CSCI 1010
Models of Computation

Lecture 15

FSM Languages are Regular
Overview

• Regular languages are recognized by FSMs.
• We now show that every language recognized by an FSM is regular.
• **Pumping Lemma** – a technique to show that some languages not regular.
• When is a regular language finite or infinite?
• We give algorithms for decision problems on regular languages.
The Objective

The goal is to show the following

**Theorem:** If the language $L$ is recognized by an FSM $M = (\Sigma, Q, \delta, s, F)$, then it is described by a regular expression.

**Proof Approach** Let $Q = \{q_1, \ldots, q_n\}$. For each state $q_i$, we describe the strings causing $M$ to move from $s$ to $q_i$ with a regular expression. We then form the union of the regular expressions taking $M$ from $s$ to each state in $F$. 
Proof

- $R_{i,j}^{(k)}$ is the set strings taking $M$ from $q_i$ to $q_j$ visiting only states in $Q^{(k)} = \{q_1, q_2, \ldots, q_k\}$ ($Q^{(0)} = \phi$).
- Since $Q^{(n)} = Q$, $R_{i,j}^{(n)} = \text{all strings taking } q_i \text{ to } q_j$.
- Let $r_{i,j}^{(k)}$ be an r.e. denoting strings in $R_{i,j}^{(k)}$.
- $R_{t,a}^{(n)} \cup R_{t,b}^{(n)} \cup \ldots \cup R_{t,h}^{(n)}$ is the set of strings taking $M$ from the start state $q_t = s$ to accept states $q_a, q_c, \ldots, q_h$.
- The r.e. $r_{t,a}^{(n)} + r_{t,b}^{(n)} + \ldots + r_{t,h}^{(n)}$ describes these accepted strings.
Proof (cont.)

• $R_{i,j}^{(0)} = \text{strings taking } M \text{ from } q_i \text{ to } q_j \text{ thru no other states. } a \text{ in } R_{i,j}^{(0)} \text{ if } \delta(q_i, a) = q_j. \ \varepsilon \text{ is in } R_{i,j}^{(0)} \text{ if } i = j.$

• $r_{i,j}^{(0)} = a + \ldots + z \text{ if } i \neq j \text{ and } a, \ldots, z \text{ take } M \text{ from } q_i \text{ to } q_j$

• Add $\varepsilon$ to $r_{i,j}^{(0)}$ if $i=j$. 
Proof (cont.)

- Because $R_{i,j}^{(k)}$ is set of strings taking $M$ from $q_i$ to $q_j$ passing only through $Q^{(k)} = \{q_1, q_2, ..., q_k\}$,

  $$R_{i,j}^{(k)} = R_{i,j}^{(k-1)} \cup R_{i,k}^{(k-1)} (R_{k,k}^{(k-1)})^* R_{k,j}^{(k-1)}$$

- **In words** $R_{i,j}^{(k)} = \{\text{strings taking } M \text{ from } q_i \text{ to } q_j \text{ via } Q^{(k-1)}\}$ plus $\{\text{strings taking } M \text{ from } q_i \text{ to } q_k \text{ via } Q^{(k-1)}\}$ · $\{\text{strings taking } M \text{ from } q_k \text{ to } q_k \text{ via } Q^{(k-1)}\}$ · $\{\text{strings taking } M \text{ from } q_k \text{ to } q_j \text{ via } Q^{(k-1)}\}$.

- This is a *dynamic programming algorithm*. 
Recursive Decomposition of $R_{i,j}^{(k)}$

$$R_{i,j}^{(k)} = R_{i,j}^{(k-1)} \cup R_{i,k}^{(k-1)} (R_{k,k}^{(k-1)}) * R_{k,j}^{(k-1)}$$
R.E. For FSM Language

\[ R_{i,j}^{(k)} = R_{i,j}^{(k-1)} \cup R_{i,q}^{(k-1)} (R_{q,q}^{(k-1)})^* R_{q,j}^{(k-1)} \]

- We give a recursive regular expression for \( R_{i,j}^{(k)} \).
  - Base: If \( R_{i,j}^{(0)} = \{x_1, x_2, \ldots, x_m\} \), let \( r_{i,j}^{(0)} = x_1 + x_2 + \ldots + x_m \).
  - Inductive assumption: \( r_{i,j}^{(k-1)} \) denotes \( R_{i,j}^{(k-1)} \)
    \[ r_{i,j}^{(k)} = r_{i,j}^{(k-1)} + r_{i,k}^{(k-1)} (r_{k,k}^{(k-1)})^* r_{k,j}^{(k-1)} \]
  - Hence \( r_{i,j}^{(k)} \) denotes \( R_{i,j}^{(k)} \).

