2 UNIT

Regular Expressions and Languages

CONTENTS

Part-1	:	Regular Expressions, Transition Graph	**2–2A to 2–4A**
Part-2	:	Kleene's Theorem	**2–5A to 2–9A**
Part-3	:	Finite Automata and Regular Expression : Arden's Theorem, Algebraic Method Using Arden's Theorem	**2–9A to 2–14A**
Part-4	:	Regular and Non-Regular Languages : Closure Properties of Regular Languages	**2–14A to 2–17A**
Part-5	:	Pigeonhole Principle, Pumping Lemma, Application of Pumping Lemma, Decidability: Decision Properties, Finite Automata and Regular Languages, Regular Language and Computers, Simulation of Transition Graph and Regular Language	**2–17A to 2–23A**

PART-1

Regular Expressions, Transition Graph.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 2.1. **What do you mean by regular expression ?**

OR

Write the formal recursive definition of a regular expression.

Answer

1. Regular expression is a formula in a special language that is used for specifying simple classes of strings.
2. A string is a sequence of symbols; for the purpose of most text-based search techniques, a string is any sequence of alphanumeric characters (letters, numbers, spaces, tabs, and punctuation).

Formal recursive definition of regular expression :

Formally, a regular expression is an algebraic notation for characterizing a set of strings.

1. Any terminals, *i.e.*, the symbols belong to S are regular expression. Null string ($\wedge$ or ε) and null set (ϕ) are also regular expression.
2. If P and Q are two regular expressions then the union of the two regular expressions denoted by $P + Q$ is also a regular expression.
3. If P and Q are two regular expressions then their concatenation denoted by PQ is also a regular expression.
4. If P is a regular expression then the iteration (repetition or closure) denoted by P^* is also a regular expression.
5. If P is a regular expression then (P) is a regular expression.
6. The expressions got by repeated application of the rules from (1) to (5) over Σ are also regular expression.

Que 2.2. **Explain the basic operations and also write the operator precedence in regular expression.**

Answer

Basic operations in regular expression :

1. **Union :** If R_1 and R_2 are two regular expressions over Σ then $L(R_1 \cup R_2)$ is denoted by $L(R_1 + R_2)$.
   $L(R_1 \cup R_2)$ is a string from R_1 or a string from R_2.
   $L(R_1 + R_2) = L(R_1) \cup L(R_2)$.

2. **Concatenation :** If R_1 and R_2 are two regular expressions over Σ then $L(R_1 \cap R_2)$ is denoted by $L(R_1 R_2)$.
   $L(R_1 \cap R_2)$ is a string from R_1 followed by a string from R_2.
   $L(R_1 R_2) = L(R_1)\, L(R_2)$.
3. **Closure :** If R_1 and R_2 are two regular expressions over Σ then $L(R^*)$ is a string obtained by concatenating n elements for $n \geq 0$.

Operator precedence in regular expression :

Precedence	Operator	Description
Highest	(), (? :)	Parentheses and other grouping operators
↓	+, *, {m, n}	Repetition
↓	^ abc	Sequence
Lowest	|	Alternation

Que 2.3. **Discuss the identities for regular expression.**

Answer

1. Two regular expression P and Q are equivalent ($P = Q$) if P and Q represent the same set of strings.
2. The identities for regular expressions are useful for simplifying regular expressions.

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

Que 2.4. **Simplify the regular expression $(r(r + s)^*)$.**

Answer

$\Rightarrow (r(r + s)^*)$
$\Rightarrow rr^* (sr^*)^*$ $[(r + s)^* = r^* (sr^*)^*]$
$\Rightarrow r(\varepsilon + r)\, r^* (sr^*)^*$ $[(\varepsilon + r)\, r^* = r^*]$
$\Rightarrow (r + rr)\, r^* (sr^*)^*$ $[r + r\varepsilon$ and $\bullet$ distributes over $+]$
$\Rightarrow (r + rr)(r + s)^*$ $[(r + s)^* = r^* (sr^*)^*]$

Que 2.5. **From the identities of regular expression prove that**

i. $(1 + 100^*) + (1 + 100^*)(0 + 10^*)(0 + 10^*) = 10^* (0 + 10^*)^*$

ii. $\varepsilon + 1^* (011)^* [1^* (011)^*]^* = (1 + 011)^*$

Answer

i. $(1 + 100^*) + (1 + 100^*)(0 + 10^*)(0 + 10^*)^*$

L.H.S $= (1 + 100^*) [\varepsilon + (0 + 10^*)(0 + 10^*)^*]$

$= (1 + 100^*)(0 + 10^*)^*$ [According to $\varepsilon + RR^* = R$

$= 1(\varepsilon + 00^*)(0 + 10^*)^*$

$= 10^*(0 + 10^*)^*$ = R.H.S.

ii. Take $1^* (011)^*$ as P then the LHS becomes $\varepsilon + PP^*$

$= P^*$

$= [1^*(011)^*]^* = (1 + 011)^*$ = R.H.S. [As $(P + Q)^* = (P^* Q^*)^*$

Que 2.6. What are transition diagrams/transition graphs ?

Answer

A transition diagram/transition graphs is a graph which consists of series of states and there is a successful path beginning at some start state and ending at a final.

Thus, we can describe the transition graph as collection of following :

1. A finite set of states, one of them designated as the start state and some of which are designated as final state.
2. The starting state is represented by a circle having an arrow and final state is represented by two concentric circles. The diagrammatic representation is as follows :

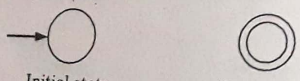


Fig. 2.6.1.

3. An alphabet Σ of possible input letters from which input string is formed.
4. A finite set of transitions (labeled edge) that shows how to go from one state to another state, based on reading specified substrings of input letters (possibly even the null string).
5. The null string may be represented by 'ε' or '∧'.
6. In transition diagram the initial state is represented as $\rightarrow (q_0)$ and final state as (q_f).

For example :

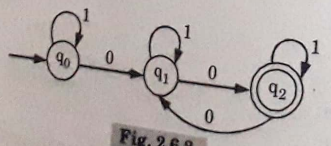


Fig. 2.6.2.

PART-2

Kleene's Theorem.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 2.7. State and prove Kleene's theorem with example.

AKTU 2016-17, Marks 10

Answer

Kleene's theorem states that any regular language is accepted by an FA and conversely that any language accepted by an FA is regular.

Theorem 1 (Part 1 of Kleene's theorem) : Any regular language is accepted by a finite automaton.

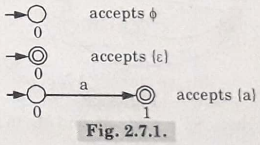


Fig. 2.7.1.

Proof :

Basis step : As shown in Fig. 2.7.1 the languages ϕ, {ε} and {a} for any symbol a in Σ are accepted by an FA.

Inductive step : We are going to show that for any languages L_1 and L_2 if they are accepted by FAs, then L_1L_2 , L_1L_2 and L_1^* are accepted by FAs. Since any regular language is obtained from {ε} and {a} for any symbol a in Σ by using union, concatenation and Kleene star operations, that together with the basis step would prove the theorem.

Suppose that L_1 and L_2 are accepted by FAs $M_1 = < Q_1, \Sigma, q_{1,0}, \delta_1, A_1 >$ and $M_2 = < Q_2, \Sigma, q_{2,0}, \delta_2, A_2 >$, respectively. We assume that $Q_1 \cap Q_2 = \phi$ without loss of generality since states can be renamed if necessary.

Then $L_1 \cup L_2$, $L_1 L_2$ and L_1^* are accepted by the FAs $M_u = < Q_u, \Sigma, q_{u,0}, \delta_u, A_u >$, $M_c = < Q_c, \Sigma, \delta_{c,0}, \delta_c, A_c >$ and $M_k = < Q_k, \Sigma, q_{k,0}, \delta_k, A_k >$, respectively, *i.e.*,

For $M_u = < Q_u, \Sigma, q_{u,0}, \delta_u, A_u >$:

$Q_u = Q_1 \cup Q_2 \cup \{q_{u,0}\}$, where $q_{u,0}$ is a state which is neither in Q_1 nor in Q_2.

$\delta_u = \delta_1 \cup \delta_2 \cup \{(q_{u,0}, \varepsilon, \{q_{1,0}, q_{2,0}\})\}$, that is $\delta_u(q_{u,0}, \varepsilon) = \{q_{1,0}, q_{2,0}\}$. *i.e.*, $\delta_u(q_{u,0}, a) = \phi$ for all a in Σ.

$A_u = A_1 \cup A_2$

For $M_c = < Q_c, \Sigma, q_{c,0}, \delta_c, A_c >$:

$Q_c = Q_1 \cup Q_2$

$q_{c,0} = q_{1,0}$

$\delta_c = \delta_1 \cup \delta_2 \cup \{(q, \varepsilon, \{q_{2,0}\}) \mid q \in A_1\}$

$A_c = A_2$

For $M_k = < Q_k, \Sigma, q_{k,0}, \delta_k, A_k >$:

$Q_k = Q_1 \cup \{q_{k,0}\}$, where $q_{k,0}$ is a state which is not in Q_1.

$\delta_k \quad \delta_1 \cup \{(q_{k,0}, \varepsilon, \{q_{1,0}\})\} \cup \{(q, \varepsilon, \{q_{k,0}\}) \mid q \in A_1\}$

$A_k = \{q_{k,0}\}$

These NFA-ε are illustrated in Fig. 2.7.2.

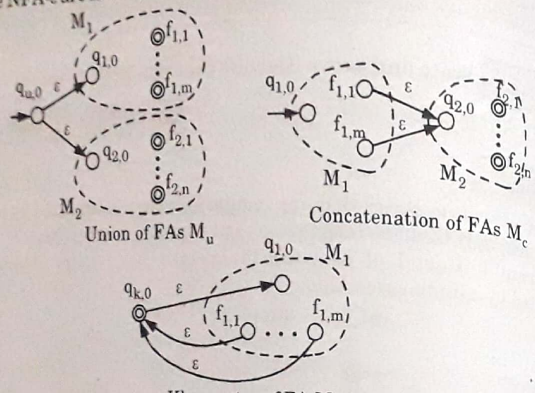


Fig. 2.7.2.

It can be proven, that these NFA-ε, M_u, M_c and M_k, in fact accept $L_1 \cup L_1$, L_1 and L_1^* respectively.

Example :

An NFA-ε that accepts the language represented by the regular expression $(aa + b)^*$ can be constructed as follows :

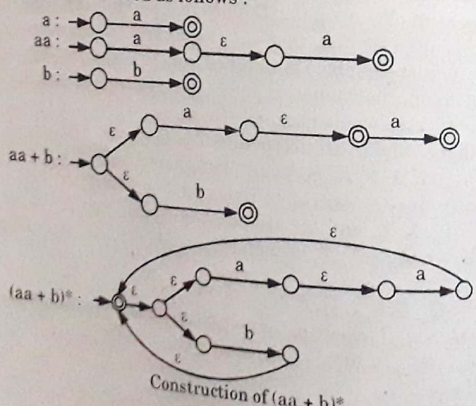


Fig. 2.7.3.

Que 2.8. **For the given languages $L_1 = \phi$, $L_2 = \varepsilon$, and $L_3 = \{0, 1\}^*$. Compute $L_1 . L_2$ and $L_2 \cup L_3$.**

Answer

$L_1 = \phi, L_2 = \varepsilon, L_3 = \{0, 1\}^*$

Concatenation : $L_1 L_2 = \phi . \varepsilon = \{\varepsilon\}$

Union : $L_2 \cup L_3 = \varepsilon \cup \{0, 1\}^* = \{0, 1\}^*$

Que 2.9. **Let L = {0, 1} {0, 1}*; construct NFA with ε-moves that accept L^2.**

Answer

The L includes strings here = {0, 1, 00, 01, 10, 11 .....}

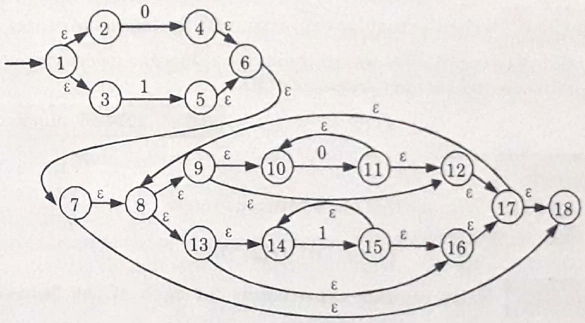


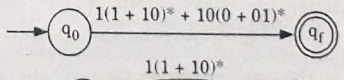
Fig. 2.9.1.

