Last time: Automata  
Today: Automata and Regular Expressions

From last class, a non-deterministic finite automaton (NFA) can be expressed as follows:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B, C</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td>C</td>
<td>A</td>
<td></td>
</tr>
</tbody>
</table>

A simple way of converting this NFA into a deterministic finite automaton (DFA) can be represented as:

A string is accepted as long as it leads to the final state C. Both DFA and NFA are equally powerful, but there is one significant difference. NFA has states, while DFA has subsets of NFA’s states as its states. Therefore, if there are n states in NFA, there will be less than or equal to 2ⁿ states in DFA.
**Theorems:**

**Theorem 1:**
Given machine $M = \langle Q, I, f, s_o, F \rangle$ there exists a type 3 grammar $G$ such that $L(G) = T(M)$.

**Construction:**

$G = \langle V_N, V_T, P, S \rangle$

Obviously, $V_N = Q$ and $V_T = I$ also $S = s_o$

$P = \{ q \rightarrow aq' \mid f(q, a) = q' \} \cup \{ q \rightarrow a \mid f(q, a) \in F \}$ where $a \in I$

**Theorem 2:**

Given type 3 grammar, $G = \langle V_N, V_T, P, S \rangle$ there exists an NFA $M = \langle Q, I, f, s_o, F \rangle$ such that $T(M) = L(G)$.

$Q = V_N \cup \{ A \}$; where $\{ A \}$ is new symbol, i.e. a symbol not present either in $V_N$ or in $V_T$.

$I = V_T$

$s_0 = S$

$F = \begin{cases} \{ A, S \} \text{ if } S \rightarrow \lambda \in P \\ \{ A \} \text{ otherwise} \end{cases}$

$f(B, a) = \{ C \mid B \rightarrow aC \in P \} \cup \{ A \mid B \rightarrow a \in P \}$

$f(A, a) = \phi \forall a \in I$

where $B \in V_N$

**Example:**

The following can be depicted as an NFA.

- $S \rightarrow aB$
- $S \rightarrow a$
- $A \rightarrow aB$
- $B \rightarrow bA$
- $A \rightarrow a$
- $B \rightarrow b$

Here $AA$ is the new symbol.

A DFA can be constructed from this NFA.
Regular expressions:

Definition: Let $S$ be an alphabet. Let $(), ., *, U, \phi, \lambda$ be symbols not in $S$. The language of regular expressions is defined inductively as:

1. $\lambda, \phi$ are regular expressions
2. $\forall a \in S$, $a$ is a regular expression (RE)
3. If $\alpha, \beta$ are RE so are $(\alpha U \beta), (\alpha \beta), (\alpha)^*$

Examples:

$(a U b)^*$ - set of all strings including $\phi$
$a(a U b)b^* - \{aa, ab, aab, abb\ldots\}$

RE is something that represents a language, but restricted to only certain type of languages.

1. $\lambda$ and $\phi$ denote $\{\lambda\}$ and $\phi$.
2. $a$ denotes $\{a\}$.
3. $\alpha$ and $\beta$ denote sets $A$ and $B$.

$\alpha \cup \beta \Leftrightarrow A \cup B$
$\alpha \beta \Leftrightarrow AB$
$\alpha^* \Leftrightarrow A^*$

Note:

$(a \cup b)^*, (\{a\} \cup \{b\})^*, (\{a, b\})^*$ are all identical.

- Regular expressions
- Type 3 grammars
- Finite automata

represent/generate/accept regular languages

Stochastic grammars:

Extension to regular grammar we studied so far.

$\theta$ generates strings $x$ with probability $P(x \mid \theta)$ and constraint $\Sigma x P(x \mid \theta) = 1$.

Example:

$S \rightarrow aB$
$S \rightarrow bA \quad \Sigma$ to 1

Note:
It can be shown that stochastic regular grammar is equivalent to Hidden Markov Model. (without proof)