Week 3, Spring 2009

recognizes

$(a \cup b)^* (aa \cup bb)(a \cup b)^*$

$(a \cup b)(a \cup b) b (a \cup b)^*$
even no of a's & even no of b's

\[(aa \cup bb \cup (ab \cup ba)(aa \cup bb)^*abub\}^*)\]

One of our goals is to prove that

FA recognize exactly the languages

that regular expressions denote.

Consideration of how to do that leads
to the following:
**Theorem** 1.25, Page 45

**Definition.** A language is regular if and only if it can be recognized by a finite automaton.

**Theorem.** If $A_1$ and $A_2$ are regular languages, so is $A_1 \cup A_2$. (We assume $A_1$ and $A_2$ are languages over the same alphabet $\Sigma$).

**Proof.** Let $M_1$ recognize $A_1$ and $M_2$ recognize $A_2$.

$$M_1 = (Q_1, \Sigma, q_1, F_1, \delta_1)$$

and

$$M_2 = (Q_2, \Sigma, q_2, F_2, \delta_2).$$

We construct $M = (Q, \Sigma, q, F, \delta)$ to recognize $A_1 \cup A_2$:

1. $Q = \{ (r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2 \}$.
2. $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$.
3. $q = (q_1, q_2)$.
4. $F = \{ (r_1, r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2 \}$. 
Example:

M_1:

M_2:

M:

Note: We don't need the state \([x_2, y_2]\) because it is not reachable from the start state.

\[L(M_1) = (b^* a b)^* a a (a u b)^*\]

\[L(M_2) = a^* b (b^* u a a (a u b))^*\]

\[L(M) = L(M_1) u L(M_2) : r_1 u r_2.\]
Recall algorithm for union. Observe that regular languages are closed under intersection and complements.

Let \( M = (Q, \Sigma, \delta, q_0, F) \) be a finite automaton. Define \( \delta : Q \times \Sigma^* \to Q \) such that \( \delta(q, w) \) is the current state of \( M \) after reading \( w \) starting in state \( q \).

1. \( \delta(q, \lambda) = q \)
2. \( \delta(q, wa) = \delta(\delta(q, w), a) \).

Letting \( w = a \) in step 2,
\[
\delta(q, a) = \delta(\delta(q, \lambda), a) = \varepsilon(q, a).
\]
So, \( \delta \) and \( \varepsilon \) are identical on symbols of the alphabet. We will write \( \delta \) instead of \( \varepsilon \).

Note \( M \) accepts \( w \) if \( \delta(q_0, w) \in F \).
We also want to show that regular languages are closed under concatenation and star. So, we introduce a new concept.

Undecidability

The finite automaton we studied is deterministic. Given a state and symbol scanned, the next state is uniquely determined.

DFA

In a nondeterministic finite automaton NFA the next state is not uniquely determined. In state q, reading symbol a, the NFA may move into any one of several possible states. The machine may move from one state to another without reading an input word. The machine in state q, reading a, might even have no move.

\[(01)^* (11 u 101) (01)^*\]
HW 1.7 and 1.11

\[(0^3)^* \cup (0^2)^*\]
For any set \( A \), \( P(A) \) denotes \( \{ s \mid s \subseteq A \} \).

A non-deterministic finite automaton (NFA) is a 5-tuple
\[ N = (Q, \Sigma, \delta, q_0, F) \]

1. \( Q \) is a finite set of states.
2. \( \Sigma \) is a finite alphabet.
3. \( \delta : Q \times (\Sigma \cup \{\lambda\}) \to P(Q) \) is the transition function.
4. \( q_0 \in Q \) is the start state.
5. \( F \subseteq Q \) is the set of accept states.

\[ Q = \{ q_1, q_2, q_3, q_4 \} \quad \Sigma = \{0, 1, 3\} \]

\( q_1 \) is the start state; \( F = \{ q_4 \} \)

\[
\begin{array}{c|ccc}
   & \emptyset & 1 & \lambda \\
q_1 & [q_1, q_2, q_3] & \emptyset & \emptyset \\
q_2 & [q_3] & \emptyset & [q_3] \\
q_3 & \emptyset & [q_4] & \emptyset \\
q_4 & [q_1, q_3] & \emptyset & \emptyset
\end{array}
\]
\[ a, \varepsilon \in \Sigma \{ \lambda \} \]

**N accepts** \( w = a_1 \ldots a_n \), if there is a sequence of states \( r_0, r_1, \ldots, r_n \) in \( Q \) such that

1. \( r_0 = q_0 \)
2. \( r_{i+1} \in \delta(r_i, a_{i+1}) \), for \( i = 0, \ldots, n-1 \),

and
3. \( r_n \in F \)

**Eg.** \( w = 111 \) is accepted by the NFA we just looked at, but we must represent \( w \) as \( w = 1 \lambda 1 \lambda 1 \) or \( w = 11 \lambda 1 \).

\[ \begin{align*}
q_2 & \in \delta(q_1, 1) \\
q_3 & \in \delta(q_2, 6) \\
q_4 & \in \delta(q_3, 1) \\
q_5 & \in \delta(q_5, 1) \cap F, \\
\end{align*} \]

So, \( N \) accepts \( \lambda 1 \lambda 1 \).

\( N \) accepts, \( L = \{ w \mid N \) accepts \( w \} \).

(*Test uses "recognizes."*)
Let \( N = (Q, \Sigma, \delta, q_0, F) \) be an NFA that has no \( \lambda \)-moves.
Define \( \hat{\delta}: Q \times \Sigma^* \to P(Q) \)
so that \( \hat{\delta}(q, w) \) is the set of possible current states of \( N \) after scanning \( w \) starting in state \( q \).

\[
\hat{\delta}(q, \lambda) = \{ q \} \quad \text{ (No \( \lambda \)-moves, disallows change)}
\]
\[
\hat{\delta}(q, aw) = \{ p \} \quad \text{ for some state } r \in \hat{\delta}(q, w), p \in \delta(r, a)
\]
\[
= \bigcup_{r \in \hat{\delta}(q, w)} \delta(r, a)
\]

As before, \( \hat{\delta}(q, a) = \{ p \} \) for some \( r \in \hat{\delta}(q, a) = \{ q \}, p \in \delta(r, a) \)
\[
= \hat{\delta}(q, a)
\]

Then, \( N \) accepts \( w \Leftrightarrow \hat{\delta}(q_0, w) \cap F \neq \emptyset \).
For any set of states \( P \subseteq Q \), we can define,
\[
\hat{\delta}(P, w) = \bigcup_{g \in P} \hat{\delta}(g, w)
\]
Two machines are equivalent if they accept the same language.

**Theorem** Every NFA has an equivalent DFA

**Proof** Let \( M = (Q, \Sigma, \delta, q_0, F) \) be an NFA. We construct a DFA that accepts the same language as \( M \). First, suppose \( M \) has no \( \lambda \)-moves.

\[ M' = (Q', \Sigma, \delta', q'_0, F') \]

1. \( Q' = \mathcal{P}(Q) \).
2. For \( R \subseteq Q' \) and \( a \in \Sigma \),

\[ \delta'(R, a) = \{ q \mid \exists r \in R \text{ such that } q \in \delta(r, a) \} = \bigcup_{r \in R} \delta(r, a) = \delta(R, a) \]

3. \( q'_0 = \{ q_0 \} \)
4. \( F' = \{ R \subseteq Q' \mid R \text{ contains an accept state of } M \} \)