/OFF)

• $S =$ Source button (cycles through Radio/Bluetooth/USB)
  
  Only works when stereo is ON, but source remembered when stereo is OFF

• Starts OFF in Radio mode

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

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

<table>
<thead>
<tr>
<th>$P$</th>
<th>$S$</th>
<th>$P$</th>
<th>$S$</th>
<th>...</th>
</tr>
</thead>
</table>

Control scans left-to-right

"State diagram"

- Different states control can be in
- How transitions between states
- How it decides to accept or reject
A DFA for the car stereo problem

[Diagram showing a DFA with states labeled 'ON R', 'OFF R', 'ON B', 'OFF B', and 'OFF U', with transitions labeled 'S' between 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?

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

- States
- Transitions
- Accepting or final states

$q_0$, $q_1$, $q_2$, $q_3$
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
- \( \delta : Q \times \Sigma \rightarrow Q \) 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?

$\Sigma = \{a, b\}$ \hspace{1cm} $L = \{w \mid w$ contains an even number of $a$’s\}

State set $Q = \{q_0, q_1, q_2\}$

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>$q_1$</td>
<td>$q_0$</td>
</tr>
<tr>
<td>$q_1$</td>
<td>$q_1$</td>
<td>$q_0$</td>
</tr>
</tbody>
</table>

Start state $q_0$

Set of accept states $F = \{q_0, q_2\}$
Formal Definition of DFA Computation

A DFA \( M = (Q, \Sigma, \delta, q_0, F) \) accepts a string \( w = w_1 w_2 \cdots w_n \in \Sigma^* \) (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. \( r_n \in F \)

\[ L(M) = \text{the language of machine } M \]
\[ = \text{set of all (finite) strings machine } M \text{ accepts} \]
\( M \text{ recognizes the language } L(M) \)
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 $w$?
Automata Tutor

http://automatatutor.com/
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 programs recognize 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\}.

$$S = C G T A C A A A A A$$

A **gene** $g$ is a special substring over this alphabet.

$$g = T A C$$

A **genetic test** searches a DNA sequence for a gene.

Is $T A C$ a substring of $S$?

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

$$S \in \text{GENETICTEST}_g$$

**Theorem.** **GENETICTEST**$_g$ is regular for every gene $g$. 
Arithmetic

LET \( \Sigma_3 = \{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \} \)

• A string over \( \Sigma_3 \) has three ROWS (ROW\(_1\), ROW\(_2\), ROW\(_3\))
• Each ROW \( b_0b_1b_2 \ldots b_N \) represents the integer
  \[ b_0 + 2b_1 + \ldots + 2^Nb_N. \]
• Let ADD = \( \{ S \in \Sigma_3^* \mid \text{ROW}_1 + \text{ROW}_2 = \text{ROW}_3 \} \)

**Theorem.** ADD is regular.
Regular Operations
An Analogy

In algebra, we try to identify operations which are common to many different mathematical structures

**Example:** The integers $\mathbb{Z} = \{... -2, -1, 0, 1, 2, ... \}$ are **closed** under
- Addition: $x + y$
- Multiplication: $x \times y$
- Negation: $-x$
- ...but **NOT** Division: $x / y$  \[ x = 1, \ y = 2 \quad x / y = \frac{1}{2} \]

We’d like to investigate similar closure properties of the class of regular languages
Regular operations on languages

Let $A, B \subseteq \Sigma^*$ be languages. Define

**Union:** $A \cup B = \{ w \in \Sigma^* | w \in A \text{ or } w \in B \}$

**Concatenation:** $A \circ B = \{ wv | w \in A \text{ and } v \in B \}$

**Star:** $A^* = \{ w_1w_2 \ldots w_n | w_i \in A, i = 1, \ldots, n, n \geq 0 \}$

$= \varepsilon^* \cup A \cup AA \cup AAA \cup \ldots$
Other operations

Let $A, B \subseteq \Sigma^*$ be languages. Define

Complement: $\bar{A} = \{w \in \Sigma^* | w \notin A\}$

Intersection: $A \cap B = \{w \in \Sigma^* | w \in A \text{ and } w \in B\}$

Reverse: $A^R = \{w_1 w_2 \ldots w_n | w_n w_{n-1} \ldots w_2 w_1 \in A\}$
Closure properties of the regular languages

