Regular Expressions

Reading: Chapter 3
Operations on Languages

• Let L, L_1, L_2 be subsets of Σ^*

• Concatenation: L_1L_2 = {xy | x is in L_1 and y is in L_2}

• Concatenating a language with itself: L^0 = {ε}
  \[ L^i = LL^{i-1}, \text{ for all } i \geq 1 \]

• Kleene Closure: L^* = \bigcup_{i=0}^{\infty} L^i = L^0 U L^1 U L^2 U…

• Positive Closure: L^+ = \bigcup_{i=1}^{\infty} L^i = L^1 U L^2 U…

• Question: Does L^+ contain ε?
Regular Expressions

A regular expression is:
• used to specify a language
• a finite length string of symbols
• very precisely, intuitive, and useful in a lot of contexts
• easy to convert to an NFA-ε, algorithmically; and consequently to an NFA, a DFA, and a corresponding program
Definition of a Regular Expression

• If \( r \) is a regular expression, then \( L(r) \) is used to denote the corresponding language.

• \( \Sigma \) be an alphabet. The regular expressions over \( \Sigma \) are:

  – \( \emptyset \) Represents the empty set \( \{ \} \)
  – \( \varepsilon \) Represents the set \( \{\varepsilon\} \)
  – \( a \) Represents the set \( \{a\} \), for any symbol \( a \) in \( \Sigma \)

Let \( r \) and \( s \) be regular expressions.

  – \( r+s \) Represents the set \( L(r) \cup L(s) \)
  – \( rs \) Represents the set \( L(r)L(s) \)
  – \( r^* \) Represents the set \( L(r)^* \)
  – \( (r) \) Represents the set \( L(r) \)

• Note that the operators are listed in increasing precedence.
• **Examples:** Let $\Sigma = \{0, 1\}$

- $(0 + 1)^*$ All strings of 0’s and 1’s
- $0(0 + 1)^*$ All strings of 0’s and 1’s, beginning with a 0
- $(0 + 1)^*1$ All strings of 0’s and 1’s, ending with a 1
- $(0 + 1)^*0(0 + 1)^*$ All strings of 0’s and 1’s containing at least one 0
- $(0 + 1)^*0(0 + 1)^*0(0 + 1)^*$ All strings of 0’s and 1’s containing at least two 0’s
- $(0 + 1)^*01*01^*$ All strings of 0’s and 1’s containing at least two 0’s
- $1^*(01*01^*)^*$ All strings of 0’s and 1’s containing an even number of 0’s
- $(1*01*0)^*1^*$ All strings of 0’s and 1’s containing an even number of 0’s
- $(1 + 01*0)^*$ All strings of 0’s and 1’s containing an even number of 0’s

• **Question:** Is there a unique minimum regular expression for a given language?

• **How do the above regular expressions “parse” based on the formal definition?**
• Other examples:

\[ 010 + 1100 + \varepsilon \]

\[ 010 + 1100 + \emptyset \]

\[ \varepsilon 0(0 + 1)^* \]

\[ (0 + 1 + \varepsilon)^*1 \]

\[ \emptyset (0 + 1)^* \]

\[ (\emptyset + 1)^* \]

\[ \emptyset (0 + \varepsilon)^* + \varepsilon^* \emptyset^* \]
• An almost completely useless program…

• Generating a Random String for a Regular Expression (Example):

\[1^*(01*01^*)^*\]

```java
// generate something from 1*
int n = random(0, inf);
for (int i=0; i<=n-1; i++) {
    print('1');
}

// generate something from (01*01^*)*
int m = random(0, inf);
for (int i=0; i<=m-1; i++) {
    // generate a single 0
    print('0');
    // generate something from 1*
    int k = random(0, inf);
    for (int i=0; i<=k-1; i++) {
        print('1');
    }
    // generate a single 0
    print('0');
    // generate something from 1*
    int k = random(0, inf);
    for (int i=0; i<=k-1; i++) {
        print('1');
    }
}
```
• Algebraic Laws for Regular Expressions:

1. \( u + v = v + u \)  
   commutativity
2. \((u + v) + w = u + (v + w)\)  
   associativity
3. \((uv)w = u(vw)\)  
   associativity
4. \( \emptyset + u = u + \emptyset = u \)  
   identity
5. \( \varepsilon u = u \varepsilon = u \)  
   identity
6. \( \emptyset u = u \emptyset = \emptyset \)  
   annihilator
7. \( u(v+w) = uv+uw \)  
   distributive
8. \((u+v)w = uw+vw\)  
   distributive
9. \( u + u = u \)  
   idempotent
10. \((u^*)^* = u^*\)  

11. \( \emptyset^* = \varepsilon \)  
    \[ L^* = \bigcup_{i=0}^{\infty} L^i = L^0 \cup L^1 \cup L^2 \cup \ldots \]  
12. \( \varepsilon^* = \varepsilon \)

• Such laws can be used to prove equivalences between regular expressions:

\[ \varepsilon + 1^* = 1^* \]
\[ 0 + 01^* = (\varepsilon + 1^*)0 \]
\[ (0 + 1)^* = (0^* + 10^*)^* \]
Other Laws:

1. 
   
   \[(uv)^*u = u(vu)^*\]

2. 
   
   \[(u+v)^* = (u^*+v)^*\]
   
   \[= u^*(u+v)^*\]
   
   \[= (u+vu^*)^*\]
   
   \[= (u^*v^*)^*\]
   
   \[= u^*(vu^*)^*\]
   
   \[= (u^*v^*)u^*\]

3. 
   
   \[L^+ = LL^* = L^*L\]

4. 
   
   \[L^* = L^+ + \epsilon\]
Equivalence of Regular Expressions and NFA-εs

• Note:
Throughout the following, keep in mind the definition of string acceptance for an NFA-ε…what is it?

Lemma 1: Let r be a regular expression. Then there exists an NFA-ε M such that $L(M) = L(r)$. Furthermore, M has exactly one final state with no transitions out of it.

Proof: (by induction on the number of operators, denoted by OP(r), in r).
**Basis:** $OP(r) = 0$

Then $r$ is either $\emptyset$, $\varepsilon$, or $a$, for some symbol $a$ in $\Sigma$

For $\emptyset$:

```
    q_0
    \rightarrow
```

For $\varepsilon$:

```
    q_f
    \rightarrow
```

For $a$:

```
    q_0 \quad a \quad q_f
    \rightarrow \quad \rightarrow
```

**Inductive Hypothesis:** Suppose there exists a $k \geq 0$ such that for any regular expression $r$ where $0 \leq \text{OP}(r) \leq k$, there exists an NFA-$\varepsilon$ such that $L(M) = L(r)$. Furthermore, suppose $M$ has exactly one final state with no transitions out of it.

**Inductive Step:** Let $r$ be a regular expression with $k + 1$ operators ($\text{OP}(r) = k + 1$). Since $k \geq 0$, it follows that $k + 1 \geq 1$, and therefore $r$ has at least one operator.

Case 1) $r = r_1 + r_2$

Since $\text{OP}(r) = k + 1$, it follows that $0 \leq \text{OP}(r_1) \leq k$ and $0 \leq \text{OP}(r_2) \leq k$. By the inductive hypothesis there exist NFA-$\varepsilon$ machines $M_1$ and $M_2$ such that $L(M_1) = L(r_1)$ and $L(M_2) = L(r_2)$. Furthermore, both $M_1$ and $M_2$ have one final state.

Construct $M$ as:
Case 2) $r = r_1 r_2$

Since $\text{OP}(r) = k+1$, it follows that $0 \leq \text{OP}(r_1) \leq k$ and $0 \leq \text{OP}(r_2) \leq k$. By the inductive hypothesis there exist NFA-$\epsilon$ machines $M_1$ and $M_2$ such that $L(M_1) = L(r_1)$ and $L(M_2) = L(r_2)$. Furthermore, both $M_1$ and $M_2$ have exactly one final state.

