Compiler Design

1

Compiler Design

BOOKS AND REFERENCES

1. A.V. Aho, R. Sethi, J.D. Ullman, “Compilers Principles, Techniques and Tools”, Addison-Wesley, 1986.

2. Santanu Chattopadhyay, “Compiler Design”, PHI Learning Pvt. Ltd., 2015.

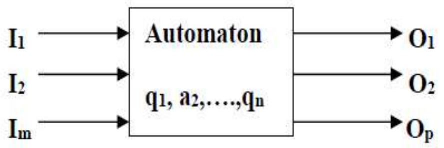
RANJAN JANA, Dept. of IT, RCC Institute of Information Technology

2

Formal Language & Automata Theory

Automation: An automation is defined as a system where energy, materials and information are transformed, transmitted and used for performing some functions without direct participation of man. Examples are automatic machine tools, automatic packing machines, and automatic photo printing machines.

An automation in which the output depends only on the input is called an automation without a memory. An automation in which the output depends on the states also is called automation with a finite memory.

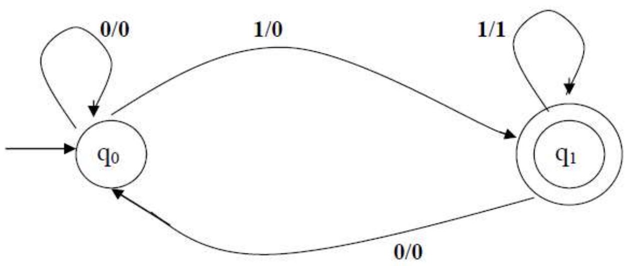


Formal language: A formal language is an abstraction of the general characteristics of programming languages.

Formal Language & Automata Theory

Transition systems: A transition graph or a transition system is a finite directed labelled graph in which each vertex (or node) represents a state and the directed edges indicate the transition of a state and the edges are labelled with input/output.

Example



The initial state is represented by a circle with an arrow pointing towards it, the final state by two concentric circles, and the other states are represented by just a circle. The edge is labelled by input/output.

Formal Language & Automata Theory

Finite Automation: A finite automaton can be represented by a 5-tuple (Q, ∑, δ, q_0, F), where

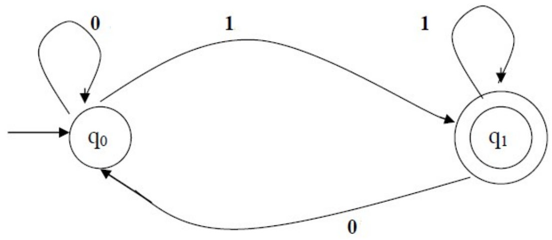
i. Q is a finite nonempty set of states;

ii. ∑ is a finite nonempty set of inputs called input alphabet;

iii. δ is a function which maps Q x∑ into Q and is usually called direct transition function.

iv. q_0 Є Q is the initial state; and

v. F ⊆ Q is the set of final states. It is assumed that there may be more than one final state.



Formal Language & Automata Theory

Nondeterministic finite automation (NDFA):

A nondeterministic finite automaton is a 5-tuple (Q, ∑, δ, q_0, F), where

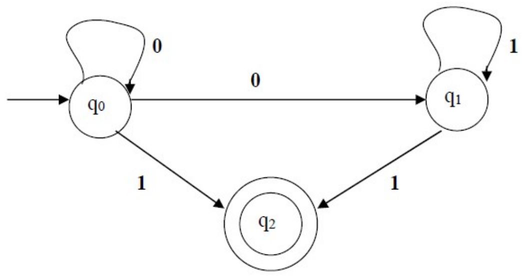
i. Q is a finite nonempty set of states;

ii. ∑ is a finite nonempty set of inputs;

iii. δ is the transition function mapping from Q x∑ into 2^Q which is the power set of Q, the set of all subsets of Q;

iv. q_0 Є Q is the initial state; and

v. F ⊆ Q is the set of final states.



Formal Language & Automata Theory

Transition Table:

States/Σ	Inputs 0	1
→ q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

Formal Language & Automata Theory

Construct a deterministic automaton equivalent to $M = (\{q_0, q_1\}, \{0,1\}, \delta, q_0, \{q_0\})$. δ is given by its state table.