Que 2.10. **Draw finite automata recognizing the following language :**

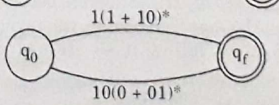
$1(1 + 10)^* + 10(0 + 01)^*$

Answer

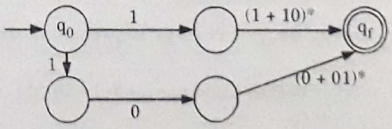
Step 1 :



Step 2 :



Step 3 :



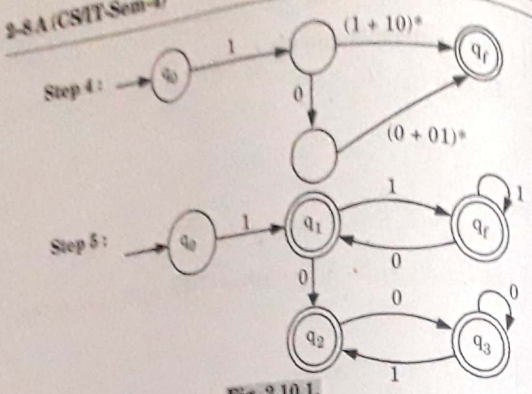


Fig. 2.10.1.

Que 2.11. **Write the regular expression for the language contain the strings over {0, 1} in which there are at least two occurrences 1's between any two occurrences of 0's.**

AKTU 2014-15, Marks 10

Answer

If (11)* 0 (11)* it means that one 0 between even 1.

∴ Regular expression will be :

$$r = (0\,(11)^*\,0\,(11)^*\,0)^*$$

Que 2.12. **Write regular expressions for each of the followin languages over the alphabet (0, 1) :**

i. **The set of all string in which every pair of adjacent zero's appea before any pair of adjacent one's.**

ii. **The set of all strings not containing 101 as substring.**

Answer

i. According to question, the pair (0011) may exists zero, one, two, ... times. Therefore the regular expression will be :

$$(0011)^*$$

ii. When we analyse the set of strings then it becomes clear that strin may start with '0' and 0's may be repeated, wherever one is encountere then no single 0 must follow it so strings are like 0000010 000001111001001.....

RE is = $(0^*1^*00)^*0^*1^*$

Que 2.13. **Let r_1 and r_2 be two regular expressions defined a follows :**

Prove that $r_1 = (00^*1)^*1$ and $r_2 = 1 + 0(0 + 10)^*11$ $r_1 = r_2$.

Answer

$r_1 = (00^*1)^*1$

Set of string $= \{1, 011, 0011, 00011, 010101011, ...\}$

$r_2 = 1 + 0\,(0 + 10)^*\,11$

Set of string $= \{1, 011, 0011, 01011, 010101011, ...\}$

Since both regular expressions produce same set of string

So, $r_1 = r_2$

PART-3

Finite Automata and Regular Expression : Arden's Theorem, Algebraic Method Using Arden's Theorem.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 2.14. **State and prove Arden's theorem. Explain its application.**

Answer

Arden's theorem : Let P and Q are two regular expression over alphabet Σ. P does not contain null string ε. Then equation $R = Q + RP$ has a unique solution given by

$$R = Q.P^*$$

Proof : Let us consider the equation $R = Q + RP$ ...(2.14.1)

Put value of R in equation (2.14.1)

$$R = Q + (Q + RP)P \quad ...(2.14.2)$$

Put value of R in equation (2.14.2)

$$R = Q + QP + RP^2 \quad ...(2.14.3)$$

When we put the value of R again and again we get the following equation :

$R = Q + QP + QP^2 + QP^3 + ...$

$R = Q\,(\varepsilon + P + P^2 + P^3 + ...)$ [Since $\varepsilon + a + a^2 + ... = a^*$]

$R = Q.P^*$ [By definition of closure operation for regular expression]

Application :

1. Arden's theorem is used to determine the regular expression represented by transition diagram.
2. There are following assumptions corresponding to transition diagram :
   a. The transition system should not have any ε-move.
   b. It has only one starting state.
   c. The vertices are $q_1, q_2, q_3, ... q_n$.
   d. R_i is the regular expression representing the set of string accepted by the transition system, if q_i is the final state.

e. w_{ij} denotes the regular expression representing the set of labels of edges from q_i to q_j. We can get the following set of equation in $q_1, q_2, \ldots q_n$.

$$q_1 = q_1 w_{11} + q_2 w_{21} + \ldots + q_n w_{n1} + \varepsilon$$
$$q_2 = q_1 w_{12} + q_2 w_{22} + \ldots + q_n w_{n2}$$
$$\ldots\ldots$$
$$q_n = q_1 w_{1n} + q_2 w_{2n} + \ldots + q_n w_{nn}$$

We solve these equations for q_i in terms of w_{ij}'s and it will be required regular expression. We add ε (null string) in the equation start with starting state q_i and we solve equation to find out q_i (final state) in terms of w_{ij}'s it is one of the regular expression for given deterministic finite automata.

Que 2.15. **Find the regular expression of given FA using Arden's theorem.**

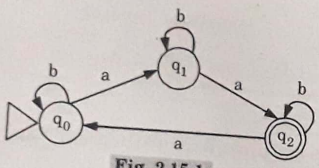


Fig. 2.15.1.

AKTU 2018-19, Marks 10

Answer

$$q_0 = q_0 b + q_2 a + \varepsilon$$
$$q_1 = q_1 b + q_0 a \quad \ldots(2.15.1)$$
$$q_2 = q_2 b + q_1 a \quad \ldots(2.15.2)$$

First solve equation (2.15.3) $\quad \ldots(2.15.3)$

According to theorem, $R = Q + RP$ gives $R = QP^*$

$$q_2 = q_1 a + q_2 b$$

$\therefore$ We will get $\quad Q = q_1 a \quad R = q_2 \quad P = b$

$$q_2 = q_1 ab^*$$

Now solve equation (2.15.2) $\quad \ldots(2.15.4)$

$$q_1 = q_0 a + q_1 b$$

$\therefore$ We will get $\quad Q = q_0 a \quad R = q_1 \quad P = b$

$$q_1 = q_0 ab^*$$

Put value of q_1 in equation (2.15.4), $\quad \ldots(2.15.5)$

$$q_2 = q_0 ab^* ab^*$$

Put value of q_2 in equation (2.15.1)

$$q_0 = q_0 b + q_0 ab^* ab^* a + \varepsilon = q_0 (b + ab^* ab^* a) + \varepsilon$$

According to Arden's theorem,

$$Q = \varepsilon \quad R = q_0 \quad P = (b + ab^* ab^* a)$$
$$R = QP^* = \varepsilon (b + ab^* ab^* a)^*$$
$$= (b + ab^* ab^* a)^* \qquad [\because \varepsilon R = R]$$

Que 2.16. **Find the regular expression using Arden's theorem of FA given below.**

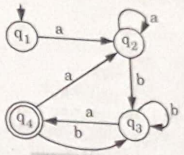


Fig. 2.16.1.

AKTU 2014-15, Marks 05

Answer

We directly apply the Arden's theorem as the graph does not contain any null (ε) moves.

We get the following equations :

$$q_1 = \varepsilon$$
$$q_2 = q_1 a + q_2 a + q_4 a$$
$$q_3 = q_2 b + q_3 b + q_4 b$$
$$q_4 = q_3 a$$

Therefore, $\quad q_3 = q_2 b + q_3 b + q_3 ab = q_2 b + q_3 (b + ab)$

Applying Arden's theorem, $q_3 = q_2 b (b + ab)^*$

Now, $\quad q_2 = a + q_2 a + q_3 aa = a + q_2 a + q_2 b (b + ab)^* aa$

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

By Arden's theorem, $\quad q_2 = a (a + b (b + ab)^* aa)^*$

$$q_4 = q_3 a = a (a + b (b + ab) aa)^* b (b + ab)^* a$$

As q_4 is only final state, the RE corresponding to given FA is,

$a(a + b (b + ab)^* aa)^* b (b + ab)^* a$.

Que 2.17. **State recursive definition of regular expression and construct a regular expression corresponding to the state transition diagram as shown in Fig. 2.17.1.**

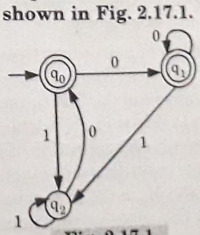


Fig. 2.17.1.

AKTU 2017-18, Marks 07

Answer

Recursive definition of regular expression : Refer Q. 2.1, Page 2–2A, Unit-2.

Numerical :

$$q_0 = q_2 0 + \varepsilon \quad ...(2.17.1)$$
$$q_1 = q_0 0 + q_1 0 \quad ...(2.17.2)$$
$$q_2 = q_0 1 + q_1 1 + q_2 1 \quad ...(2.17.3)$$

Put the value of q_0 and q_1 in equation (2.17.3)

$$\begin{aligned} q_2 &= q_0 1 + (q_0 0 + q_1 0)\, 1 + q_2 1 \\ &= q_0 1 + q_0 01 + q_1 01 + q_2 1 \\ &= q_0 1 + q_0 01 + (q_0 0 + q_1 0)\, 01 + q_2 1 \\ &= q_0 1 + q_0 01 + q_0 001 + q_1 001 + q_2 1 \\ &= q_0 (1 + 01 + 001) + q_1 001 + q_2 1 \\ &= q_0 (1 + 01 + 001) + (q_0 0 + q_1 0) 001 + q_2 1 \\ &= q_0 (1 + 01 + 001) + q_0 0001 + q_1 0001 + q_2 1 \end{aligned}$$

If we solve further we get q_1 in every step. So, we cannot find the value of q_2 in term of q_0.

Hence, question cannot be solved.

Que 2.18. Explain the condition in which two machines M_1 and M_2 are said to be equivalent. Show that the following automatas are not equivalent.

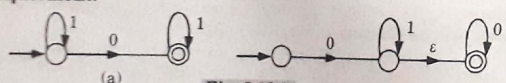


Fig. 2.18.1.

AKTU 2014-15, Marks 05

Answer

Comparison method :

1. Let M_1 and M_2 be two finite automata over Σ. We construct a comparison table consisting of $n + 1$ columns, where n is the number of input symbols.
2. The first column consists of pairs of vertices of the form (q, q'), where $q \in M_1$ and $q' \in M_2$.
3. If (q, q') appears in some row of the first column, then the corresponding entry in the a-column ($a \in \Sigma$) is (q_a, q'_a), where q_a and q'_a are reachable from q and q', respectively on application of a (*i.e.*, by a-paths).
4. The comparison table is constructed by starting with the pair of initial vertices q_{in}, q'_{in} of M_1 and M_2 in the first column.
5. The first elements in the subsequent columns are (q_a, q'_a), where q_a and q'_a are reachable by a-paths from q_{in} and q'_{in}.
6. We repeat the construction by considering the pairs in the second and subsequent columns which are not in the first column.

7. The row-wise construction is repeated. There are two cases :

Case 1 : If we reach a pair (q, q'_a) such that q is a final state of M_1, and q'_a is a non-final state of M_2 or vice versa, we terminate the construction and conclude that M_1 and M_2 are not equivalent.

Case 2 : Here the construction is terminated when no new element appears in the second and subsequent columns which are not in the first column (*i.e.*, when all the elements in the second and subsequent columns appear in the first column). In this case we conclude that M_1 and M_2 are equivalent.

Numerical :

Let M_1 and M_2 be the following automatas :

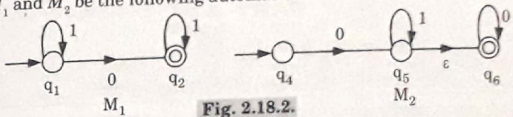


Fig. 2.18.2.

The initial states in M_1 and M_2 are q_1 and q_4 respectively.

Hence, the first element of the first column in the comparison table must be (q_1, q_4).

Table 2.18.1.

(q, q')	(q_0, q'_0)	(q_1, q'_1)
(q_1, q_4)	(q_2, q_5)	(q_2, q_5)
(q_2, q_5)	–	(q_2, q_5)

As q_2 is final state in M_1 and q_5 is non-final state in M_2, so M_1 and M_2 are not equivalent.

Que 2.19. From the given NFAs, decide whether the two accept the same language.

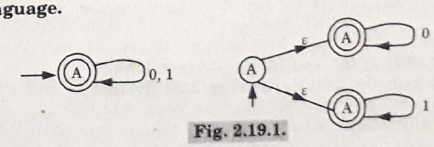


Fig. 2.19.1.

Answer

1. Let

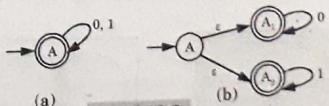


Fig. 2.19.2.

2. In Fig. 2.19.2(a) the NFA accept all string which is combination of alphabets

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

3. In Fig. 2.19.2(b) the NFA has ε-move and we know that ε-transition is a null transition. So we can consider state A, A_1 and A_2 as same state *i.e.*, A and we get

Fig. 2.19.3.

4. So the given NFAs (Fig. 2.19.1) accept same language.

PART-4

Regular and Non-Regular Languages : Closure Properties of Regular Languages.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 2.20. **Prove that the language L_1 and L_2 are closed under intersection and complementation if they are regular.**

Answer

Intersection :

If L_1 and L_2 are regular languages, then $L_1 \cap L_2$ is also regular language. In other words the set of the regular language is closed under intersection.