Construct $M$ as:

![Diagram 1]

Case 3) $r = r_1^*$

Since $\text{OP}(r) = k+1$, it follows that $0 \leq \text{OP}(r_1) \leq k$. By the inductive hypothesis there exists an NFA-$\epsilon$ machine $M_1$ such that $L(M_1) = L(r_1)$. Furthermore, $M_1$ has exactly one final state.

Construct $M$ as:

![Diagram 2]
• Note that the previous proof is “constructive” in that it shows us how to construct the NFA-$\varepsilon$ from the regular expression.

• Given a regular expression, first decompose it based on the recursive definition:

\[
r = 0(0+1)^* \\
r = r_1 r_2 \\
r_1 = 0 \\
r_2 = (0+1)^* \\
r_2 = r_3^* \\
r_3 = 0+1 \\
r_3 = r_4 + r_5 \\
r_4 = 0 \\
r_5 = 1
\]
\[ r = 0(0+1)^* \]

\[ r = r_1 r_2 \]

\[ r_1 = 0 \]

\[ r_2 = (0+1)^* \]

\[ r_2 = r_3^* \]

\[ r_3 = 0+1 \]

\[ r_3 = r_4 + r_5 \]

\[ r_4 = 0 \]

\[ r_5 = 1 \]
\[ r = 0(0+1)^* \]

\[ r = r_1 r_2 \]

\[ r_1 = 0 \]

\[ r_2 = (0+1)^* \]

\[ r_2 = r_3^* \]

\[ r_3 = 0+1 \]

\[ r_3 = r_4 + r_5 \]

\[ r_4 = 0 \]

\[ r_5 = 1 \]
\[ r = 0(0+1)^* \]

\[ r = r_1 r_2 \]

\[ r_1 = 0 \]

\[ r_2 = (0+1)^* \]

\[ r_2 = r_3^* \]

\[ r_3 = 0+1 \]

\[ r_3 = r_4 + r_5 \]

\[ r_4 = 0 \]

\[ r_5 = 1 \]
\[ r = 0(0+1)^* \]
\[ r = r_1r_2 \]
\[ r_1 = 0 \]
\[ r_2 = (0+1)^* \]
\[ r_2 = r_3^* \]
\[ r_3 = 0+1 \]
\[ r_3 = r_4 + r_5 \]
\[ r_4 = 0 \]
\[ r_5 = 1 \]
\[ r = 0(0+1)^* \]

\[ r = r_1 r_2 \]

\[ r_1 = 0 \]

\[ r_2 = (0+1)^* \]

\[ r_2 = r_3^* \]

\[ r_3 = 0+1 \]

\[ r_3 = r_4 + r_5 \]

\[ r_4 = 0 \]

\[ r_5 = 1 \]
\( r = 0(0+1)^* \)

\( \mathbf{r} = \mathbf{r}_1 \mathbf{r}_2 \)

\( \mathbf{r}_1 = 0 \)

\( \mathbf{r}_2 = (0+1)^* \)

\( \mathbf{r}_2 = \mathbf{r}_3^* \)

\( \mathbf{r}_3 = 0+1 \)

\( \mathbf{r}_3 = \mathbf{r}_4 + \mathbf{r}_5 \)

\( \mathbf{r}_4 = 0 \)

\( \mathbf{r}_5 = 1 \)
Definitions Required to Convert a DFA to a Regular Expression

- Let \( M = (Q, \Sigma, \delta, q_1, F) \) be a DFA with state set \( Q = \{q_1, q_2, \ldots, q_n\} \), and define:

  \[ R_{i,j} = \{ x \mid x \text{ is in } \Sigma^* \text{ and } \delta(q_i, x) = q_j \} \]

  \( R_{i,j} \) is the set of all strings that define a path in \( M \) from \( q_i \) to \( q_j \).

- Note that states have been numbered starting at 1!

