Theory of Computation
COMP3630/COMP6363

Prerequisites: COMP1140 and COMP 1600 (Foundations of Computing)

Textbook: Introduction to Automata Theory, Languages and Computation
John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman [HMU].

Course assumes one knows:
- Sets, functions, relations
- Mathematical induction
- Any other background material related to Chapter 1 of HMU.
The First Half of the Course...

Covered by Badri Vellambi
badri.vellambi@anu.edu.au

Models of Computation and Languages:

- Automata and Regular Languages [1.5 weeks]

- Pushdown Automata and Context-free Languages [1.5 weeks]

- Turing Machines and Recursively Enumerable Languages [1.5 weeks]

Computational Problems:

- Decidability, Undecidability, and Intractable Problems [1.5 weeks]
This Lecture Covers Finite Automata (Chapter 2 of HMU)

- Deterministic Finite Automaton
- Nondeterministic Finite Automaton
- NFA with $\epsilon$-transitions
- An Equivalence among the above three.

Reading (from HMU): All of Chapter 2.
Preliminary Concepts

• **Alphabet** $\Sigma$: A finite set of **symbols**

  E.g., $\Sigma = \{0, 1\}$ (**binary** alphabet)
  $\Sigma = \{a, b, \ldots, z\}$ (**Roman** alphabet)

• **String** (or **word**) is a finite sequence of symbols
  - Usually represented without commas, e.g., 0011 instead of (0, 0, 1, 1)

• **Concatenation** of strings $x$ and $y$ is the string $xy = x$ followed by $y$

  $\epsilon$ is the identity element for concatenation, i.e., $\epsilon x = x \epsilon = x$.

  Concatenation of sets of strings: $AB = \{ab : a \in A, b \in B\}$

  Concatenation of the same set: $A^2 = AA; A^3 = (AA)A$, etc

• **Kleene * or closure operator**: $\Sigma^* = \{\epsilon\} \cup \Sigma \cup \Sigma^2 \cup \Sigma^3 \cdots$ denotes the **set of all strings**.

• A (**formal**) **language** is a subset of $\Sigma^*$. 
Deterministic Finite Automaton

Informally:

- The device consisting of: (a) input tape; (b) reading head; and (c) finite control (Finite-state machine)
- The input is read from left to right
- Each read operation changes the internal state of the FSM
- Input is accepted/rejected based on the final state after reading all symbols
Deterministic Finite Automaton (DFA)

- A DFA $A = (Q, \Sigma, \delta, q_0, F)$

  $Q$: A finite set (of internal states)
  
  $\Sigma$: The alphabet corresponding to the input
  
  $\delta : Q \times \Sigma \rightarrow Q$ (Transition Function)
  
  [If present state is $q \in Q$, and $a \in \Sigma$ is read, the DFA moves to $\delta(q, a)$.

  $q_0$: The (unique) starting state of the DFA (prior to any reading). ($q_0 \in Q$)

  $F \subset Q$ is the set of final (or accepting) states

  Transition Table:

  \[
  \begin{array}{c|cc}
    & 0 & 1 \\
    \hline
    q_0 & q_2 & q_0 \\
    q_1 & q_1 & q_1 \\
    q_2 & q_2 & q_1 \\
  \end{array}
  \]

  $F = \{q_1\}$

  $\delta(q_0, 0) = q_2$

  $\delta(q_0, 1) = q_0$

  Transition Diagram:

  Remark: Each state has exactly one outgoing edge labelled by a symbol
The language $L(A)$ accepted by a DFA $A = (Q, \Sigma, \delta, q_0, F)$ is:

The set of all input strings that move the state of the DFA from $q_0$ to a state in $F$

This is formalized via the **extended** transition function $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$:

Basis:  
1) $\hat{\delta}(q, \epsilon) = q$  
[No state change]

2) $\hat{\delta}(q, s) = \delta(q, s)$  
$s \in \Sigma$

Induction:  
3) if $\hat{\delta}(q, w) = p$, then $\hat{\delta}(q, ws) = \delta(p, s)$.  
$s_1 \in \Sigma$, $w \in \Sigma^*$

$L(A) :=$ all strings that take $q_0$ to some final state

$= \{w \in \Sigma^* : \hat{\delta}(q_0, w) \in F\}$.

