Theory of Computer Science

MODULE-1

Alphabet

A finite set of symbols is denoted by ∑.

Language

A language is defined as a set of strings of symbols over an alphabet.

Language of a Machine

Set of all accepted strings of a machine is called language of a machine.

Grammar

Grammar is the set of rules that generates language.

CHOMSKY HIERARCHY OF LANGUAGES

Type 0-Language:-Unrestricted Language, Accepter:-Turing Machine, Generetor:-Unrestricted Grammar

Type 1- Language:-Context Sensitive Language, Accepter:- Linear Bounded Automata, Generetor:-Context Sensitive Grammar

Type 2- Language:-Context Free Language, Accepter:-Push Down Automata, Generetor:-Context Free Grammar

Type 3- Language:-Regular Language, Accepter:- Finite Automata, Generetor:-Regular Grammar

Finite Automata

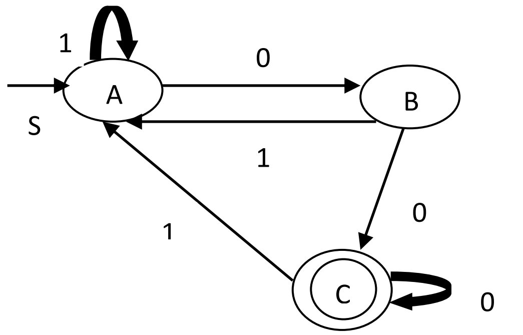
Finite Automata is generally of two types :(i)Deterministic Finite Automata (DFA)(ii)Non Deterministic Finite Automata(NFA)

DFA

DFA is represented formally by a 5-tuple (Q,Σ,δ,q_0,F), where:

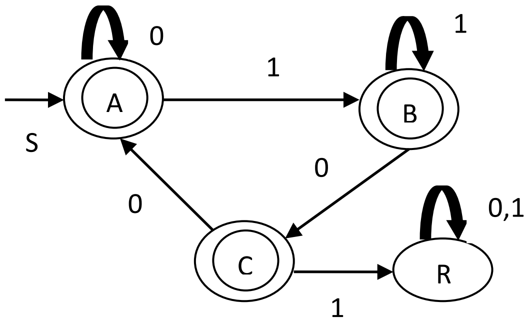
- Q is a finite set of states.
- Σ is a finite set of symbols, called the alphabet of the automaton.
- δ is the transition function, that is, $\delta: Q \times \Sigma \rightarrow Q$.
- q_0 is the start state, that is, the state of the automaton before any input has been processed, where $q_0 \in Q$.
- F is a set of states of Q (i.e. $F \subseteq Q$) called accept states or Final States

1. Construct a DFA that accepts set of all strings over $\sum=\{0,1\}$, ending with 00 ?



State/Input	0	1
→A	B	A
B	C	A
*C	C	A

2. Construct a DFA that accepts set of all strings over $\sum=\{0,1\}$, not containing 101 as a substring ?



State/Input	0	1
→*A	A	B
*B	C	B
*C	A	R
R	R	R

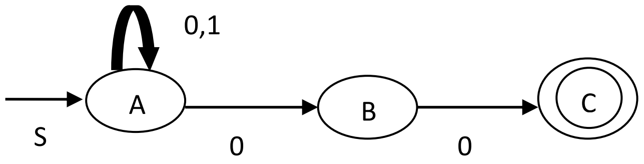
NFA

NFA is represented formally by a 5-tuple (Q,Σ,δ,q_0,F), where:

- Q is a finite set of states.
- Σ is a finite set of symbols, called the alphabet of the automaton.
- δ is the transition function, that is, $\delta: Q \times \Sigma \rightarrow 2^Q$.

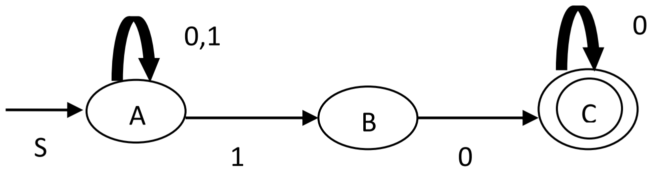
- q_0 is the start state, that is, the state of the automaton before any input has been processed, where $q_0 \in$ Q.
- F is a set of states of Q (i.e. F $\subseteq$ Q) called accept states or Final States

1. Construct a NFA that accepts set of all strings over ∑={0,1}, ending with 00 ?



State/Input	0	1
→A	A,B	A
B	C	_
*C	_	_

2. Construct a NFA that accepts set of all strings over ∑={0,1}, containing 10 as substring ?



State/Input	0	1
→A	A	A,B
B	C	_
*C	C	C

NFA to DFA conversion

Every DFA is an NFA, it is clear that the class of languages accepted by NFA's includes the languages accepted by DFA's.

