CSE 322, Fall 2010

Nondeterministic
Finite State Machines
Defn: For any $X, Y \subseteq \Sigma^*$, define

$$X \cdot Y = \{ xy \mid x \in X \& y \in Y \}$$

Ex:

$X = \{ a, ab \}$

$Y = \{ \varepsilon, b, bb \}$

$X \cdot Y = \{ a, ab, abb, abbb \}$

$Y \cdot X = \{ a, ab, ba, bab, bba, bbab \}$

note $|X \cdot Y| \leq |X| \cdot |Y|$
Powers

\[ L^2 = L \cdot L \]
\[ L^3 = L \cdot L \cdot L \]
\[ \vdots \]
\[ L^n = L \]
\[ L = 3 \cdot 3 \]

\[ \text{for } n \geq 0 \]
\[ L^n = \begin{cases} 
    L \cdot L^{n-1} & \text{if } n > 1 \\
    3 & \text{if } n = 0 
\end{cases} \]

\[ \sum_{g \in S} \sigma^2 = \{ w | 1w| = 2 \} \]
\[ \sum_{n=1} L = \{ w | 1w| = n \} \]
\[ (\sum_{n=1} L^n) \sum_{n=1} \{ w | 1w| \leq n \} \]
\[ X, Y \leq \Sigma \]
\[ X \cdot Y = \exists x \cdot y \mid x \in X \land y \in Y \]

Examples:
- \( \text{oddparity} \cdot \text{oddparity} = \text{even} \)
\( X, Y \subseteq \Sigma \)

\( X \cdot Y = \{ x \cdot y \mid x \in X \text{ and } y \in Y \} \)

**Examples**

\( L_{\text{odd parity}} \cdot L_{\text{odd parity}} = L_{\text{even}} \)

\( L_{\text{odd parity}} \cdot L_{\text{even}} = L_{\text{odd}} \)
\[ x, y \in \Sigma^* \]
\[ x \cdot y = \exists x \cdot y | x \in X \wedge y \in Y \]

Examples:
\[ \text{odd parity} \cdot \text{odd parity} = \text{even} \]
\[ \text{odd parity} \cdot \text{even} = \text{odd} \]

\[ A \cdot B \]
\[ \text{probability} \]
\[ \text{possible?} \]
$x, y \leq \Sigma^*$

$\exists \cdot : x \in X \land y \in Y$  

**Examples**

$L_{odd parity} \cdot L_{odd parity} = L_{even}$  
$L_{odd parity} \cdot L_{even} = L_{odd}$

$A \cdot B$  

Refute finite  

Finite  

Possible?  

$2^* \cdot \emptyset = \emptyset$  

Yes
\[ x, y \leq \Sigma^* \]
\[ x \cdot y = \exists x, y \mid x \in X \land y \in Y \]

Examples:
- \( \text{odd parity} \cdot \text{odd parity} = \text{even} \)
- \( \text{odd parity} \cdot \text{even} = \text{odd} \)
- \( A \cdot B \)
- Suppose both are fruit
- \( Z \cdot \Phi = \Phi \)
- \( x \cdot y \leq x \cdot x \)
- \( x \cdot y = y \cdot x \)
- \( ? \text{ always?} \)
$x, y \leq \Sigma^*$

$x \cdot y = \exists x, y \mid x \in X \land y \in Y$

**Examples**

- $\text{odd parity} \cdot \text{odd parity} = \text{even}$
- $\text{odd parity} \cdot \text{even} = \text{odd}$

**A \cdot B**

- $\uparrow$
- $\text{fruit}$
- $\text{fruit}$

possible?

$2^x \cdot \emptyset = \emptyset$

$X \cdot Y \leq Y \cdot X$

\{03, {13} \} \neq \{13 \cdot 303\}

always true?

no
• Is the class of regular languages closed under concatenation?

• Again, for Java programs, say, it’s not too hard to prove this.

• What about finite automata? Inability to back up the input tape is one issue...
An idea for closure under concatenation, but not clear how to do it – may need to stay in $M_1$ for several visits to $F$ before jumping to $M_2$.

E.g.:

\{even parity\} • \{exactly 5 1’s\}

which 1 is 5th from end?
1 DFA as a recognizer.
2 generator
3 A different kind of generator.