Proof : By De-Morgan's law for sets of any kind (regular languages or not) :

$$L_1 \cap L_2 = \overline{(\overline{L_1} + \overline{L_2})}$$

This can be justify by the Venn diagrams as follows :

So, it is clear from Fig. 2.20.1(*a*) and Fig. 2.20.1(*b*) that

$$L_1 \cap L_2 = \overline{(\overline{L_1} + \overline{L_2})}$$

Because L_1 and L_2 are regular so $\overline{L}_1$ and $\overline{L}_2$ are regular. So is $\overline{L}_1 + \overline{L}_2$. And because $\overline{L}_1 + \overline{L}_2$ is regular then so is $\overline{(\overline{L_1} + \overline{L_2})}$ which means $\overline{L}_1 \cap \overline{L}_2$ is regular.

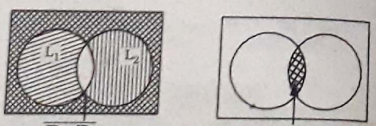


Fig. 2.20.1.

Complementation :

If L is regular then complement of L that is $\overline{L}$ is also regular. In other words, the set of regular languages is closed under complementation.

Proof :

1. To show closure under complementation, let $M = (Q, \Sigma, \delta, q_0, F)$ be DFA that accepts L, then DFA accepts $\overline{L}$.

$$\overline{M} = (Q, \Sigma, \delta, q_0, Q - F)$$

2. In DFA M accepts the language L in which some of the states of FA are final states and some are not.
3. In DFA $\overline{M}$ status of each state is reversed, that is if it is final state, make it a non-final state, and if it is non-final state, make it a final state.
4. If an input string formally ended in a non-final state, it now ends in a final state and vice versa.
5. This new machine accepts all input strings that were not accepted by the original FA (all the strings in $\overline{L}$) and rejects all the input strings that the FA used to accept (the strings in L). Therefore, machine accepts exactly the language $\overline{L}$. So, $\overline{L}$ is also regular language.

Que 2.21. **State whether following statement is true or false. Justify your answer :**

i. **If L and M are non-regular languages then intersection of L and M is also non-regular.**

ii. **If L and M are regular languages then $L - M$ is also regular language.**

Answer

i. True

Proof : Let L and M are two non-regular language.

The intersection of L and M is represented by $L \cap M$.

By De-Morgan's law $L \cap M = \overline{\overline{L} \cup \overline{M}}$

We know that complement of a non-regular language is non-regular and union of two non-regular languages is non-regular. Therefore $\overline{\overline{L} \cup \overline{M}}$ is non-regular.

Here $L \cap M$ is non-regular language.

ii. True

Proof : $L_1 - L_2 = L_1 \cap \overline{L}_2$

[Replace L_1 by L and L_2 by M]

We know that complement of a regular language is regular and we also know that the intersection of two regular languages is regular.

Hence, $L_1 \cap \overline{L}_2$ will be regular.

Therefore $L_1 - L_2$ is regular.

Que 2.22. **Prove that the complement, homomorphism and inverse homomorphism, closure of a regular language is regular.**

AKTU 2016-17, Marks 10

Answer

1. **Regular languages are closed under complementation :** Refer Q. 2.20, Page 2–14A, Unit-2.
2. **Regular languages are closed under homomorphism, *i.e.*, if L is a regular language and h is a homomorphism, then $h(L)$ is also regular :**

   Proof :

   We will use the representation of regular languages in terms of regular expressions to prove this :

   Let E be the regular expression for L.

   Apply h to each symbol in E.

   Language of resulting RE is $h(L)$.

   Example :

   Let $h(0) = ab$; $h(1) = \varepsilon$.

   Let L be the language of regular expression $01^* + 10^*$.

   Then $h(L)$ is the language of regular expression $ab\varepsilon^* + \varepsilon(ab)^*$.

   $ab\varepsilon^* + \varepsilon(ab)^*$ can be simplified.

   $\varepsilon^* = \varepsilon$, so $ab\varepsilon^* = ab\varepsilon$.

   ε is the identity under concatenation *i.e.*, $\varepsilon E = E\varepsilon = E$ for any RE E.

   Thus, $ab\varepsilon^* + \varepsilon(ab)^* = ab\varepsilon + \varepsilon(ab)^* = ab + (ab)^*$.

   Finally, $L(ab)$ is contained in $L((ab)^*)$, so a RE for $h(L)$ is $(ab)^*$.
3. **Regular languages are closed under inverse homomorphism, *i.e.*, if L is regular and h is a homomorphism then $h^{-1}(L)$ is regular :**

   Proof :

   a. We will use the representation of regular languages in terms of DFA to argue this.

      Start with a DFA A for L.

   b. For a given a DFA M recognizing L, construct a DFA M' that accepts $h^{-1}(L)$.

      Construct a DFA B for $h^{-1}(L)$ with the same set of states, the same start state, the same final states and input alphabet (the symbols to which homomorphism h applies).

   c. The transitions for B are computed by applying h to an input symbol a and seeing where A would go on sequence of input symbols $h(a)$. Formally, $\delta_B(q, a) = \delta_A(q, h(a))$.

   Example :

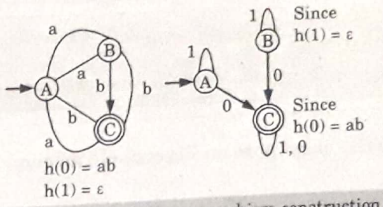


Fig. 2.22.1. Inverse homomorphism construction.

Que 2.23. **Discuss closure properties *i.e.*, concatenation, union, intersection, complement of regular languages.**

AKTU 2017-18, Marks 07

Answer

Union : If $L1$ and if $L2$ are two regular languages, their union $L1 \cup L2$ will also be regular.

For example :

Let $L1 = \{a^n \mid n \geq 0\}$ and $L2 = \{b^n \mid n \geq 0\}$

$L3 = L1 \cup L2 = \{a^n \cup b^n \mid n \geq 0\}$ is also regular.

Concatenation : If $L1$ and if $L2$ are two regular languages, their concatenation $L1.L2$ will also be regular.

For example :

Let $L1 = \{a^n \mid n \geq 0\}$ and $L2 = \{b^n \mid n \geq 0\}$

$L3 = L1.L2 = \{a^m \,.\, b^n \mid m \geq 0$ and $n \geq 0\}$ is also regular.

Intersection : If $L1$ and if $L2$ are two regular languages, their intersection $L1 \cap L2$ will also be regular.

For example :

Let $L1 = \{a^m b^n \mid n \geq 0$ and $m \geq 0\}$ and $L2 = \{a^m b^n \cup b^n a^m \mid n \geq 0$ and $m \geq 0\}$

$L3 = L1 \cap L2 = \{a^m\, b^n \mid n \geq 0$ and $m \geq 0\}$ is also regular.

Complement : If $L(G)$ is regular language, its complement $L'(G)$ will also be regular. Complement of a language can be found by subtracting strings which are in $L(G)$ from all possible strings. **For example :**

$$L(G) = \{a^n \mid n > 3\}$$
$$L'(G) = \{a^n \mid n <= 3\}$$

PART-5

Pigeonhole Principle, Pumping Lemma, Application of Pumping Lemma, Decidability : Decision Properties, Finite Automata and Regular Languages, Regular Language and Computers, Simulation of Transition Graph and Regular Language.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 2.24. **Write short note on Pigeonhole principle.**

Answer

Pigeonhole principle :

1. It states that "If n pigeons are assigned to m pigeonholes then at least one pigeonhole contains two or more pigeons ($m < n$)".
2. The pigeonhole principle is sometime useful in counting methods.

Proof :

1. Let m pigeonholes be numbered with the numbers 1 through m.
2. Beginning with the pigeon 1, each pigeon is assigned in order to the pigeonholes with the same number.
3. Since $m < n$ *i.e.*, the number of pigeonhole is less than the number of pigeons, $n - m$ pigeons are left without having assigned a pigeonhole.
4. Thus, at least one pigeonhole will be assigned to a more than one pigeon.
5. We note that the pigeonhole principle tells us nothing about how to locate the pigeonhole that contains two or more pigeons.
6. It only asserts the existence of a pigeon hole containing two or more pigeons.
7. To apply the principle one has to decide which objects will play the role of pigeon and which objects will play the role of pigeonholes.

Que 2.25. **State and prove the pumping lemma for regular language.**

Answer

1. Pumping Lemma is a powerful tool for proving certain language non-regular.
2. It is also useful in the development of algorithm to answer certain questions concerning, finite automata, such as whether the language accepted by a given FA is finite or infinite.

Statement : Let L be a regular language. Then there exist a constant n (which depends on L) such that for every string w in L, such that $|w| \geq n$, we can break w into three substrings, $w = xyz$, such that :

1. $y \neq \varepsilon$
2. $|xy| \leq n$
3. For all $i \geq 0$, the string xy^iz is also in L.

That is we always find a non-empty string y not too far from the beginning of w that can be "pumped"; that is repeating y any number of times keeps the resulting string in the language.

Proof :

1. Suppose L is regular. Then $L = L(M)$ for some DFA, M. Suppose M has n states.
2. Now, consider any string w of length n or more. Let $w = a_1, a_2, \ldots a_m$, where $m \geq n$ and each a_i is an input symbol, for $i = 0, 1, 2, \ldots, m$ define state p_i, to be $\hat{\delta}(q_0, a_1a_2 \ldots a_i)$, where $\hat{\delta}$ is the transition function of M, and q_0 is the start state of M. That is p_i is the state, M is in after reading the first i symbols of w.
3. By the Pigeonhole principle, it is not possible for the $n + 1$ different p_i's for $i = 0, 1, \ldots n$ to be distinct, since there are only n different states.
4. Thus we can find two different integers i and j, with $0 \leq i < j \leq n$, such that $p_i = p_j$. Now we can break $w = xyz$ as follows :
   1. $x = a_1a_2 \ldots a_i$
   2. $y = a_{i+1}a_{i+2} \ldots a_j$.
   3. $z = a_{j+1}a_{j+2} \ldots a_m$.

   That is x takes us to p_i once; y takes us from p_i back to p_i (since p_i is also p_j), and z is balance of w.
5. The relationship among the strings and states are discussed in Fig. 2.25.1.

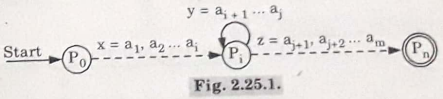


Fig. 2.25.1.

   x may be empty, in the case that $i = 0$. Also, z may be empty if $j = n = m$. However y cannot be empty, since i is strictly less than j.
6. Now consider what happens if the automaton M receives xy^iz for any $i \geq 0$. If $i = 0$, then automaton M goes from the start state q_0 to P_i on input x. Since p_i is also p_j, it must be that M goes from p_i to the accepting state in Fig. 2.20.2 on input z. Thus, in accepts xz.
7. If $i > 0$, then M goes from p_0 to p_i on input string x, circles from p_i to p_i, i times on input y_i, and then goes to the accepting state on input z. Thus, for any $i \geq 0$, xy^iz is also accepted by M; that is, xy^iz is in L.

Que 2.26. **Explain the application of pumping lemma.**

Answer

Application of the pumping lemma are :

The pumping lemma is extremely useful in proving that certain sets are non-regular. The general steps in its application are :

Step 1 : Assume L is a regular. Let n be the number of states in corresponding FA.

Step 2 : Choose a string w such that $|w| \geq n$. Use pumping lemma to write $w = xyz$, with $|yz| \leq n$ and $|y| > n$.

Step 3 : Find a suitable integer i such that $xy^iz \notin L$. This contradicts our assumption. Hence, L is not regular.

Que 2.27. **State pumping lemma for regular languages. Use pumping lemma to prove that the language L, defined as follows, is not regular.**
$L = \{0^m 1^n \mid m$ and n are positive integers and $m \neq n\}$

Answer

Pumping lemma for regular language : Refer Q. 2.25, Page 2–18A, Unit-2.

Proof :

Let $L = \{0^m 1^n \mid m \neq n\}$ be a language. Assume that L is regular language. Now consider following case :

Case 1 :

Assume $z = 0001111 \in L$

By pumping lemma if $z = uvw$ then if we pump some string to make $z = uv^iw$ then if $z \in L$ then such a language is called regular.

Let

$$z = \underset{u}{0}\ \underbrace{0\ 1}_{v}\ \underbrace{1\ 1}_{w}$$

if $z = uv^iw$ and if $i = 2$ then $z = uvvw$

$$z = \underset{u}{0}\ \underbrace{(0\ 1}_{v}\ \underbrace{1)\ (0\ 1}_{v}\ \underbrace{1)\ 1\ 1}_{w}$$

$\therefore$ $z = 0^3 1^2 0^2 1^4 \notin 0^m 1^n \notin L$

Case 2 :

Assume $L \in z$ such that

$$z = \underset{u}{0}\ \underbrace{0\ 0}_{v}\ \underbrace{0\ 1\ 1\ 1\ 1\ 1\ 1}_{w}$$

By pumping lemma, $z = uv^iw \in L$ for a regular language. If $i = 2$ then,

$z = uvvw$

$$z = \underset{u}{0}\ \underbrace{0\ 0}_{v}\ \underbrace{0\ 0}_{v}\ \underbrace{0\ 1\ 1\ 1\ 1\ 1}_{w}$$

$z = 0^6 1^5 \in 0^m 1^n$ because $m = n$.