- This construction produces \( r_{i,j}^{(k)} \) for \( k = 0, 1, \ldots, n \) from which the r.e. for the regular language recognized by M is generated. Q.E.D.
Example

- To construct $T^{(1)}$ use $r_{ij}^{(1)} = r_{ij}^{(0)} + r_{i1}^{(0)} (r_{11}^{(0)})^* r_{1j}^{(0)}$. Because $r_{1,1}^{(0)} = \varepsilon$ and $r_{i,1}^{(0)} = \emptyset$ for $i \geq 2$, $r_{ij}^{(1)} = r_{ij}^{(0)}$ and $T^{(1)} = T^{(0)}$. 
Example

Using $r_{i,j}^{\{2\}} = r_{i,j}^{\{1\}} + r_{i,2}^{\{1\}} (r_{2,2}^{\{1\}})^* r_{2,j}^{\{1\}}$ and $(r_{2,2}^{\{1\}})^* = \epsilon$, we have the following table.

\[
\begin{array}{ccccc}
\text{i} & \text{j} & 1 & 2 & 3 & 4 & 5 \\
1 & \epsilon & 0 & 1 & \emptyset & \emptyset & \\
2 & \emptyset & \epsilon & 0 & 1 & \emptyset & \\
3 & \emptyset & \emptyset & \epsilon + 0 + 1 & \emptyset & \emptyset & \\
4 & \emptyset & \emptyset & 1 & \epsilon & 0 & \\
5 & \emptyset & 0 & \emptyset & 1 & \epsilon & \\
\end{array}
\]

\[
\begin{array}{ccccc}
\text{i} & \text{j} & 1 & 2 & 3 & 4 & 5 \\
1 & \epsilon & 0 & 1+00 & 01 & \emptyset & \\
2 & \emptyset & \epsilon & 0 & 1 & \emptyset & \\
3 & \emptyset & \emptyset & \epsilon + 0 + 1 & \emptyset & \emptyset & \\
4 & \emptyset & \emptyset & 1 & \epsilon & 0 & \\
5 & \emptyset & 0 & 00 & 1+01 & \epsilon & \\
\end{array}
\]
Example

Using \( r_{i,j}^{(3)} = r_{i,j}^{(2)} \cup r_{i,3}^{(2)} (r_{3,3}^{(2)})^* r_{3,j}^{(2)} \) and \((r_{3,3}^{(2)})^* = (0+1)^*\), we have the following table.

\[
\begin{array}{|c|c|c|c|c|c|}
\hline
i\&j & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & \epsilon & 0 & 1+00 & 01 & \emptyset \\
2 & \emptyset & \epsilon & 0 & 1 & \emptyset \\
3 & \emptyset & \emptyset & \epsilon + 0 + 1 & \emptyset & \emptyset \\
4 & \emptyset & \emptyset & 1 & \epsilon & 0 \\
5 & \emptyset & 0 & 00 & 1+01 & \epsilon \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|c|c|c|c|}
\hline
i\&j & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & \epsilon & 0 & (1+00)(0+1)^* & 01 & \emptyset \\
2 & \emptyset & \epsilon & 0(0+1)^* & 1 & \emptyset \\
3 & \emptyset & \emptyset & (0+1)^* & \emptyset & \emptyset \\
4 & \emptyset & \emptyset & 1(0+1)^* & \epsilon & 0 \\
5 & \emptyset & 0 & 0(0+1)^* & 1+01 & \epsilon \\
\hline
\end{array}
\]
Example

Using $r_{i,j}^{(4)} = r_{i,j}^{(3)} \cup r_{i,4}^{(3)} (r_{4,4}^{(3)})^* r_{4,j}^{(3)}$ and $(r_{4,4}^{(3)})^* = \varepsilon$, we have following table.

<table>
<thead>
<tr>
<th>$i\backslash j$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\varepsilon$</td>
<td>0</td>
<td>$(1+00)(0+1)^*$</td>
<td>01</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>2</td>
<td>$\emptyset$</td>
<td>$\varepsilon$</td>
<td>0$(0+1)^*$</td>
<td>1</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>3</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$(0+1)^*$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>4</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>1$(0+1)^*$</td>
<td>$\varepsilon$</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>$\emptyset$</td>
<td>0</td>
<td>0$(0+1)^*$</td>
<td>1+01</td>
<td>$\varepsilon$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$i\backslash j$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\varepsilon$</td>
<td>0</td>
<td>$(1+00+011)(0+1)^*$</td>
<td>01</td>
<td>010</td>
</tr>
<tr>
<td>2</td>
<td>$\emptyset$</td>
<td>$\varepsilon$</td>
<td>$(0+1)(0+1)^*$</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$(0+1)^*$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>4</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>1$(0+1)^*$</td>
<td>$\varepsilon$</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>$\emptyset$</td>
<td>0</td>
<td>0$(0+11+011)(0+1)^*$</td>
<td>1+01</td>
<td>$\varepsilon$ + 10+010</td>
</tr>
</tbody>
</table>
Example