- This has been done simply for convenience, and it is “without loss of generality.”
• Example:

![Diagram with states and transitions]

\[ R_{2,3} = \{0, 001, 00101, 011, \ldots\} \]
\[ R_{1,4} = \{01, 00101, \ldots\} \]
\[ R_{3,3} = \{11, 100, \ldots\} \]
Another definition:

\[ R^k_{i,j} = \{ x \mid x \text{ is in } \Sigma^* \text{ and } \delta(q_i, x) = q_j, \text{ and for no } u \text{ where } 1 \leq |u| < |x| \text{ and } x = uv \text{ is it the case that } \delta(q_i, u) = q_p \text{ where } p > k \} \]

for any i, j, k, where 1 \leq i, j \leq n and 0 \leq k \leq n

In other words, \( R^k_{i,j} \) is the set of all strings that define a path in \( M \) from \( q_i \) to \( q_j \) but that pass through no state numbered greater than k.

Here, the phrase \textit{pass through a state }q\textit{ means that the machine enters the state }q\textit{ at some point, and then (subsequently) leaves that state }q\textit{.}

Consequently it may be the case that i > k or j > k for \( R^k_{i,j} \).
Example:

\[ R_{2,3}^4 = \{0, 1000, 011, \ldots\} \]
111 is not in \( R_{2,3}^4 \)

\[ R_{2,3}^1 = \{0\} \]
111 is not in \( R_{2,3}^1 \)
101 is not in \( R_{2,3}^1 \)

\[ R_{2,3}^2 = \{\} \]

\[ R_{2,3}^5 = R_{2,3} \]
• Observations:

1) $R_{i,j}^n = R_{i,j}$

   -- More generally, $R_{i,j}^k = R_{i,j}$ for any $k \geq n$.

2) $R_{i,j}^{k-1}$ is a subset of $R_{i,j}^k$

3) $L(M) = \bigcup_{q \in F} R_{1,q}^n$

4) $R_{i,j}^0 = \begin{cases} \{a | \delta(q_i, a) = q_j, i \neq j\} & i \neq j \\ \{a | \delta(q_i, a) = q_j\} \cup \{\varepsilon\} & i = j \end{cases}$

   -- Easily computed from the DFA!

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

   For $k \geq 1$
• Explanation of 5:

5) $R^k_{i,j} = R^{k-1}_{i,k} (R^{k-1}_{k,k})^* R^{k-1}_{k,j} U R^{k-1}_{i,j}$

• Consider paths represented by the strings in $R^k_{i,j}$:

- If $x$ is a string in $R^k_{i,j}$ then no state numbered > $k$ is passed through when processing $x$.

- Any state numbered $\leq k$, on the other hand, may or may not appear on the path while processing $x$; this includes, in particular, state $q_k$.

• So there are two cases:
  - $q_k$ is not passed through, i.e., $x$ is in $R^{k-1}_{i,j}$
  - $q_k$ is passed through one or more times, i.e., $x$ is in $R^{k-1}_{i,k} (R^{k-1}_{k,k})^* R^{k-1}_{k,j}$
• **Lemma 2:** Let $M = (Q, \Sigma, \delta, q_1, F)$ be a DFA. Then there exists a regular expression $r$ such that $L(M) = L(r)$.

• **Proof:**
  First we will show (by induction on $k$) that for all $i,j$, and $k$, where $1 \leq i,j \leq n$ and $0 \leq k \leq n$, there exists a regular expression $r$ such that $L(r) = R_{i,j}^k$.

  Throughout the following, the regular expression representing $R_{i,j}^k$ will be denoted by $r_{i,j}^k$. 
Basis: $k=0$

$R^0_{i,j}$ contains single symbols, one for each transition from $q_i$ to $q_j$, and possibly $\varepsilon$ if $i=j$.