From these cases, we get $z \notin L$. Thus our assumption of L being regular is wrong.

Hence given language $\{L = 0^m 1^n \mid m \neq n\}$ is not regular.

Que 2.28. **Prove that the language $L = \{0^n \mid n$ is prime$\}$ is not regular.**

OR

State Pumping Lemma for regular sets. Show that the set $L = \{a^p \mid p$ is a prime$\}$ is not regular. **AKTU 2015-16, 2017-18; Marks 10**

Answer

Pumping lemma : Refer Q. 2.25, Page 2–18A, Unit-2.

Numerical :

Step 1 : We suppose L is regular. Let q be the number of states in the finite automaton accepting L.

Step 2 : Let p be a prime number greater than p. Let $w = a^q$. By pumping lemma, w can be written as $w = xyz$, with $|xy| \leq q$ and $|y| > 0$. x, y, z are simply strings of a's. So, $y = a^m$ for some $m \geq 1$ (and $\leq q$).

Step 3 : Let $i = p + 1$. Then $|xy^iz| = |xyz| + |y^{i-1}| = p + (i-1)m = p + pm$. By pumping lemma, $xy^iz \in L$. But $|xy^iz| = p + pm = p(1 + m)$, and $p(1 + m)$ is not a prime. So, $xy^iz \in L$. This is a contradiction. Thus, L is not regular.

Que 2.29. **Prove that following are not regular languages :**

i. **$\{0^n \mid n$ is perfect square$\}$.**

ii. **The set of strings of form $0^i 1^i$ such that the greatest common divisor of i and j is 1.**

OR

Using Pumping lemma for regular languages prove that language $L = 0^{n^2}$, $n >= 1$ is not regular. **AKTU 2018-19, Marks 10**

Answer

i. **$\{0^n \mid n$ is perfect square$\}$.**

1. Let us suppose L be a regular, let p be a constant provided by pumping lemma. Let w be the string of 0^{p^2}.
2. This string is in L and its length is at least p. So we can write $|xy| \leq p$ and $y \neq \varepsilon$ and each xy^kz also in L.
3. Since $|xy| \leq p$, y consists of no more than p 0's and $y \neq \varepsilon$, so there is at least one '0' in y.
4. Now choose k to be 2 and the resulting string is $xyyz$.
5. We have to prove that the length of $xyyz$ is not perfect square. In fact we have to show that it lies most strictly in between two consecutive perfect squares, namely p^2 and $(p-1)^2$.
6. Let 'q' stand for length of y. Then the length of $xyyz$ is $p^2 + q$. So $p^2 < p^2 + q < (p+1)^2$.
7. The first inequality holds since $q > 0$. For second inequality, we compute $(p+1)^2 = p^2 + 2p + 1 > p^2 + q$. Since $q < p$, therefore $xyyz$ is not in L.
8. This contradicts the pumping lemma, so our assumption, that 'L' is regular is incorrect.

ii. Let $L = 0^i1^j$ such that gcd $(i, j) = 1$.

We assume that this is a regular language. Now consider three cases

Case 1 : Let $z = \underbrace{0 \ldots 0}_{i}\ \underbrace{1 \ldots 1}_{j}$ be a string $\in L$.

According to pumping lemma if $z = uvw$ is a string $\in L$, then $z = uv^iw$, $\in L$.

That means even if we pump some substring the string $z \in L$.
We map $z = uvw$ as this is a case in which v consists of substring 0^n1^n.
By pumping lemma $z = uv^iw$ if $i = 2$, $z = uvvw$

Then we get $z = 0^{i-m}(01)^m 1^{j-m} \notin L = \{0^i\ 1^j\}$

Case 2 : Let, $z = \underbrace{\overbrace{0\ldots0}^{i}}_{uv}\ \underbrace{\overbrace{1\ldots1}^{j}}_{w} \in L$

By pumping lemma $z = uv^iw$. If $i = 2$ then,

$$z = \underbrace{\overbrace{0\ldots0}^{i}\ \ldots0}_{u}\ \underbrace{1\ldots}_{v}\ \underbrace{\overbrace{1\ldots1}^{j}}_{w}$$

$$z = uvvw$$

i.e., $z = 0^{i+m}1^j$

But, gcd $(i+m, j) \neq 1$.

Hence, $z \notin [L = \{0^i 1^j\}]$

Case 3 : Let,

$$z = \underbrace{\overbrace{0\ldots0}^{i}}_{u}\ \underbrace{\overbrace{1\ldots1}^{j}}_{vw}$$

By pumping lemma, $z = uv^iw$. If $i = 2$ then,

$$z = uvvw$$

i.e., $z = 0^i 1^{j+m}$

But gcd $(i, j+m) \neq 1$.

Hence, $z \notin [L = \{0^i 1^j\}]$

From these 3 cases, we can prove that $L = 0^i1^j | \gcd(i, j) = 1$ is not a regular language.

Que 2.30. **Prove that the language $L = \{0^n | n$ is perfect cube$\}$ is not regular.**

AKTU 2014-15, Marks 05

Answer

1. Let us assume L is regular, let p be a constant provided by the pumping lemma.
2. Let w be the string 0^{p^3}. This string is in L, and is of length at least p. So w can be written as xyz with $|xy| \leq p$ and y not ε.
3. Pumping lemma states that if $xyz \in L$ then xy^kz also in L. y contains at least one 0 in y since y is not ε.
4. Let us assume for $k = 2$, the resulting string is $xyyz$.
5. We have to show that the length of $xyyz$ is not a perfect cube. Let q be the length of y, then the length of $xyyz$ is $p^3 + q$.
6. So it suffices to show that $p^3 < p^3 + q < (p + 1)^3$. The first inequality holds since $q > 0$ (since y was not ε).
7. For second inequality, we compute $(p + 1)^3 = p^3 + 3p^2 + 3p + 1$, and this is greater than $p^3 + q$ since $q < p$ (since $|xy| \leq p$).

8. The claim is established, so $xyyz$ is not in L.
9. This contradicts the pumping lemma, so our original assumption, that L was regular is not incorrect.

Que 2.31. **Explain decision problem and decidability properties in regular language.**

Answer

1. A decision problem is a restricted type of an algorithmic problem where for each input there are only two possible outputs.
2. A decision problem is a function that associates with each input instance of the problem a truth value true or false.
3. A decision algorithm is an algorithm that computes the correct truth value for each input instance of a decision problem. The algorithm has to terminate on all inputs.
4. A decision problem is decidable if there exists a decision algorithm for it. Otherwise it is undecidable.

Decidability properties of regular languages :

1. **DFA membership :** Let $M = (Q, \Sigma, \delta, q_0, F)$ denote a deterministic finite automaton and x is a sequence of finite-domain variables $(x1, x2, \ldots, xn)$ with respective domains $D1, D2, \ldots, Dn \subseteq \Sigma$. Under a regular language membership constraint regular (x, M), any sequence of values taken by the variables of x must belong to the regular language recognized by M.
2. **DFA emptiness :** DFA is said to be empty if there is no path from initial state to final state.
3. **DFA equivalence :** Let $M_1 = (Q, \Sigma, \delta, q_0, F)$ and $M_2 = (Q', \Sigma', \delta', q_0', F')$ are two DFA. M_1 and M_2 are said to be equivalent if and only if $L(M_1) = L(M_2)$.
4. **DFA finiteness :** Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. String accept by M is said to be finite if and only if there is no cycle between any two state q_1 and q_2.

VERY IMPORTANT QUESTIONS

Following questions are very important. These questions may be asked in your SESSIONALS as well as UNIVERSITY EXAMINATION.

Q. 1. State and prove Kleene's theorem with example.

Ans. Refer Q. 2.7.

Q. 2. State and prove Arden's theorem. Explain its application.

Ans. Refer Q. 2.14.

Q. 3. Find the regular expression of given FA using Arden's theorem.

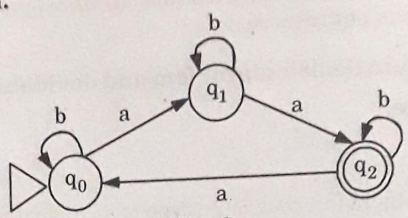
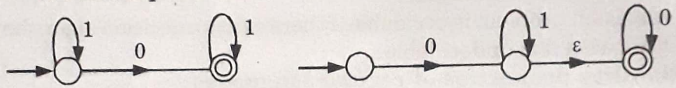


Fig. 1.

Ans. Refer Q. 2.15.

Q. 4. Explain the condition in which two machines M_1 and M_2 are said to be equivalent. Show that the following automatas are not equivalent.



(a)

Ans. Refer Q. 2.18.

Q. 5. Prove that the complement, homomorphism and inverse homomorphism, closure of a regular language is regular.

Ans. Refer Q. 2.22.

Q. 6. Discuss closure properties *i.e.*, concatenation, union, intersection, complement of regular languages.

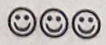
Ans. Refer Q. 2.23.

Q. 7. Prove that the language $L = \{0^n \mid n$ is perfect cube$\}$ is not regular.

Ans. Refer Q. 2.30.

Q. 8. Prove that the language $L = \{0^n \mid n$ is prime$\}$ is not regular.

Ans. Refer Q. 2.28.



4 UNIT

Push Down Automata and Properties of Context Free Languages

CONTENTS

Part-1	:	Non-Deterministic Pushdown Automata (NPDA) : Definition, Moves, A Language Accepted by NPDA, Deterministic Pushdown Automata (DPDA) and Deterministic Context Free Languages	**4–2A to 4–9A**
Part-2	:	Pushdown Automata for Context Free Language	**4–9A to 4–15A**
Part-3	:	Context Free Grammar for Pushdown Automata	**4–15A to 4–17A**
Part-4	:	Two Stack Pushdown Automata	**4–18A to 4–21A**
Part-5	:	Pumping Lemma for CFL, Closure Properties of CFL, Decision Problems of CFL, Programming Problems based on the Properties of CFLs	**4–22A to 4–27A**

PART-1

Non-Deterministic Push Down Automata (NPDA) : Definition Move
A Language Accepted by NPDA, Deterministic Push Down Automa
(DPDA) and Deterministic Context Free Languages (DCFL).

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 4.1. What is Push Down Automata (PDA) ? Explain with diagram.

Answer

1. PDA is the machine format of Context Free Language (CFL).
2. It is one type of finite state machine (FSM) which is used to accept only CFLs.

Diagram :

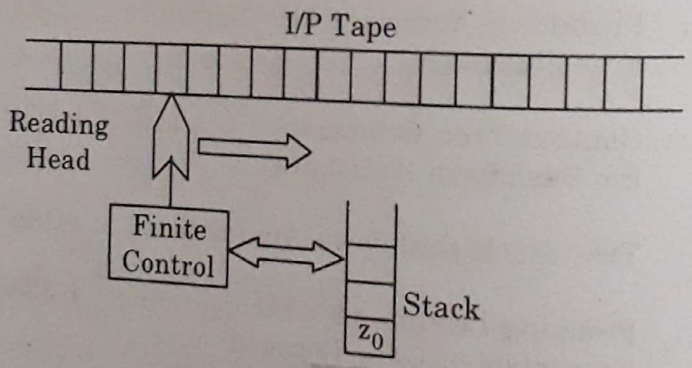


Fig. 4.1.1.

a. Input tape :

i. Input tape contains the input symbols.

ii. The tape is divided into a number of squares. Each square contains a single input character.

iii. The string placed in the input tape is traversed from left to right.

iv. Two end sides of the input string contain infinite number of blank symbol.

b. Reading head :

i. The head scans each square in the input tape and reads the input from the tape.

ii. The head moves from left to right.

iii. The input scanned by the reading head is sent to the finite control of the PDA.

c. Finite control :

i. Finite control can be considered as a control unit of a PDA.

ii. The reading head scans the input from the input tape and sends it to finite control.

iii. Depending on the input take from input tape and input from stack top, finite control decides in which state the PDA will move and which stack symbol it will push to the stack or pop from the stack or do nothing on the stack.

d. Stack :

i. A stack is a temporary storage of stack symbols.

ii. Every move of PDA indicates one of the followings to the stack.

1. One stack symbol may be added to the stack.
2. One stack symbol may be deleted from the top of the stack.

iii. Stack is the component of PDA which differentiates it from finite automata.

iv. In stack there is always a symbol z_0 which denotes the bottom of the stack.

Que 4.2. Explain definition of push down automata and why it is named push down ?

Answer

A push down automata is a system, which is mathematically defined as follows :

$$P = (Q, \Sigma, \Gamma, \delta, s, F)$$

Where, Q : non-empty finite set of states

Σ : non-empty finite set of input symbols

Γ : is finite set of pushdown symbols

s : is the initial state, $s \in Q$

F : is the set of final states and $F \subseteq Q$

δ : it is a transition function or transition relation which maps.