To convert NFA to DFA we can carry out the following algorithm:

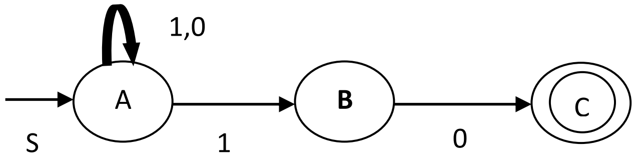
1- begin with $\{q_0\}$, start state, and calculate $\delta(\{q_0\},a)$ for all a in Σ this gives a number of new states Q^-.

2- For each new states Q^-, we again calculate $\delta(Q^-,a)$ for all a in Σ and introduce new states if necessary.

3- Repeat step2 until there are no new states.

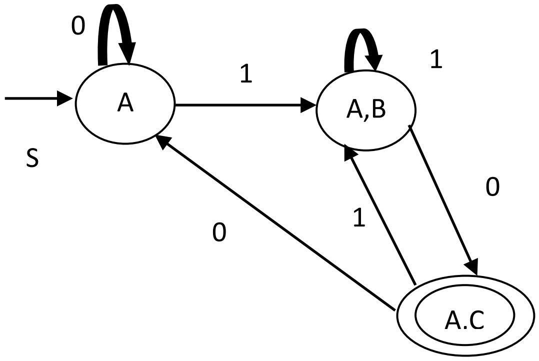
4- Final states of new DFA are the states that contain any final state of the previous NFA.

NFA State Transition Diagram



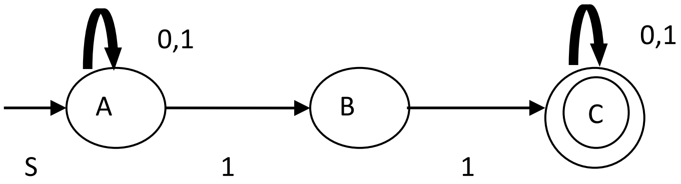
State Transition Table of DFA

State/Input	0	1
→A	A	[A,B](New)
[A,B]	[A,C](New)	[A,B]
* [A,C]	A	[A,B]



Construct a DFA that accepts set of all strings over ∑={0,1}, not containing 11 as substring ?

1. Construct a NFA containing 11 as a substring say N.
2. Covert N to equivalent DFA say D.
3. Construct ID by converting the nonfinal states to final states and final states to nonfinal states in D.

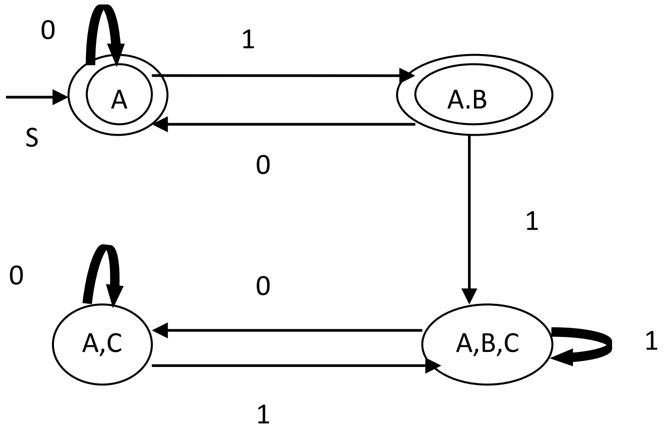


State Transition Table of DFA containing 11 as substring.

State/Input	0	1
→A	A	[A,B](New)
[A,B]	A	[A,B,C](New)
* [A,B,C]	[A,C](New)	[A,B,C]
*[A,C]	[A,C]	[A,B,C]

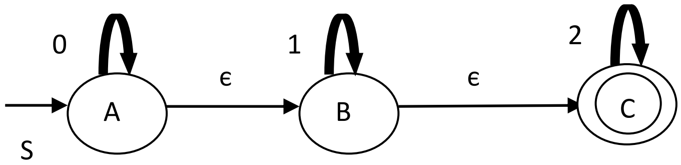
State Transition Table of DFA not containing 11 as substring.

State/Input	0	1
→*A	A	[A,B](New)
* [A,B]	A	[A,B,C](New)
[A,B,C]	[A,C](New)	[A,B,C]
[A,C]	[A,C]	[A,B,C]