\[ \begin{array}{c|c|c|c|c|c}
\text{i} & \text{j} & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & \varepsilon & 0 & (1+00+011)(0+1)^* & 01 & 010 \\
2 & \emptyset & \varepsilon & (0+11)(0+1)^* & 1 & 10 \\
3 & \emptyset & \emptyset & (0+1)^* & \emptyset & \emptyset \\
4 & \emptyset & \emptyset & 1(0+1)^* & \varepsilon & 0 \\
5 & \emptyset & 0 & (00+11+011)(0+1)^* & 1+01 & \varepsilon + 10+010 \\
\end{array} \]

• Result \( r = r_{1,1}^{(5)} + r_{1,4}^{(5)} + r_{1,5}^{(5)} \) where

\[ r_{i,j}^{(5)} = r_{i,j}^{(4)} + r_{i,5}^{(4)} \left( r_{5,5}^{(4)} \right)^* r_{5,j}^{(4)} \]

• \( (r_{5,5}^{(5)})^* = (\varepsilon + 10 + 010)^* = (10+010)^* \). Thus, \( r_{1,1}^{(5)} = \varepsilon \). Also, \( r_{1,4}^{(5)} = 01 + (010)(10+101)^*(1+01) \). Finally, \( r_{1,5}^{(5)} = (010)(10+010)^* \).
Properties of Regular Languages

• **Definition** A class $C$ of languages is **closed** under an operation if applying the operation on a language in $C$ gives another one in $C$.

• The **complement** $\overline{L}$ of a language $L$ over an alphabet $\Sigma$ is the set of strings in $\Sigma^* - L$. 
Properties of Regular Languages

• **Theorem** The class of regular languages is closed under
  1. Concatenation
  2. Union
  3. Kleene closure
  4. Complementation
  5. Intersection

• **Proof** 1-3 follow from definition of r.e.’s. For 4, swap accepting/rejecting states. For 5 use the identity $L_1 \cap L_2 = (\overline{L_1} \cup \overline{L_2})$
Pumping Lemma

• **Theorem** Let L be a regular language recognized by an FSM M with \( m \) states. If \( w \) is a string accepted by M and \( |w| \geq m \) then there are strings \( r, s, t \in \Sigma^* \) with \( |s| > 0 \) and \( |rs| \leq m \) such that \( w = rst \) and for all \( n \geq 0 \), \( rs^n t \) is also accepted by M.

• **Remark:** If \( |w| \geq m \), \( w = rst \), then L is infinite and contains \( rt, rst, rsst, rssst, \ldots \). L contains a string pumped down (\( rt \)) and those pumped up.
Pumping Lemma

- **Proof** Let $k = |w|$. Then $w$ takes $M$ through states $q_0, q_1, q_2, ..., q_k$. Since this sequence has $k+1$ states and $k \geq m$, by the pigeon-hole principle, some state of $M$ is repeated. Let $q'$ be first repeated state. Let $r$ take $M$ from $q_0$ to $q'$ and let $s$ take $M$ from the first to second instance of $q'$. $|rs| \leq m$ because no state other than $q'$ is repeated. Let $t$ be the rest of $w$. Since $M$ accepts $w$, it also accepts $rs^nt$ for any $n \geq 0$.

- **Remark** We can pump between any repeated states.
Applications of the Pumping Lemma

**Lemma A:** $L = \{0^p1^p \mid p \geq 1\}$ is not regular.

**Proof (by contradiction)** Assume $L$ is regular and show a contradiction. If $L$ is regular, it can be recognized by an FSM $M$ with $m$ states. $0^p1^p$ is in $L$ for all $p$. Apply pumping lemma for $p \geq m$. Thus, we write $0^p1^p = rst$.

Since $rs = 0^k$, $k \leq m \leq p$, $s = 0^\alpha$, $|s| = \alpha \leq k$, and $\alpha > 0$. $rs^n t$ must also be in $L$. With $n = 0$, $L$ must contain $0^{p-\alpha}1^p$. But this contradicts the definition of the language.

We conclude that $L$ is not regular. That is, the machine recognizing $L$ cannot have a finite number of states.