**Theorem:** The class of regular languages is closed under all three regular operations (union, concatenation, star), as well as under complement, intersection, and reverse.

i.e., if $A$ and $B$ are regular, applying any of these operations yields a regular language
Proving Closure Properties
Complement

Complement: $\tilde{A} = \{ w | w \notin A \}$

**Theorem:** If $A$ is regular, then $\tilde{A}$ is also regular

Proof idea:
Union

Union: \( A \cup B = \{ w \mid w \in A \text{ or } w \in B \} \)

**Theorem:** If \( A \) and \( B \) are regular, then so is \( A \cup B \)

**Proof:**

Let \( M_A = (Q_A, \Sigma, \delta_A, q_0^A, F_A) \) be a DFA recognizing \( A \) and \( M_B = (Q_B, \Sigma, \delta_B, q_0^B, F_B) \) be a DFA recognizing \( B \).

**Goal:** Construct a DFA \( M = (Q, \Sigma, \delta, q_0, F) \) that recognizes \( A \cup B \).
Example

\[ M = ? \]
Closure under union proof (cont’d)

Idea: Run both $M_A$ and $M_B$ at the same time

“Cross-product construction”

\[ Q = Q_A \times Q_B = \{(q_A, q_B) \mid q_A \in A \text{ and } q_B \in B\} \]

\[ \delta ( (q_A, q_B), \sigma) = (\delta_A(q_A,\sigma), \delta_B(q_B,\sigma)) \]

\[ q_0 = (q_0^A, q_0^B) \]

\[ F = \{(q_A, q_B) \mid q_A \in F_A \text{ or } q_B \in F_B\} = F_A \times Q_B \cup Q_A \times F_B \]
Example (cont’d)

$M_A$

$M_B$

$M$

$q_0^A \xleftrightarrow{0} q_1^A$

$q_0^A \xleftrightarrow{1} q_1^A$

$q_0^B \xleftrightarrow{1} q_1^B$

$q_0^B \xleftrightarrow{0} q_1^B$

$q_0 \xleftrightarrow{a, b} q_1$
Intersection

Intersection: \( A \cap B = \{ w \mid w \in A \text{ and } w \in B \} \)

**Theorem:** If \( A \) and \( B \) are regular, then so is \( A \cap B \)

**Proof:**

Let \( M_A = (Q_A, \Sigma, \delta_A, q_0^A, F_A) \) be a DFA recognizing \( A \) and \( M_B = (Q_B, \Sigma, \delta_B, q_0^B, F_B) \) be a DFA recognizing \( B \)

**Goal:** Construct a DFA \( M = (Q, \Sigma, \delta, q_0, F) \) that recognizes \( A \cap B \)
Intersection

Intersection: $A \cap B = \{ w \mid w \in A \text{ and } w \in B \}$

**Theorem:** If $A$ and $B$ are regular, then so is $A \cap B$

**Another Proof:**

$$A \cap B = \overline{A} \cup \overline{B}$$
Reverse

Reverse: \( A^R = \{ w_1w_2 \cdots w_n | w_n \cdots w_1 \in A \} \)

**Theorem:** If \( A \) is regular, then \( A^R \) is also regular

**Proof idea:**

Let \( M = (Q, \Sigma, \delta, q_0, F) \) be a DFA recognizing \( A \)

**Goal:** Construct a DFA \( M' = (Q', \Sigma, \delta', q'_0, F') \) that recognizes \( A^R \)

Define \( M' \) as \( M \) but

- With the arrows reversed
- With start and accept states swapped
Example (Reverse)

\[ M \]

\[ M' \]
Closure under reverse

\( M' \) is not always a DFA!

- It might have many start states
- Some states may have too many outgoing edges, or none at all

1/27/2020 CS332 - Theory of Computation 33
Nondeterminism

A **Nondeterministic Finite Automaton** (NFA) accepts if there is a way to make it reach an accept state.