In other words,

(a) $\epsilon \in L(A) \iff q_0 \in F$

(b) For $k > 0$,

$w = s_1s_2 \cdots s_k \in L(A) \iff q_0 \xrightarrow{s_1} P_1 \xrightarrow{s_2} P_2 \cdots \xrightarrow{s_k} P_k \in F$
An Example

Is 00 accepted by $A$?
- Need to determine $\delta(q_0, 00)$

$$
\begin{align*}
q_0 \rightarrow q_2 \rightarrow q_2 \notin F
\end{align*}
$$

Thus, 00 is not accepted by $A$

Is 001 accepted by $A$?

$$
\begin{align*}
q_0 \rightarrow q_2 \rightarrow q_2 \rightarrow q_1 &\in F
\end{align*}
$$

Thus, 001 is accepted by $A$.

- The only way one can reach $q_1$ from $q_0$ is if the string contains 01.
- $L(A)$ is the set of strings containing 01.

Remark 1: In general, each string corresponds to a unique path of states.

Remark 2: The converse isn’t true. For example, 0010 and 0011 have the same sequence of states.
Limitations of DFAs

• Can all languages be accepted by DFAs?

• DFAs have a finite number of states (and hence finite memory).

• Given a DFA, there is always a long pattern it cannot 'remember' or 'track'

  e.g., \( L = \{0^n1^n : n \in \mathbb{N}\} \) cannot be accepted by any DFA.

• Can generalize DFAs in one of many ways:

  - Allow transitions to multiple states at each symbol reading.

  - Allow transitions without reading any symbol

  - Allow the device to have an additional tape to store symbols

  - Allow the device to edit the input tape

  - Allow bidirectional head movement
Non-deterministic Finite Automaton (NFA)

- Allow transitions to multiple states at each symbol reading.

  - Multiple transitions allows the device to:
    
    (a) *clone* itself, traverse through and consider all possible parallel outcomes.
    
    (b) *hypothesize/guess* multiple eventualities concerning its input.

- Seems bizarre, but aids the implication of describing the automaton.

• Formal Definition: \( A = (Q, \Sigma, \delta, q_0, F) \)

  \[
  \delta : Q \times \Sigma \rightarrow 2^Q \quad \text{[Transition Function]}
  \]

Remark 3: \( \delta(q, s) \) can be a set with two or more states, or even be empty!

Remark 4: If \( \delta(\cdot, \cdot) \) is a singleton for all argument pairs, then NFA is a DFA.

  [So every DFA is an NFA, by definition!]
Language Accepted by an NFA

- This is formalized via the **extended** transition function \( \hat{\delta} : Q \times \Sigma^* \rightarrow 2^Q \):

  **Basis:**
  1) \( \hat{\delta}(q, \epsilon) = \{q\} \)  
  2) \( \hat{\delta}(q, s) = \delta(q, s) \)  

  **Induction:**
  3) \( \hat{\delta}(q, ws) = \bigcup_{i=1}^{k} \delta(p_i, s), \quad \hat{\delta}(q, w) = \{p_1, \ldots, p_k\} \)  

  \[ s_1 \in \Sigma, w \in \Sigma^* \]

\[ \hat{\delta}(q; ws) = \{q\} \]

\[ \hat{\delta}(q; s) = \delta(q; s) \]

\[ \hat{\delta}(q; w) = \{p_1, \ldots, p_k\} \]

- \( L(A) := \{w \in \Sigma^* : \hat{\delta}(q_0, w) \cap F \neq \emptyset\} \).

In other words,

- (a) \( \epsilon \in L(A) \iff q_0 \in F \)
- (b) For \( k > 0 \),
  
  \[ w = s_1 s_2 \cdots s_k \in L(A) \iff q_0 \rightarrow P_1 \rightarrow P_2 \rightarrow \cdots \rightarrow P_k \in F \]
An Example

- \( L(A) = \{ w : \text{penultimate symbol in } w \text{ is a 1} \} \).

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_0 )</td>
<td>( q_0 )</td>
<td>( q_0 ) ( q_1 )</td>
</tr>
<tr>
<td>( q_1 )</td>
<td>( q_2 )</td>
<td>( q_2 )</td>
</tr>
<tr>
<td>( \ast q_2 )</td>
<td>( 0 )</td>
<td>( 0 )</td>
</tr>
</tbody>
</table>

\[
\hat{\delta}(q_0, 00) = \{q_0\} \quad \hat{\delta}(q_0, 01) = \{q_0, q_1\} \quad \hat{\delta}(q_0, 10) = \{q_0, q_2\} \quad \hat{\delta}(q_0, 100) = \{q_0\}
\]

- An input can move the state from \( q_0 \) to \( q_2 \) only if it ends in 10 or 11.
- Each time the NFA reads a 1 (in state \( q_0 \)) it considers two parallel possibilities:
  
  (a) the 1 is the penultimate symbol.
  [These paths die if the 1 is not actually the penultimate symbol]
  
  (b) the 1 is not the penultimate symbol.
Non-determinism was introduced to increase the computational power.