Conversion of FA with ϵ-moves to NFA with out ϵ-moves

1. Convert the given FA with ϵ-moves to NFA.



Find ϵ^* (Epsilon Closure) of all states

$\epsilon^*(A)=\{A,B,C\}$

$\epsilon^*(B)=\{B,C\}$

$\epsilon^*(C)=\{C\}$

Pre and Post ϵ^* treatment

Find $\delta(A,0)$

$\epsilon^*(A)=\{A,B,C\}$

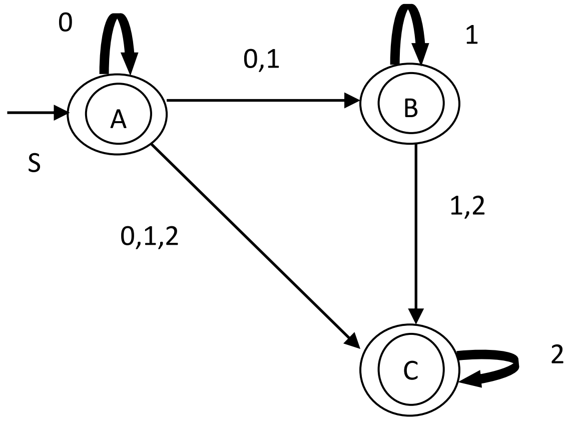
$\text{Input}(\epsilon^*(A),0)=\{A\}$

$\epsilon^*(\ \text{Input}(\epsilon^*(A),0))=\{A,B,C\}$

Like that $\delta(A,1)=\{B,C\}$, $\delta(A,2)=\{C\}$, $\delta(B,0)=\{\}$, $\delta(B,1)=\{A,B\}$, $\delta(B,2)=\{C\}$, $\delta(C,0)=\{\}$, $\delta(C,1)= \{\}$, $\delta(C,2)=\{C\}$.

State Transition Table of NFA with out ϵ-moves

State/Input	0	1	2
→*A	A,B,C	B,C	C
* B	_	B,C	C
*C	_	_	C



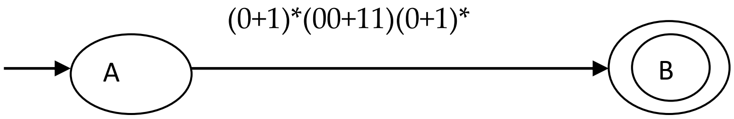
CONSTRUCTION OF FINITE AUTOMATA EQUIVALENT TO A REGULAR EXPRESSION

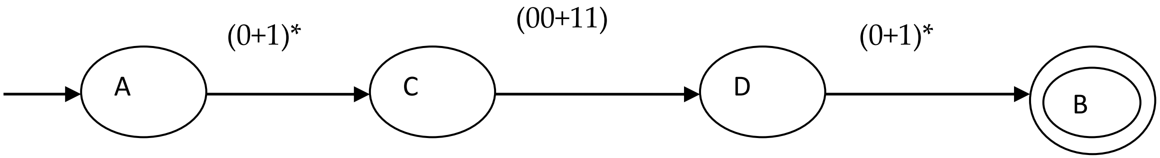
1. Decompose the regular expression into parts.

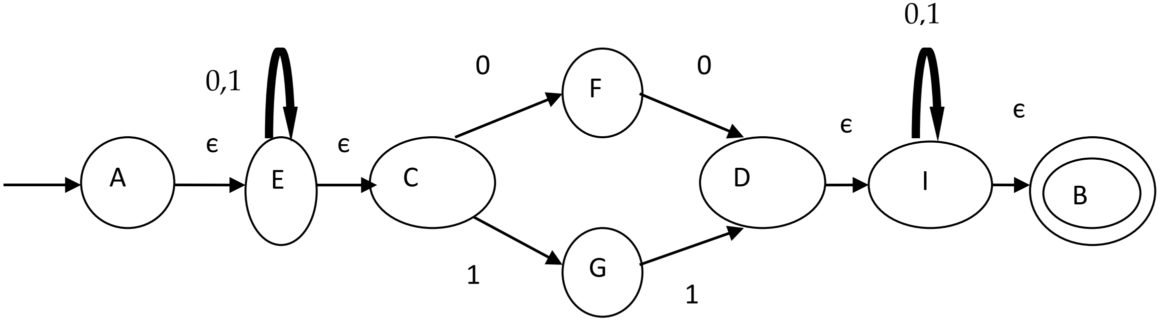
2. Construct the parts and then combine different parts using ϵ transitions.

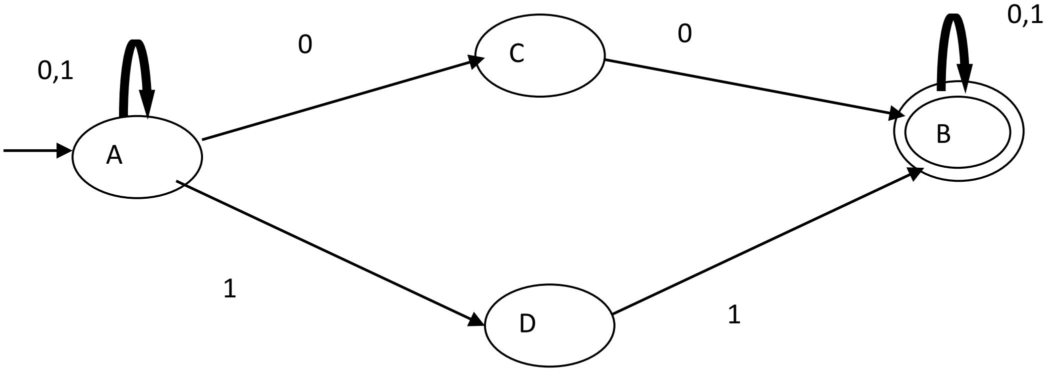
3. Remove unnecessary ϵ transitions.