Figure 1.27

<table>
<thead>
<tr>
<th>Transition</th>
<th>State 1</th>
<th>State 2</th>
<th>State 3</th>
<th>State 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,1</td>
<td></td>
<td></td>
<td></td>
<td>0,1</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>0,ε</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0,1</td>
<td></td>
<td></td>
<td></td>
<td>0,1</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Transitions:
- 0101, 11
- 0101, 01
- 00101, 0011
- 111101, 11111
- 1010, 0110
- 10100, 01101
1. DFA as a recognizer.
2. ... generator
   0001 D 001
3. A different kind of generator.

**Figure 1.27**

- Q: What would it mean/how could we define an equivalent recognizer?
- A: Non determinism
Figure 1.29
A finite state machine
\[ M = (Q, \Sigma, \delta, q_0, F) \]
where:
- \( Q \) is a set of states
- \( q_0 \in Q \) is the start state
- \( \Sigma \) is a finite set of alphabet
- \( F \subseteq Q \) is the set of final states or accepting states

\[ \delta : Q \times \Sigma \rightarrow 2^Q \]

\text{E.g. for } M \text{ with } Q = \{ q_0, q_1, q_2 \}
\[ \delta(q_0, \text{start}) = \{ q_1 \} \]
\[ \delta(q_1, 0) = \{ q_2 \} \]
\[ \delta(q_1, 1) = \{ q_1, q_2 \} \]
\[ \delta(q_2, 0) = \{ q_0, q_2 \} \]
\[ L = \{ w \in \{a,b\}^* \mid \text{3rd letter from the right end of } w \text{ is } "a" \} \]
L = { w in \{a,b\}^* | 3rd letter from the right end of w is "a" }
DEF: 

\[ M \text{ ends in state } q \text{ after reading } w \in \Sigma^* \text{ if } \]

1. \( w = w_1 w_2 \cdots w_n \) where \( w_i \in \Sigma \cup \{\varepsilon\} \)

2. \( \exists \text{ state } r_0, r_1, r_2 \cdots r_n \in Q \)

   (a) \( r_0 = q_0 \)
   (b) \( \forall 1 \leq i \leq n \)

\[ r_i \in S(r_{i-1}, w_i) = r_i \]

(c) \( r_n = q \)

Fact: \( q \) is unique because \( S \) is a function, basically
Defn
M accepts \( w \in \Sigma^* \) \( \iff \) some State, \( q \), reached by \( M \) after reading \( w \) is an accepting state, i.e., \( q \in F \).

Defn
The language recognized by \( M \), \( L(M) = \{ w \in \Sigma^* \mid M \text{ accepts } w \} \).

Note
Every \( M \) recognizes exactly one language. Implicitly, it “recognizes” both strings it must accept and those it must reject.

Very important note that “might be in a non-final state” does not imply “reject.”

- to show \( M \) on \( w \):
  Accepts—show one path ending in \( F \)
  Rejects—show all paths fail to end in \( F \)
Example “guess & check”:  
$L = \{ a^n \mid n \text{ is a multiple of } 2, 3, 5 \text{ or } 7 \}$

Note: equiv DFA has $2 \times 3 \times 5 \times 7$ states and messy set of final states.
\[ L = \{ w \in \{a,b\}^* \mid \text{3rd letter from the right end of } w \text{ is "a"} \} \]
(Non-)Example

\[ L = \{ a^p \mid p \text{ is prime} \} \]

Q: is \( M \) deterministic?
Q: Does \( M \) accept \( a^p \) for every prime \( p \)?
Q: does \( L(M) = L \)?
Q: but, doesn’t it always guess right?
Nondeterminism: How

- View it as a generator of a language
- View it as a recognizer of a language
  - “build the tree”
  - explore all paths
  - guess-and-check
Nondeterminism: Why

- **Specifications**: say, clearly & concisely, *what*, not *how*
- Precise, and often **concise specification**
  - “do A or B, but I don’t yet know/don’t want clutter of saying which”
  - Sometimes *exponentially* more concise - “3rd letter from end”
- Natural model of incompletely specified/partially known systems
  - if correct wrt a partial spec, then correct wrt any implementation consistent with that spec
  - “is state ‘reactor boiling / control rods out’ unreachable, even allowing for unknown behavior of subsystem X”?
Kleene Star

• Defn: \( L^* = \bigcup_{n \geq 0} L^n \)

• Examples

  i) \( \Sigma^* \): a simple special case

  ii) \( L = \{ a^p b \mid p \text{ is prime} \} \)

      \( L^* = \{ \varepsilon \} \cup \{ a^{p_1} b a^{p_2} b \ldots b a^{p_k} b \mid k \geq 1, \)

      and each \( p_i \) is prime}
Closure under union
Given NFA M, can build one for $L(M)^*$?
Given NFA $M$, can build one for $L(M)^*$?
Given NFA $M$, can build one for $L(M)^*$?