So is there a language $L$ that is accepted by an NDA, but not by any DFA?

Theorem 1: Every Language $L$ that is accepted by an NFA is also accepted by some DFA.
Proof of Theorem 1

1) Let $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ generate the given language $L$

**Idea:** Devise a DFA $D$ such that at any time instant the state of the DFA is the set of all states that NFA $N$ can be in.

2) Define DFA $D = (Q_D, \Sigma, \delta_D, q_{D,0}, F_D)$ from $N$ using the following **subset construction**:

$$Q_D = 2^{Q_N}$$

$$q_{D,0} = \{q_0\}$$

$$F_D = \{S \subseteq Q_N : S \cap F_N \neq \emptyset\}$$

**Example:**

$$\begin{align*}
N: & 
\begin{array}{c}
0, 1 \\
\circlearrowleft \\
q_0 \\
\rightarrow \\
1 \\
q_1 \\
\rightarrow \\
0, 1 \\
\rightarrow \\
q_2 \\
\end{array} \\
D: & 
\begin{array}{c}
\emptyset \\
\{q_2\} \\
\{q_0, q_1\} \\
\{q_0, q_1, q_2\} \\
\{q_1\} \\
\{q_0, q_2\} \\
\{q_0\} \\
\{q_1, q_2\} \\
\end{array}
\end{align*}$$

3) Hence,

$$\epsilon \in L(N) \iff q_0 \in F$$

$$\iff \{q_0\} \in F_D \iff \epsilon \in L(D)$$
Proof of Theorem 1

4) To define $\delta_D(P, s)$ for each $P \subseteq Q$ and $s \in \Sigma$:

- Assume NFA $N$ is simultaneously in all states of $P$
- Let $P'$ be the states that $N$ can transition upon reading $s$
- Set $\delta_D(P, s) := P' = \bigcup_{p \in P} \delta_N(p, s)$.

5) Let $s_1 \cdots s_k \in \Sigma^k$ be given for some $k \geq 1$.

Hence, $L(N) = L(D)$
ε-Transitions

- State transitions occur without reading any symbols.

- An ε-Nondeterministic Finite Automaton is a 5-tuple \((Q, \Sigma, \delta, q_0, F)\), where:

\[
Q, \Sigma, q_0, \text{ and } F \text{ are as in an NFA}
\]

\[
\delta : Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q
\]

Example:

Without reading any input symbols, the state of the ε-NFA can transition

- From \(q_0\) to \(q_1, q_4, q_2, \) or \(q_3\).
- From \(q_2\) to \(q_3\).
- From \(q_1\) to \(q_2, \) or \(q_3\).
- From \(q_5\) to \(q_6\).
\textbf{Language accepted by an $\epsilon$-NFA}

- \textit{\$\epsilon$-closure} of a state

$\text{ECLOSE}(q) =$ all states that are reachable from $q$ by $\epsilon$-transitions alone.

\begin{align*}
\text{ECLOSE}(q_0) &= \{q_0, q_1, q_4, q_2, q_3\} \\
\text{ECLOSE}(q_1) &= \{q_1, q_2, q_3\} \\
\text{ECLOSE}(q_2) &= \{q_2, q_3\} \\
\text{ECLOSE}(q_3) &= \{q_3\} \\
\text{ECLOSE}(q_4) &= \{q_4\} \\
\text{ECLOSE}(q_5) &= \{q_5, q_6\} \\
\text{ECLOSE}(q_6) &= \{q_6\}
\end{align*}
Language accepted by an $\varepsilon$-NFA

Given $\varepsilon$-NFA $N = (Q, \Sigma, \delta, q_0, F)$

- **extended** transition function $\hat{\delta} : Q \times \Sigma^* \to 2^Q$ by

**Basis:**

1) $\hat{\delta}(q, \varepsilon) = \text{ECLOSE}(q)$

\[
q \xrightarrow{\varepsilon} q_1 \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} q' = \varepsilon = \varepsilon^2 = \varepsilon^3 = \cdots
\]

