Chapter 2. L.A.
I. Basics
II. Deterministic Finite State Machine
III. Non-Deterministic Finite State Machine
IV. Equivalence of DFSM and NFSM
V. Regular Expression
VI. RE and FSM
VII. Application of DFSM to LA

I. Basics
a) Tasks of a LA
   - Tokens
   - Overpass comments and white spaces
   - Case conversion if needed
   - Preprocessing
b) Token VS. Lexeme
   Token is a generic type of a “meaningful unit”.
   Lexeme is an actual instance.
   Example:
   Source: if (x>y) then …
   Token         Lexeme
   Keyword       if
   Separator     (      
   Identifier    x
   Operator      >
   Identifier    y
   Separator     )      
   Keyword       then
   …

II. DFSM
a.) Definition:
   DFSM = (Σ, Q, q₀, F, N)
   Where:
   Σ is a finite set of input symbols (Alphabets)
   Q is a finite set of states
   F ⊆ Q is a set of accepting states
   q₀ ∈ Q is the starting state
   N is the transition function
   (Q × Σ) → Q

   Example: M = ( 
   Σ = {a, b, c}, Q = {1, 2, 3, 4}, q₀ = 1, F = {3, 4}, 
   )
Example: Vending Machine for 25 cents candy.

\[ M = (\Sigma = \{n, d, q, s\}, (n=\text{nickel}, d=\text{dime}, q=\text{quarter}, s=\text{select}) \]
\[ Q = \{0,1,2,3,4,5,6\}, q_0 = 0, \]
\[ F = \{5, 6\} \]

Graphical Representation of M: See book page 21, Figure 2.1.

b.) Acceptance of a string by M
Definition: M accepts (recognizes) a string if (1) the entire string has been read; (2) the machine M is in any accepting state.
Graphical representation of acceptance: A string \( \Omega \) is accepted by M, if and only if there is a path from \( q_0 \) to any accepting state whose edges spell \( \Omega \).

Example: Acceptance of a string.
\[ M = (\Sigma = \{0,1\}, Q = \{A, B, C, D\}, q_0 = A, F = \{B, D\}, \]
\[ \begin{array}{c|cc}
   & 0 & 1 \\
   \hline
   A & B & A \\
   B & C & C \\
   C & D & B \\
   D & A & D \\
\end{array} \]

\( \omega = 11001 \)

use trace diagram:

\[
\begin{array}{cccccc}
   A & \rightarrow & A & \rightarrow & A & \rightarrow \\
   & & & & & \rightarrow B \\
\end{array}
\]

M accepts string \( \omega \).

\( \omega_1 = 011100 \).

Use trace diagram, we can get that it is not accepted by M.

c.) Implementation of a DFSM.

Pseudocode:

\[
\text{Table}_N = \text{array}[1..\text{Nstates}, 1..\text{Nalphabets}] \text{ of integer}; \\
\text{State} = 1; \\
\text{For } i = 1 \text{ to length}(\omega) \text{ do} \\
\quad \text{Begin} \\
\quad \quad \text{Col} = \text{get_col_number}(\omega[i]); \\
\quad \quad \text{State} = \text{Table}[\text{state}, \text{col}]; \\
\quad \text{End}; \\
\quad \text{If state is in F then return True} \\
\quad \text{Else return False};
\]

III. NFSM

a.) Definition: \( M = (\Sigma, Q, q_0, F, N) \)

Where:

\( \Sigma \) is a finite set of input symbols (Alphabets)
\( Q \) is a finite set of states
\( F \subseteq Q \) is a set of accepting states
\( q_0 \in Q \) is the starting state
\( N \) is the transition function

\( (Q \times (\Sigma \cup \epsilon)) \rightarrow P(Q) \)

Acceptance of a string in NFSM: same as in DFSM.

Example 1 of NFSM

\( M = (\Sigma = \{a, b\}, Q = \{1, 2, 3\}, q_0 = 1, F = \{2, 3\}, \)
N: a  b
1 {1, 2}  {1}
2 {2}  {1,3}
3 {2,3}  {1,3}

Graphical representation:

$\omega = baabab$, is $\omega$ accepted? Use trace diagram, we know that it is accepted.

Example 2 of NFSM. (allow $\varepsilon$-transitions)
Use book’s notation for M:

M | a   | b   | $\varepsilon$
---|-----|-----|-----
$q_0$ = 1  {1,2}  {3}  {}  
2          {3}  {2,3}  {4}
3          {3,4}  {2}  {1,4}
4          {1}  {2}  {}

$\omega = aba$. It is accepted by M.
IV. Equivalence of DFSM and NFSM.
Claim: For any NFSM, there exists an equivalent DFSM.
Question: How?
Answer: two cases:
Case 1: Exclude ε-transitions.
There are two methods.
Method 1:
Task: M = (Σ, Q, q₀, F, N), which is an NFSM.
→ M₀ = (Σ’, Q’, q₀’, F’, N’), which is a DFSM.
Method: Σ’ = Σ.
Q’ = the power set of states in M in [ ].
F’ = all states in Q’ that contain at least one accepting state in M.
q₀’ = [q₀].
N’ : N’([P₁, P₂, ..., Pₙ], x ∈ Σ’)
= N(P₁, x) ∪ N(P₂, x) ∪ ... ∪ N(Pₙ, x) in [ ].
Example:
M = (Σ = {a, b}, Q = {1, 2, 3}, q₀ = 1, F = {2})
N:
<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{1,2}</td>
<td>{1}</td>
</tr>
<tr>
<td>2</td>
<td>{2}</td>
<td>{1,3}</td>
</tr>
<tr>
<td>3</td>
<td>{2,3}</td>
<td>{1,3}</td>
</tr>
</tbody>
</table>
M’ = (Σ’ = {a,b}, Q’ = {[1], [2], [3], [1,2], [1,3], [2,3], [1,2,3], [ ]}, q₀’=[1]
F’ = {[2], [1,2], [2,3], [1,2,3]}
N’:
<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>[1,2]</td>
<td>[1]</td>
</tr>
<tr>
<td>[2]</td>
<td>[2]</td>
<td>[1,3]</td>
</tr>
<tr>
<td>[3]</td>
<td>[2,3]</td>
<td>[1,3]</td>
</tr>
<tr>
<td>[1,2]</td>
<td>[1,2]</td>
<td>[1,3]</td>
</tr>
<tr>
<td>[1,3]</td>
<td>[1,2,3]</td>
<td>[1,3]</td>
</tr>
<tr>
<td>[2,3]</td>
<td>[2,3]</td>
<td>[1,3]</td>
</tr>
<tr>
<td>[1,2,3]</td>
<td>[1,2,3]</td>
<td>[1,3]</td>
</tr>
<tr>
<td>[ ]</td>
<td>[ ]</td>
<td>[ ]</td>
</tr>
</tbody>
</table>
Observations:
1. Some states are not needed.
2. 2^Q states, when Q = 3, there are 8 states; when Q = 10, there are 1024 states.
Method 2:
Given M = (Σ, Q, q₀, F, N), which is an NFSM.
→ M₀ = (Σ’, Q’, q₀’, F’, N’), which is a DFSM
Method:
Σ’ = Σ, q₀’ = [q₀].
Computation of N’:
Start with q₀’,
While incomplete rows remaining do
Let x = [P₁, P₂, ..., Pₙ] be the current row,
For each $a \in \Sigma'$, do
Find $y = N'(x, a)$
Add $y$ to $N'$
If $y$ not in $N'$, then add $y$
End for
End while

$Q'$ = the states in $N'$.
$F'$ = all states in $Q'$ that contain at least one accepting state in $M$.
Example: Using the same NFSM as the last example:

$M' = (\Sigma' = \{a, b\}, q_0' = [1], Q' = \{[1], [1,2], [1,3], [1,2,3]\},$
$F' = \{[1,2], [1,2,3]\}$

<table>
<thead>
<tr>
<th>$N'$</th>
<th>$a$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>[1,2]</td>
<td>[1]</td>
</tr>
<tr>
<td>[1,2]</td>
<td>[1,2]</td>
<td>[1,3]</td>
</tr>
<tr>
<td>[1,3]</td>
<td>[1,2,3]</td>
<td>[1,3]</td>
</tr>
<tr>
<td>[1,2,3]</td>
<td>[1,2,3]</td>
<td>[1,3]</td>
</tr>
</tbody>
</table>

Case 2: consider $\varepsilon$-transition. (Only use method 2).
Definition: The $\varepsilon$-closure of a state $q \in Q$ is the set of all states that can be reached from $q$
by means of $\varepsilon$-transitions including $q$.
Example:

<table>
<thead>
<tr>
<th></th>
<th>$a$</th>
<th>$b$</th>
<th>$\varepsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{1}</td>
<td>{2}</td>
<td>{2,3}</td>
</tr>
<tr>
<td>2</td>
<td>{2}</td>
<td>{3}</td>
<td>{}</td>
</tr>
<tr>
<td>3</td>
<td>{2}</td>
<td>{1}</td>
<td>{2}</td>
</tr>
</tbody>
</table>

$\varepsilon$-closure(1) = \{2, 3, 1\}
$\varepsilon$-closure(2) = \{2\}
$\varepsilon$-closure(3) = \{2, 3\}.