1) No transitions from $q_i$ to $q_j$ and $i \neq j$

$$r^0_{i,j} = \emptyset$$

2) At least one ($m \geq 1$) transition from $q_i$ to $q_j$ and $i \neq j$

$$r^0_{i,j} = a_1 + a_2 + a_3 + \ldots + a_m \quad \text{where} \quad \delta(q_i, a_p) = q_j, \quad \text{for all} \ 1 \leq p \leq m$$

3) No transitions from $q_i$ to $q_j$ and $i = j$

$$r^0_{i,j} = \varepsilon$$

4) At least one ($m \geq 1$) transition from $q_i$ to $q_j$ and $i = j$

$$r^0_{i,j} = a_1 + a_2 + a_3 + \ldots + a_m + \varepsilon \quad \text{where} \quad \delta(q_i, a_p) = q_j, \quad \text{for all} \ 1 \leq p \leq m$$
**Inductive Hypothesis:**
Suppose there exists a $k \geq 1$ such that $R^k_{i,j}$ can be represented by a regular expression, for all $1 \leq i,j \leq n$. Let that regular expression be denoted by $r^{k-1}_{i,j}$.

**Inductive Step:**
Consider $R^k_{i,j} = R^{k-1}_{i,k} (R^{k-1}_{k,k})^* R^{k-1}_{k,j} U R^{k-1}_{i,j}$.

By the inductive hypothesis $R^{k-1}_{i,k}$ can be represented by a regular expression, denoted $r^{k-1}_{i,k}$.

Similarly, $R^{k-1}_{k,k}$, $R^{k-1}_{k,j}$, and $R^{k-1}_{i,j}$ can all be represented by regular expressions, denoted $r^{k-1}_{k,k}$, $r^{k-1}_{k,j}$, and $r^{k-1}_{i,j}$, respectively.

Thus, if we let

$$r^k_{i,j} = r^{k-1}_{i,k} (r^{k-1}_{k,k})^* r^{k-1}_{k,j} + r^{k-1}_{i,j}$$

then $r^k_{i,j}$ is a regular expression generating $R^k_{i,j}$, i.e., $L(r^k_{i,j}) = R^k_{i,j}$. 
• Finally, if $F = \{q_{j_1}, q_{j_2}, \ldots, q_{j_r}\}$, then

$$r_{1j_1}^n + r_{1j_2}^n + \ldots + r_{1j_r}^n$$

is a regular expression generating $L(M)$.

• Not only does this prove that the regular expressions generate the regular languages, but it also provides an algorithm for computing it!
• Example:

First table column is computed from the DFA.

<table>
<thead>
<tr>
<th></th>
<th>k = 0</th>
<th>k = 1</th>
<th>k = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_{1,1}^k$</td>
<td>$\varepsilon$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r_{1,2}^k$</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r_{1,3}^k$</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r_{2,1}^k$</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r_{2,2}^k$</td>
<td>$\varepsilon$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r_{2,3}^k$</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r_{3,1}^k$</td>
<td>$\emptyset$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r_{3,2}^k$</td>
<td>0 + 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r_{3,3}^k$</td>
<td>$\varepsilon$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
All remaining columns are computed from the previous column using the formula.

\[ r^1_{2,3} = r^0_{2,1} (r^0_{1,1}) \cdot r^0_{1,3} + r^0_{2,3} \]
\[ = 0 (\varepsilon) \cdot 1 + 1 \]
\[ = 01 + 1 \]

<table>
<thead>
<tr>
<th>k = 0</th>
<th>k = 1</th>
<th>k = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r^k_{1,1} )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
</tr>
<tr>
<td>( r^k_{1,2} )</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( r^k_{1,3} )</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( r^k_{2,1} )</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( r^k_{2,2} )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon + 00 )</td>
</tr>
<tr>
<td>( r^k_{2,3} )</td>
<td>1</td>
<td>01 + 1</td>
</tr>
<tr>
<td>( r^k_{3,1} )</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>( r^k_{3,2} )</td>
<td>( 0 + 1 )</td>
<td>( 0 + 1 )</td>
</tr>
<tr>
<td>( r^k_{3,3} )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
</tr>
</tbody>
</table>
\[ r_{1,3}^2 = r_{1,2}^1 (r_{2,2}^1)^* r_{1,3}^1 + r_{1,3}^1 \]
\[ = 0 (\varepsilon + 00)^* (1 + 01) + 1 \]
\[ = 0^*1 \]