Q. Construct FA for the RE=(0+1)*(00+11)(0+1)* .







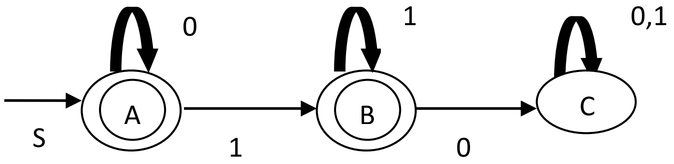


CLOSURE PROPERTY OF REGULAR SET

The closure property of a set signifies the containment of the elements of the set under a particular operation is in the same set. So it defines the set of operations on the regular set that results to an element of regular set.

- ➔ If L_1 is regular and L_2 is regular then $L_1 U L_2$,L_1L_2 and $L_1 \cap L_2$ are also regular.
- ➔ The class of regular sets is closed under substitution.
- ➔ The class of regular sets is closed under kleene closure .
- ➔ The class of regular set is closed under homomorphism and inverse homomorphism.
- ➔ The class of regular sets is closed under complementation.

Regular Expression from FA



Reachability Equations for each State

$A=\epsilon+A0$-------------------(1)

$B=A1+B1$---------------------(2)

$C=B0+C(0+1)$-----------------(3)

Solving (1) by Arden's Lemma

$A= \epsilon 0^*=0^*$--------------------(4)

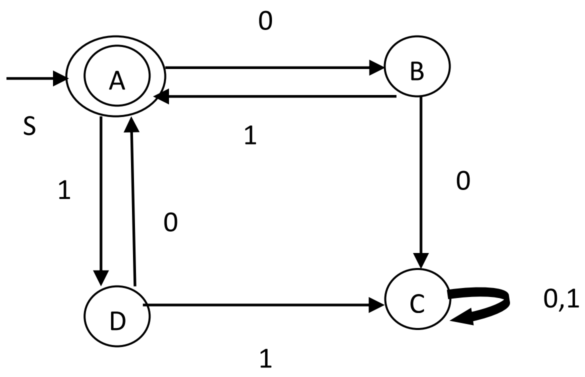
Put (4) in (2)

$B=0^*1+B1$------------------(5)

Solving (5) by Arden's Lemma

$B=0^*11^*=0^*1^+$---------------(6)

Regular Expression is $A+B=0^*(\epsilon+1^*)=0^*1^*$



A=є+B1+D0--------------(1)

B=A0---------------------(2)

C=B0+D1+C(0+1)-------------(3)

D=A1------------------(4)

Put (2) and (4) in (1)

A= є+A01+A10

= є+A(01+10)

= є(01+10)*

=(01+10)*

So the Regular Expression is (01+10)*.

IDENTITIES FOR REGULAR EXPRESSIONS

Φ+R=R

ΦR=RΦ= Φ

єR=Rє=R

є*=є

φ*=є

R+R=R

R*R*=R*

RR*=R*R

(R*)*=R*

Є+RR*=R*=є+R*R

(PQ)*P=P(QP)*

(P+Q)*=(P*Q*)*=(P*+Q*)*

(P+Q)R=PR+QR

R(P+Q)=RP+RQ

Q. Prove (1+00*1)+(1+00*1)(0+10*1)*(0+10*1)=0*1(0+10*1)*

LHS=(1+00*1)+(1+00*1)(0+10*1)*(0+10*1)

=(1+00*1)(є+(0+10*1)*(0+10*1))

=(1+00*1)(0+10*1)*

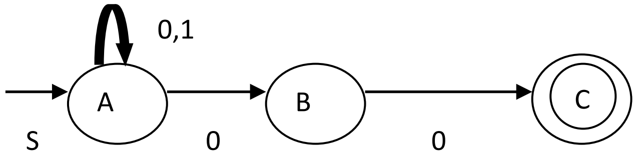
=(є+00*)1(0+10*1)*

=0*1(0+10*1)*=RHS.

<u>Regular Grammar</u>

Right Linear Grammar and Left Linear Grammar From FA

Find RLG and LLG from the given FA.



RLG

A→0A|0B|1A

B→0C|0

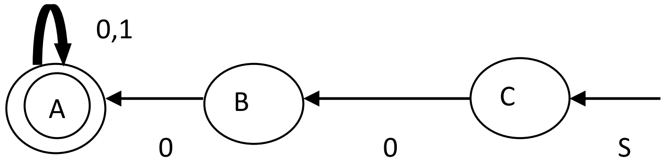
Updating RLG

A→0A|0B|1A

B→0

Constructing LLG

1. Construct the reverse machine of the Given FA(Covert initial state to final state and vice versa).
2. Find RLG.
3. Reverse the position of Nonterminal and terminal at the RHS of the productions.



C→0B

B→0A|0

A→0A|1A|0|1

LLG

C→B0

B→A0|0

A→A0|A1|0|1

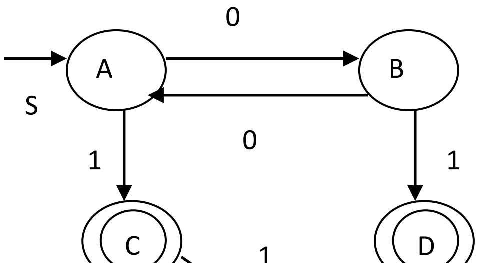
MINIMIZATION OF FA

It is desirable to reduce the number of states as far as possible because it requires less space and is easily processed. Here we are using partitioning method for minimization.

Partitioning Method.

1. At first states are divided into final states and non-final states.
2. Then inputs are applied on the states.
3. Again the groups are converted into groups by looking at their transition to same group of different group. If two states are transiting to same groups by the same input then they are kept in same group or separate groups.

State Minimization of FA

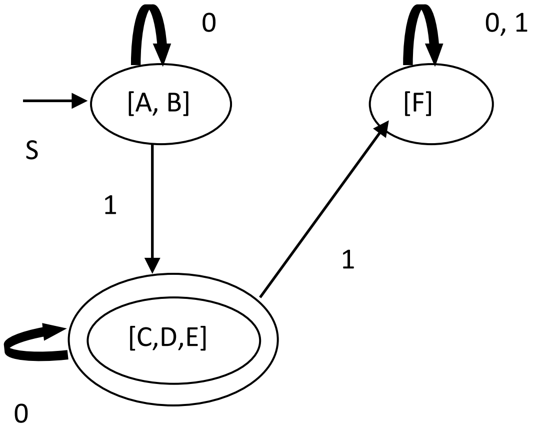


State/Input	0	1
→A	B	*C
B	A	*D
*C	*E	F
*D	*E	F
*E	*E	F
F	F	F

By Partitioning Algorithm

{[A, B, F],*[C, D, E]}

{[A, B], [F],*[C, D, E]}



PUMPING LEMMA FOR REGULAR SET

If L is a Regular Language accepted by some finite automaton D with n number of state, with a string w in L written as w=xyz,then

a. $|y|\geq 1$

b. $|xy|\leq n$

c. $xy^iz \in L$,for all $i\geq 0$.

To prove a language as non regular we have to find out a I value such that condition (c) is not satisfied.

Non regular Languages

$\{a^{2n}|n\geq 1\}$ is not regular.

$\{a^p|P$ is a prime$\}$ is not regular.

$\{a^nb^n|n\geq 0\}$ is not regular.

Q. Show that $L=\{a^p|P$ is a prime$\}$ is not regular.

Solution:

$L=\{a^2,a^3,a^5\ldots\ldots..\}$. let us consider w=aaa.

Let us consider X=a, y=a,z=a

Let us consider i=2

So xy^iz=aaaa doesn't belong to L.

Hence $L=\{a^p|P$ is a prime$\}$ is not regular.

MODULE-2

Generally the context free languages are used for parser design. Also useful in deriving the block structure in programming languages. Context free language is generated by context free grammar and accepted by Push down automata.

Context Free Grammar

A CFG denoted G(V,T,P,S) where V is a finite set of nonterminals,T is a finite set of terminals,P is the finite set of production rules and S is a starting symbol.

Here production rule is of $A \rightarrow \alpha$ form where $A \epsilon V, \alpha \epsilon (VUT)^*$.