$(Q \times \Sigma^* \times \Gamma^*) \rightarrow (Q \times \Gamma^*)$

Why push down :

1. Push is an operation related to stack. By this operation one symbol is added to the stack top.
2. In finite automata, states act as a form of primitive memory. States memorize the non-terminals encountered at the time of derivation of a string.

3. Hence only state is suitable for traversing a regular expression as in the case of finite automata.

Example :

a. Let's consider a case of $L = \{a^n b^n$, where $n \geq 1\}$. It is not a regular expression but a CFL. Here n is any number.

b. In the string there is equal number of 'a' and 'b'. In the string 'a' will occur before 'b'.

c. In case of traversing $a^n b^n$, the machine has to remember n number of a's to traverse equal number of 'b'. The 'n' is any number.

d. Therefore, to memorize number of a's in the string the machine requires infinite number of states, *i.e.*, the machine will not be finite state machine.

e. To remove this difficulty we need to add an auxiliary memory in the form of stack.

f. For each occurrence of 'a' in the string one symbol is pushed into the stack.

g. For each occurrence of b (after finishing 'a') one symbol will be popped from the stack. By this process the matching of 'a' will be done. By adding a stack to a finite automata, PDA is generated.

Que 4.3. **Design PDA for palindrome strips.**

AKTU 2014-15, Marks 05

Answer

PDA for palindrome :

We define the PDA M as follows :

$$M = (\{q_0, q_1, q_2\}, \{a, b\}, \{a, b, Z_0\}, \delta, q_0, Z_0, \{q_2\})$$

where δ is defined by

$$\delta(q_0, a, Z_0) = \{(q_0, aZ_0), (q_1, Z_0)\}$$
$$\delta(q_0, b, Z_0) = \{(q_0, bZ_0), (q_1, Z_0)\}$$
$$\delta(q_0, a, a) = \{(q_0, aa), (q_1, a)\}$$
$$\delta(q_0, b, a) = \{(q_0, ba), (q_1, a)\}$$
$$\delta(q_0, a, b) = \{(q_0, ab), (q_1, b)\}$$
$$\delta(q_0, b, b) = \{(q_0, bb), (q_1, b)\}$$
$$\delta(q_0, \varepsilon, Z_0) = \{(q_1, Z_0)\}$$
$$\delta(q_0, \varepsilon, a) = \{(q_1, a)\}$$
$$\delta(q_0, \varepsilon, b) = \{(q_1, b)\}$$
$$\delta(q_1, a, a) = \{(q_1, \varepsilon)\}$$
$$\delta(q_1, b, b) = \{(q_1, \varepsilon)\}$$
$$\delta(q_1, \varepsilon, Z_0) = \{(q_2, Z_0)\}$$

Que 4.4. **What are the condition to declare a string accepted by PDA ?**

OR

Define Pushdown Automata (PDA). Differentiate PDA by empty stack and final state by giving their definitions.

AKTU 2015-16, Marks 10

Answer

PDA : Refer Q. 4.1, Page 4–2A, Unit-4.

There are two ways to declare a string accepted by a PDA :

1. **Accepted by empty stack (store) :**
   a. In each PDA, there is a stack attached.
   b. In the stack at the bottom there is a symbol called stack bottom symbol.
   c. In each move of the PDA, one symbol called stack symbol is either pushed in or popped from the stack. But the symbol z_0 still remains in the stack.
   d. A string w may be declared accepted by empty stack after processing all the symbol of w, if the stack is empty after reading the rightmost input character of the string w.
   e. In mathematical notation, we can say $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ be a PDA, the string w is declared accepted by empty stack if
   $\{w \in \Sigma^* / (q_0, w, z_0)^* \rightarrow (q, \varepsilon, \varepsilon)$ for some $q \in Q\}$.
   f. In general, we can say that a string is accepted by a PDA by empty stack if both the following conditions are fulfilled.
      i. The string is finished (Totally traversed)
      ii. The stack is empty.

2. **Accepted by final state :**
   a. A string w may be declared accepted by final state if after total traversal of the string w the PDA enters into its final state.
   b. In mathematical notation, we can say $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ be a PDA, the string w is declared accepted by final state if
   $\{w \in \Sigma^* / (q_0, w, z_0)^* \rightarrow (q_f, \varepsilon, z_0)$ for $q_f \in F$ and z_0 is the stack bottom symbol$\}$.
   c. In general, we can say that a string is accepted by a PDA by final state if both the following conditions are fulfilled.
      i. The string is finished (Totally traversed)
      ii. The PDA enters into its final state.

Que 4.5. **Define Push Down Automata (PDA). Design a PDA for the following language :**

$L = \{a^i b^j c^k \mid i = j \text{ or } j = k\}$

AKTU 2014-15, Marks 10

Answer

PDA : Refer Q. 4.1, Page 4–2A, Unit-4.

Numerical :

We have PDA M as follows :

$M = (\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}, \{a, b, c\}, \{z_0, x\}, \delta, q_0, z_0, \{q_1, q_3\})$

δ is given by :

$\delta(q_0, \varepsilon, z_0) = \{(q_1, z_0), (q_2, z_0), (q_3, z_0)\}$
$\delta(q_1, c, z_0) = (q_1, z_0)$
$\delta(q_2, a, z_0) = (q_0, xz_0)$
$\delta(q_2, a, x) = (q_2, xx)$
$\delta(q_2, b, x) = (q_4, b, x) = (q_4, \varepsilon)$
$\delta(q_4, \varepsilon, z_0) = (q_1, z_0)$
$\delta(q_3, a, z_0) = (q_3, z_0)$
$\delta(q_3, b, z_0) = (q_5, xz_0)$
$\delta(q_5, b, x) = (q_5, xx)$
$\delta(q_5, c, x) = (q_6, c, x) = (q_6, \varepsilon)$
$\delta(q_6, \varepsilon, z_0) = (q_3, \varepsilon) \rightarrow$ Accepted

Que 4.6. **How many types of PDA are there ? Explain each of them.**

Answer

There are two types of PDA :

1. **Deterministic PDA (DPDA) :**
   a. A PDA is said to be a DPDA if all derivations in the design give only single move.
   b. If a PDA being in a state with a single input and single stack symbol gives a single move, then the PDA is called Deterministic PDA.
2. **Non-deterministic PDA (NPDA) :**
   a. A PDA is called non-deterministic if one of the derivations generates more than one move.
   b. If a PDA being in a state with a single input and single stack symbol gives more than one move for any of its transitional functions, then the PDA is called non-deterministic PDA.

Que 4.7. **Construct a PDA that accepts $L = \{ww^R \mid w = (a + b)^*\}$**

AKTU 2016-17, Marks 10

Answer

From analysis of problem we find that we have to design a PDA, which accepts all palindrome of even length. The string (palindrome) must content only a, b and ε the strings accepted by PDA will be like bb, aa, $abba$, $baab$ etc.

Now we can define PDA and the δ given by

$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{a, b, \varepsilon\}$
$q_0 = \{q_0\}$
$Z_0 = \{X\}$
$\Gamma = \{X, A, B\}$
$F = \{q_2\}$

$\delta(q_0, a, X) \rightarrow (q_0, AX)$
$\delta(q_0, b, X) \rightarrow (q_0, BX)$
$\delta(q_0, a, A) \rightarrow (q_0, AA)$
$\delta(q_0, a, A) \rightarrow (q_1, \varepsilon)$
$\delta(q_0, a, B) \rightarrow (q_0, AB)$
$\delta(q_0, b, A) \rightarrow (q_0, BA)$
$\delta(q_0, b, B) \rightarrow (q_0, BB)$
$\delta(q_0, b, B) \rightarrow (q_1, \varepsilon)$
$\delta(q_1, a, A) \rightarrow (q_1, \varepsilon)$
$\delta(q_1, b, B) \rightarrow (q_1, \varepsilon)$
$\delta(q_1, \varepsilon, X) \rightarrow (q_2, \varepsilon) \rightarrow$ Accepted

Que 4.8. **Show that if L is a language of Deterministic PDA (DPDA) and R is regular then $L \cap R$ is a language of DPDA.**

Answer

1. To run the PDA for L and the DFA for R in parallel, we construct a DFA for the intersection of two regular languages by running both their DFAs in parallel and accepting only if both machines accept.
2. Formally, suppose the PDA for L has finite states Q_1, stack alphabet Γ, transition function δ_1, and accepting states F_1, and the DFA for R has similarly Q_2, δ_2, and F_2.
3. Assume for simplicity that the PDA for L reads a input symbol on every step.
4. Then the PDA for $L \cap R$ will have states $Q = Q_1 \times Q_2$ and transition function $\delta : Q \times \varepsilon \times \Gamma \rightarrow Q \times \Gamma$, where the new state is $(\delta_1(q_1, a), \delta_2(q_2, a))$ and we use δ_1 to determine what, if anything, we push on the stack.
5. Our set of accepting states is $F = F_1 \times F_2 \subset Q$. The resulting PDA is deterministic if the original PDA for L is regular, so $L \cap R$ is a CFL or DCFL if L is regular.

Que 4.9. **Construct a PDA to accept the language $L = \{a^n b^n c^m d^m$, where $m, n \geq 1\}$ by empty stack and by final state.**

Answer

1. Here number of 'a' number of 'b' are same and number of 'c' and number of 'd' are same state change will occur in transition from 'a' to 'b', 'b' to 'c' and 'c' to 'd'.
2. Upon traversing 'a', stack symbol z_1 will be pushed to the stack. Number of 'b' are same as number of 'a'. Number of z_1 pushed to the stack will be equal to number of 'a', *i.e.*, number of 'b'.
4. For traversal of each 'b' in the input tape one z_1 will be popped from the stack.
5. When first 'c' will be traversed the stack top will be z_0. Upon traversing 'c', stack symbol z_2 will be pushed to the stack. Upon traversing 'd' those z_2's will be popped from the stack.
6. When first 'c' will be traversed the stack top will be z_0. Upon traversal 'c', stack symbol z_2 will be pushed to the stack. Upon traversing 'd' those z_2's will be popped from the stack.

By these processes all the string will be traversed. The PDA for accepting the string $a^n b^n c^m d^m / m, n \geq 1$ will be

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

$\Sigma = \{a, b, c, d\}$

$\Gamma = \{z_0, z_1, z_2\}$

$q_0 = \{q_0\}$

$z_0 = \{z_0\}$

$F = \{q_f\}$

The transition function of constructing PDA will be

$\delta(q_0, a, z_0) = (q_0, z_1 z_0)$

$\delta(q_0, a, z_1) = (q_0, z_1 z_1)$

$\delta(q_0, b, z_1) = (q_1, \varepsilon)$

$\delta(q_1, b, z_1) = (q_1, \varepsilon)$

$\delta(q_1, c, z_0) = (q_2, z_2 z_0)$

$\delta(q_2, c, z_2) = (q_2, z_2 z_2)$

$\delta(q_2, d, z_2) = (q_3, \varepsilon)$

$\delta(q_3, d, z_2) = (q_3, \varepsilon)$

$\delta(q_3, \varepsilon, z_0) = (q_3, \varepsilon)$ accepted by empty stack,

$\delta(q_3, \varepsilon, z_0) = (q_f, z_0)$ accepted by final state.

Que 4.10. **What is difference between deterministic and non-deterministic PDA ? Give example of each.**

Answer

S. No.	Deterministic PDA	Non-deterministic PDA
1.	A deterministic PDA is one for which every string has a unique path.	A non-deterministic PDA is one for which at certain time, we may have to select a particular path among the existing ones.
2.	There is at most one legal sequence of transitions that can be followed for any input.	There can be more than one sequence of transition that can be followed for any input.
3.	DPDA contains one-stack.	Non DPDA has two-stack.
4.	There is only one-stack in DPDA for energy state.	Every state has its own stack.
5.	The example of DPDA is PDA accepting string like wcw^R where 'w' is a string and w^R is reverse of string.	The example of NDPDA is a PDA accepting string like ww^R where w^R is reverse of w.

PART-2

Pushdown Automata for Context Free Language.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 4.11. **Obtain PDA to accept all strings generated by the language $\{a^n b^m a^n, m, n > = 1\}$.** **AKTU 2015-16, Marks 10**

Answer

Let PDA be $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$.

Here δ is defined as follows :

1. $(q_0, a, \varepsilon) = (q_0, a)$
2. $(q_0, a, a) = (q_0, a)$
3. $(q_0, b, a) = (q_1, a)$
4. $(q_1, b, a) = (q_1, a)$
5. $(q_1, a, a) = (q_2, \varepsilon)$

6. $(q_2, a, a) = (q_2, \varepsilon)$
7. $(q_2, \varepsilon, \varepsilon) = (f, \varepsilon)$

$Q = \{q_0, q_1, q_2, f\}$
$\Sigma = \{a, b\}, \Gamma = \{a, b\}$
$F = \{f\}$

Here we are reading all a's first and putting them on the stack. After it all b's are read without making any change on stack. Now again we are reading a's from input tape and one 'a' from stack is popped for every single 'a' of input tape. If all a's are popped and there is no 'a' on input tape then final state is achieved.