2) $\hat{\delta}(q, s) = \bigcup_{p \in \text{ECLOSE}(q)} \left( \bigcup_{p' \in \delta(p, s)} \text{ECLOSE}(p') \right)$

\[
q \xrightarrow{\varepsilon} q_1 \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} q' \xrightarrow{s} p' \xrightarrow{\varepsilon} p_1 \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} p
\]

**Induction:**

3) $\hat{\delta}(q, ws) = \bigcup_{p \in \delta(q, w)} \left( \bigcup_{p' \in \delta(p, s)} \text{ECLOSE}(p') \right)$

- $w \in L(N)$ if and only if $\hat{\delta}(q_0, w) \cap F \neq \emptyset$
Language accepted by an \( \varepsilon \)-NFA

- \( w \in L(N) \) if and only if \( \hat{\delta}(q_0, w) \cap F \neq \emptyset \)

In other words,

(a) \( \epsilon \in L(N) \iff \text{ECLOSE}(q_0) \cap F \neq \emptyset \)

\[ q_0 \xrightarrow{\epsilon} p_1 \xrightarrow{\epsilon} \ldots \xrightarrow{\epsilon} p_r \in F \]

(b) For \( k > 0 \),

\[ w = s_1 s_2 \ldots s_k \in L(A) \iff \]

\[ q_0 \xrightarrow{\epsilon} \ldots \xrightarrow{\epsilon} s_1 \xrightarrow{\epsilon} p_1 \]

\[ \ldots \]

\[ p_{k-1} \xrightarrow{\epsilon} \ldots \xrightarrow{\epsilon} s_k \xrightarrow{\epsilon} p_k \]

\[ p_k \xrightarrow{\epsilon} \ldots \xrightarrow{\epsilon} q_F \in F \]
Theorem 2: Every Language $L$ that is accepted by an $\epsilon$-NFA is also accepted by some DFA.
Theorem 2: Every Language $L$ that is accepted by an $\varepsilon$-NFA is also accepted by some DFA.

Proof: Given $L$ that is accepted by some $\varepsilon$-NFA, we must find an NFA that accepts $L$.

[NFA to DFA conversion can be done as in Theorem 1].

Let $\varepsilon$-NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ accept $L$.

Let us devise NFA $N' = (Q_{N'}, \Sigma, \delta_{N'}, q_0', F_{N'})$ as follows:

- $Q_{N'} = Q_N$. $q_0' = q_0$. $F_{N'} = \{q \in Q_N : \text{ECLOSE}(q) \cap F_N \neq \emptyset\}$

$\delta_{N'} : Q_{N'} \times \Sigma \rightarrow 2^{Q_{N'}}$ defined by:

$\delta_{N'}(q, s) = \bigcup_{p \in \text{ECLOSE}(q)} \delta(p, s)$

$N : q \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} p \xrightarrow{s} p'$

$N : q$ can transition to $p'$ after a few $\varepsilon$-transitions, and a single read of $s \in \Sigma$.

$\Downarrow$

$N' : q \xrightarrow{s} p'$

$N' : q$ can transition to $p'$ after reading $s$.
(Proof continued)

Using the definitions of the extended transition functions of NFAs and $\epsilon$-NFAs, one can formalize the following intuition.

$s_1 \ldots s_k$ is accepted by $\epsilon$-NFA $N$

\[
\begin{align*}
q_0 & \xrightarrow{\epsilon} \cdots \xrightarrow{\epsilon} s_1 \xrightarrow{\epsilon} p_1 \\
p_1 & \xrightarrow{\epsilon} \cdots \xrightarrow{\epsilon} s_2 \xrightarrow{\epsilon} p_2 \\
\vdots & \\
p_{k-1} & \xrightarrow{\epsilon} \cdots \xrightarrow{\epsilon} s_k \xrightarrow{\epsilon} p_k \\
p_k & \xrightarrow{\epsilon} \cdots \xrightarrow{\epsilon} q_F \in F
\end{align*}
\]

$s_1 \ldots s_k$ is accepted by NFA $N'$

\[
\begin{align*}
q & \xrightarrow{s_1} p_1 \\
\vdots & \\
p_k & \xrightarrow{s_k} p_k \\
\end{align*}
\]

ECLOSE($p_k$) \cap $F_N \neq \emptyset$
To Summarise...

Languages accepted by DFAs = Languages accepted by NFAs = Languages accepted by $\epsilon$-NFAs