(What kind of machine will recognize this language?)
Another Application

**Lemma B:** $L = \{a^p \mid p \text{ is prime}\}$ is not regular.

**Proof** Suppose it is regular. Then, for some prime $p$, $a^p = rst$ where $|s| \geq 1$. By pumping lemma, $rs^nt$ is in $L$ for all $n \geq 0$. Here $r = a^x$, $s = a^y$, and $t = a^z$ for integers $x$, $y$, $z$ with $y \geq 1$. If $a^{x+ny+z}$ is in $L$, then $x+ny+z$ is prime for each $n \geq 1$. However if we set $n = x+2y+z+2$, then,

$$x+ny+z = x+(x+2y+z+2)y+z \quad = x+xy+2y^2+zy+2y+z$$

or

$$x+ny+z = (y+1)(x+2y+z)$$

Since $(y+1)(x+2y+z)$ is not a prime, $L$ is not regular.
More on Regular Languages

- **Lemma C1**: Let $L$ be recognized by an $m$-state DFSM. $L$ is non-empty if and only if $L$ contains a string of length less than $m$.

- **Proof** Let $w$ be a shortest string in $L$. If $|w| < m$, done. If $|w| \geq m$, we can use pumping lemma to shorten it, which means it wasn’t the shortest string and $L$ is empty.
More on Regular Languages

• **Lemma C2**: Let $L$ be recognized by an $m$-state DFSM. $L$ is infinite if and only if it contains a string $w$ of length at least $m$ and at most $2m-1$.

• **Proof** If $L$ contains a string of length between $m$ and $2m-1$, it can pumped up to create an infinite set of strings. If $L$ is infinite, consider a shortest string $w$ of length $m$ or more. If its length is at most $2m-1$, done. If not, by pumping lemma, it can be shortened by removal of a string of length between 1 and $m$. This can be repeated until string has length between $m$ and $2m-1$. 
Application of Properties

- **Lemma D:** The language $L = \{w \mid w$ has an equal number of 0’s and 1’s$\}$ is not regular.

- **Proof** We use the fact that the intersection of two regular languages is regular. $L$ contains strings such as $01$, $1010$, $1100$, $000111$. The language $0^*1^*$ is clearly regular. If $L$ is regular, then so is $L \cap 0^*1^*$.

If $L \cap 0^*1^*$ is not regular, $L$ could not be regular because the intersection of two regular languages is regular.

But the strings in $L \cap 0^*1^*$ are of the form $0^n1^n$. But in Lemma A we have shown that $L \cap 0^*1^* = \{0^n1^n\}$ is not regular using the pumping lemma.
Decision Problems on Reg. Langs.

- **Theorem**: There are algorithms for each of the following decision problems:
  
  (a) For FSM $M$ and string $w$, is $w$ in $L(M)$?
  
  (b) For FSM $M$, is $L(M) = \emptyset$?
  
  (c) For FSM $M$, is $L(M) = \Sigma^*$?

- **Proof** (a) Decide in $|w|$ steps by giving to $M$. It is in $L(M)$ if $M$ accepts.

- (b) From Lemma C1, $L(M) = \emptyset$ only if $M$ does not accept any strings of length less than $m$, the number of states of $M$. There are at most $|\Sigma|^m-1$ strings to test.

- (c) Same as asking if $L(M) = \emptyset$. We answer it by constructing the complement to $M$ and applying Lemma C1 to it.
Decision Problems on Reg. Langs.

- **Theorem:** There are algorithms for each of the following decision problems:
  - (d) For FSM’s $M_1$ and $M_2$, is $L(M_1) \subseteq L(M_2)$?
  - (e) For FSM’s $M_1$ and $M_2$, is $L(M_1) = L(M_2)$?

- **Proof (d)** This question is the same as $L(M_1) \cap L(M_2) = \emptyset$ because the only way that $L(M_1)$ is contained in $L(M_2)$ is if $L(M_1)$ is not contained in the complement of $L(M_2)$. (See Figure.)

- (e) If $L(M_1) = L(M_2)$, then $L(M_1) \subseteq L(M_2)$ and $L(M_2) \subseteq L(M_1)$. Apply (d) to each subproblem.
Overview

• Regular languages are recognized by FSMs.
• We’ve shown that languages recognized by FSMs are regular.
• Pumping Lemma used to show that some languages not regular.
  – Can the lemma be generalized to show that a substring of length $\leq m$ anywhere in an accepted string can be pumped?
• Conditions for a regular language to be finite or infinite.
• We gave algorithms for decision problems on regular languages.