Q. Construct a CFG to generate set of palindromes over alphabet {a,b}.

Ans. S→aSa

S→bSb

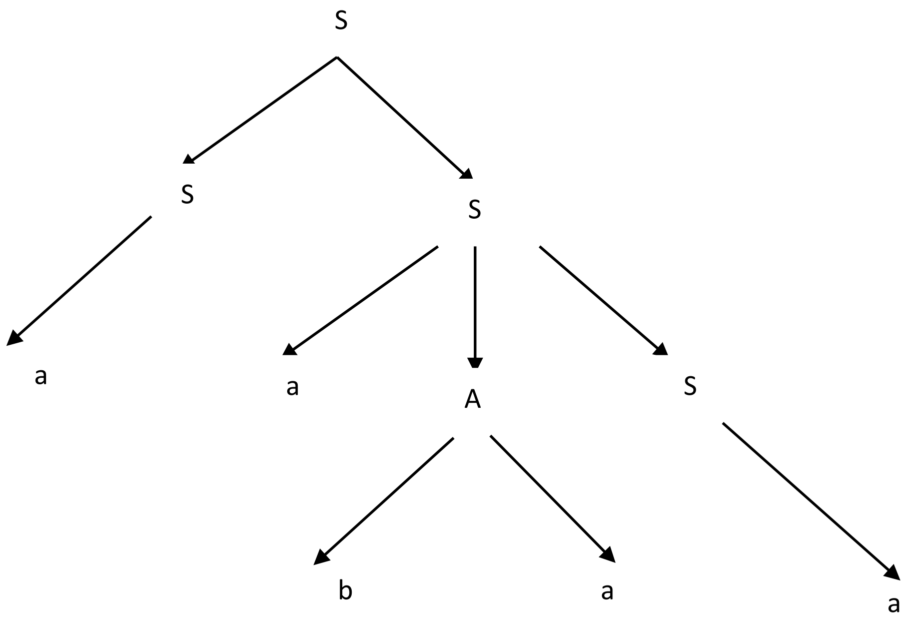
S→ϵ|a|b

Derivation Tree(Parse Tree)

Deviation Tree is a tree representation of derivation steps required to derive a string from the starting symbol S of the CFG G.

Q. Derive aabaa from the grammar G({S,A},{a,b},P,S),where P consists of S→aAS/a/SS,A→SbA/ba

.



The step wise derivation shown in the above tree.

S➔SS

➔aaAS

➔aabaS

➔aabaa

LEFT MOST DERIVATION(LMD) AND RIGHT MOST DERIVATION(RMD)

If at each step in a derivation a production is applied to the left most nonterminal, then the derivation is called LMD, if it is applied to right most nonterminal then the derivation is called RMD.

Q. Derive abaa from the grammar G({S,A},{a,b},P,S),where P consists of S→aAS/a/SS,A→SbA/ba/b

LMD

S➔aAS

➔abaS

➔abaa

RMD

S➔SS

➔Sa

➔aASa

➔aAaa

➔abaa

AMIGUOUS GRAMMAR

A CFG G is called ambiguous iff there exist some words wєL(G), that can be expressed by more than one parse tree. Also there exist more than one LMD or RMD.

Q. Show that the CFG G({S},{a,b,+,*},P,S)where p consists of S→S+S|S*S|a|b.

Let us consider a word a+a*b.

Two LMDs

S→S+S→a+S→a+S*S→a+a*S→a+a*b

S→S*S→S+S*S→a+S*S→a+a*S→a+a*b

SIMPLIFICATION OF CFG

A CFG G can be simplified by eliminating not useful symbols from G. This can be achieved by eliminating useless symbols, Null productions and unit productions.

Eliminating useless symbols

Consider a grammar G having productions

S→AB/a

A→a

Here S→AB is useless so remove it.

So the Simplified Grammar is

S→a

A→a

Eliminating Null production

Consider a grammar G having productions

S→aS|bA,A→aA|є

A→є is a null production after removing it the Simplified Grammar is

S→aS|bA|b,A→aA|a

Eliminating Unit Production

A production of the form A→B ,where A,B єV is called unit production.

Consider a grammar G having productions

S→A|bb

A→B|b

B→S|a

After Eliminating unit productions S→A,A→B,B→S the simplified Grammar is

S→b|a|bb

A→a|bb|b

B→bb|b|a

NORMAL FORMS

According to context free grammar the right hand side of the production can be any combination of terminals and nonterminals. The Normal Forms are used to restrict the right hand side of a production rule.

Two widely used normal forms are

1) Chomsky Normal Form(CNF)
2) Greibach Normal Form(GNF)

Chomsky Normal Form

A context free grammar G is inCNF if all the production obey the form A→BC or A→a.

Greibach Normal Form

A context free grammar G is inCNF if all the production obey the form A→aα , where αϵ(V)*.

Q. Find the equivalent grammar in CNF for the grammar G=({S,A,B,D},{a,b},P,S)

Productions are S→aAD,A→aB|bAB,B→b,D→d.

S→XM

X→a

M→AD

A→XB

A→YN

Y→b

N→AB

B→b

D→d

Model of pushdown automata

PDA is defined as 7-tuple notation.

$M(Q,\sum, \Gamma,\delta,q_0,z_0,F)$

Q=Finite set of states

$\sum$=Finite set of input alphabet

Γ=Finite set of stack alphabet

$\delta =Qx\{\sum U\{\epsilon\}\}x\ \Gamma \rightarrow Qx\ \Gamma^*$

<u>Deterministic Push Down Automata</u>

A PDA is said to be deterministic if all derivations(ID) in the design has to give only single move.

<u>NonDeterministic Push Down Automata</u>

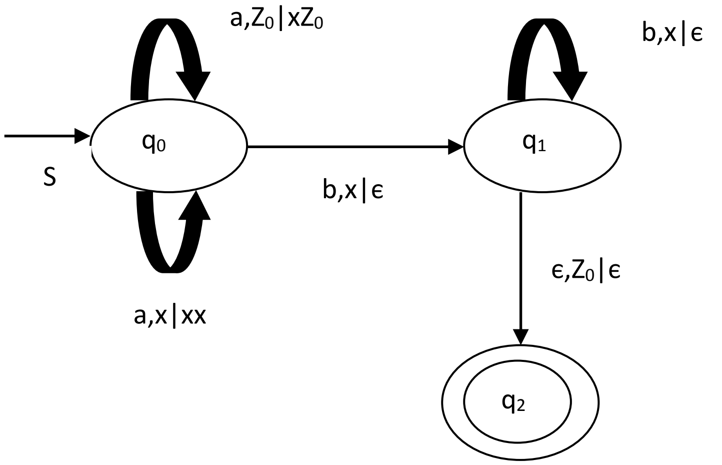
A PDA is said to be nondeterministic ,if derivation generates more than one move in the designing of particular task.

Q.Construct PDA that accepts L={$a^n b^n$|n>0}

Instantaneous Description

($q_0$aaabbb,Z_0)|-------- ($q_0$aabbb,xZ_0)|----------($q_0$abbb,xxZ_0)|---------($q_0$bbb,xxxZ_0)|-----------

($q_1$bbb,xxxZ_0)|---------- ($q_1$bb,xxZ_0)|--------(q_1b,xZ_0)|------ (q_1,Z_0) |---------- (q_f,___)

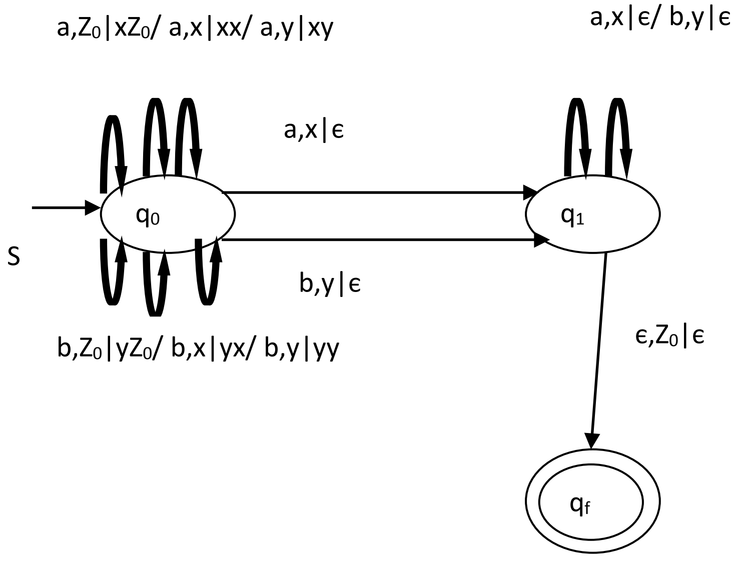


State Transition Table

	a		b		ϵ	
	x	Z_0	x	Z_0	x	Z_0
q_0	(q_0,xx)	(q_0,xZ_0)	(q_1,ϵ)	________	_________	________
q_1	__________	________	(q_1,ϵ)	________	_________	(q_f ,__)
q_2	__________	________	__________	________	__________	_________

NonDeterministic Push Down Automata

Q. Construction of NPDA for $L=\{ww^R|w\epsilon(a+b)^+\}$


a,Z_0|xZ_0/ a,x|xx/ a,y|xy
a,x|ϵ/ b,y|ϵ
a,x|ϵ
q_0
q_1
S
b,y|ϵ
b,Z_0|yZ_0/ b,x|yx/ b,y|yy
ϵ,Z_0|ϵ
q_f

Cocke,Younger & Kasami Algorithm

This algorithm is useful for testing membership in a CFL(Context Free Language). This is based on the idea of Dynamic programming. The CFG should be in CNF. The table is filled row-by-row in bottom to top. The bottom row correspond to the substring of length 1. The second row from bottom row corresponds to the substring of length 2, like on. It takes O(n) time to compute any one entry of the table and here n(n+1)/2 entries are there so T.C. is $O(n^3)$.

X_{15}				
X_{14}	X_{25}			
X_{13}	X_{24}	X_{35}		
X_{12}	X_{23}	X_{34}	X_{45}	
X_{11}	X_{22}	X_{33}	X_{44}	X_{55}
a_1	a_2	a_3	a_4	a_5

In this above figure the word of length 5 say $a_1a_2\ a_3\ a_4\ a_5$ is tested for its membership under a CFG say G. If we find the starting symbol of the CFG in X_{15} then we can say $a_1a_2\ a_3\ a_4\ a_5$ is a member of L(G).