No
(may reject $\varepsilon$)
Given NFA $M$, can build one for $L(M)^*$?
Given NFA M, can build one for $L(M)^*$?

No, may accept extra stuff (if M can loop back to start before reaching F)
Given NFA $M$, can build one for $L(M)^*$?
Given NFA $M$, can we build one for $L(M)^*$?
Closure under $\ast$

General strategy: such proofs are usually constructive, i.e., given a (generic) NFA $N_1$, we construct a “new” NFA, $N$. In this case:

[Notation changed slightly to match Thm 1.49 in Sipser; see it for careful description of $N$ vs $N_1$]

Then prove the correctness of the construction, i.e., that $L(N) = (L(N_1))^\ast$. Proof idea: connect computation trace(s) of “old” NFA to ones in “new” NFA, where a “trace” means, recalling the definition of “$M$ could be in state $q$ after reading $w$,” the/a sequence of states/transitions/edges $M$ follows/could follow on some input.
Closure under $*$, II

For the correctness proof, there are usually 2 directions, namely:

$(L(N_1))^* \subseteq L(N)$ and $L(N) \subseteq (L(N_1))^*$

1) $(L(N_1))^* \subseteq L(N)$, or, equivalently, given any $k \geq 0$ and any $k$ strings $x_1, x_2, ..., x_k$, each in $L(N_1)$, show that their concatenation $x_1 \cdot x_2 \cdot ... \cdot x_k = x$ is in $L(N)$. For this direction, let $r_{i0}, r_{i1}, r_{i2}, ..., r_{in_i}$ be an accepting trace (in $N_1$) for $x_i$, $1 \leq i \leq k$. Note $q_1 = r_{i0}$, (why?) and $r_{in_i} \in F$ (why?) The key idea is that you can glue these together using the new start state and the new $\varepsilon$ transitions (green state/arrows) to build an accepting trace in $N$ for $x$.

Namely: $q_0, r_{10}, r_{11}, r_{12}, ..., r_{1n_1}, r_{20}, r_{21}, r_{22}, ..., r_{2n_2}, ..., r_{k0}, ..., r_{kn_k} \in F$. This is a valid accepting trace in $N$ since all transitions in that sequence are either transitions of $N_1$, hence in $N$, or are $\varepsilon$ transitions from a final state of $N_1$ to $N_1$’s start state $q_1 = r_{i0} = r_{20} = ..., $ hence again in $N$.

$\therefore x \in L(N)$.

Trace really should be $r_{i0}, a_{i0}, r_{i1}, a_{i1}, ...$, i.e. alternately $\in Q, \in \Sigma \cup \{\varepsilon\}$, but slides are small & I’m being lazy.
Closure under *,

2) \( L(N) \subseteq (L(N_1))^* \), or equivalently, given any \( x \) in \( L(N) \), show that it can be broken into \( k \geq 0 \) substrings \( x_1, x_2, ..., x_k \) (i.e., \( x = x_1 \cdot x_2 \cdot ... \cdot x_k \)) so that each is in \( L(N_1) \). For this direction, suppose \( q_0 = r_0, r_1, r_2, ..., r_n \in F \) is an accepting trace (in \( N \)) for \( x \). Note that \( r_1 = q_1 \), since the only transition leaving \( q_0 \) goes to \( q_1 \) (and is labeled \( \varepsilon \)). Let \( x_1 \) be the concatenation of all edge labels up to (but excluding) the next green edge (i.e., an \( \varepsilon \)-move from a final state back to \( q_1 \)). Note that \( x_1 \in L(N_1) \), since the included transitions are all present in \( N_1 \) and run from its start state to a final state, so they are an accepting trace in \( N_1 \). Similarly, let \( x_2 \) be the concatenation of all edge labels up to the next green edge, ..., and \( x_k \) those after the last green edge. By the same reasoning, each \( x_i \in L(N_1) \), for each \( 1 \leq i \leq k \). Finally, note that \( x = x_1 \cdot x_2 \cdot ... \cdot x_k \) since the excluded transitions are all \( \varepsilon \)-moves. \( \therefore x \in (L(N_1))^* \)  

QED
Closure under *, Leftovers

There are a few points in the proof above that I deliberately didn’t address. I strongly suggest that you think about them and see if you can fill in missing details and/or explain why they actually are covered, even if not explicitly mentioned. I suggest you write it (but no need to turn it in).

