Section: Regular Languages

Regular Expressions

Method to represent strings in a language

+ union (or)
  ◦ concatenation (AND) (can omit)
  * star-closure (repeat 0 or more times)

Example:

\[(a + b)^* \circ a \circ (a + b)^* = (a+b)(a+b)^*\]

Strings over \(\sum\) that contain at least one \(a\)

Example:

\[(aa)^*\]

Strings with an even number of \(a\)'s
Definition Given $\Sigma$,

1. $\emptyset, \lambda, a \in \Sigma$ are R.E.

2. If $r$ and $s$ are R.E. then
   - $r+s$ is R.E.
   - $rs$ is R.E.
   - $(r)$ is a R.E.
   - $r^*$ is R.E.

3. $r$ is a R.E. iff it can be derived from (1) with a finite number of applications of (2).
Definition: \( L(r) = \) language denoted by R.E. \( r \).

1. \( \emptyset, \{ \lambda \}, \{a\} \) are \( L \) denoted by a R.E.

2. if \( r \) and \( s \) are R.E. then
   
   (a) \( L(r+s) = L(r) \cup L(s) \)
   
   (b) \( L(rs) = L(r) \circ L(s) \)
   
   (c) \( L((r)) = L(r) \)
   
   (d) \( L((r)^*) = (L(r)^*) \)
Precedence Rules

* highest

Example:

$$ab^* + c = (a(b^*)) + c$$
Examples:

1. $\Sigma = \{a, b\}, \{w \in \Sigma^* | w \text{ has an odd number of } a\text{'s followed by an even number of } b\text{'s}\}.$
   
   \[(aa)^*a(bb)^*\]

2. $\Sigma = \{a, b\}, \{w \in \Sigma^* | w \text{ has no more than 3 } a\text{'s and must end in } ab\}\}.

3. Regular expression for all integers (including negative)

   $\mathbb{Z} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \ldots\}$

   \[b^*(a+b)b^*(a+b)baba\]

   \[b^*(ab^* + abab^* + \lambda)ab\]

   \[0 + (- + \lambda)(1+2+\ldots+9)0|1+\ldots+9\]
Section 3.2 Equivalence of DFA and R.E.

Theorem Let $r$ be a R.E. Then $\exists$ NFA $M$ s.t. $L(M) = L(r)$.

Proof:

- $\emptyset$
- $\{\lambda\}$
- $\{a\}$

Suppose $r$ and $s$ are R.E.

1. $r + s$
2. $r \circ s$
3. $r^*$

New final state
Example

$ab^* + c$
Theorem Let \( L \) be regular. Then \( \exists \) R.E. \( r \) s.t. \( L = L(r) \).

Proof Idea: remove states sucessively until two states left

\( \bullet \) Proof:

- \( L \) is regular
  \[ \Rightarrow \exists \text{ NFA } M \text{ s.t. } L = L(M) \]

1. Assume \( M \) has one final state and \( q_0 \not\in F \)

2. Convert to a generalized transition graph (GTG), all possible edges are present. If no edge, label with \( \emptyset \)

Let \( r_{ij} \) stand for label of the edge from \( q_i \) to \( q_j \)
3. If the GTG has only two states, then it has the following form:

In this case the regular expression is:

\[ r = (r_{ii}^* r_{ij} r_{jj}^* r_{ji})^* r_{ii}^* r_{ij} r_{jj}^* \]
4. If the GTG has three states then it must have the following form:
<table>
<thead>
<tr>
<th>REPLACE</th>
<th>WITH</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_{ii}$</td>
<td>$r_{ii} + r_{ik}r^*<em>{kk}r</em>{ki}$</td>
</tr>
<tr>
<td>$r_{jj}$</td>
<td>$r_{jj} + r_{jk}r^*<em>{kk}r</em>{kj}$</td>
</tr>
<tr>
<td>$r_{ij}$</td>
<td>$r_{ij} + r_{ik}r^*<em>{kk}r</em>{kj}$</td>
</tr>
<tr>
<td>$r_{ji}$</td>
<td>$r_{ji} + r_{jk}r^*<em>{kk}r</em>{ki}$</td>
</tr>
</tbody>
</table>

**remove state $q_k$**
5. If the GTG has four or more states, pick a state $q_k$ to be removed (not initial or final state).