States/Σ	Inputs 0	1
→ q_0	q_0	q_1
q_1	q_1	q_0, q_1

For the deterministic automaton M_1,

I. **The states are subsets of $\{q_0, q_1\}$, i.e. Φ, $[q_0]$, $[q_0, q_1]$, $[q_1]$;**
II. **$[q_0]$ is the initial state;**
III. **$[q_0]$ and $[q_0, q_1]$ are the final states as these are the states containing q_0;**
IV. **δ is defined by the state table given below**

States/Σ	0	1
→ $[q_0]$	$[q_0]$	$[q_1]$
$[q_1]$	$[q_1]$	$[q_0, q_1]$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1]$

q_0 and q_1 appear in the rows corresponding to q_0 and q_1 and the column corresponding to 0. So, $\delta([q_0,q_1], 0) = [q_0, q_1]$.

Formal Language & Automata Theory

Construct a deterministic finite automaton equivalent to $M = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \delta, q_0, \{q_3\})$. δ is given below.

States/Σ	a	b
→ q_0	q_0, q_1	q_0
q_1	q_2	q_1
q_2	q_3	q_3
q_3		q_2

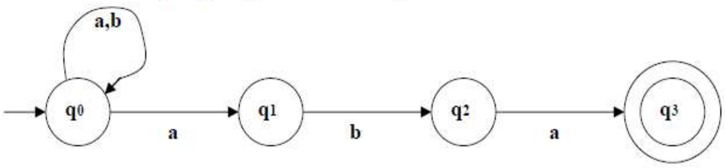
The deterministic automaton M_1 equivalent to M is given by $M_1 = (2^Q, \{a, b\}, \delta, [q_0], F)$, where F consists of $[q_0, q_1, q_3]$, and $[q_0, q_1, q_2, q_3]$. δ is given in below.

States/Σ	a	b
→ $[q_0]$	$[q_0, q_1]$	$[q_0]$
$[q_0, q_1]$	$[q_0, q_1, q_2]$	$[q_0, q_1]$
$[q_0, q_1, q_2]$	$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_3]$
$[q_0, q_1, q_3]$	$[q_0, q_1, q_2]$	$[q_0, q_1, q_2]$
$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_2, q_3]$

Formal Language & Automata Theory

Construct a nondeterministic finite automaton accepting the set of all strings over {a, b} ending with aba. Use it to construct a DFA accepting the same set of strings.

The NDFA accepting the given set of strings is



The corresponding state table is

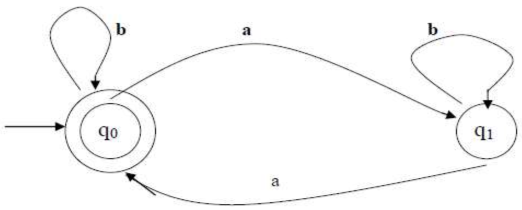
States/Σ	a	b
→ q_0	q_0, q_1	q_0
q_1		q_2
q_2	q_3	
q_3		

The state table of the corresponding DFA is

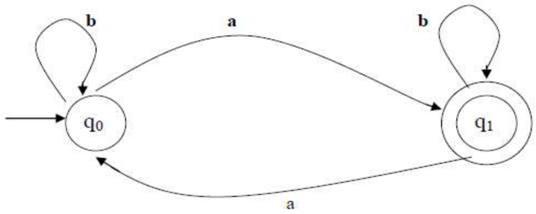
States/Σ	a	b
→ $[q_0]$	$[q_0, q_1]$	$[q_0]$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_2]$
$[q_0, q_2]$	$[q_0, q_1, q_3]$	$[q_0]$
$[q_0, q_1, q_3]$	$[q_0, q_1]$	$[q_0, q_2]$

Formal Language & Automata Theory

Draw the transition diagram of a finite state automation M that accepts the string of even number of a's over {a, b}.



Draw the transition diagram of a finite state automation M that accepts the string of odd number of a's over {a, b}.



Formal Language & Automata Theory

Assignment No: 1

Draw the transition diagram of a finite state automation M that accepts the string of eve number of a's and even number of b's over {a, b}.

Formal Language & Automata Theory

Regular Expressions: A formal recursive definition of regular expressions over Σ as follows:

1. Any terminal symbol (i.e. an element of Σ), ^, and Ø are regular expressions.
2. The union of two regular expressions R1 and R2, written as R1+R2, is also a regular expression.
3. The concatenation of two regular expressions R1 and R2, written as R1R2, is also a regular expression.
4. The iteration (or closure of) a regular expression R, written as R*, is also a regular expression.
5. If R is a regular expression, then (R) is also a regular expression.
6. The regular expressions over Σ are precisely those obtained recursively by the application of the rules 1-5 once or several times.