• Are $x = \epsilon / k = 0$ correctly handled, or do you need to say more?

• Is it a problem if $N_1$’s start state is a final state?

• Is it a problem if $N_1$ includes $\epsilon$-moves from (some or all states in) $F$ to $q_1$?

• Is there anything else I omitted?
Closure under Concatenation

2 NFA's $M_1, M_2$ $L_i = L(M_i)$

$L_1 \cdot L_2$

Final states of $M_1$
no longer final
NFA == DFA,

or not?
Symbol read

0
1
0
1
0

FIGURE 1.29
**Definition**

$M_1$ & $M_2$ equivalent if $L(M_1) = L(M_2)$

**Theorem 1.39**

A nfa $N$ is equivalent dfa $M$

given $N = (Q, \Sigma, \delta, q_0, F)$

build $M = (Q', \Sigma, \delta', q'_0, F')$

(warm up: no $\varepsilon$-moves)

$Q' = 2^Q$

$q'_0 = \{ q_0 \}$

$F' = \{ R \subseteq Q \mid R \cap F \neq \emptyset \}$

$\forall a \in \Sigma, \forall R \subseteq Q: S'(R, a) = \bigcup_{r \in R} S(r, a)$
\[ L = \{ w \in \{a,b\}^* \mid \text{3rd letter from the right end of } w \text{ is } "a" \} \]
A 3-state NFA

The equivalent $2^3 = 8$-state DFA, built as in Theorem 1.39 (4 states on left are not reachable from start state but are part of the DFA.)

An example transition:

$$\delta'(\{1,2,3\}, b) = \delta(1, b) \cup \delta(2, b) \cup \delta(3, b) = \{1\} \cup \{3\} \cup \emptyset = \{1,3\}$$
\[ L = \{ w \in \{a,b\}^* \mid \text{3rd letter from the right end of } w \text{ is "a"} \}\]

Exercise: apply the construction to the NFA below.
Note: You will not get the DFA above (but it will be equivalent).

**Figure 1.31**
Simulation of NFAs by DFAs: Notes on the Proof of Theorem 1.39

W. L. Ruzzo

The text’s assertion that the construction given in the proof of Theorem 1.39 (1st ed: 1.19) is “obviously correct” is a little breezy. Here is an outline of a somewhat more formal correctness proof. I will only handle the case where the NFA has no ε-transitions. Notation is as in the book.

For any \( x \in \Sigma^* \), define

\[
Q_{N,x} = \{ r \in Q \mid N \text{ could be in state } r \text{ after reading } x \}, \text{ and}
\]

\[
Q_{M,x} = \text{ the state } R \in Q' \text{ that } M \text{ would be in after reading } x.
\]

The key idea in the proof is that these two sets are identical, i.e., that the single state of the DFA faithfully reflects the complete range of possible states of the NFA. The proof is by induction on \(|x|\).

**Basis**: \(|x| = 0\). Obviously \( x = \epsilon \). Then

\[
Q_{N,\epsilon} = \{ q_0 \} = q'_0 = Q_{M,\epsilon}.
\]

The first and third equalities follow from the definitions of “moves” for NFAs and DFAs, respectively, and the middle equality follows from the construction of \( M \).

**Induction**: \(|x| = n > 0\). Suppose \( Q_{N,y} = Q_{M,y} \) for all strings \( y \in \Sigma^* \) with \(|y| < n\), and let \( x \in \Sigma^* \) be an arbitrary string with \(|x| = n > 0\). Since \( x \) is not empty, there must be some \( y \in \Sigma^* \) and some \( a \in \Sigma \) such that \( x = ya \). For any \( r \in Q \),

\[
N \text{ could be in state } r \text{ after reading } x = ya
\]

\[\Leftrightarrow \] there is some \( r' \in Q \) such that \( N \text{ could be in } r' \text{ after reading } y \) and \( r \in \delta(r', a) \)

\[\Leftrightarrow r \in \bigcup \delta(r', a) \]
reflects the complete range of possible states of the NFA. The proof is by induction on $|x|$.

