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

Input

\[
\begin{array}{cccc}
 P & S & P & S & \ldots \\
\end{array}
\]

Finite control

Control scans left-to-right
A DFA for the car stereo problem
A DFA for Parity

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

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

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

**State set** $Q =$

**Alphabet** $\Sigma =$

**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** $F =$
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. $r_n \in F$

$L(M) = \text{the language of machine } M$

$= \text{set of all (finite) strings machine } M \text{ accepts}$

$M \text{ recognizes the language } L(M)$
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

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

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.

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

**Theorem.** \(GENETICTEST_g\) is regular for every gene \(g\).
Arithmetic

\[
\text{LET } \Sigma_3 = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\}
\]

- A string over \( \Sigma_3 \) 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 = \( \{ S \in \Sigma_3^* \mid \text{ROW}_1 + \text{ROW}_2 = \text{ROW}_3 \} \)

\textbf{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: $x / y$

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:** $A \cup B =$

**Concatenation:** $A \circ B =$

**Star:** $A^* =$
Other operations

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

Complement: $\overline{A} =$

Intersection: $A \cap B =$

Reverse: $A^R =$
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: $\overline{A} = \{ w \mid w \notin A \}$

**Theorem:** If $A$ is regular, then $\overline{A}$ 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)
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
A Nondeterministic Finite Automaton (NFA) accepts if there is a way to make it reach an accept state.
Example

\[ L(M) = \]
Example

\[ L(M) = \]

A transition diagram is shown with states labeled and transitions marked with symbols 0, 1, and \( \varepsilon \). The diagram is a sequence of states connected by arrows indicating the transition symbols.
Formal Definition of a NFA

An **NFA** is a 5-tuple \( M = (Q, \Sigma, \delta, q_0, F) \)

- \( Q \) is the set of states
- \( \Sigma \) is the alphabet
- \( \delta : Q \times \Sigma \rightarrow P(Q) \) is the transition function
- \( q_0 \in Q \) is the start state
- \( F \subseteq Q \) is the set of accept states

**\( M \) accepts** a string \( w \) if **there exists** a path from \( q_0 \) to an accept state that can be followed by reading \( w \).
Example

\[ M = (Q, \Sigma, \delta, Q_0, F) \]
\[ Q = \{ q_0, q_1, q_2, q_3, q_4 \} \]
\[ \Sigma = \{ 0, 1 \} \]
\[ F = \{ q_4 \} \]
\[ \delta(q_2, 1) = \]
\[ \delta(q_3, 1) = \]
Example

\[ N = (Q, \Sigma, \delta, q_0, F) \]

\[ Q = \{q_0, q_1, q_2, q_3\} \]

\[ \Sigma = \{0, 1\} \]

\[ F = \{q_3\} \]

\[ \delta(q_0, 0) = \]

\[ \delta(q_0, 1) = \]

\[ \delta(q_1, \varepsilon) = \]

\[ \delta(q_2, 0) = \]
Nondeterminism

Ways to think about nondeterminism
- (restricted) parallel computation
- tree of possible computations
- guessing and verifying the “right” choice

Deterministic Computation
- accept or reject

Nondeterministic Computation
- accept
- reject