<table>
<thead>
<tr>
<th></th>
<th>k = 0</th>
<th>k = 1</th>
<th>k = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r_{1,1}^k )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
<td>( (00)^* )</td>
</tr>
<tr>
<td>( r_{1,2}^k )</td>
<td>0</td>
<td>0</td>
<td>0(00)*</td>
</tr>
<tr>
<td>( r_{1,3}^k )</td>
<td>1</td>
<td>1</td>
<td>0*1</td>
</tr>
<tr>
<td>( r_{2,1}^k )</td>
<td>0</td>
<td>0</td>
<td>0(00)^*</td>
</tr>
<tr>
<td>( r_{2,2}^k )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon + 00 )</td>
<td>( (00)^* )</td>
</tr>
<tr>
<td>( r_{2,3}^k )</td>
<td>1</td>
<td>1 + 01</td>
<td>0*1</td>
</tr>
<tr>
<td>( r_{3,1}^k )</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
<td>( (0 + 1)(00)^*0 )</td>
</tr>
<tr>
<td>( r_{3,2}^k )</td>
<td>( 0 + 1 )</td>
<td>( 0 + 1 )</td>
<td>( (0 + 1)(00)^* )</td>
</tr>
<tr>
<td>( r_{3,3}^k )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon )</td>
<td>( \varepsilon + (0 + 1)0^*1 )</td>
</tr>
</tbody>
</table>
To complete the regular expression, we compute:

$$r^3_{1,2} + r^3_{1,3}$$

<table>
<thead>
<tr>
<th></th>
<th>k = 0</th>
<th>k = 1</th>
<th>k = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r^k_{1,1}$</td>
<td>$\varepsilon$</td>
<td>$\varepsilon$</td>
<td>(00)*</td>
</tr>
<tr>
<td>$r^k_{1,2}$</td>
<td>0</td>
<td>0</td>
<td>0(00)*</td>
</tr>
<tr>
<td>$r^k_{1,3}$</td>
<td>1</td>
<td>1</td>
<td>0*1</td>
</tr>
<tr>
<td>$r^k_{2,1}$</td>
<td>0</td>
<td>0</td>
<td>0(00)*</td>
</tr>
<tr>
<td>$r^k_{2,2}$</td>
<td>$\varepsilon$</td>
<td>$\varepsilon + 00$</td>
<td>(00)*</td>
</tr>
<tr>
<td>$r^k_{2,3}$</td>
<td>1</td>
<td>1 + 01</td>
<td>0*1</td>
</tr>
<tr>
<td>$r^k_{3,1}$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>(0 + 1)(00)*0</td>
</tr>
<tr>
<td>$r^k_{3,2}$</td>
<td>$0 + 1$</td>
<td>$0 + 1$</td>
<td>(0 + 1)(00)*</td>
</tr>
<tr>
<td>$r^k_{3,3}$</td>
<td>$\varepsilon$</td>
<td>$\varepsilon$</td>
<td>$\varepsilon + (0 + 1)0*1$</td>
</tr>
</tbody>
</table>
• **Theorem:** Let $L$ be a language. Then there exists an a regular expression $r$ such that $L = L(r)$ if and only if there exits a DFA $M$ such that $L = L(M)$.

• **Proof:**

(if) Suppose there exists a DFA $M$ such that $L = L(M)$. Then by Lemma 2 there exists a regular expression $r$ such that $L = L(r)$.

(only if) Suppose there exists a regular expression $r$ such that $L = L(r)$. Then by Lemma 1 there exists a DFA $M$ such that $L = L(M)$.

• **Corollary:** The regular expressions define the regular languages.

• **Note:** With the completion of Lemma 1, the conversion from a regular expression to a DFA and a program accepting $L(r)$ is now complete, and fully automated!