We need to make 2 following modifications to our methods in case 1 in order to work
with $\varepsilon$-transition.
1. For starting state $q_0'$, find $\varepsilon$-closure of $q_0$.
2. For each new state $[P_1, P_2, \ldots, P_n]$, find the $\varepsilon$-closure of $[P_1, P_2, \ldots, P_n]$
   $= \varepsilon$-closure($P_1$) $\cup$ $\varepsilon$-closure($P_2$) $\cup \ldots$ $\cup$ $\varepsilon$-closure($P_n$) in [ ].

Example of case 2 using method 2.

<table>
<thead>
<tr>
<th>$N'$</th>
<th>$a$</th>
<th>$b$</th>
<th>$\varepsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{1}</td>
<td>{3}</td>
<td>{2}</td>
</tr>
<tr>
<td>2</td>
<td>{3}</td>
<td>{2}</td>
<td>{}</td>
</tr>
<tr>
<td>3</td>
<td>{1}</td>
<td>{3}</td>
<td>{2}</td>
</tr>
</tbody>
</table>

$q_0 = 1$.
$\varepsilon$-closure(1) = \{1,2\}; $\varepsilon$-closure(2) = \{2\}; $\varepsilon$-closure(3) = \{2,3\}.
V. Regular Expressions (RE)

Example: Identifier token
First character = letter; any # of letters or digits.
1 (1 | d)*

Definition: RE over an alphabet Σ is defined as follows:
(i) any character in Σ is a RE;
(ii) ε is a RE;
(iii) if R and S are REs, then:
    RS is a RE /* Concatenation */
    R|S is a RE /* Union */
    R* is a RE, S* is a RE /* Kleene Closure */
    R+ is a RE, S+ is a RE.

Example: Σ = {0, 1}
(i) 0 is a RE; 1 is a RE;
(ii) ε is a RE;
(iii) 01, 11, 10, 00 are REs.
     0|1 is a RE;
     0 or more times of R : R*
        0* = 0 or more times of 0 => ε, 0, 00, 000, … are REs.
     1 or more times of R: R+
        0+ = 0, 00, 000, … are REs.

Example of an interesting RE:
(ε|+-) (0|d*) . (0|d)*

Equivalence Rules:
R(ST) = (RS)T /* Associativity */
R/(S|T) = (R/S)|T
R|S = S|R /* Commutativity */
Note: RS ≠ SR.
R(S|T) = RS | RT /* Distributivity */
R+ = RR*
R*|ε = R*
Rε = εR = R /* Identity */
VI. RE and FSM

Claim: There exists a FSM for each RE.

Q: How do we build such a FSM for a RE?

A: Using the Thompson Construction Method.

How? We are building FSM inductively. (Small $\rightarrow$ Large)

In this method, we provide rules for building “bigger” FSM.

Rules:

(i) FSM $M$ that recognizes $\varepsilon$.

(ii) FSM $M$ that recognizes $a \in \Sigma$.

(iii) FSM that recognizes $RS$.

Way 1:

Way 2:

(iv) FSM that recognizes $R|S$.

(v) FSM that recognizes $R^*$

Way 1:
Way 2:

(vi) FSM that recognizes $R^+$.

Example 1: Build a FSM for the RE: $(a^*|bc)^+$.

Example 2: Build a FSM for the RE: $(ab|ab^*a)^*$

VII. Application of FSM to LA
Building a Lexer = Constructing a FSM.

Step 1: Write tokens in REs.
Step 2: Build an NFSM for the REs in Step 1.
Step 3: Convert NFSM to DFSM and code.

Example: Build a LA for a language with the following ID and integer tokens.

ID: has a letter with any number of letters or digits.
Integer: has any number of digits.

Step 1: Identifier RE: l(|d)*
Step 2: build a NFSM for REs.

\[\begin{array}{c|ccc}
1 & ABK & CDFHI & LKI \\
2 & CDFHI & EHCDF & GHICDF \\
3 & LKI & & LKI \\
4 & EHCDF & EHCDF & GHICDF \\
5 & GHICDF & EHCDF & GHICDF \\
6 & & & \\
\end{array}\]

Re-write the machine:

\[\begin{array}{c|cc}
q_0 & 1 & d \\
1 & 2 & 3 \\
2 & 4 & 5 \\
3 & 6 & 3 \\
4 & 4 & 5 \\
5 & 4 & 5 \\
6 & 6 & 6 \\
\end{array}\]

Example string \(\omega = a12\).

\[\begin{array}{c}
1 \quad a \\
2 \quad 1 \\
5 \quad 2 \\
5 \\
\end{array}\]

Accepted.

Suggested Homework Assignments for Chapter 2:
2.1 c; 2.2 a; 2.3 a; 2.8; 2.10; 2.14.