For all $o \neq k, p \neq k$ use the rule
$r_{op}$ replaced with $r_{op} + r_{ok}r_{kk}^*r_{kp}$
with different values of $o$ and $p$.

When done, remove $q_k$ and all its edges. Continue eliminating states until only two states are left. Finish with step 3.
6. In each step, simplify the regular expressions r and s with:

\[ r + r = r \]
\[ s + r^* s = r^* s \]
\[ r + \emptyset = r \]
\[ r\emptyset = \emptyset \]
\[ \emptyset^* = \emptyset \]
\[ r\lambda = r \]
\[ (\lambda + r)^* = r^* \]
\[ (\lambda + r)r^* = r^* \]

and similar rules.
Example:
Edit the regular expression below:

```
((aa*b)*(a+aa*b)b)*(aa*b)*(a+aa*b)
```
Grammar $G=(V,T,S,P)$

$V$ variables (nonterminals)
$T$ terminals
$S$ start symbol
$P$ productions

Right-linear grammar:

all productions of form

$A \rightarrow xB$
$A \rightarrow x$

where $A, B \in V$, $x \in T^*$
Left-linear grammar:

all productions of form

\[ A \rightarrow Bx \]
\[ A \rightarrow x \]

where \( A,B \in V, \ x \in T^* \)

Definition:

A regular grammar is a right-linear or left-linear grammar.
Example 1:

\[ G = (\{S\}, \{a,b\}, S, P), \quad P = \]
\[
S \rightarrow abS \\
S \rightarrow \lambda \\
S \rightarrow Sab
\]

NOT regular Grammar
Example 2:

\[ G = (\{ S, B \}, \{ a, b \}, S, P), \]
\[ P = \]
\[ S \to aB \mid bS \mid \lambda \]
\[ B \to aS \mid bB \]

\[ L(G) = \{ \text{strings with an even no. of } a \text{'s} \} \]
Theorem: $L$ is a regular language iff $\exists$ regular grammar $G$ s.t. $L=L(G)$.

Outline of proof:

$(\iff)$ Given a regular grammar $G$
  Construct NFA $M$
  Show $L(G)=L(M)$

$(\implies)$ Given a regular language $L$
  $\exists$ DFA $M$ s.t. $L=L(M)$
  Construct reg. grammar $G$
  Show $L(G)=L(M)$
Proof of Theorem:

\((\Leftarrow)\) Given a regular grammar \(G\)

\(G=(V,T,S,P)\)

\(V=\{V_0, V_1, \ldots, V_y\}\)

\(T=\{v_0, v_1, \ldots, v_z\}\)

\(S=V_0\)

Assume \(G\) is right-linear

(see book for left-linear case).

Construct NFA \(M\) s.t. \(L(G)=L(M)\)

If \(w\in L(G), w=v_1v_2\ldots v_k\)
\[ M = (V \cup \{V_f\}, T, \delta, V_0, \{V_f\}) \]

- \( V_0 \) is the start (initial) state
- For each production, \( V_i \rightarrow aV_j \),

For each production, \( V_i \rightarrow a \),

Show \( L(G) = L(M) \)

Thus, given R.G. G,

L(G) is regular
Given a regular language $L$
\exists DFA $M$ s.t. $L = L(M)$
$M = (Q, \Sigma, \delta, q_0, F)$
$Q = \{q_0, q_1, \ldots, q_n\}$
$\Sigma = \{a_1, a_2, \ldots, a_m\}$

Construct R.G. $G$ s.t. $L(G) = L(M)$
$G = (Q, \Sigma, q_0, P)$
\text{if } \delta(q_i, a_j) = q_k \text{ then }
\text{if } q_k \in F \text{ then }

Show $w \in L(M) \iff w \in L(G)$
Thus, $L(G) = L(M)$.
QED.
Example

\[ G=([S, B], \{a, b\}, S, P), \quad P= \]
\[
S \to aB \mid bS \mid \lambda \\
B \to aS \mid bB
\]
Example: