Contents

Chapter 1

Finite Automata

In This Book We Learn :

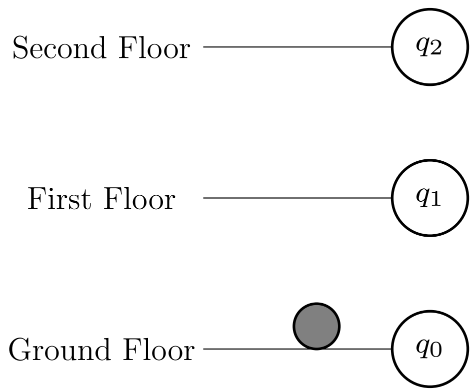
1. The Theory Of Automata
2. Formal Languages
3. Regular Sets and Regular Grammars
4. Context Free Languages
5. Pushdown Automata
6. LR(K) Grammars
7. Turing Machine

1.1 Definition of Finite Automata

F.A. can be represented by a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where :

- Q is finite non-empty set of states
- Σ is finite non-empty set of input alphabets
- δ is the transition function which maps : $Q \times \Sigma \rightarrow Q$
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ is the set of final states

1.2 Example of lift control



$$Q = \{q_0, q_1, q_2\}$$

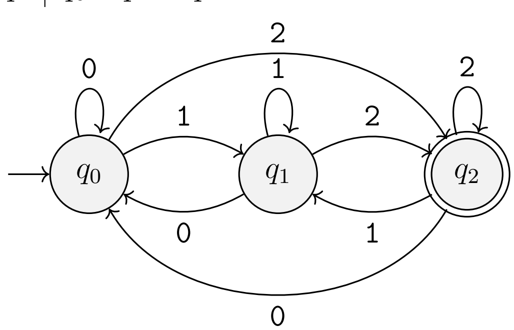
$$\Sigma = \{0, 1, 2\}$$

$$q_0 = initial\ \ state, q_0 \in Q$$

$$F = \{q_2\} \subseteq Q$$

Transition Table

	0	1	2
q_0	q_0	q_1	q_2
q_1	q_0	q_1	q_2
q_2	q_0	q_1	q_2



1.3 Acceptability of a string by DFA

A string $x \in \Sigma^*$ is accepted by a Finite automaton $M = (Q, \Sigma, \delta, q_0, F)$ if $\delta(q_0, x) = q$ for some $q \in F$.

1.4 Properties of Transition Function

1. $\delta(q, \wedge) = q$
2. For all strings $\omega \in Z^*$ and input symbol a :

$$\delta(q, a\omega) = \delta(\delta(q, a), \omega)$$

$$\delta(q, \omega a) = \delta(\delta(q, \omega), a)$$

1.4.1 Example

Transition table given below, is string 110101 accepted by this machine ?

Transition Table

	$x = 0$	$x = 1$
q_1	q_3	q_1
q_2	q_4	q_1
q_3	q_1	q_4
q_4	q_2	q_3

Answer : Yes, because

$$\begin{aligned}\delta(q_1, \underbrace{1}_{a}\underbrace{10101}_{\omega}) &= \delta(q_2, 10101) \\ &= \delta(q_1, 0101) \\ &= \delta(q_3, 101) \\ &= \delta(q_4, 01) \\ &= \delta(q_2, 1) \\ &= \delta(q_1, \wedge) \\ &= q_1\end{aligned}$$

so the string is accepted .

1.5 Definition of NFA

A Non-deterministic Finite Automata (NDFA or NFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where :

- Q is a finite non-empty set of states
- Σ is a finite non-empty set of input alphabets
- δ is the transition function mapping from $Q \times \Sigma \to 2^Q$, where 2^Q is the power-set of all subsets of Q
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ is the set of final states

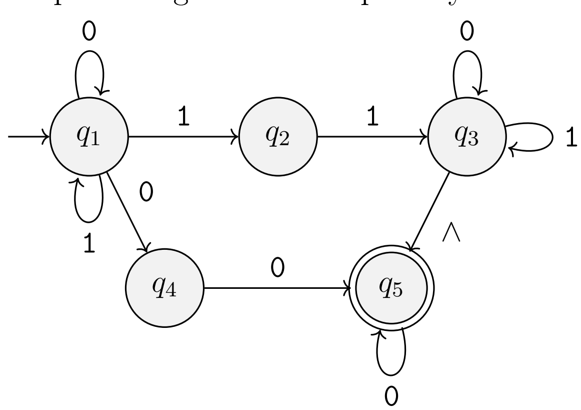
1.6 Acceptability by NFA

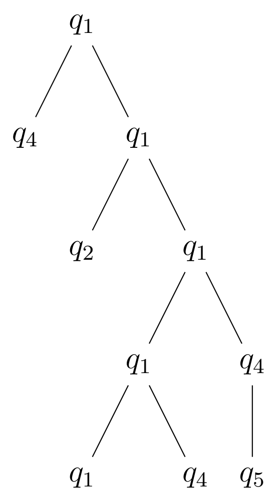
A string $\omega \in \Sigma^*$ is accepted by NFA M if $\delta(q_0, \omega)$ contains some final state.

The set accepted by an automaton M(Deterministic or Non-Deterministic) is the set of all input strings accepted by M. It is denoted by T(M) .

1.6.1 Example

Is the input string 0100 is accepted by NFA below :



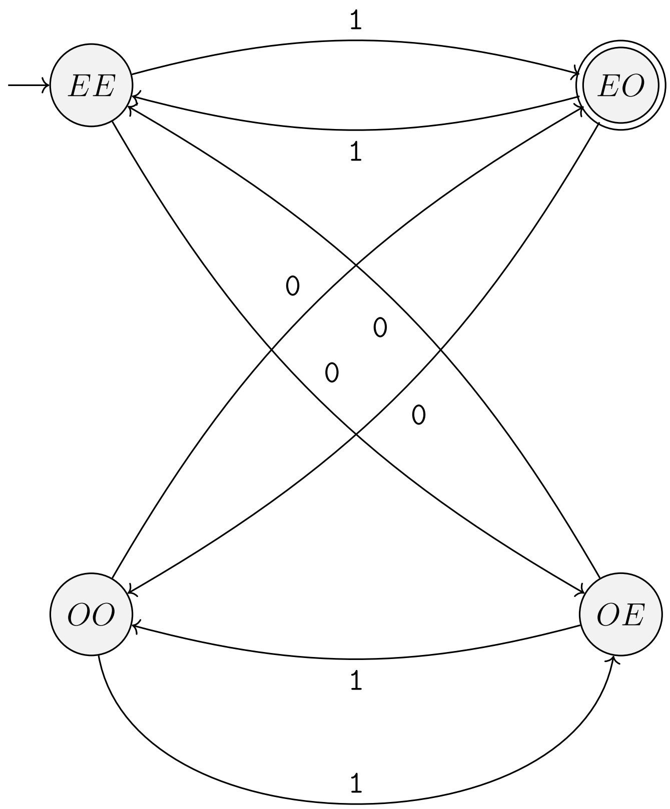


$q_5 \in F$ so the string is accepted

1.6.2 Example

Design a DFA which take 0s and 1s as input strings and accepts that string which will have even number of 0s and odd number of 1s ?

EO : Even Number of 0's - Odd Number of 1's



Suppose :

$$EE \to q_0$$

$$OO \to q_1$$

$$OE \to q_2$$

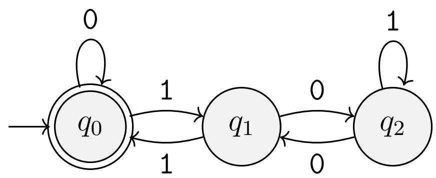
$$EO \to q_3$$

Transition Table

	$x = 0$	$x = 1$
$\to q_0$	q_2	q_3
q_1	q_3	q_2
q_2	q_0	q_1
$(f)\ q_3$	q_1	q_0

1.6.3 Example

Design one DFA which Takes 0s and 1s as input string and accepts that binary number which is divisible by 3 ?



We Suppose :

$q_0 \equiv (\%3 == 0)$	$q_1 \equiv (\%3 == 1)$	$q_2 \equiv (\%3 == 2)$
0	1	10
11	100	101
110	111	1000
1001	1010	1011

	$x = 0$	$x = 1$
$(f) \rightarrow q_0$	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2

1.7 NFA to DFA Conversion

Construct a DFA equivalent to $M = (\{q_1, q_2, q_3, q_4\}, \{0, 1\}, \delta, q_1, \{q_4\})$ where δ is given below :

	0	1
$\rightarrow q_1$	q_1, q_2	q_1
q_2	q_3	q_2
q_3	q_4	q_4
q_4	$-$	q_3

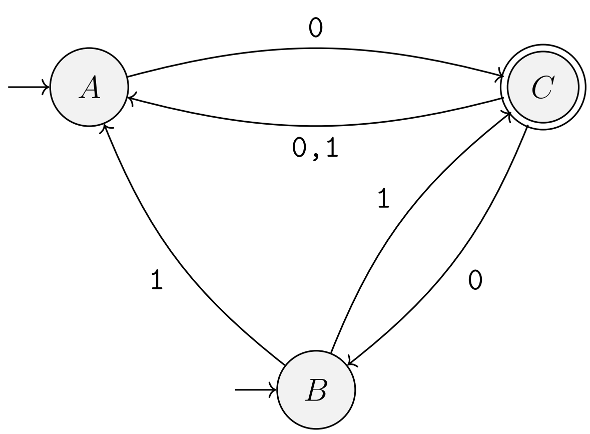
Answer :

	0	1
$\rightarrow [q_1]$	$[q_1, q_2]$	$[q_1]$
$[q_1, q_2]$	$[q_1, q_2, q_3]$	$[q_1, q_2]$
$[q_1, q_2, q_3]$	$[q_1, q_2, q_3, q_4]$	$[q_1, q_2, q_4]$
(f)$[q_1, q_2, q_3, q_4]$	$[q_1, q_2, q_3, q_4]$	$[q_1, q_2, q_3, q_4]$
(f)$[q_1, q_2, q_4]$	$[q_1, q_2, q_3]$	$[q_1, q_2, q_3]$

note : q_4 is the final state so every where q_4 is in the set is the final state
.

1.8 NFA to DFA Conversion

Construct a DFA against the following NFA :



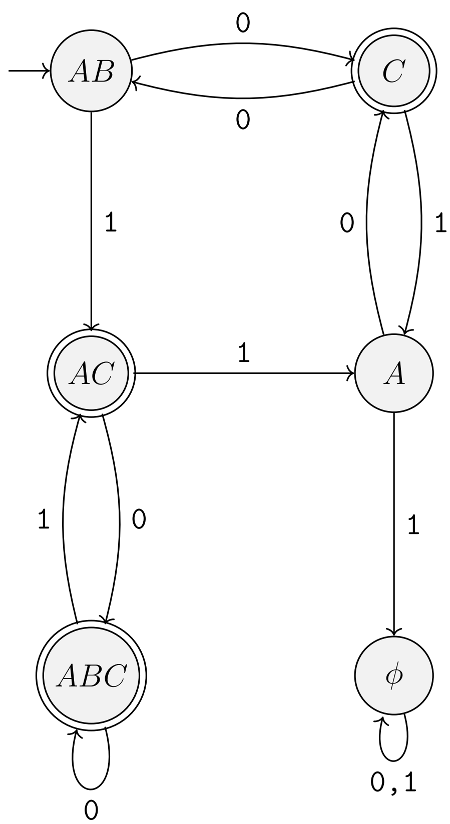
Answer : let's create the transition table :

	0	1
$\rightarrow A$	C	$-$
$\rightarrow B$	$-$	AC
$(f)C$	AB	A

Now let's create Transition table for DFA :

	0	1
$\rightarrow AB$	C	AC
$(f)C$	AB	A
$(f)AC$	ABC	A
A	C	ϕ
$(f)ABC$	ABC	AC
ϕ	ϕ	ϕ

note : for initial state we combine all the initial states
and here is the transition diagram for DFA :



1.9 Mealy Machine

a Mealy Machine is a 6-tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$, where :

- Q is a finite set of states

- Σ is the set of input alphabets
- Δ is the set of output alphabets
- δ is the transition function $Q \times \Sigma \rightarrow Q$
- λ is the output function mapping $Q \times \Sigma \rightarrow \Delta$
- q_0 is the initial state, $q_0 \in Q$

and :

$$Z(t) = \lambda(q(t), x(t))$$

which Z is the output, λ is the output function, $q(t)$ is the present state, $x(t)$ is the present input

1.9.1 Example of Mealy Machine

	$a = 0$		$a = 1$	
	state	*output*	*state*	*output*
$\rightarrow q_0$	q_2	0	q_3	0
q_1	q_3	1	q_3	0
q_2	q_0	1	q_2	1
q_3	q_1	0	q_1	0

$$Q = \{q_0, q_1, q_2, q_3\}$$

$$\Sigma = \{0, 1\}$$

$$\Delta = \{0, 1\}$$

1.10 Moore Machine

a Moore Machine is a 6-tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$, where :

- Q is a finite set of states
- Σ is the finite set of input alphabets
- Δ is the finite set of output alphabets

- δ is the transition function : $Q \times \Sigma \to Q$
- λ is the output function mapping $Q \to \Delta$
- q_0 is the initial state, $q_0 \in Q$

$$Z(t) = \lambda(q(t))$$

Z is the output , λ is the output function, $q(t)$ is the present state

1.10.1 Example of a Moore Machine

	$a = 0$	$a = 1$	λ
$\to q_0$	q_2	q_0	0
q_1	q_3	q_0	1
q_2	q_0	q_2	1
q_3	q_1	q_3	0

$$Q = \{q_0, q_1, q_2, q_3\}$$

$$\Sigma = \{0, 1\}$$

$$\Delta = \{0, 1\}$$

1.11 Moore to Mealy Convertion

Convert the following Moore Machine to Mealy Machine :

	$a = 0$	$a = 1$	*output*
$\to q_1$	q_4	q_2	0
q_2	q_2	q_3	1
q_3	q_3	q_4	0
q_4	q_4	q_1	0

Answer : for each place we have q_2 we put 1 and other places we put 0.

	$a = 0$		$a = 1$	
	state	*output*	*state*	*output*
$\to q_1$	q_4	0	q_2	1
q_2	q_2	1	q_3	0
q_3	q_3	0	q_4	0
q_4	q_4	0	q_1	0

1.12 Mealy to Moore Convertion

Convert the following Mealy Machine to Moore Machine :

	$a=0$		$a=1$	
	state	*output*	*state*	*output*
$\rightarrow q_0$	q_2	0	q_1	0
q_1	q_0	1	q_3	0
q_2	q_1	1	q_0	1
q_3	q_3	1	q_2	0

Answer :

First We Clearize this Table to :

	$a=0$		$a=1$	
	state	*output*	*state*	*output*
$\rightarrow q_0$	q_2	0	q_{10}	0
q_{10}	q_0	1	q_{30}	0
q_{11}	q_0	1	q_{30}	0
q_2	q_{11}	1	q_0	1
q_{30}	q_{31}	1	q_2	0
q_{31}	q_{31}	1	q_2	0

And Then we reach to Moore Machine :

	$a=0$	$a=1$	*output*
$\rightarrow q_0$	q_2	q_{10}	1
q_{10}	q_0	q_{30}	0
q_{11}	q_0	q_{30}	1
q_2	q_{11}	q_0	0
q_{30}	q_{31}	q_2	0
q_{31}	q_{31}	q_2	1

1.13 Minimization of Finite Automata

Definition : Two states q_1 and q_2 are equivalent (denoted by $q_1 \equiv q_2$) if both $\delta(q_1, x)$ and $\delta(q_2, x)$ are final states or both of them are non-final states for all $x \in \Sigma^*$.

Definition : Two states q_1 and q_2 are k equivalent ($k \geq 0$) if both $\delta(q_1, x)$ and $\delta(q_2, x)$ are final states or both are non-final states. for all strings x of length k or less.

1.13.1 Construction of minimum automata

	0	1
$\rightarrow q_1$	q_2	q_6
q_2	q_7	q_3
q_3	q_1	q_3
q_4	q_3	q_7
q_5	q_8	q_6
q_6	q_3	q_7
q_7	q_7	q_5
q_8	q_7	q_3

note : string with length 0 is : $\wedge$

and $\delta(q, \wedge) = q$

now we have :

$$\pi_0 = \{\{q_3\}, \{q_1, q_2, q_4, q_5, q_6, q_7, q_8, \}\}$$

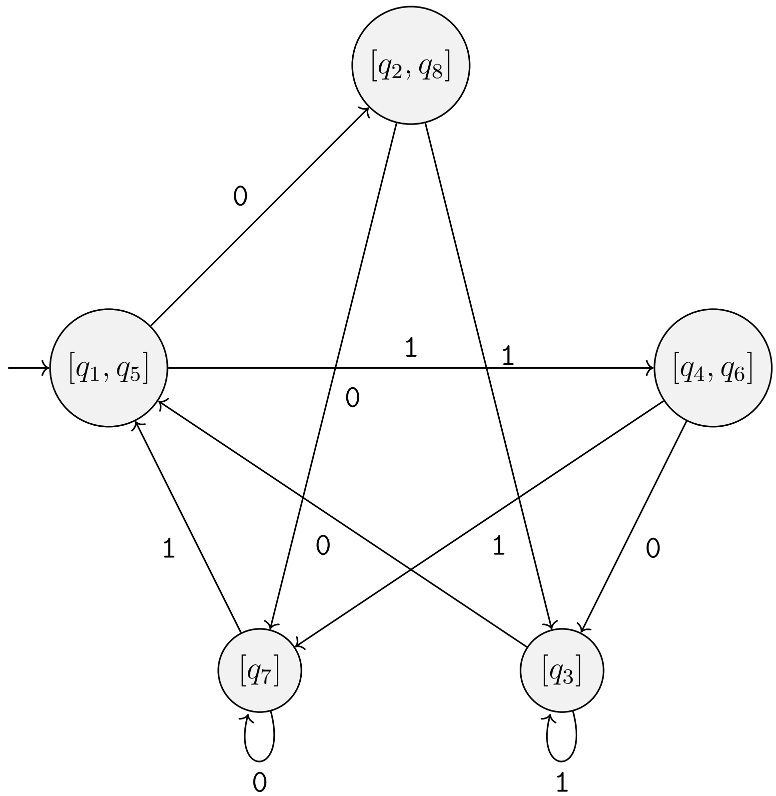
$$\pi_1 = \{\{q_3\}, \{q_4, q_6\}, \{q_2, q_8\}, \{q_1, q_5, q_7\}\}$$

$$\pi_2 = \{\{q_3\}, \{q_4, q_6\}, \{q_2, q_8\}, \{q_1, q_5\}, \{q_7\}\}$$

$$\pi_3 = \{\{q_3\}, \{q_4, q_6\}, \{q_2, q_8\}, \{q_1, q_5\}, \{q_7\}\}$$

$$\pi_2 = \pi_3 \Rightarrow 2 - equivalent$$

	0	1
$\rightarrow [q_1, q_5]$	$[q_2, q_8]$	$[q_4, q_6]$
$[q_2, q_8]$	$[q_7]$	$[q_3]$
$[q_4, q_6]$	$[q_3]$	$[q_7]$
$[q_7]$	$[q_7]$	$[q_1, q_5]$
$[q_3]$	$[q_1, q_5]$	$[q_3]$



1.14 Definition of a Grammar

a Grammar is (V, Σ, S, P), where :

- V is a finite non-empty set whose elements are variables
- Σ or T is a finite non-empty set whose elements are terminals

$$V \cap \Sigma = \phi$$

- S is a start symbol, where $S \in V$
- P is a finite set whose elements are $\alpha \to \beta$, known as production rules, where, $\alpha, \beta \in (V \cup \Sigma)^*$, α should contain at least one symbol from V.

1.14.1 Example

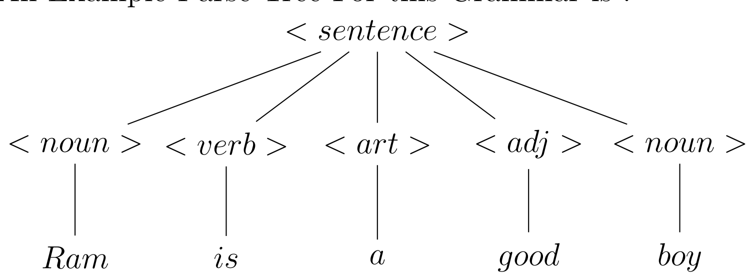
$G = (V, \Sigma, S, P)$ is a grammar where :

$$V = \{< sentence >, < noun >, < adj >, < verb >, < art >\}$$
$$\Sigma = \{Ram, Rita, Azad, is, are, a, an, good, bad, boy, girl\}$$
$$S =< sentence >$$

P consists of the following production rules :

$$< sentence > \rightarrow < noun >< verb >< art >< adj >< noun >$$
$$< noun > \rightarrow Ram|Rita|Azad|boy|girl$$
$$< verb > \rightarrow is|are$$
$$< art > \rightarrow a|an$$
$$< adj > \rightarrow good|bad$$

An Example Parse Tree For this Grammar is :



1.14.2 Example

Determine the Grammer G Where :

$$L(G) = \{0^n 1^n | n \geq 0\}$$

Answer :
$S \rightarrow 0S1|\lambda$
sample : $0^3 1^3$

$S \rightarrow 0S1 \rightarrow 00S11 \rightarrow 000S111 \rightarrow 000111 \equiv 0^3 1^3$

$$\begin{aligned}
&G = (V, \Sigma, S, P) \\
&V = \{S\} \\
&\Sigma = \{0, 1\} \\
&S = S \\
&P = \{S \rightarrow 0S1, S \rightarrow \lambda\}
\end{aligned}$$

1.14.3 Example

Determine the Grammar G Where :

$$L(G) = \{0^n 1^n | n \geq 1\}$$

Answer :
$S \rightarrow 0S1|01$

$$\begin{aligned}
&G = (V, \Sigma, S, P) \\
&V = \{S\} \\
&\Sigma = \{0, 1\} \\
&S = S \\
&P = \{S \rightarrow 0S1, S \rightarrow 01\}
\end{aligned}$$

1.14.4 Example

Determine the Grammar G Where :

$$L(G) = \{a^n b^m c^k | n, k > 0 \, and \, m \geq 0\}$$

Answer :

$$
\begin{aligned}
S &\to S_1S_2S_3 \\
S_1 &\to aS_1|a \\
S_2 &\to bS_2|\lambda \\
S_3 &\to cS_3|c
\end{aligned}
$$

$$
\begin{aligned}
&G = (V, \Sigma, S, P) \\
&V = \{S, S_1, S_2, S_3\} \\
&\Sigma = \{a, b, c\} \\
&S = S \\
&P = \{S \to S_1S_2S_3, S_1 \to aS_1|a, S_2 \to bS_2|\lambda, S_3 \to cS_3|c\}
\end{aligned}
$$

1.14.5 Example

Determine the Grammar G Where :

$$L(G) = \{a^n b^m c^k | n \geq 0\,, k > 1\,, m = n + k\}$$

Answer :

$$
\begin{aligned}
S &\to S_1S_2 \\
S_1 &\to aS_1b|\lambda \\
S_2 &\to bS_2c|bbcc
\end{aligned}
$$

$$
\begin{aligned}
&G = (V, \Sigma, S, P) \\
&V = \{S, S_1, S_2\} \\
&\Sigma = \{a, b, c\} \\
&S = S \\
&P = \{S \to S_1S_2, S_1 \to aS_1b|\lambda, S_2 \to bS_2c|bbcc\}
\end{aligned}
$$

1.14.6 Example

Determine the Grammar G Where :

$$L(G) = \{a^n b^m | n > m \geq 1\}$$

Answer :

$$S \to aS|aSb|aab$$

$$\begin{aligned}
&G = (V, \Sigma, S, P) \\
&V = \{S\} \\
&\Sigma = \{a, b\} \\
&S = S \\
&P = \{S \to aS, S \to aSb, S \to aab\}
\end{aligned}$$

1.14.7 Example

Determine the Grammar G Where :

$$L(G) = \{(ab)^n c^n | n \geq 1\}$$

Answer :

$$S \to abSc|abc$$

$$\begin{aligned}
&G = (V, \Sigma, S, P) \\
&V = \{S\} \\
&\Sigma = \{a, b, c\} \\
&S = S \\
&P = \{S \to abSc, S \to abc\}
\end{aligned}$$

1.14.8 Example

Determine the Grammar G Where :

$$L(G) = \{(x)^{2n}y^n | n \geq 1\}$$

Answer :

$$S \rightarrow xxSy | xxy$$

$$\begin{aligned} &G = (V, \Sigma, S, P) \\ &V = \{S\} \\ &\Sigma = \{x, y\} \\ &S = S \\ &P = \{S \rightarrow xxSy, S \rightarrow xxy\} \end{aligned}$$

1.14.9 Example

Determine the Grammar G Where :

$$L(G) = \{\omega c \omega^T | \omega \in (a, b)^*\}$$

Sample :

$$\underbrace{abb}_{\omega} c \underbrace{bba}_{\omega^T}$$

Answer :

$$S \rightarrow aSa | bSb | c$$

$$\begin{aligned}
&G = (V, \Sigma, S, P) \\
&V = \{S\} \\
&\Sigma = \{x, y\} \\
&S = S \\
&P = \{S \to aSa, S \to bSb, S \to c\}
\end{aligned}$$

1.14.10 Example

Determine the Grammar G Where :

$$L(G) = \{\omega c \omega^T | \omega \in (a, b)^+\}$$

Answer :

$$S \to aSa|bSb|aca|bcb$$

$$\begin{aligned}
&G = (V, \Sigma, S, P) \\
&V = \{S\} \\
&\Sigma = \{a, b, c\} \\
&S = S \\
&P = \{S \to aSa, S \to bSb, S \to aca, S \to bcb\}
\end{aligned}$$

1.14.11 Example

Determine the Grammar G Where :

$$L(G) = \{a^n b^m |\, where\, n\, is\, even\, and\, m\, is\, odd\, \}$$

Answer :

$$\begin{aligned} S &\rightarrow S_1S_2 \\ S_1 &\rightarrow aaS_1|\wedge \\ S_2 &\rightarrow bbS_2|b \end{aligned}$$

note : one extra b cause odd number of b

$$\begin{aligned} &G = (V, \Sigma, S, P) \\ &V = \{S, S_1, S_2\} \\ &\Sigma = \{a, b\} \\ &S = S \\ &P = \{S \rightarrow S_1S_2b, S_1 \rightarrow aaS_1|\wedge, S_1 \rightarrow bbS_2|b\} \end{aligned}$$

1.14.12 Example

Determine the Grammar G Where :

$$L(G) = \{a^nb^mc^md^n|n \geq 1, m \geq 0\}$$

Answer :

$$\begin{aligned} S &\rightarrow aSd|aS_1d \\ S_1 &\rightarrow bS_1c|\wedge \end{aligned}$$

$$\begin{aligned} &G = (V, \Sigma, S, P) \\ &V = \{S, S_1\} \\ &\Sigma = \{a, b, c, d\} \\ &S = S \\ &P = \{S \rightarrow aSd, S_1 \rightarrow aSd|aS_1d, S_1 \rightarrow bS_1c|\wedge, S \rightarrow bcb\} \end{aligned}$$

1.14.13 Example

Determine the Grammar G Where :

$$L(G) = \{a^n b^n c^n | n \geq 1\}$$

Answer :
if n = 3 then $\omega = aaabbbccc \equiv a^3b^3c^3$.
let us apply $S \rightarrow aSBC$ for $(n-1)$ number of times .

$$\begin{aligned} S &\rightarrow aSBC \\ &\rightarrow aaSBCBC \end{aligned}$$

now apply $S \rightarrow aBC$ once :

$$\rightarrow aaaBCBCBC$$

now apply $CB \rightarrow BC$:

$$\begin{aligned} &\rightarrow aaaB\underbrace{CB}_{BC}\underbrace{CB}_{BC}C \\ &\rightarrow aaaBB\underbrace{CB}_{BC}CC \\ &\rightarrow aaaBBBCCC \end{aligned}$$

we shall apply $aB \rightarrow ab$:

$$\begin{aligned} &\rightarrow aa\underbrace{aB}_{ab}BBCCC \\ &\rightarrow aaabBBCCC \end{aligned}$$

Now we apply $bB \rightarrow bb$:

$$\begin{aligned}
&\to aaa\underbrace{bB}_{ab}BCCC \\
&\to aaab\underbrace{bB}_{bb}CCC \\
&\to aaabbbCCC
\end{aligned}$$

Now we apply $bC \to bc$:

$$\begin{aligned}
&\to aaabb\underbrace{bC}_{bc}CC \\
&\to aaabbbcCC
\end{aligned}$$

Now we apply $cC \to cc$:

$$\begin{aligned}
&\to aaabbb\underbrace{cC}_{cc}C \\
&\to aaabbbc\underbrace{cC}_{cc} \\
&\to aaabbbccc
\end{aligned}$$

$G = (V, \Sigma, S, P)$
$V = \{S, B, C\}$
$\Sigma = \{a, b, c\}$
$S = S$
$P = \{S \to aSBC, S \to aBC, CB \to BC, aB \to ab, bB \to bb, bC \to bc, cC \to cc\}$

1.14.14 Example

Find the language generated the following Grammar :

$$S \to 0S1$$
$$S \to 0A1$$
$$A \to 1A$$
$$A \to 1$$

answer :

$$L(G) = \{0^m 1^n | n > m \geq 1\}$$

Chapter 2

Regular Expressions and Identities

we are mainly concerned with the characterization of sets of strings recognized by finite automata . It is therefore appropriate to develope a compact language for describing such sets of strings, the language thus developed is known as type-3 language or as the language of regular expressions . some sample string are 101, (01+10)11 , . . .

note :

$$1^* = \lambda + 1 + 11 + 111 + 1111 + \ldots$$

$$1^+ = 1 + 11 + 111 + 1111 + \ldots$$

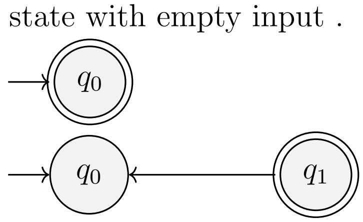
$$\Rightarrow 1^* = 1^+ \cup \lambda$$

2.1 Identities of Regular Expressions

$$
\begin{aligned}
I_1 &: \phi + R = R \\
I_2 &: \phi R + R\phi = \phi \\
I_3 &: \wedge R + R\wedge = R \\
I_4 &: \wedge^* = \wedge \ \ and \ \ \phi^* = \wedge \\
I_5 &: R + R = R \\
I_6 &: R^* R^* = R^* \\
I_7 &: RR^* = R^* R = R^+ \\
I_8 &: (R^*)^* = R^* \\
I_9 &: \wedge + RR^* = R^* = \wedge + R^* R \\
I_{10} &: (PQ)^* P = P(QP)^* \\
I_{11} &: (P+Q)^* = (P^* Q^*)^* = (P^* + Q^*)^* \\
I_{12} &: (P+Q)R = PR + QR \\
&\ and \\
&: R(P+Q) = RP + RQ
\end{aligned}
$$

2.2 difference between $\wedge$ and ϕ

you can't reach to final state when $R = \phi$ but when $R = \wedge$ you can reach at final state with empty input .

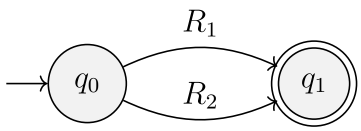


$R = \wedge$

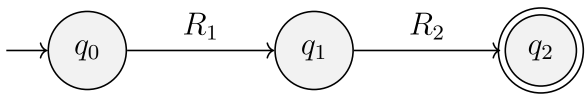
$R = \phi$

2.3 Regular Expressions and Transition Systems

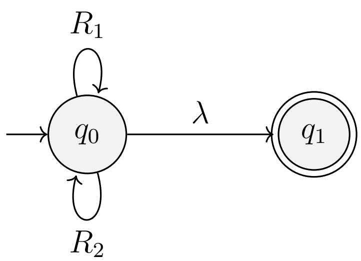
1. $(R_1 + R_2)$:



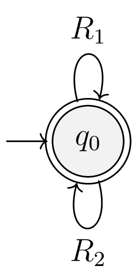
2. (R_1R_2) :



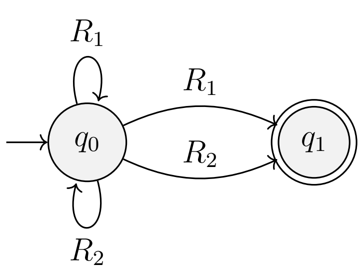
3. $(R_1 + R_2)^*$:



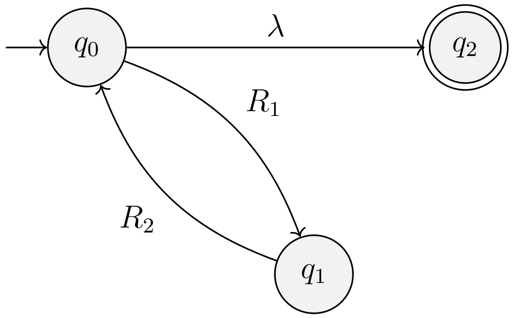
OR



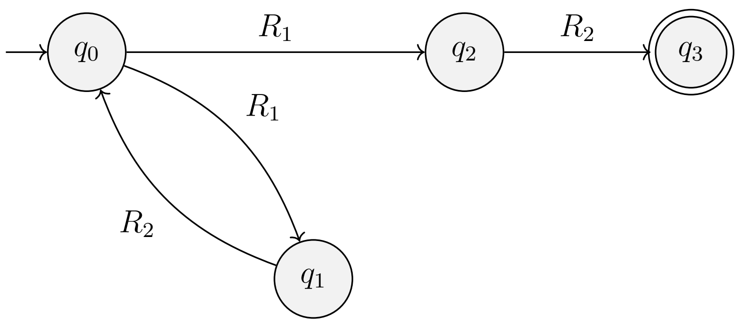
4. $(R_1 + R_2)^+$:



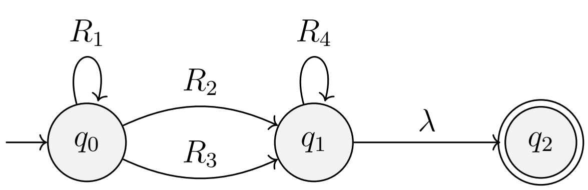
5. $(R_1R_2)^*$:



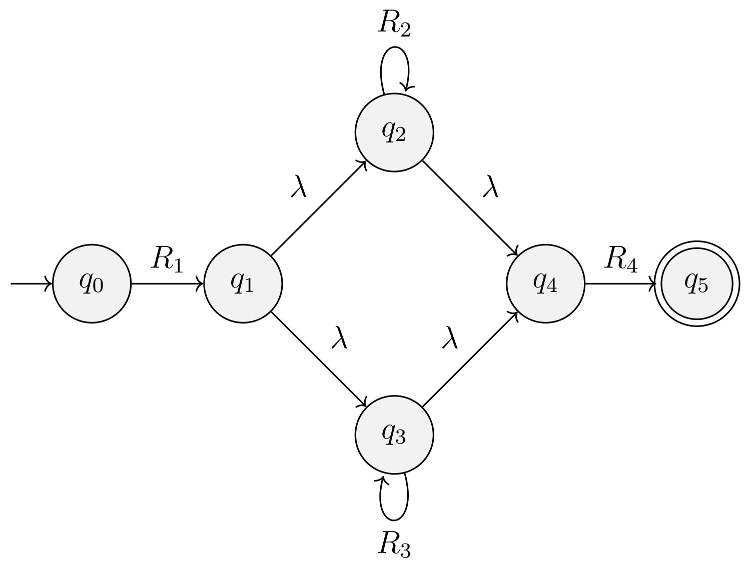
6. $(R_1R_2)^+$:



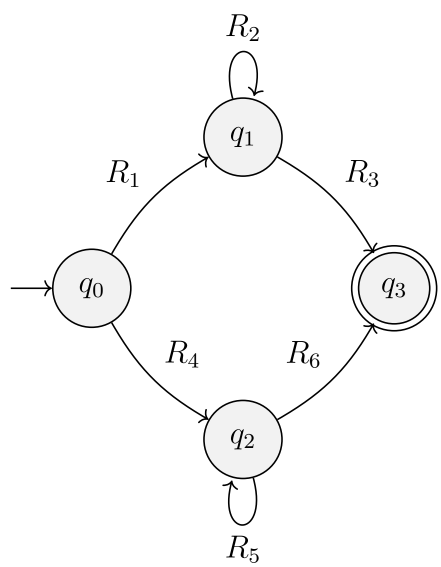
7. $R_1^*(R_2 + R_3)R_4^*$:



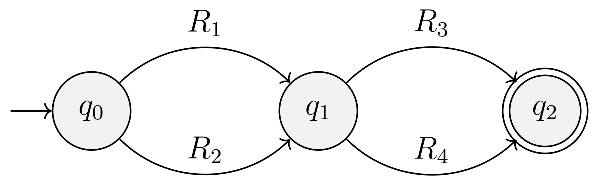
8. $R_1(R_2^* + R_3^*)R_4$:



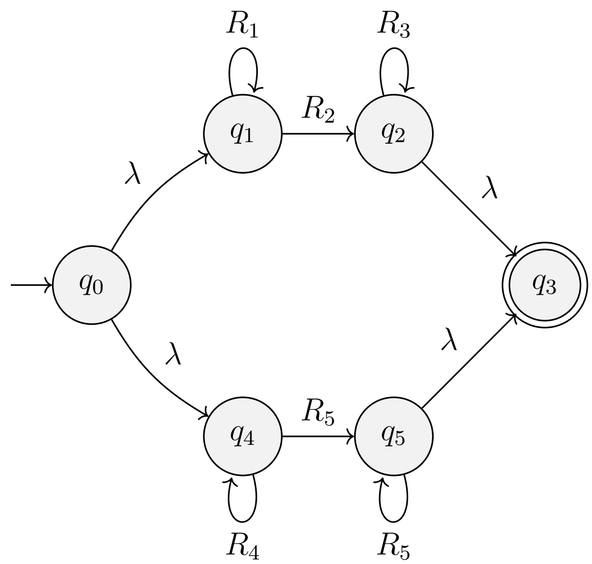
9. $R_1 R_2^* R_3 + R_4 R_5^* R_6$:



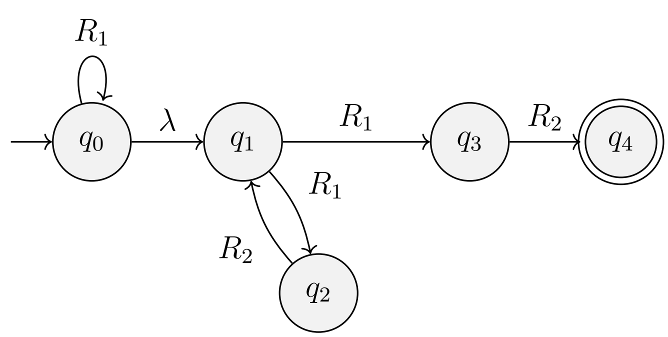
10. $(R_1 + R_2)(R_3 + R_4)$:



11. $R_1^* R_2 R_3^* + R_4^* R_5 R_6^*$:



12. $R_1^*(R_1R_2)^+$:

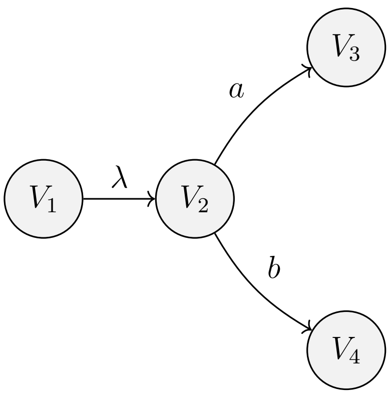


2.4 λ transition elimination

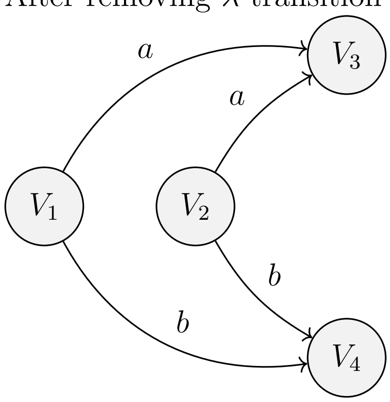
Rules :

- Find all the edges starting from V_2
- Duplicate all these edges starting from V_1 , without changing the edge labels
- If V_1 is the initial state, make V_2 also initial state
- If V_2 is the final state, make V_1 as final state

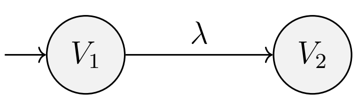
2.4.1 Example



After removing λ transition :



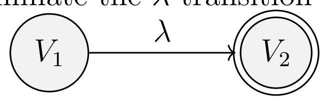
2.4.2 Example



after :

2.4.3 Example

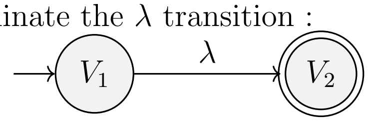
eliminate the λ transition :



after :

2.4.4 Example

eliminate the λ transition :



after :

2.4.5 Example

Simple the Regular expressions below :

$$10 + (1010)^*[\lambda^* + \underbrace{\lambda(1010)^*}_{(1010)^*}]$$

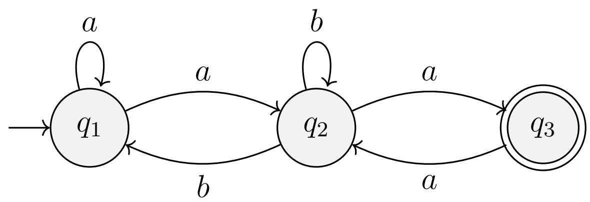
$$\rightarrow 10 + (1010)^*[\underbrace{\lambda^* + (1010)^*}_{(1010)^*}]$$

$$\rightarrow 10 + \underbrace{(1010)^*(1010)^*}_{(1010)^*}$$

$$\rightarrow 10 + (1010)^*$$

2.4.6 Example

Consider the following transition system



Find out which regular expressions can be deducted from this transition system :

$$
\begin{aligned}
q_1 &= q_1 a + q_2 b + \wedge && (2.1)\\
q_2 &= q_1 a + q_2 b + q_3 a && (2.2)\\
q_3 &= q_2 a && (2.3)
\end{aligned}
$$

using (2.1) and (2.2) we have :

$$
\begin{aligned}
q_2 &= q_1 a + q_2 b + q_2 aa\\
&= \underbrace{q_1 a}_{Q} + \underbrace{q_2}_{R} \underbrace{(b + aa)}_{P}
\end{aligned}
$$

Arden's Theorem :

$$R = Q + RP \rightarrow R = QP^*$$

and we have :

$$q_2 = q_1 a (b + aa)^*$$

$$
\begin{aligned}
\underbrace{q_1}_{R} &= q_1 a + q_1 a (b + aa)^* b + \wedge\\
&= \underbrace{q_1}_{R} \underbrace{(a + a(b + aa)^* b)}_{P} + \underbrace{\wedge}_{Q}
\end{aligned}
$$

According to Arden's Theorem :

$$\begin{aligned} q_1 &= \wedge(a + a(b + aa)^*b)^* \\ &= (a + a(b + aa)^*b)^* \end{aligned}$$

$$\begin{aligned} q_2 &= q_1 a(b + aa)^* \\ &= (a + a(b + aa)^*b)^* a(b + aa)^* \end{aligned}$$

$$\begin{aligned} q_3 &= q_2 a \\ &= (a + a(b + aa)^*b)^* a(b + aa)^* a \end{aligned}$$

2.5 Arden's Theorem

Let P and Q be two regular expressions over Σ if P does not contain $\wedge$, then the following equation in R :

$$R = Q + RP$$

has unique solution (one and only one) :

$$R = QP^*$$

Proof : put $R = QP^*$ in the $R = Q + RP$ formula

$$\begin{aligned} QP^* &= Q + (QP^*)P \\ &= Q(\wedge + \underbrace{P^*P}_{P^*}) \\ &= Q(\underbrace{\wedge + P^*}_{P^*}) \qquad\qquad = QP^* \end{aligned}$$

2.5.1 Example

Construct a Finite Automata equivalent to the regular expressoin :

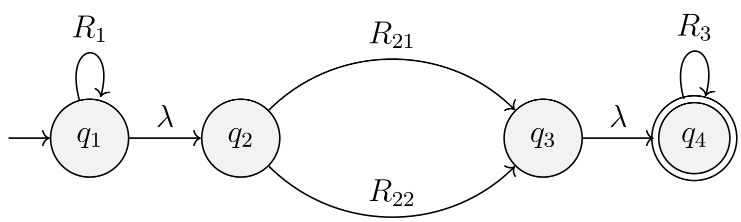
$$(0 + 1)^*(00 + 11)(0 + 1)^*$$

Suppose :

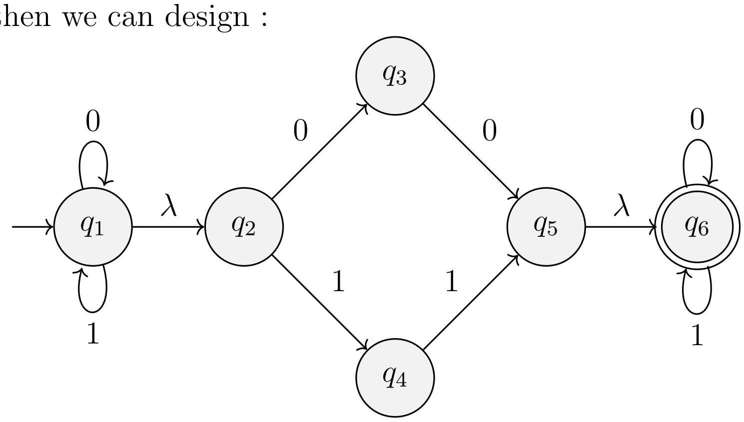
$$R_1^* R_2 R_3^*$$

$$R_1^*(R_{21} + R_{22})R_3^*$$

now we can have :



then we can design :



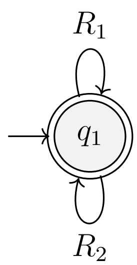
2.5.2 Example

Construct a Finite Automata equivalent to the regular expressions :

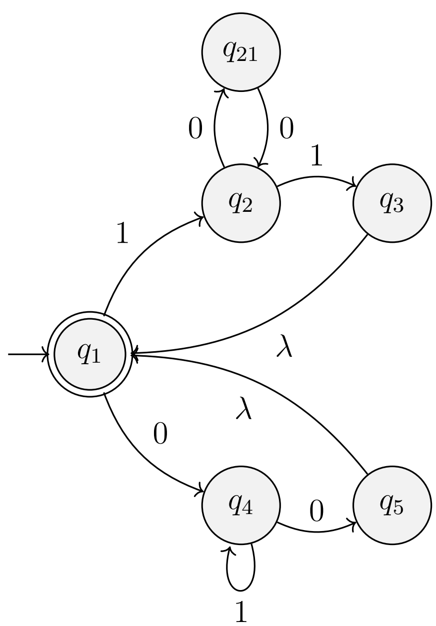
$$R = (1(00)^* + 01^*0)^*$$

Suppose we have :

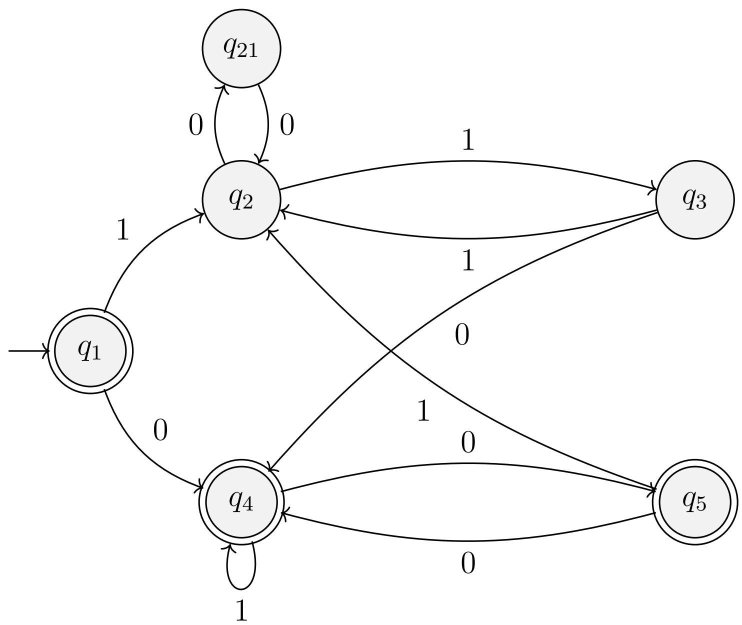
$$(R_1 + R_2)^*$$



so we have :



if we want to do $\lambda - transition$ elimination we have :



2.5.3 Example

Construct a Finite Automata equivalent to the regular expression :

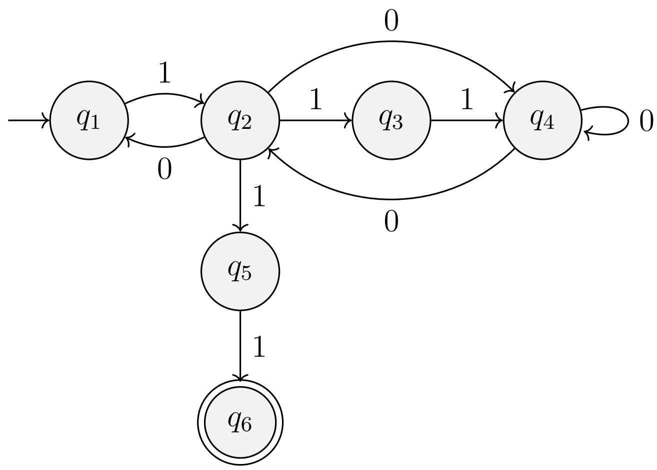
$$R = (01 + (11 + 0)1^*0)^*11$$

Suppose :

$$R_1^* R_2$$

$$(R_{11} + R_{22})^* R_2$$

so we have :



2.6 Left Most and Right Most Derivation in CFG

Definition : a derivation $A \xrightarrow{*} \omega$ is called a left most derivation if we apply a production only to the left most variable at every step.

Definition : a derivation $A \xrightarrow{*} \omega$ is called a right most derivation if we apply a production only to the right most variable at every step.

2.6.1 Example of left most derivation

$$
\begin{aligned}
A &\rightarrow X_1X_2X_3\dots X_m \\
&\xrightarrow{*} \omega_1X_2\dots X_m \\
&\xrightarrow{*} \omega_1\omega_2\dots X_m \\
&\xrightarrow{*} \omega_1\omega_2\dots \omega_m
\end{aligned}
$$

Thus :

$$A \xrightarrow[G]{*} \omega$$

2.6.2 Example of right most derivation

$$\begin{aligned} A &\rightarrow X_1X_2X_3 \ldots X_m \\ &\xrightarrow{*} X_1X_2 \ldots \omega_m \\ &\xrightarrow{*} X_1\omega_2 \ldots \omega_m \\ &\xrightarrow{*} \omega_1\omega_2 \ldots \omega_m \end{aligned}$$

Thus :

$$A \xrightarrow[G]{*} \omega$$

2.6.3 Exercise

Consider the following Grammar :

$$\begin{aligned} &S \rightarrow aAS \\ &S \rightarrow a \\ &A \rightarrow SbA \\ &A \rightarrow SS \\ &A \rightarrow ba \end{aligned}$$

for input string "aabbaa" find :

- left most derivation
- right most derivation
- derivation tree

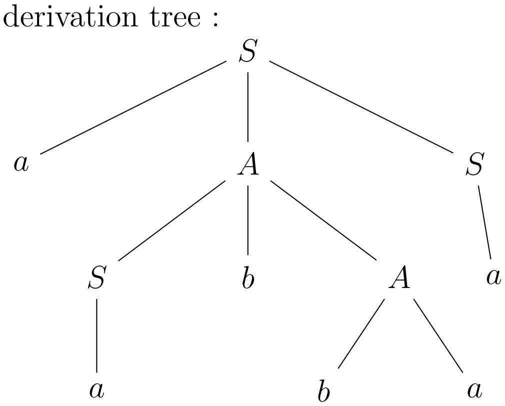
Answer :
left most derivation :

$$
\begin{aligned}
S &\rightarrow aAS \\
&\rightarrow aSbAS \\
&\rightarrow aabAS \\
&\rightarrow aabbaS \\
&\rightarrow aabbaa
\end{aligned}
$$

right most derivation :

$$
\begin{aligned}
S &\rightarrow aAS \\
&\rightarrow aAa \\
&\rightarrow aSbAa \\
&\rightarrow aSbbaa \\
&\rightarrow aabbaa
\end{aligned}
$$

derivation tree :



2.7 Left Linear Grammar

Left Linear Grammar :

In a Grammar if all productions are in form $A \rightarrow B\alpha$ of $A \rightarrow \alpha$ where $A, B \in V$ and $\alpha \in \Sigma^*$, then the gammar is called left linear grammar .

Example :

$$A \rightarrow Aa|Bb|b$$

2.8 Right Linear Grammar

Right Linear Grammar :

In a Grammar if all productions are in form $A \rightarrow \alpha B$ of $A \rightarrow \alpha$ where $A, B \in V$ and $\alpha \in \Sigma^*$, then the gammar is called right linear grammar .

Example :

$$A \rightarrow aA|bB|b$$

2.9 Ambiguity in CFG

Definition : a terminal sting $\omega \in L(G)$ is ambiguous if there exists two or more left most derivation of ω .

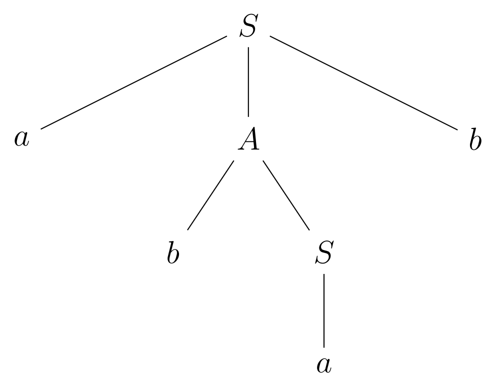
a CFG called G is ambiguous if there exists some $\omega \in L(G)$ which is ambiguous .

Example : show the grammar below is ambiguous ?

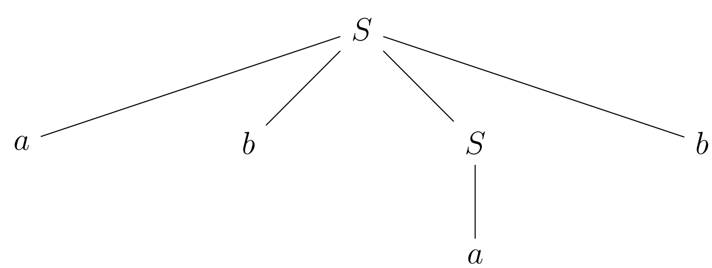
$$S \rightarrow a$$
$$S \rightarrow abSb$$
$$S \rightarrow aAb$$
$$A \rightarrow bS$$
$$A \rightarrow aAAb$$

Answer : you can reach the string "abab" with two different parse tree's so the grammar is ambiguous

$$S \to aAb$$
$$S \to abSb$$
$$S \to abab$$



$$S \to abSb$$
$$S \to abab$$



2.10 CFG Examples

Let $M = (Q, \Sigma, \delta, S, F)$ be the Finite State Machine, where :

$$\begin{array}{ll} Q = \{A, B\} & \delta(A, a) = A \\ \Sigma = \{a, b\} & \delta(A, b) = B \\ S = A & \delta(B, a) = B \\ F = \{B\} & \delta(B, b) = A \end{array}$$

design a grammar to generate the language accepted by M can be specified as $G = (V, \Sigma, S, P)$ where $V = Q \cup \Sigma$ and $S = A$, built the Grammar L(G) = L(M) ?

Answer :

$$\begin{array}{ccc} \delta(A, a) = A & \Rightarrow & A \to aA \\ \delta(A, b) = B & \Rightarrow & A \to bB \\ \delta(B, a) = B & \Rightarrow & B \to aB \\ \delta(B, b) = A & \Rightarrow & B \to bA \end{array}$$

B is initial state $\Rightarrow B \to \wedge$

$$\Rightarrow P = \{A \to aA, A \to bB, B \to aB, B \to bA, B \to \wedge\}$$

2.11 Simplification of Context-Free-Grammar

In a CFG G, it may not be necessary to use all the symbols in $V \cap \Sigma$, or all the scentences in P for deriving scentences .

Sample : Consider the grammar

$$G = (\{S, A, B, C, E\}, \{a, b, d\}, S, P)$$

where,

$$P = \{S \to AB, A \to a, B \to b, B \to C, E \to d|\lambda\}$$

- C does not derive any terminal string
- E and d do not appear in any result
- $E \to \wedge$ is a null production
- $B \to C$ simply replace B by C

2.12 Construction of Reduced Grammar - Procedure1

Theorem : If G is a CFG such that $L(G) \neq \phi$, we can find an equivalent grammar G', such that each variable in G' derives some terminal string where $G = (V, \Sigma, S, P)$ and $G' = (V', \Sigma, S, P')$

2.12.1 step-1 : Construction of V'

$\omega_1 = \{\ A \in V|$ there exists a production $A \rightarrow \omega$ where $\omega \in \Sigma^*\ \}$

$\omega_{i+1} = \omega_i \cup \{\ A \in V|$ there exists some production $A \rightarrow \alpha$ with $\alpha \in (\Sigma \cup \omega_i)^*\ \}$

$\omega_i \subseteq \omega_{i+1} for all i.$

2.12.2 step-2 : Construction of P'

$P' = \{\ A \rightarrow \alpha|$ A, $\alpha \in (V' \cup \Sigma)^*\ \}$

2.12.3 step-3

for each $A \in V'$, then $A \xrightarrow[G]{*} \omega; \omega \in \Sigma^*$,

for each $A \xrightarrow[G]{*} \omega$, then $A \in V'$

$$L(G') = L(G)$$

2.12.4 Example

Let $G = (V, \Sigma, S, P)$ be given by the productions :

$$\begin{aligned}
&S \rightarrow AB \\
&A \rightarrow a \\
&B \rightarrow b \\
&B \rightarrow C \\
&E \rightarrow d
\end{aligned}$$

Find G' derives some terminal string
Construction of V' :
$\omega_1 = \{$ A, B , E $\}$ since :

$$\begin{aligned} A &\to a \\ B &\to b \\ E &\to d \end{aligned}$$

$$\begin{aligned} \omega_2 &= \omega_1 \cup \{A_1 \in V | A \to \alpha; for \alpha \in (\Sigma \cup \{A, B, E\})^*\} \\ &= \omega_1 \cup \{S\} \\ &= \{A, B, E, S\} \end{aligned}$$

$$\begin{aligned} \omega_3 &= \omega_2 \cup \phi \\ &= \omega_2 \end{aligned}$$

$$\Rightarrow V' = \{A, B, E, S\}$$

Construction of P' :

$$\begin{aligned} P' &= \{A_1, \alpha \in (V' \cup \Sigma)^*\} \\ &= \{S \to AB, A \to \alpha, B \to b, E \to d\} \end{aligned}$$

$$G' = (\{S, A, B, C\}, \{a, b, c\}, S, P')$$

2.12.5 Example

Let $G = (V, \Sigma, S, P)$ be given by the productions :

$$
\begin{aligned}
S &\to AB \\
A &\to CA \\
B &\to BC \\
B &\to AB \\
A &\to a \\
C &\to aB \\
C &\to b
\end{aligned}
$$

Answer :
note : ω_1 is a subset that directly derives terminal string

$$\omega_1 = \{A, C\}$$

note : ω_2 is a subset that directly derives ω_1

$$
\begin{aligned}
\omega_2 &= \omega_1 \cup \{S\} \\
&= \{S, A, C\}
\end{aligned}
$$

$$
\begin{aligned}
\omega_3 &= \omega_2 \cup \phi = \omega_2 \\
&= \{S, A, C\}
\end{aligned}
$$

Thus :

$$\Rightarrow V' = \{S, A, C\}$$

and

$$\Rightarrow P' = \{S \to CA, A \to a, C \to b\}$$

2.13 Construction of Reduced Grammar : Procedure-2

Theorem : For every CFG with Grammar $G = (V, \Sigma, S, P)$, we can construct an equivalent Grammar $G' = (V', \Sigma', S, P')$ such that every symbol in $V' \cup \Sigma'$ appears in some result .

Method : We construct $G' = (V', \Sigma', S, P')$ as follows :

a) Construction of ω_i for $i \geq 1$

$$\begin{aligned} \omega_i &= \{S\} \\ \omega_{i+1} &= \omega_i \cup \{X \in V \cup \Sigma\} \end{aligned}$$

$$\omega_i \subseteq V \cup \Sigma$$

$$\omega_i \subseteq \omega_{i+1}$$

b) Construction of V', Σ', P'

$$\begin{aligned} V' &= V \cap \omega_k \\ \Sigma' &= \Sigma \cap \omega_k \\ P' &= \{A \rightarrow \alpha | A \in \omega_k\} \end{aligned}$$

2.13.1 Example

Let $G = (\{S, A, B, E\}, \{a, b, c\}, S, P)$ where P consists of :

$$\begin{aligned} S &\rightarrow AB \\ A &\rightarrow a \\ B &\rightarrow b \\ E &\rightarrow d \end{aligned}$$

$\omega_1 = \{S\}$
$\omega_2 = \{S\} \cup \{X \in V \cup \Sigma | there\ exists\ a\ production\ A \to \alpha\ with\ A \in \omega_i and\ \alpha\ containing\ X$
$= \{S\} \cup \{A, B\}$
$\omega_3 = \{S, A, B\} \cup \{a, b\}$
$\omega_4 = \omega_3$

so :

$$V' = \{S, A, B\}$$
$$\Sigma' = \{a, b\}$$
$$P' = \{S \to AB, A \to a, B \to b\}$$

Thus the reduced Grammar is :

$$G' = (V', \Sigma', S, P')$$

2.14 Construction of Reduced Grammar : Combining Precedure 1 and 2

Theorem : For every CFG, G there exists a reduced grammar G' which is equivalent to G .

Method : We construct that reduced grammar in two steps .

step 1 : We construct a grammar G, equivalent to the grammar G, so that every variable in G, derives some terminal strings . (i.e : the theorem steps mentioned in procedure 1)

step 2 : We construct a grammar

$$G' = (V', \Sigma', S, P')$$

equivalent to G, so that every symbol in G' appears in some scentence form of G' . (i.e : the theorem steps mentioned in procedure 2)

2.14.1 Example

Construct a reduced grammar equivalent to grammar as mentioned below :

$$
\begin{aligned}
S &\rightarrow aAa \\
A &\rightarrow Sb \\
A &\rightarrow bCC \\
A &\rightarrow DaA \\
C &\rightarrow abb \\
C &\rightarrow DD \\
E &\rightarrow aC \\
D &\rightarrow aDA
\end{aligned}
$$

Answer :

$$
\begin{aligned}
\omega_1 &= \{C\} & & C \rightarrow abb \\
\omega_2 &= \{C\} \cup \{A, E\} & & E \rightarrow aC \quad A \rightarrow bCC \\
\omega_3 &= \{A, E, C\} \cup \{S\} & & S \rightarrow aAa \\
\omega_4 &= \omega_3 \cup \phi \\
&= \{S, A, C, E\}
\end{aligned}
$$

so :

$$
P' = \{S \rightarrow aAa, A \rightarrow Sb, A \rightarrow bCC, C \rightarrow abb, E \rightarrow aC\}
$$

note : for the second step considered P' only .

$$
\begin{aligned}
\omega_1 &= \{S\} \\
\omega_2 &= \{S\} \cup \{a, A\} \\
\omega_3 &= \{S, A, a\} \cup \{b, C\} \\
\omega_4 &= \{S, A, C, a, b\}
\end{aligned}
$$

$\Rightarrow$

$$P'' = \{S \to aAa, A \to Sb, A \to bCC, C \to abb\}$$

so reduced Grammar is :

$$G = (\{S, A, C\}, \{a, b\}, P'', S)$$

2.15 Chomsky Normal Form (CNF)

Definition : a CFG is in CNF, if every production is of the form $A \to a$ or $A \to BC$ and $S \to \wedge$ is in G, if $\wedge \in L(G)$, we assume that S does not appear on the Right Hand Side of any production .

Example :

If $S \to AB|\wedge$, $A \to a$, $B \to b$ is in G, then G is in CNF

Theorem : for every CFG, there is an equivalent grammar G' in CNF .

2.15.1 How to make some Grammar to CNF

step1: Elimination of null and unit production .

step2 : Elimination of terminals on the Right Hand Side.

step3 : Restricting the number of variables on the Right Hand Side

2.15.2 Example

Reduce the following grammar to CNF :

G is :

$$S \to aAD$$
$$A \to aB|bAB$$
$$B \to b$$
$$E \to d$$

Answer :

step1 : There is no null and unit production. step2 : let's create the productions according to chomsky normal form :

$$
\begin{aligned}
S &\rightarrow aAD & & \\
C_a &\rightarrow a & & S \rightarrow C_aAD \\
A &\rightarrow aB & & \\
A &\rightarrow C_aB & & \\
A &\rightarrow bAB & & \\
C_b &\rightarrow b & & A \rightarrow C_bAB \\
B &\rightarrow b & & \\
E &\rightarrow d & &
\end{aligned}
$$

so we have :

$V' = \{S, A, B, E, C_a, C_b\}$
$P_1 = \{S \rightarrow C_aAD, A \rightarrow C_aB, A \rightarrow C_bAB, B \rightarrow b, E \rightarrow d, C_a \rightarrow a, C_b \rightarrow b\}$

step3 :

$$
\begin{aligned}
S &\rightarrow C_aAD \\
S &\rightarrow C_aC_1 \\
C_1 &\rightarrow AD \\
A &\rightarrow C_bAB \\
A &\rightarrow C_bC_2 \\
C_2 &\rightarrow AB
\end{aligned}
$$

2.15.3 Example on CNF

Reduce the following Grammar to CNF

G is :

$$\begin{aligned}
S &\rightarrow aAbB \\
A &\rightarrow aA \\
A &\rightarrow a \\
B &\rightarrow bB \\
B &\rightarrow b
\end{aligned}$$

step2 :

$$\begin{array}{lll}
S \rightarrow aAbB & C_a \rightarrow a & C_b \rightarrow b \\
S \rightarrow C_aAC_bB & & \\
A \rightarrow aA & A \rightarrow C_aA & \\
B \rightarrow bB & B \rightarrow C_bB &
\end{array}$$

step3 :

$$\begin{aligned}
S &\rightarrow C_aAC_bB \\
S &\rightarrow C_aC_1 \\
C_1 &\rightarrow AC_bB \\
C_1 &\rightarrow AC_2 \\
C_2 &\rightarrow C_bB
\end{aligned}$$

so G is :

$$\begin{aligned}
V &= \{A, B, S, C_a, C_b, C_1, C_2\} \\
\Sigma &= \{a, b\} \\
S &= startstate
\end{aligned}$$

and P is : P = {

$$
\begin{aligned}
A &\rightarrow a \\
B &\rightarrow b \\
C_a &\rightarrow a \\
C_b &\rightarrow b \\
A &\rightarrow C_a A \\
B &\rightarrow C_b B \\
S &\rightarrow C_a C_1 \\
C_1 &\rightarrow A C_b B \\
C_1 &\rightarrow A C_2 \\
C_b &\rightarrow C_b B
\end{aligned}
$$

}

2.16 Greibach Normal Form

a CFG is in GNF if every production is of form :

$$A \rightarrow a\alpha$$

where $\alpha \in V^*$ and $\alpha \in \Sigma$, $S \rightarrow \wedge$ is allowed in G, if $\wedge \in L(G)$, we assume that S does not appear on the Right Hand Side of any production .

Example :

a grammar G with following production rules is in GNF :

$$
\begin{aligned}
S &\rightarrow b \\
S &\rightarrow aBC \\
S &\rightarrow \wedge \\
B &\rightarrow bBC \\
C &\rightarrow c \\
B &\rightarrow b
\end{aligned}
$$

Chapter 3

Push Down Automata

3.1 Definition of Push Down Automata

Definition : a PDA is a 7-tuple $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ where :

- a finite non-empty set of states denoted by Q
- a finite non-empty set of input symbols denoted by Σ
- a finite non-empty set of push down symbols denoted by Γ
- a special state q_0, called initial state, where $q_0 \in Q$
- a special push down symbols called initial symbol on the push down store denoted by Z_0
- the set of final states, a subset of Q denoted by F
- the transition function δ from $Q \times (\Sigma \times \{\lambda\}) \times \Gamma$ to the set if finite subsets of $Q \times \Gamma^*$

$$Q \times (\Sigma \times \{\lambda\}) \times \Gamma \rightarrow Q \times \Gamma^*$$

3.1.1 Example

Design a PDA

$$A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

which accepts input strings over "a" & "b", but the input string should contain even number of a's .

Sample : abbaabab

Answer :

$$\delta = \begin{cases} \delta(\overbrace{q_0}^{state}, \overbrace{a}^{input}, \overbrace{Z_0}^{stack-top}) = (q_0, aZ_0) \\ \delta(q_0, a, a) = (q_0, \wedge) \\ \delta(q_0, b, a) = (q_0, a) \\ \delta(q_0, b, Z_0) = (q_0, Z_0) \\ \delta(q_0, \wedge, Z_0) = (q_f, Z_0) \end{cases}$$

note : reaching to final state means the string is accepted

so :

$$Q = \{q_0, q_f\}$$
$$\Sigma = \{a, b\}$$
$$\Gamma = \{Z_0, a\}$$
$$F = \{q_f\}$$

3.2 PDA : Accepting of a string

- Acceptance by Final state
- Acceptance by Null Store

3.2.1 Definition 1 - Acceptance by Final state

Let

$$A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

be a PDA, The set accepted by PDA by final state is defined by

$$T(A) = \{\omega \in \Sigma^* | (q_0, \omega, Z_0) \vdash (q_f, \wedge, \alpha) for\ some\ q_f \in F\ \ and\ \ \alpha \in \tau^*\}$$

3.2.2 Definition 2 - Acceptance by Null Store

Let

$$A = (Q, \Sigma, \tau, \delta, q_0, Z_0, F)$$

be a PDA, The set accepted by PDA by null state (or empty store) is defined by

$$N(A) = \{\omega \in \Sigma^* | (q_0, \omega, Z_0) \vdash^* (q, \wedge, \wedge) for\ some\ q \in Q\}$$

3.3 PDA to Context-Free-Grammars

Theorem : if L is a CFL, then we can construct a PDA A accepting L by empty store, i.e : N(A)

Method : Let $L = L(G)$, where $G = (V, \Sigma, S, P)$ is a Context-Free-Grammar, we construct a PDA A as

$$A = (\{q\}, \Sigma, V \cup \Sigma, \delta, q, S, \phi)$$

where δ is defined by the following rules :

$$R_1 : \delta(q, \wedge, A) = \{(q, \alpha) | A \to \alpha\ is\ in\ P\}$$
$$R_2 : \delta(q_1, a, a) = \{(q, \wedge) | for\ every\ a\ in\ \Sigma\}$$

3.3.1 Example

Construct a PDA which is equivalent to the following CFG :

$$S \to 0CC$$
$$C \to 0S$$
$$C \to 1S$$
$$C \to 0$$

test whether 010*4 is accepted by N(A) ?

Solution :

δ is defined by the following rules :

$$
\begin{aligned}
R_1 : \delta(q, \wedge, S) &= \{(q, 0CC)\} \\
\delta(q, \wedge, C) &= \{(q, 0S), (q, 1S), (q, 0)\} \\
R_2 : \delta(q, 1, 1) &= \{(q, \wedge)\} \\
\delta(q, 0, 0) &= \{(q, \wedge)\}
\end{aligned}
$$

so :

$$
\begin{aligned}
(q, 010^4, S) &\vdash (q, 010^4, 0CC) \\
&\vdash (q, 10^4, CC) \\
&\vdash (q, 10^4, 1SC) \\
&\vdash (q, 0^4, SC) \\
&\vdash (q, 0^4, 0CCC) \\
&\vdash (q, 0^3, CCC) \\
&\vdash (q, 0^3, 0CC) \\
&\vdash (q, 0^2, CC) \\
&\vdash (q, 0^2, 0C) \\
&\vdash (q, 0, C) \\
&\vdash (q, 0, 0) \\
&\vdash (q, \wedge, \wedge)
\end{aligned}
$$

So $010^4 \in N(A)$

3.3.2 Example

Design a PDA which accepts

$$T(A) = \{\omega \in \omega^T | where\ \omega \in (a, b)^+\}$$

sample :

$$\underbrace{abb}_{\omega} c \underbrace{bba}_{\omega^T}$$

Answer :

$$\delta = \begin{cases} \delta(q_0, a, Z_0) = (q_0, aZ_0) \\ \delta(q_0, b, Z_0) = (q_0, bZ_0) \\ \delta(q_0, a, a) = (q_0, aa) \\ \delta(q_0, a, b) = (q_0, ab) \\ \delta(q_0, b, a) = (q_0, ba) \\ \delta(q_0, b, b) = (q_0, bb) \\ \delta(q_0, c, a) = (q_1, a) \\ \delta(q_0, c, b) = (q_1, b) \\ \delta(q_1, a, a) = (q_1, \wedge) \\ \delta(q_1, b, b) = (q_1, \wedge) \\ \delta(q_1, \wedge, Z_0) = (q_f, Z_0) \end{cases}$$

3.3.3 Example

Design a PDA which accepts

$$T(A) = \{a^n b^n | n > 0\}$$

sample : if n = 3 $\rightarrow$ ω = aaabbb
Answer :

$$\delta = \begin{cases} \delta(q_0, a, Z_0) = (q_0, aZ_0) \\ \delta(q_0, a, a) = (q_0, aa) \\ \delta(q_0, b, a) = (q_1, \wedge) \\ \delta(q_1, b, a) = (q_1, \wedge) \\ \delta(q_0, \wedge, Z_0) = (q_f, Z_0) \end{cases}$$

3.3.4 Example

Design a PDA which accepts

$$N(A) = \{a^n b^m a^n | n, m \geq 1\}$$

note : N(A) means Null-Terminating .

sample : $\underbrace{aaa}_{q_0}\underbrace{bb}_{q_1}\underbrace{aaa}_{q_2}$

Answer :

$$\delta = \begin{cases} \delta(q_0, a, Z_0) = (q_0, aZ_0) \\ \delta(q_0, a, a) = (q_0, aa) \\ \delta(q_0, b, a) = (q_1, a) \\ \delta(q_1, b, a) = (q_1, a) \\ \delta(q_1, a, a) = (q_2, \wedge) \\ \delta(q_2, a, a) = (q_2, \wedge) \\ \delta(q_2, \wedge, Z_0) = (q_2, \wedge) \end{cases}$$

3.3.5 Example

Design a PDA which accepts

$$N(A) = \{a^n b^{2n} | n \geq 1\}$$

note : for every a we should push two a at top of the stack .

Answer :

$$\delta = \begin{cases} \delta(q_0, a, Z_0) = (q_0, aaZ_0) \\ \delta(q_0, a, a) = (q_0, aaa) \\ \delta(q_0, b, a) = (q_1, \wedge) \\ \delta(q_1, b, a) = (q_1, \wedge) \\ \delta(q_1, \wedge, Z_0) = (q_1, \wedge) \end{cases}$$

3.3.6 Example

Design a PDA which accepts

$$N(A) = \{a^m b^m c^n | m, n \geq 1\}$$

Answer :

$$\delta = \begin{cases} \delta(q_0, a, Z_0) = (q_0, aZ_0) \\ \delta(q_0, a, a) = (q_0, aa) \\ \delta(q_0, b, a) = (q_1, \wedge) \\ \delta(q_1, b, a) = (q_1, \wedge) \\ \delta(q_1, \wedge, Z_0) = (q_1, \wedge) \end{cases}$$

note : we don't care how many times we see 'c' .

3.3.7 Example

Design a PDA which accepts

$$N(A) = \{a^m b^n | m > n \geq 1\}$$

note : because $m > n$, the stack should remain 'a' at top of the stack
Answer :

$$\delta = \begin{cases} \delta(q_0, a, Z_0) = (q_0, aZ_0) \\ \delta(q_0, a, a) = (q_0, aa) \\ \delta(q_0, b, a) = (q_1, \wedge) \\ \delta(q_1, b, a) = (q_1, \wedge) \\ \delta(q_1, \wedge, a) = (q_2, \wedge) \\ \delta(q_2, \wedge, a) = (q_2, \wedge) \\ \delta(q_2, \wedge, Z_0) = (q_2, \wedge) \end{cases}$$

3.4 LL(K) Grammar

suppose LL(1) Grammar :
First L $\xrightarrow{means}$ Reading input string from left to right
Second L $\xrightarrow{means}$ Left Most Derivation
1 $\xrightarrow{means}$ looking ahead terminal symbols in the input string

Chapter 4

Turing Machine

4.1 Turing Machine

Definition : a Turing Machine, M is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, b, F)$ where :

- Q is a finite non-empty set of states.
- Γ is a finite non-empty set of tape symbols.
- $b \in \Gamma$ is the blank.
- Σ is a finite non-empty set of input symbols . Σ is a subset of Γ and $b \notin \Sigma$.
- δ is the transition function mapping the states of finite automaton and tape symbols and movement of the head .

  i.e : $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
- $q_0 \in Q$ is the initial state.
- $F \subseteq Q$ is the set of final states.

* The acceptability of a string is decided by the reachability from the initial state to same final state, so final states are also called as accepting states .

* δ may not be defined for some elements of $Q \times \Gamma$

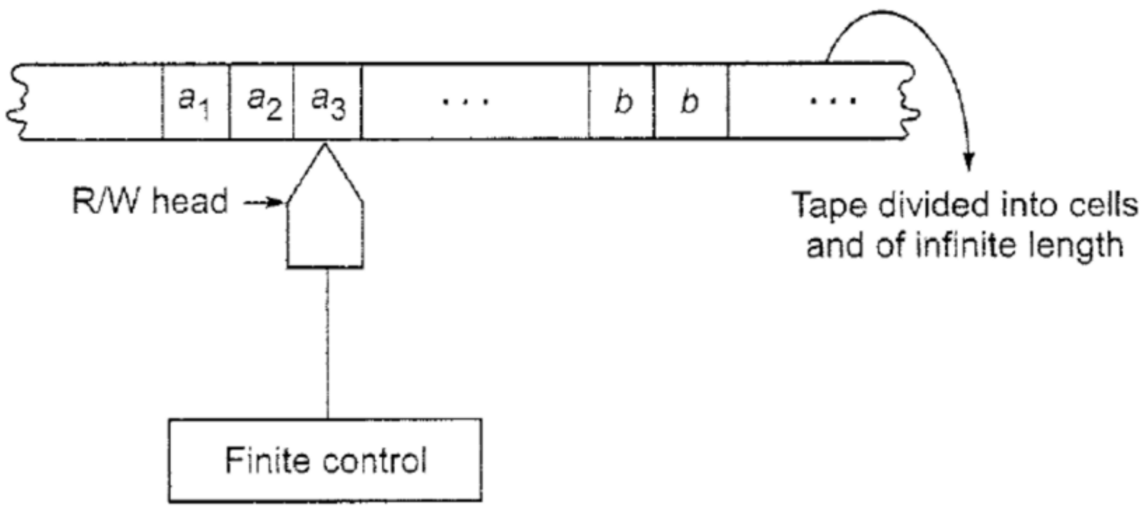


Fig. 9.1 Turing machine model.

* Tape devided into cells containing tape symbols .
* Head can move left or right .

4.2 Turing Maching : Instanteneous Description (ID)

Definition : ID of a Turing Machine, is a snapshot of TM to describe the current situation of the TM .

4.2.1 Transitions

let the initial ID of a TM is

$$x_1x_2\dots x_{i-1}\underbrace{x_i}_{q}x_{i+1}\dots x_n$$

So :

$$x_1x_2\dots x_{i-1}\underbrace{x_i}_{q}x_{i+1}\dots x_n \xrightarrow{\delta(q,x_i)=(p,y,L)} x_1x_2\dots \underbrace{x_{i-1}}_{p}yx_{i+1}\dots x_n$$

$$x_1x_2\dots x_{i-1}\underbrace{x_i}_{q}x_{i+1}\dots x_n \xrightarrow{\delta(q,x_i)=(p,y,R)} x_1x_2\dots x_{i-1}y\underbrace{x_{i+1}}_{p}\dots x_n$$

4.3 TM : Aceptance through transition Table

	0	1	x	y	b
$\rightarrow q_1$	xRq_2	$-$	$-$	$-$	bRq_5
q_2	$0Rq_2$	yLq_3	$-$	yRq_2	$-$
q_3	$0Lq_4$	$-$	xRq_5	yLq_3	$-$
q_4	$0Lq_4$	$-$	xRq_1	$-$	$-$
q_5	$-$	$-$	$-$	yRq_5	bRq_6
$(f)q_6$	$-$	$-$	$-$	$-$	$-$

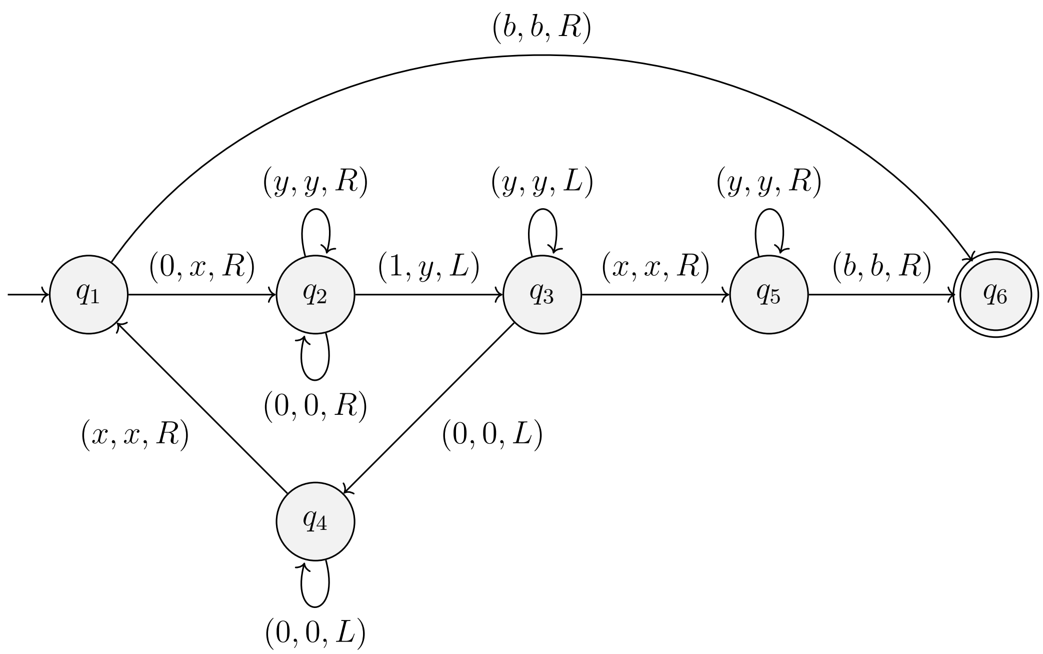
check wether input string 0011 is accepted or not by the given turing machine shown above ?

Answer :

$$
\begin{aligned}
\underbrace{0}_{q_1}011 &\vdash x\underbrace{0}_{q_2}11 \\
&\vdash x0\underbrace{1}_{q_2}1 \\
&\vdash x\underbrace{0}_{q_3}y1 \\
&\vdash \underbrace{x}_{q_4}0y1 \\
&\vdash x\underbrace{0}_{q_1}y1 \\
&\vdash xx\underbrace{y}_{q_2}1 \\
&\vdash xxy\underbrace{1}_{q_2} \\
&\vdash xx\underbrace{y}_{q_3}y \\
&\vdash x\underbrace{x}_{q_3}yy \\
&\vdash xx\underbrace{y}_{q_5}y \\
&\vdash xxy\underbrace{y}_{q_5} \\
&\vdash xxyy\underbrace{b}_{q_5} \\
&\vdash xxyyb\underbrace{b}_{q_6}
\end{aligned}
$$

* q_6 is the final state so the string is accepted .

4.4 TM : Acceptance through transition system



check the above TM accepts input string 0011 or not ?

Answer :

$$
\begin{aligned}
\underbrace{0}_{q_1}011 &\xrightarrow{(0,x,R)} x\underbrace{0}_{q_2}11 \\
&\xrightarrow{(0,0,R)} x0\underbrace{1}_{q_2}1 \\
&\xrightarrow{(1,y,L)} x\underbrace{0}_{q_3}y1 \\
&\xrightarrow{(0,0,L)} \underbrace{x}_{q_4}0y1 \\
&\xrightarrow{(x,x,R)} x\underbrace{0}_{q_1}y1 \\
&\xrightarrow{(0,x,R)} xx\underbrace{y}_{q_2}1 \\
&\xrightarrow{(y,y,R)} xxy\underbrace{1}_{q_2} \\
&\xrightarrow{(1,y,L)} xx\underbrace{y}_{q_3}y \\
&\xrightarrow{(y,y,L)} x\underbrace{x}_{q_3}yy \\
&\xrightarrow{(x,x,R)} xx\underbrace{y}_{q_5}y \\
&\xrightarrow{(y,y,R)} xxy\underbrace{y}_{q_5} \\
&\xrightarrow{(y,y,R)} xxyy\underbrace{b}_{q_5} \\
&\xrightarrow{(b,b,R)} xxyyb\underbrace{b}_{q_6}
\end{aligned}
$$

* q_6 is the final state so the TM accepts the string .

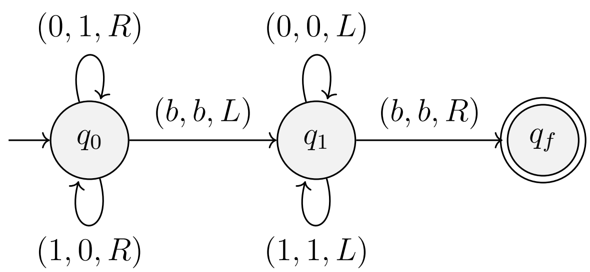
4.4.1 Example

TM : 1's complement of a binary number
result :

$$b01101b \xrightarrow[TM]{*} b10010b$$

	Tape Symbols		
	0	1	b
$\rightarrow q_0$	$1Rq_0$	$0Rq_0$	bLq_1
q_1	$0Lq_1$	$1Lq_1$	bRq_f
q_f	$-$	$-$	$-$

* The Machine HALT at q_f



4.5 Example

TM : Even number of 0's and Odd number of 1's .

EE $\xrightarrow{means}$ Even number of 0's and Even number of 1's
OE $\xrightarrow{means}$ Odd number of 0's and Even number of 1's
OO $\xrightarrow{means}$ Odd number of 0's and Odd number of 1's
EO $\xrightarrow{means}$ Even number of 0's and Odd number of 1's

$$EE \rightarrow q_0$$
$$OE \rightarrow q_1$$
$$OO \rightarrow q_2$$
$$EO \rightarrow q_f$$

		Tape Symbols		
		0	1	b
EE	$\rightarrow q_0$	$0Rq_1$	$1Rq_f$	$-$
OE	q_1	$0Rq_0$	$1Rq_2$	$-$
OO	q_2	$0Rq_f$	$1Rq_1$	$-$
EO	q_f	$0Rq_2$	$1Rq_0$	$-$