Que 4.12. **Construct PDA to accept.**

$L = \{0^n 1^n : n \geq 0\}$ **AKTU 2016-17, Marks 7.5**

Answer

1. By the analysis of language, it is clear that PDA for the given language will accept the set of strings in which every string start with '0' followed by any number of 0's and followed by same number of 1's.
2. The PDA should accept the string like 01, 0011, 00011 ..... and reject ε, 00, 11, 001, 0101, 1100 etc.

Let the PDA

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

Here, consider the stack starting symbol is Z.

The tuples for PDA

$P = (\{q_0, q_1, q_2\}, \{0, 1\}, q_0, \{Z\}, \delta, Z, q_2)$

i.e., $Q = \{q_0, q_1, q_2\}$
$\Sigma = \{0, 1\}$
$\Gamma = \{A, Z\}$
$q_0 = q_0$
$Z_0 = Z$
$F = q_2$

$\delta(q_0, 0, Z) = (q_0, 0Z)$
$\delta(q_0, 0, 0) = (q_0, 00)$
$\delta(q_0, 1, 0) = (q_1, \varepsilon)$
$\delta(q_1, \varepsilon, 0) = (q_2, \varepsilon) \rightarrow$ Accepted

Que 4.13. **Construct PDA for following :**

$L = \{a^n c\, b^{2n} \mid n >= 1\}$ over alphabet $\Sigma = \{a, b, c\}$. Specify the acceptance state.

Answer

This PDA has input set $\{a, b, c\}$ and the number of b's following total number of a's is double and the a's and b's are separated by single c.

Step 1 : Initially we will read string of a's and push those a's onto the stack.

$\delta(q_0, a, Z_0) = (q_0, aZ_0)$
$\delta(q_0, a, a) = (q_0, aa)$

Step 2 : Now we will read single c. We will not perform any push or pop operation at this step.

$\delta(q_0, c, a) = (q_1, a)$

At this step the state is changed to q_1.

Step 3 : Being in state q_1 if we read input symbol b then we will read two consecutive b's without pushing or popping anything, simply read two b's and

$\delta(q_1, b, a) = (q_2, a)$
$\delta(q_2, b, a) = (q_3, a)$

Then read the third consecutive b and pop single a from the stack. This means checking presence of two b's against single a.

$\delta(q_3, b, a) = (q_3, \varepsilon)$

Step 4 : When the input read is a blank character and stack is empty.

Then we reach to accept $\delta(q_3, \varepsilon, z_0) = (q_4, \varepsilon)$.
The state q_4 is accept state.

Que 4.14. **Construct a deterministic PDA for the following language :**

$L = \{x \in \{a, b\}^* \mid n_a(x) \neq n_b(x)\}$
where $n_a(x)$: number of a's in the string x
$n_b(x)$: number of b's in the string x

Answer

The tuples will be :

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}, \Sigma = \{a, b\}$
$\Gamma = \{z_0, x\}, q_0 = \{q_0\}, z_0 = \{z_0\}, F = \{q_1, q_3\}$
$\delta(q_0, \varepsilon, z_0) = \{(q_1, z_0), (q_2, z_0), (q_3, z_0)\}$
$\delta(q_1, a, z_0) = (q_1, xz_0)$
$\delta(q_1, a, x) = (q_1, xx)$
$\delta(q_1, b, x) = (q_3, b, x) = (q_3, \varepsilon)$
$\delta(q_3, \varepsilon, z_0) = (q_1, z_0)$
$\delta(q_2, a, z_0) = (q_2, z_0)$
$\delta(q_2, b, z_0) = (q_4, xz_0)$
$\delta(q_2, b, x) = (q_4, x\,x) = (q_4, \varepsilon)$
$\delta(q_5, \varepsilon, z_0) = (q_2, \varepsilon)$

Que 4.15. **Design PDA for Language WcW^R, $W \in (a, b)^*$.**

AKTU 2018-19, Marks 10

OR

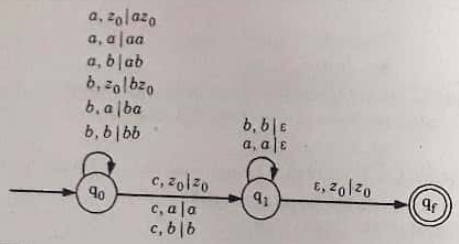
What is Push Down Automata (PDA) ? Design the PDA for the language $L = \{wcw^R \mid w \in \{a, b\}^*\}$ **AKTU 2017-18, Marks 07**

Answer

PDA : Refer Q. 4.1, Page 4–2A, Unit-4.

$L = \{WcW^R \mid W \in \{a, b\}^*\}$

PDA for L is given as :



Que 4.16. **How to convert an equivalent PDA of a context free grammar ?**

Answer

Step I : Convert the PDA into Greibach Normal Form (GNF).

Step II : First the start symbol S of the CFG is put to the stack by transition function

$\delta(q_0, \varepsilon, z_0) \rightarrow (q_1, Sz_0)$

Step III : For a production in the form $\{NT_1\} \rightarrow \{\text{Single } T\}\ \{\text{String of } NT\}$, the transitional function will be

$\delta(q_1, T, NT_1) \rightarrow (q_1, \text{String of } NT)$.

Step IV : For a production in the form $\{NT_1\} \rightarrow \{\text{Single } T\}$, the transitional function will be $\delta(q_1, T, NT_1) \rightarrow (q_1, \varepsilon)$.

Step V : For accepting a string two transitional functions are added, one for accepted by empty stack and one for accepted by final state.

Que 4.17. **Convert the following CFG into PDA :**

$S \rightarrow aSa/aA/Bb$, $A \rightarrow aA/a$, $B \rightarrow Bb/A$ **AKTU 2014-15, Marks 05**

Answer

The PDA for given CFG will be :

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

$Q = \{q\}$

$\Sigma = \{a, b\}$

$\Gamma = \{S, A, B, a, b\}$

$q_0 = \{q\}$

$Z_0 = \{S\}$

$F = \{q\}$

The δ will be defined as :

$\delta(q, \varepsilon, S) = (q, aSa)$

$\delta(q, \varepsilon, S) = (q, aA)$

$\delta(q, \varepsilon, S) = (q, Bb)$

$\delta(q, \varepsilon, A) = (q, aA)$

$\delta(q, \varepsilon, A) = (q, a)$

$\delta(q, \varepsilon, B) = (q, Bb)$

$\delta(q, \varepsilon, B) = (q, A)$

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

$\delta(q, b, b) = (q, \varepsilon)$

Que 4.18. **Construct a PDA from the following CFG.**

$G = (\{S, X\}, \{a, b\}, P, S)$ where the productions are :

$S \rightarrow XS \mid \varepsilon, A \rightarrow aXb \mid Ab \mid ab$ **AKTU 2016-17, Marks 10**

Answer

Given : $G = (\{S, X\}, \{a, b\}, P, S)$

where the productions are

$S \rightarrow XS \mid \varepsilon, A \rightarrow aXb \mid Ab \mid ab$

Let the equivalent PDA,

$P = (\{q\}, \{a, b\}, \{a, b, X, S\}, \delta, q, S)$

where $\delta \rightarrow$

$\delta(q, \varepsilon, S) = \{(q, XS), (q, \varepsilon)\}$

$\delta(q, \varepsilon, X) = \{(q, aXb), (q, Xb), (q, ab)\}$

$\delta(q, a, a) = \{(q, \varepsilon)\}$

$\delta(q, 1, 1) = \{(q, \varepsilon)\}$

Que 4.19. **Consider the CFG $(\{S, A, B\}, \{a, b\}, P, S)$ where productions P are as follows : $S \rightarrow aABB/aAA$, $A \rightarrow aBB/a$, $B \rightarrow bBB/A$. Convert the given grammar to PDA that accept the same language by empty stack.** **AKTU 2015-16, Marks 10**

Answer

The PDA is given by $P(\{q\}, \{a, b\}, \{a, b, S, A, B\}, \delta, q, s,)$
where δ is given by

$$\delta(q, \varepsilon, S) \rightarrow \{(q, aABB), (q, aAA)\}$$
$$\delta(q, \varepsilon, A) \rightarrow \{(q, aBB), (q, a)\}$$
$$\delta(q, \varepsilon, B) \rightarrow \{(q, bBB), (q, A)\}$$
$$\delta(q, a, a) \rightarrow \{(q, \varepsilon)\}$$
$$\delta(q, b, b) \rightarrow \{(q, \varepsilon)\}$$

Que 4.20. **Construct a PDA M equivalent to grammar with following productions :**
$S \rightarrow aAA, A \rightarrow aS|bS|a$
Also, check whether the string '*abaaaa*' is in *M* or not.

AKTU 2018-19, Marks 10

Answer

Let the PDA for given CFG be
$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$
$Q = \{q\}$
$\Sigma = \{a, b\}$
$\Gamma = \{S, A, a, b\}$
$q_0 = \{q\}$
$z_0 = \{S\}$
$F = \{q\}$
δ will be defined as :
$\delta(q, \epsilon, S) = \{(q, aAA)\}$
$\delta(q, \epsilon, A) = \{(q, aS), (q, bS), (q, a)\}$
$\delta(q, a, a) = \{(q, \varepsilon)\}$
$\delta(q, b, b) = \{(q, \varepsilon)\}$

Check : Given string is '*abaaaa*'

$(q, abaaaa, S)$
$\vdash (q, abaaaa, aAA) \vdash (q, baaaa, AA)$
$\vdash (q, baaaa, bSA) \vdash (q, aaaa, SA)$
$\vdash (q, aaaa, aAAA) \vdash (q, aaa, AAA)$
$\vdash (q, aaa, aAA) \vdash (q, aa, AA)$
$\vdash (q, aa, aA) \vdash (q, a, A)$
$\vdash (q, a, a) \vdash (q, \varepsilon, \varepsilon)$

Thus, the string '*abaaaa*' is in *M*.

Que 4.21. **Construct PDA equivalent to the following CFG production : $S \rightarrow aAA, A \rightarrow aS|bS|a$**

AKTU 2017-18, Marks 07

Answer

PDA for given CFG will be :

$P = (Q, \Sigma, T, \delta, q_0, Z_0, F)$
$Q = (q), \Sigma = (a, b), T = (S, A, a, b), q_0 = (q)$
$Z_0 = (S), F = (q)$

The δ will be defined as :

$\delta(q, \varepsilon, S) = (q, aAA)$
$\delta(q, \varepsilon, A) = (q, aS)$
$\delta(q, \varepsilon, A) = (q, bS)$
$\delta(q, \varepsilon, A) = (q, a)$
$\delta(q, \varepsilon, a) = (q, \varepsilon)$
$\delta(q, \varepsilon, b) = (q, \varepsilon)$

PART-3

Context Free Grammar for Pushdown Automata.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 4.22. **Convert the given PDA *M* to equivalent context free grammar. The PDA *M* is defined as M $(\{q_0, q_1\}, \{0, 1\}, \{X, Z_0\}, \delta, q_0, Z_0, \{q_1\})$ where δ is given as follows :**

$$(q_0, 1, Z_0) = (q_0, XZ_0)$$
$$(q_0, 1, X) = (q_0, XX)$$
$$(q_0, 0, X) = (q_0, X)$$
$$(q_0, \varepsilon, X) = (q_1, \varepsilon)$$
$$(q_1, \varepsilon, X) = (q_1, \varepsilon)$$
$$(q_1, 0, X) = (q_1, XX)$$
$$(q_1, 0, Z_0) = (q_1, \varepsilon)$$

Answer

Step 1 : Applying Rule 1 : $S \rightarrow [q_0, Z_0, q_0] \mid [q_0, Z_0, q_1]$

Step 2 : $\delta(q_0, 1, Z_0) = (q_0, XZ_0)$ the production will be

$$[q_0, Z_0, q_0] \rightarrow 1[q_0, X, q_0] [q_0, Z_0, q_0]$$
$$[q_0, Z_0, q_0] \rightarrow 1[q_0, X, q_1] [q_1, Z_0, q_0]$$
$$[q_0, Z_0, q_1] \rightarrow 1[q_0, X, q_0] [q_0, Z_0, q_1]$$
$$[q_0, Z_0, q_1] \rightarrow 1[q_0, X, q_1] [q_1, Z_0, q_1]$$

Step 3 : $\delta(q_0, 1, X) = (q_0, XX)$ production will be

$$[q_0, X, q_0] \rightarrow 1[q_0, X, q_0] [q_0, X, q_0]$$

$[q_0, X, q_0] \rightarrow 1[q_0, X, q_1]\,[q_1, X, q_0]$
$[q_0, X, q_1] \rightarrow 1[q_0, X, q_0]\,[q_0, X, q_1]$
$[q_0, X, q_1] \rightarrow 1[q_0, X, q_1]\,[q_1, X, q_1]$

Step 4 : $\delta(q_0, 0, X) = (q_0, X)$ production will be

$[q_0, X, q_0] \rightarrow 0[q_0, X, q_0]$
$[q_0, X, q_0] \rightarrow 0[q_0, X, q_1]$
$[q_0, X, q_1] \rightarrow 0[q_0, X, q_1]$
$[q_0, X, q_1] \rightarrow 0[q_1, X, q_1]$

Step 5 : $(q_0, \varepsilon, X) = (q_1, \varepsilon)$

$[q_0, X, q_1] \rightarrow \varepsilon$

Step 6 : $\delta(q_1, \varepsilon, X) = (q_1, \varepsilon)$

$[q_1, X, q_1] \rightarrow \varepsilon$

Step 7 : $\delta(q_1, 0, X) = (q_1, XX)$

$[q_1, X, q_0] \rightarrow 0[q_1, X, q_0]\,[q_0, X, q_0]$
$[q_1, X, q_0] \rightarrow 0[q_1, X, q_1]\,[q_1, X, q_0]$
$[q_1, X, q_1] \rightarrow 0[q_1, X, q_0]\,[q_0, X, q_1]$
$[q_1, X, q_1] \rightarrow 0[q_1, X, q_1]\,[q_1, X, q_1]$

Step 8 : $\delta(q_1, 0, Z_0) = (q_1, \varepsilon)$

$[q_1, Z_0, q_1] \rightarrow \varepsilon$
$[q_0, Z_0, q_0] \rightarrow 1[q_0, X, q_1]\,[q_1, Z_0, q_1]/\varepsilon$
$[q_0, X, q_1] \rightarrow 1[q_0, X, q_1]\,[q_1, X, q_1]\,/0[q_1, X, q_1]$
$[q_1, X, q_1] \rightarrow 1$
$[q_0, Z_0, q_0] \rightarrow 0[q_0, Z_0, q_0]$
$[q_1, Z_0, q_1] \rightarrow \varepsilon$
$[q_1, X, q_1] \rightarrow 0[q_0, X, q_1]\,[q_0, X, q_1]$

Let,
$(q_0, Z_0, q_0) = S$
$(q_0, X, q_1) = A$
$(q_1, Z_0, q_1) = B$
$(q_1, X, q_1) = D$
$(q_1, Z_0, q_1) = E$

CFG for given PDA

$S \rightarrow 1AE \mid \varepsilon \mid 0S$
$A \rightarrow 1AD \mid 0D$
$D \rightarrow 1 \mid 0AD$
$E \rightarrow \varepsilon$

Que 4.23. **Consider given PDA :**

$M = (\{q_0\}, \{0, 1\}, \{a, b, z_0\}, \delta, q_0, z_0, \phi)$ δ is defined as :

$\delta\ (q_0, a, z_0) = \{(q_0, az_0)\}$

$\delta\ (q_0, 1, z_0) = \{(q_0, bz_0)\}$
$\delta\ (q_0, 0, z_0) = \{(q_0, aa)\}$
$\delta\ (q_0, 1, b) = \{(q_0, bb)\}$
$\delta\ (q_0, 0, b) = \{(q_0, \varepsilon)\}$
$\delta\ (q_0, 1, a) = \{(q_0, \varepsilon)\}$
$\delta\ (q_0, \varepsilon, z_0) = \{(q_0, \varepsilon)\}$

Convert given PDA M to corresponding CFG.

Answer

Step 1 :

$S \rightarrow [q_0, z_0, q_0]$

Step 2 : For move

$\delta(q_0, a, z_0) = (q_0, az_0)$ we get the following production

$(q_0, z_0, q_0) \rightarrow a\,(q_0, z_0, q_0)\,(q_0, a_0, q_0)$

Step 3 : For move

$\delta(q_0, 1, z_0) = (q_0, bz_0)$ we get the following production

$(q_0, z_0, q_0) \rightarrow 1\,(q_0, z_0, q_0)\,(q_0, b, q_0)$

Step 4 : For move

$\delta(q_0, 0, z_0) = (q_0, a\,a)$

$(q_0, z_0, q_0) \rightarrow 0\,(q_0, a, q)\,(q_0, a, q_0)$

Step 5 : $\delta(q_0, 1, b) = (q_0, bb)$

$(q_0, b, q_0) \rightarrow 1\,(q_0, b, q_0)\,(q_0, b, q_0)$

Step 6 : For move

$\delta(q_0, 0, z_0) = (q_0, \varepsilon)$

$(q_0, b, q_0) \rightarrow 0$

Let
$(q_0, z_0, q_0) = A$
$(q_0, a, z_0) = B$
$(q_0\, b\, z_0) = C$

Step 7 : $(q_0, a, z_0) = 1$

$S \rightarrow A$
$A \rightarrow aAB \mid 1AC \mid 0BB \mid \varepsilon$
$B \rightarrow 1$
$C \rightarrow 1\,CC \mid 0$

Step 8 : $(q_0, z_0, z_0) = \varepsilon$

CFG for given PDA

$S \rightarrow aSB \mid 1SC \mid 0BB \mid \varepsilon$
$B \rightarrow 1$
$C \rightarrow 1\,CC \mid 0$

PART-4

Two Stack Pushdown Automata.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 4.24. Explain two-stack PDA with example.

Answer

1. A two-stack push down automaton (Two-stack PDA) is similar to a PDA, but it has two stacks instead of one.
2. In each transition, we must denote the pop and push action on both stacks.
3. A two-stack pushdown automaton to be a six tuple

$$M = (K, \Sigma, \Gamma, \Delta, S, F)$$

where, K is a finite set of states,

Σ is an alphabet (the input symbols),

Γ is an alphabet (the stack symbols),

$S \in K$ is the initial state,

$F \subseteq K$ is the set of final states, and

Δ, the transition relation, is a finite subset of $(K \times (\Sigma \cup \{e\}) \times \Gamma^* \times \Gamma^*) \times (K \times \Gamma^* \times \Gamma^*)$, where the third parameter pops the first stack, the fourth parameter pops the second stack, the fifth parameter pushes a symbol onto the first stack, and the sixth parameter pushes a symbol onto the second stack.

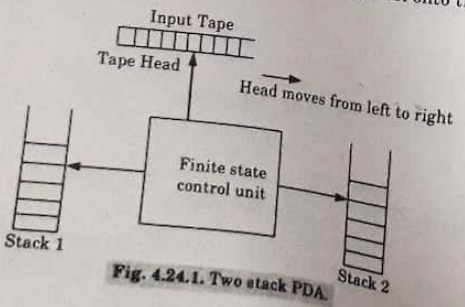


Fig. 4.24.1. Two stack PDA.

4. A move of two-stack PDA includes :
   a. Change state.
   b. The input symbol read.
   c. Replace the symbol of each stack with a string of zero or more stack symbol.
5. There are certain languages which cannot be accepted by pushdown automata as there is only one stack. This is especially when we want to perform two checks on the input string.
6. But if we use two stacks there then those languages can be accepted by PDA.

For example :

1. Let consider a language $L = \{a^n b^n c^n \mid n \geq 1\}$. Now if we use one stack here then the language cannot be accepted by PDA.
2. Suppose with one stack we will push all a's onto the stack on reading them. Then on reading b's we will pop each a. This helps in making a check on equal number of a's and b's.

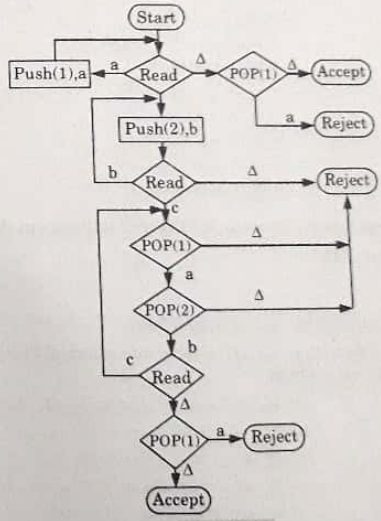


Fig. 4.24.2.

3. **But now when we read out c's the single stack is empty and we cannot make out the equality of a's and b's with c's.**

4. Thus it is not possible for us to design a PDA with one stack which accepts language $L = a^n b^n c^n$.
5. But if we use two stacks one for storing all a's. Other for storing all the b's then on reading each c we can pop each a from first stack and each b from second stack. This helps in identifying whether or not there are equal number of a's, b's and c's coming in order.
6. This also means that PDA with two stacks has more power than PDA with one stack.
7. The PDA with two stacks accepting $a^n b^n c^n$ is given in Fig. 4.24.2.

Que 4.25. **Construct a two-stack PDA that will accept the language $L = \{a^n b^n c^n \mid n \in N\}$.**

Answer

The two stack PDA will be expressed as follows :

$$\Sigma = \{a, b, c\}, \Gamma = \{a, b\}, K = \{q_0, q_1, q_2, f\},$$
$$s = \{q_0\}, F = \{f\}$$
$$\Delta = (q_0, a, \varepsilon, \varepsilon)(q_0, a, \varepsilon)$$
$$= (q_0, a, a, \varepsilon)(q_0, aa, \varepsilon)$$
$$= (q_0, \varepsilon, \varepsilon, \varepsilon)(q_1, \varepsilon, \varepsilon)$$
$$= (q_1, b, \varepsilon, \varepsilon)(q_1, \varepsilon, b)$$
$$= (q_1, b, \varepsilon, b)(q_1, \varepsilon, bb)$$
$$= (q_1, c, a, b)(q_2, \varepsilon, \varepsilon)$$
$$= (q_2, c, a, b)(q_2, \varepsilon, \varepsilon)$$
$$= (q_2, \varepsilon, \varepsilon, \varepsilon)(q_2, \varepsilon, \varepsilon)$$

Que 4.26. **Construct a two-stack PDA that will accept the language $L = \{a^n b^n a^n b^n \mid n \geq 1\}$.**

Answer

The two-stack PDA will be expressed as follows :

This is another problem that was not solvable with a normal PDA, but solvable with a TM and two-stack PDA.

$$\Sigma = \{a, b\}, \Gamma = \{a, b\}, K = \{q_0, q_1, q_2, q_3, f\},$$
$$s = \{q_0\}, F = \{f\}$$
$$\Delta = (q_0, a, \varepsilon, \varepsilon)(q_0, a, \varepsilon)$$
$$= (q_0, a, a, \varepsilon)(q_0, aa, \varepsilon)$$
$$= (q_0, \varepsilon, \varepsilon, \varepsilon)(q_1, \varepsilon, \varepsilon)$$
$$= (q_1, b, \varepsilon, \varepsilon)(q_1, \varepsilon, b)$$
$$= (q_1, b, \varepsilon, b)(q_1, \varepsilon, bb)$$
$$= (q_1, a, a, \varepsilon)(q_2, \varepsilon, \varepsilon)$$

Que 4.27. **Design a two stack PDA for the language $L = a^n b^m c^n d^m$ where $n, m \geq 1$.** **AKTU 2018-19, Marks 10**

OR

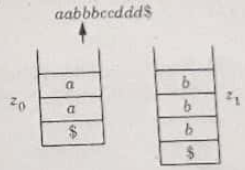
Differentiate between deterministic PDA (DPDA) and non-deterministic PDA (NPDA) with suitable example. Also discuss two stack PDA with example. **AKTU 2017-18, Marks 07**

Answer

$$L = a^n b^m c^n d^m \quad n, m \geq 1$$

The PDA accepts the string like $abcd$, $aabccd$, $aabbccdd$, $aabbbccddd$ etc.

Consider a string



Let the PDA :

$$P = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$$
$$Q = \{q_0, q_1, q_2, q_3\}$$
$$\Sigma = \{a, b\}$$
$$\Gamma = \{z\}$$
$$z = \{z_0, z_1\}$$
$$q_0 = \{q_0\}$$
$$F = \{q_f\}$$
$$\delta(q_0, a, z_0, z_1) \rightarrow (q_1, az_0, z_1)$$
$$\delta(q_1, a, a, z_1) \rightarrow (q_1, aa, z_1)$$
$$\delta(q_1, b, a, z_1) \rightarrow (q_2, aa, bz_1)$$
$$\delta(q_2, b, a, b) \rightarrow (q_2, aa, bb)$$
$$\delta(q_2, c, a, b) \rightarrow (q_3, \varepsilon, bb)$$
$$\delta(q_3, d, z_1, b) \rightarrow (q_f, \varepsilon, \varepsilon)$$
$$\delta(q_3, \$, z_0, z_1) \rightarrow (q_f, \varepsilon, \varepsilon) \rightarrow \text{Accepted}$$

Difference : Refer Q. 4.10, Page 4–8A, Unit-4.

Two stack PDA : Refer Q. 4.24, Page 4–18A, Unit-4.

PART-5

Pumping Lemma for CFL, Closure Properties of CFL, Decision Problems of CFL, Programming Problems based on the Properties of CFLs.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 4.28. State and prove pumping Lemma for CFL.

Answer

Statement : Let L be a CFL. Then we can find a natural number n such that

a. Every $z \in L$ with $z \geq n$ can be written as $w = uvwxy$, for some strings u, v, w, x, y.

b. $|vx| \geq 1$

c. $|vwx| \leq n$.

d. $uv^kwx^ky \in L$ for all $k \geq 0$.

Proof :

1. We can prove that if G is a context free grammar, then we can find a context free grammar G_1 having no null productions such that $L(G_1) = L(G) - \{\varepsilon\}$.
2. If null string (Λ or ε) belongs to the language set L, we will consider $L - \{\varepsilon\}$ and will construct CNF of the grammar G generating $L - \{\varepsilon\}$ [If null string does not belong to L, we will construct CNF of G generating L].
3. Let number of non-terminals (V_N) of the grammar is m. $|V_N| = m$ and $n = 2^m$. To prove that n is the required number, we start with a string z where $z \in L$, $|z| \geq 2^m$.
4. Construct a derivation tree or parse tree T of the string z.
5. If the length of the longest path in T is almost m, then we can write the length of the yield of T, *i.e.*, $|z| \leq 2^{m-1}$. But $|z| \geq 2^m$. So T has a path, say Γ of length greater than or equal to $m + 1$.
6. Path length of greater than or equal to in $m + 1$ needs at least $m + 2$ vertices, where the last vertex is the leaf.
7. Thus in Γ all the labels except the last one are variables. As $|N| = m$, in $m + 1$ labels of the parse tree T, some labels are repeated.
8. Let v_1 and v_2 be the vertices with repeated label (let A) obtained at the time of upward traversal from leaf of Γ.

9. Between v_1 and v_2, let v_1 is nearer to root. In Γ_1 the portion of the path from v_1 to leaf has only one label repeated (let A), so its length is almost $m + 1$.
10. Let T_1 and T_2 are two subtrees taking root as v_1 and v_2 respectively. Let z_1 and w are the yields of T_1 and T_2, respectively.
11. As Γ is the longest path in T, v_1 to leaf is the longest path of T_1 and its length is almost in $m + 1$. So the length of z_1 (yield of T_1), *i.e.*, $|z_1| \leq 2^m$.
12. As z is the yield of T and T_1 is a proper subtree of T. Yield of T_1, is z_1. So yield of T, *i.e.*, z can be written as $z = uz_1y$.

    As z_1 is the yield of T_1, and T_2 is a proper subtree of T_1. Yield of T_2 is w. So yield of T_1, *i.e.*, z_1 can be written as $z_1 = vwx$.
13. T_2 is a proper subtree of T_1, so $|vwx| > |w|$. So $|vx| \geq 1$.
14. Thus we can write $z = uvwxy$ with $|vwx| \leq n$ and $|vx| \geq 1$.

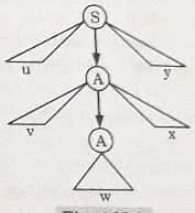


Fig. 4.28.1.

15. As T is tree with root S(S tree) and T_1, T_2 are tree with root B(B tree), we can write $S \overset{*}{\Rightarrow} uAy$, $A \overset{*}{\Rightarrow} vAx$ and, $A \overset{*}{\Rightarrow} w$.
16. As $S \overset{*}{\Rightarrow} uAy \Rightarrow uwy$, $uv^0wx^0y \in L$. For $k \geq 1$, $S \overset{*}{\Rightarrow} uAy \overset{*}{\Rightarrow} uv^kAx^k y \overset{*}{\Rightarrow} uv^kwx^k y \in L$. By this the theorem is proved.

Que 4.29. Prove that $L = \{0^p$ / where p is prime$\}$ is not a CFL.

Answer

Step I : Suppose $L = L(G)$ is context free. Let n be a natural number obtained from the pumping lemma for CFL.

Step II : Let p is a number $> n$, $z = 0^p \in L$. By using pumping lemma for CFL we can write $z = uvwxy$, where $|vr| \geq 1$ and $|vwx| \leq n$.

Step III : Let $k = 0$. From the pumping lemma for CFL, we can write uv^0wx^0y, *i.e.*, $uwy \in L$. As uvy is in the form 0^p, where p is prime; so $|uwy|$ is also a prime number. Let take it as q. Let

$|vx| = r$. Then $|uv^q wx^r y| = q + qr = q(1 + r)$. This is not prime as it has factors q, $(1+r)$ including 1 and $q(1+r)$. So $uv^q wx^q y \notin L$. This is a contradiction. Therefore L is not context free.

Que 4.30. **Find out whether the language $L = \{x^n y^n z^n \mid n \geq 1\}$ is context free or not.** **AKTU 2016-17, Marks 05**

Answer

The language L says that 'x' cannot appear after 'y' and 'z' cannot appear before 'y' and so on.

Step 1 : Assume L is context free and we get a contradiction. Let n be the natural number obtained by using pumping lemma.

Step 2 : Let $Z = x^n y^n z^n$, then $|Z| = 3n > n$. We can write $Z = abcde$, where $|bd| \geq 1$ (*i.e.*, both b and d cannot be ε).

Step 3 : $abcde = x^n y^n z^n$. As we have assumed $1 \leq |bd| \leq n$, then b or d cannot contain all three symbols x, y and z. Therefore,

i. b or d is of the form $x^i y^j$ (or $y^i z^j$) for some i, j such that $i + j \leq n$, or

ii. b or d is a string formed by the repetition of only one symbol among x, y, z.

When b or d is of the form $x^i y^j$, so $x^2 = b^2 = x^i y^j x^i y^j$. As b^2 is a sub-string of ab^2cd^2e so ab^2cd^2e cannot be of the form $x^m y^m z^m$. Therefore $ab^2cd^2e \notin L$.

When both b and d are formed by the repetition of a single symbol, for example $d = x^i$ and $b = y^j$ for some i and j, where $i, j \leq n$ then the string uwy will contain the remaining symbols, say e. Also, e^n will be a sub-string of ace as d does not occur in b or d. The number of occurrence of one of the other two symbols in ace is less than n, and n is the number of occurrence of e. Therefore $ab^0cd^0e = ace \notin L$.

Thus, for any choice of b or d we get a contradiction. Hence, L is not context free language.

Que 4.31. **Given context free grammar, how do you determine that grammar as :**

i. **Empty or non-empty**
ii. **Finite or non-finite**
iii. **Whether a string X belong to language of grammar.**

Answer

i. **Empty or non-empty :** Prove that for a given CFG, there is an algorithm to determine whether or not it can generate any word or not, or it is empty or not.

Proof :

1. We already know about nullable non-terminals, and how to decide whether the start symbol S is nullable :

$$S \overset{*}{\Rightarrow} \varepsilon ?$$

2. We also know that every CFG that does not generate ε can be convert ε-production.
3. Now let us assume that ε is not a word generated by CFG.
4. In this case we can convert the CFG to CNF, preserving the entire language.
5. If there is a production of the form

$$S \to a$$

where a is terminal, then a is word in the language if there is no such production, we then propose the following algorithm :

Step 1 :

a. For each non-terminal N that has some productions of the form

$$V_n \to a$$

where a is terminal or string of terminals, we choose one of them and put a in all other productions having V_n at right side, this eliminating non-terminal V_n altogether.

b. We have changed the grammar so that it no longer accepts the same language.

c. It may no longer be in CNF.

d. That is fine with us.

e. Every word that can be generated from the new grammar could have been generated by the old CFG.

f. If old CFG generated any words, then the new one does also.

Step 2 :

Repeat step 1 until it eliminates S or it eliminates no new non-terminals. If S has been eliminated, then the CFG produces some words, if not then not.

ii. **Finite or non-finite :**

There is an algorithm to decide whether a given CFG generates an infinite language or finite language.

Proof :

1. We assume that grammar contains no ε-productions (if there is any useless or ε-production in the grammar then we have to remove those productions) or no unit-production, and no useless symbols.
2. Now, let us assume that grammar has production like,

$$X \overset{*}{\Rightarrow} xXy$$

3. Since, grammar is assumed to have no ε-production and no unit production x and y cannot both be simultaneously empty.
4. Since X is neither nullable nor a useless symbol, we have :

$$S \overset{*}{\Rightarrow} uXv \overset{*}{\Rightarrow} w$$

and $X \overset{*}{\Rightarrow} z$

where u, v and z are in V^*_t. But then

$$S = \overset{*}{\Rightarrow} uXv \overset{*}{\Rightarrow} ux^n Xy^n v \overset{*}{\Rightarrow} ux^n zy^n v$$

is possible for all n, so that $L(G)$ is infinite.

5. If no variable can ever repeat, then the length of any derivation is bounded by $|V_n|$. In that case, $L(G)$ is finite.
6. Thus, to get an algorithm for determining whether $L(G)$ is finite, we need only to determine whether the grammar has some repeating variables.
7. This can be done simply by drawing a dependency graph for the variables in such a way that there is an edge (X, Y) whenever there is a corresponding production

$$X \to xYy$$

8. Then any variable which is at the base of a cycle is a repeating one.
9. Consequently, the grammar has a repeating variable if and only if the dependency graph has a cycle.
10. Since, we have an algorithm for deciding whether a grammar has a repeating variable, we have an algorithm for determining whether or not $L(G)$ is infinite.

iii. CYK algorithm is used to determine whether a string X belong to language of grammar.

The CYK algorithm : Let the input be a string S consisting of n characters : $a_1 \ldots a_n$. Let the grammar contain r non-terminal symbols $R_1 \ldots R_r$. This grammar contains the subset R_s which is the set of start symbols. Let $P[n, n, r]$ be an array of booleans. Initialize all elements of P to false.

For each $i = 1$ to n

For each unit production $R_j \to a_i$, set $P[i, l, j] =$ true.

For each $i = 2$ to n - - Length of span

For each $j = 1$ to $n - i + 1$ - - Start of span

For each $k = 1$ to $i - 1$ - - Partition of span

For each production $R_A \to R_B R_C$

If $P[j, k, B]$ and $P[j + k, i - k, C]$ then set $P[j, i, A] =$ true

If any of $P[1, n, x]$ is true (x is iterated over the set s, where s are all the indices for R_s).

Then S is a member of language

Else S is not member of language.

Que 4.32. **Discuss the decidability property of CFG.**

OR

Given a context free grammar G. Write an algorithm (if it exist) to determine whether $L(G)$ is infinite or not.

Answer

Decidability property : Following are the decidability properties :

i. **Decide whether grammar is empty or non-empty :** Refer Q. 4.31, Page 4–24A, Unit-4.

ii. **Decide whether grammar is finite or non-finite :** Refer Q. 4.31, Page 4–24A, Unit-4.

iii. **Decide whether a string X belong to language of grammar :** Refer Q. 4.31, Page 4–24A, Unit-4.

Algorithm :

1. Make the grammar simplified by removing unit productions and useless symbols.
2. Check whether there is any self-embedded non-terminal or not by following steps :
   a. Underline some non-terminal say X which is at L.H.S of production rule.
   b. Then put dot over all the X which are at R.H.S throughout the grammar. The dotted X and underlined X are treated as two different symbols.
   c. Put dot over any non-terminal at L.H.S whose R.H.S contains any dotted symbols. Dot this non-terminal throughout the grammar.
   d. If underlined X gets dotted then X is called self-embedded otherwise it is not self-embedded.
3. If the grammar contains any self-embedded non-terminal then it generates infinite languages otherwise it generates finite languages.

VERY IMPORTANT QUESTIONS

Following questions are very important. These questions may be asked in your SESSIONALS as well as UNIVERSITY EXAMINATION.

Q. 1. Define Pushdown Automata (PDA). Differentiate PDA by empty stack and final state by giving their definitions.

Ans. Refer Q. 4.4.

Q. 2. Define Push Down Automata (PDA). Design a PDA for the following language :

$$L = \{a^i b^j c^k \mid i = j \text{ or } j = k\}$$

Ans. Refer Q. 4.5.

Q. 3. Obtain PDA to accept all strings generated by the language $\{a^n b^m a^n, m, n >= 1\}$.

Ans. Refer Q. 4.11.

Q. 4. Convert the following CFG into PDA :

$S \rightarrow aSa/aA/Bb, \quad A \rightarrow aA/a, \quad B \rightarrow Bb/A$

Ans. Refer Q. 4.17.

Q. 5. Construct a PDA from the following CFG.

$G = (\{S, X\} \{a, b\}, P, S)$ where the productions are :

$S \rightarrow XS \mid \varepsilon, A \rightarrow aXb \mid Ab \mid ab$

Ans. Refer Q. 4.18.

Q. 6. Consider the CFG $(\{S, A, B\} \{a, b\}, \{P, S\})$ where productions P are as follows :

$S \rightarrow aABB/aAA$, $A \rightarrow aBB/a$, $B \rightarrow bBB/A$. Convert the given grammar to PDA that accept the same language by empty stack.

Ans. Refer Q. 4.19.

Q. 7. Construct a PDA M equivalent to grammar with following productions :

$S \rightarrow aAA, A \rightarrow aS \mid bS \mid a$

Also, check whether the string '$abaaaa$' is in M or not.

Ans. Refer Q. 4.20.

Q. 8. Design a two stack PDA for the language $L = a^n b^m c^n d^m$ where $n, m \geq 1$.

Ans. Refer Q. 4.27.

Q. 9. State and prove pumping Lemma for CFL.

Ans. Refer Q. 4.28.