Nondeterministic Finite Automata

Nondeterminism allows several possible next states (no one state is determined)

<table>
<thead>
<tr>
<th>Deterministic</th>
<th>S.D.</th>
<th>Nondeterministic</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Diagram of Deterministic Automaton" /></td>
<td><img src="image2.png" alt="Diagram of S.D. Automaton" /></td>
<td><img src="image3.png" alt="Diagram of Nondeterministic Automaton" /></td>
</tr>
</tbody>
</table>

Computation

NFA vs. DFA

NFA are seemingly more powerful (NOT!)

NFA allow choice of next state

When choice, NFA may be more compact

Handling $\varepsilon$

If in $q_1$ reading input symbol $a$, and take $\varepsilon$ edge, still reading $a$, but in state $q_2$.

All $\varepsilon$ edges must be included in execution tree.

Qu: Is every DFA an NFA?
Example NFA

![Example NFA Diagram]

Execution on baabb

$L(M) =$

More Example NFA's

![More Example NFA Diagram]
Formal Definition of Nondeterministic FA

A NFA $M$ is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where,
1. $Q$ is a finite set of states
2. $\Sigma$ is a finite set called the alphabet
3. $\delta : Q \times \Sigma \rightarrow P(Q)$ (power set of $Q$, include $\epsilon$)
4. $q_0 \in Q$, the start state
5. $F \subseteq Q$, set of accept (final) states

We say that $M$ accepts $w$ if there is a sequence of states $r_0, r_1, ... r_m$ in $Q$ and $w = y_1 y_2 ... y_m$, $y_i \in \Sigma \in \epsilon$ such that
1. $r_0 = q_0$
2. $r_{i+1}$ element of $\delta(r_i, y_{i+1})$
3. $r_m$ is in $F$

Equivalence of NFA and DFA

Def: Two NFA $M_1, M_2$ are equivalent if $L(M_1) = L(M_2)$.

Th.: For each NFA, there is an equivalent DFA.

Proof idea:
Start with NFA $N$. Want DFA $M$ with $L(N) = L(M)$.
We will simulate NFA with DFA.
NFA has set of possible next states; DFA just has 1.
Solution: state of the DFA will be subset of states of NFA

Proof: Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA.

Case 1: $N$ has no $\epsilon$ transitions.

$M = (Q', \Sigma, \delta', q_0', F')$
1. $Q' = P(Q)$
2. $\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$
3. $q_0' = \{q_0\}$
4. $F' = \{R \in Q' | R \cap F = \emptyset\}$
Equivalence of NFA and DFA

Case 2: $N$ has $\varepsilon$ transitions.

Proof Idea: What needs to change? Have to account for all $\varepsilon$ moves after read an input symbol.

Proof: For $R$ a subset of $Q$, define

$$E(R) = \{ q | q \text{ can be reached from some state in } R \text{ by following only } \varepsilon \text{ edges} \}$$

Change the construction in Case 1 as follows:

1. $\delta'(R, a) = \{ q \in Q | q \in E(\delta(r, a)) \text{ for some } r \in R \}$
2. $q'_0 = \{ E(q_0) \}$

$M$ keeps track of exactly the subset of states $N$ would be in, and accepts when $N$ accepts.

Equivalence of NFA and DFA

Cor.: A language is regular iff some NFA recognizes it.

Example: NFA, construct DFA (Ex. 1.21 p. 57)

Get lots of states....can we do better? States that can’t be reached from start states that have no incoming edges........More later!
The class of regular languages are closed under \( \cup \).

Let \( N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) \)
let \( N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) \) be NFA’s.

Construct NFA \( N \) such that \( L(N) = L(N_1) \cup L(N_2) \)

Proof Idea:

Proof: We define NFA \( N \) as follows....

Regular languages are closed under Union

\[
N = (Q, \Sigma, \delta, q_0, F)
\]

1. \( Q = \{q_0\} \cup Q_1 \cup Q_2 \quad (q_0 \text{ new, } Q_1 \text{ and } Q_2 \text{ disjoint}) \)

2. \( q_0 \) is start

3. \( F = F_1 \cup F_2 \)

4. \[
\delta(q, a) = \begin{cases} 
\delta_1(q, a) & \text{if } q \text{ in } Q_1 \\
\delta_2(q, a) & \text{if } q \text{ in } Q_2 \\
\{q_1, q_2\} & \text{if } q = q_0 \text{ and } a = \varepsilon \\
\phi & \text{if } q = q_0 \text{ and } a = \varepsilon 
\end{cases}
\]
Regular languages are closed under Star

Proof: Define $N = (Q, \Sigma, \delta, q_0, F)$ from $N_1$.
1. $Q = \{q_0\} \cup Q_1$  $\ \ \ \text{q0 is new}$
2. $q_0$ is start
3. $F = \{q_0\} \cup F_1$
4. $\delta(q, a) = \begin{cases} 
\delta_1(q, a) & \text{if } q \text{ in } Q_1, q \not \text{ in } F_1 \\
\delta_1(q, a) & \text{if } q \in F_1, a \not = \varepsilon \\
\delta_1(q, a) \cup \{q_1\} & \text{if } q \in F_1 \text{ and } a = \varepsilon \\
\{q_1\} & \text{if } q = q_0, a = \varepsilon \\
\phi & \text{if } q = q_0, a \not = \varepsilon 
\end{cases}$

Regular languages are closed under Concatenation

Proof Idea: Start out in $N_1$, and in any final state of $N_1$, can switch to $N_2$; accept if wind up in final state in $N_2$.

Proof: Define $N = (Q, \Sigma, \delta, q_1, F)$ from $N_1$ and $N_2$.
1. $Q = Q_1 \cup Q_2$
2. $q_1$ is start (same as start of $N_1$)
3. $F = F_2$
4. $\delta(q, a) = \begin{cases} 
\delta_1(q, a) & \text{if } q \text{ in } Q_1, q \not \text{ in } F_1 \\
\delta_1(q, a) & \text{if } q \in F_1, a \not = \varepsilon \\
\delta_1(q, a) \cup \{q_2\} & \text{if } q \in F_1 \text{ and } a = \varepsilon \\
\delta_2(q, a) & \text{if } q \in Q_2
\end{cases}$