Formal Language & Automata Theory

Regular set: Any set represented by a regular expression is called a regular set.

Describe the following sets by regular expressions

1. The set of all strings of 0's and 1's ending in 00.
   (0+1)*00
2. The set of all strings of 0's and 1's beginning with 0 and ending with 1.
   0(0+1)*1
3. The language of all strings containing exactly two 0's over alphabet set {0, 1}.
   1*01*01*
4. The language of all strings containing at least two 0's over alphabet set {0, 1}.
   (0+1)* 0 (0+1)* 0 (0+1)*
5. The set of all strings over {0, 1} having at least one pair of 0's.
   (0+1)*00(0+1)*
6. The set of all strings over {a, b} with three consecutive b's.
   (a+b)*bbb (a+b)*
7. The language of all strings that does not end with 01 over alphabet set {0, 1}.
   (0+1)* (00+10+11)

Formal Language & Automata Theory

Construct an NFA for the regular expression R=(a+b)*abb

Construct an NFA that accepts the language L= {ab, abc}*

Construct an NFA for the regular expression R=1(0+1)*0

Formal Language & Automata Theory

Assignment No: 2

Construct an FA equivalent to the regular expression

(0+1)* (00+11) (0+1)*

Formal Language & Automata Theory

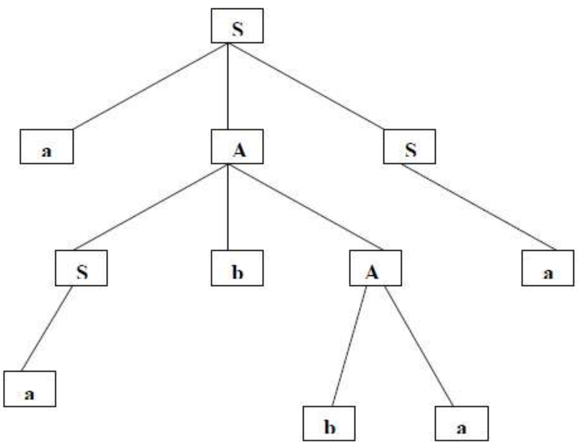
Derivation Tree: A derivation tree (also called a parse tree) for a CFG G= (V_N, Σ, P, S) is a tree satisfying the following:

1. Every vertex has a label which is a variable or terminal or ^.
2. The root has a label S.
3. The label of an internal vertex is a variable.
4. If the vertices $n_1, n_2, \ldots, n_k$ written with label $X_1, X_2, \ldots, X_k$ are the sons of vertex n with label A, the $A \rightarrow X_1X_2 \ldots X_k$ is a production in P.
5. A vertex n is a leaf if its label is a ∈ Σ or ^.

G=({S, A}, {a, b, c}, { S->aA, A->bA, A-> c}, S)

Formal Language & Automata Theory

Ex1: For the grammar G defined by S→aAS|a, A→SbA|SS|ba. Draw the derivation tree for the string aabbaa.



Formal Language & Automata Theory

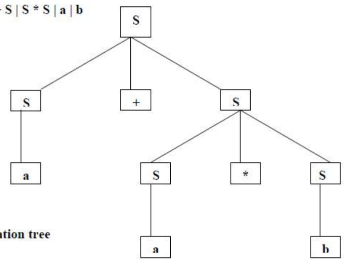
A derivation $A \overset{*}{\Rightarrow} W$ is called left-most derivation if we apply a production only to the leftmost variable at every step.

A derivation $A \overset{*}{\Rightarrow} W$ is called right-most derivation if we apply a production only to the rightmost variable at every step.

A terminal string $W \in L(G)$ is ambiguous if there exists two or more derivation trees for W.

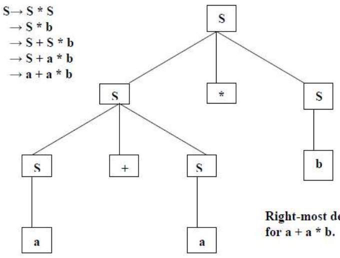
Example: $S \rightarrow S + S \mid S * S \mid a \mid b$

$S \rightarrow S + S$
$\rightarrow a + S$
$\rightarrow a + S * S$
$\rightarrow a + a * S$
$\rightarrow a + a * b$



Left-most derivation tree for a+ a * b.

$S \rightarrow S * S$
$\rightarrow S * b$
$\rightarrow S + S * b$
$\rightarrow S + a * b$
$\rightarrow a + a * b$



Right-most derivation tree for a + a * b.

THANK YOU