**Basis:** ($|x| = 0$.) Obviously $x = \epsilon$. Then

$$Q_{N,\epsilon} = \{q_0\} = q'_0 = Q_{M,\epsilon}.$$  

The first and third equalities follow from the definitions of “moves” for NFAs and DFAs, respectively, and the middle equality follows from the construction of $M$.

**Induction:** ($|x| = n > 0$.) Suppose $Q_{N,y} = Q_{M,y}$ for all strings $y \in \Sigma^*$ with $|y| < n$, and let $x \in \Sigma^*$ be an arbitrary string with $|x| = n > 0$. Since $x$ is not empty, there must be some $y \in \Sigma^*$ and some $a \in \Sigma$ such that $x = ya$. For any $r \in Q$,

$$N \text{ could be in state } r \text{ after reading } x = ya \quad (1)$$

$$\iff \text{ there is some } r' \in Q \text{ such that } N \text{ could be in } r' \text{ after reading } y \text{ and } r \in \delta(r',a) \quad (2)$$

$$\iff r \in \bigcup_{r' \in Q_{N,y}} \delta(r',a) \quad (3)$$

$$\iff r \in \delta'(Q_{N,y},a) \quad (4)$$

$$\iff r \in \delta'(Q_{M,y},a) \quad (5)$$

$$\iff r \in Q_{M,x} \quad (6)$$

The equivalence of (1) and (2) follows from the definition of “moves” for NFAs: the last step must be a move from some state reached after reading $y$. The equivalence of (2) and (3) is just set theory. The equivalence of (3) and (4) follows from the definition of $\delta'$. The equivalence of (4) and (5) follows from the induction hypothesis. The equivalence of (5) and (6) follows from the definition of “moves” for DFAs.

Given the equivalence established above, it’s easy to see that $L(N) = L(M)$, since $N$ accepts $x$ if and only if it can reach a final state after reading $x$, which will be true if and only if $Q_{N,x}$ contains a final state, which happens if and only if $Q_{M,x} \in F'$.  

46
**Defn**

\( M_1 \) & \( M_2 \) equivalent if \( L(M_1) = L(M_2) \)

**Theorem 1.39**

\( \forall \text{nfa } N \exists \text{ equivalent dfa } M \)

given \( N = (Q, \Sigma, \delta, q_0, F) \)

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

(warm up: no \( \varepsilon \)-moves)

\[ Q' = 2^Q \]

\[ q'_0 = \{ q_0 \} \]

\[ F' = \{ R \subseteq Q \mid R \cap F \neq \emptyset \} \]

\( \forall a \in \Sigma, \forall R \subseteq Q: \)

\( S'(R, a) = \bigcup_{r \in R} S(r, a) \)
**Defn**

$M_1$ & $M_2$ equivalent if $L(M_1) = L(M_2)$

**Theorem 1.39**

A nfa $N$ is equivalent to a $M$ given $N = (Q, \Sigma, \delta, s_0, F)$ build $M = (Q', \Sigma, \delta', s_0', F')$

(warm up: no $\varepsilon$-moves) → Full version: with $\varepsilon$-moves

\[ Q' = Q \setminus \varepsilon \]

\[ s_0' = E(\{ s_0 \}) \]

\[ F' = \{ R \in Q \mid R \cap F \neq \emptyset \} \]

\[ \forall a \in \Sigma, \forall R \in Q: S'((r, a)) = \bigcup_{r \in R} \varepsilon(E(r, a)) \]

\[ \forall R \in Q: E(R) = \{ q \mid q \text{ reachable by } \text{our move } \varepsilon \text{-moves from some } r \in R \} \]

**NB:** do $\varepsilon$-moves before start, after other moves, not both before & after each move.
Notes on Subset Construction:
1) only the top 6 states are reachable from the start state, but all 16 are required by the construction.
2) \( \varepsilon \) moves come after \( \Sigma \) moves. E.g., \( \delta'(\{q_2\}, 1) = \emptyset \), not \( \{q_4\} \).