Q. Given CNF grammar

$S\rightarrow AB|BC$

$A\rightarrow BA|a$

$B\rightarrow CC|b$

$C\rightarrow AB|a$

Test that the string *baaba* is in L(G).

Ans:- Dynamic Programming formulation for CYK

$$X_{ij}=\bigcup_{i\leq k<j} X_{ik}X_{k+1j}$$

First row from the bottom

$X_{11}=\{B\}$, $X_{22}=\{A,C\}$, $X_{33}=\{A,C\}$, $X_{44}=\{B\}$,$X_{55}=\{A,C\}$

Second row from the bottom

$X_{12}=X_{11}X_{22}=\{B\}\{A,C\}=\{BA,BC\}=\{A,S\}$

$X_{23}=X_{22}X_{33}=\{A,C\}\{A,C\}=\{AA,AC,CA,CC\}=\{B\}$

$X_{34}=X_{33}X_{44}=\{A,C\}\{B\}=\{AB,CB\}=\{S,C\}$

$X_{45}=X_{44}X_{55}=\{B\}\{A,C\}=\{BA,BC\}=\{A,S\}$

<u>Third row from the bottom</u>

$X_{13}=X_{11}X_{23}\ U\ X_{12}X_{33}=\{B\}\{B\}\ U\{S,A\}\{A,C\}$

$=\{BB,SA,SC,AA,AC\}=\{\}$

$X_{24}=X_{22}X_{34}UX_{23}X_{44}$

$=\{A,C\}\{S,C\}U\{B\}\{B\}$

$=\{AS,AC,CS,CC,BB\}$

$=\{B\}$

$X_{35}=X_{33}X_{45}UX_{34}X_{55}$

$=\{A,C\}(A,S)U\{S,C\}\{A,C\}$

$=\{AA,AS,CA,CS,SA,SC,CA,CC\}$

$=\{B\}$

<u>Fourth row from the bottom</u>

$X_{14}=X_{11}X_{24}UX_{12}X_{34}UX_{13}X_{44}$

$=\{B\}\{B\}U\{A,C\}\{S,C\}U\{\}\{B\}$

$=\{BB,AS,AC,CS,CC,B\}$

$=\{\}$

$X_{25}=X_{22}X_{35}\ UX_{23}X_{45}UX_{24}X_{55}$

$=\{A,C\}\{B\}U\{B\}\{S,A\}U\{B\}\{A,C\}$

$=\{AB,CB,BS,BA,BA,BC\}$

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

<u>Last row from bottom</u>

$X_{15}=X_{11}X_{25}UX_{12}X_{35}UX_{13}X_{45}UX_{14}X_{55}$

$=\{B\}\{S,A,C\}U\{S,A\}\{B\}U\{\}\{S,A\}U\{\}\{A,C\}$

$=\{BS,BA,BC,SB,AB,S,A,C\}$

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

{S,A,C}				
---------	{S,A,C}			
---------	{B}	{B}		
{S,A}	{B}	{S,C}	{S,A}	
{B}	{A,C}	{A,C}	{B}	{A,C}
b	a	a	b	a

So *baaba* is a member of L(G).

Parikh's Theorem

To understand Parikh's Theorem there is a need of some basic definitions of Linear subset, Semilinear subset and Parikh's Mapping.

Linear subset: A linear subset M, of N^n is given by tuples $t_0,t_1,\ldots\ldots,t_m \epsilon N^n$, where $m \epsilon N$ and $n \epsilon N^+$.

$M=\{t_0+l_1t_1+\ldots.+l_mt_m|l_1,l_2,\ldots..,l_m \epsilon N \}$

$=t_0+\{t_1,\ldots..t_m\}^*$

Semilinear subset: A semilinear subset,M′, of N^n, is a union of finite number of linear subsets $M_1,\ldots..M_k$, where $k \epsilon N^+$.

Parikh Mapping: if a word w is defined over $\sum=\{a_1,a_2,\ldots..,a_n\}$, where $n \epsilon N^+$, Parikh image is

$\Psi(w)=\{m_1,m_2,\ldots..m_n\}$. Each m_i represents the occurrences of a_i in w.

Example:- Suppose language L is defined by $\sum=\{a,b\}$. words are w_1=abab, w_2=babb,w_3=aaa,then $\Psi(w_1)=\{2,2\}$, $\Psi(w_2)=\{1,3\}$, $\Psi(w_3)=(3,0)$.

Parikhs Theorem

For every context-free language L. Ψ(L) is effectively semilinear. The tuples specifying Ψ(L) can be constructed effectively from a context-free grammar generating L.

Pumping Lemma for CFL

Pumping Lemma for CFL is generally used to prove a language is not context free.

Let L be any CFL. Then there is a constant n, depending only on L, such that if z is in L and $|z|\geq n$. Then we may write w=uvwxy such that

1) $|vx|\geq 1$

2) $|vwx| \leq n$

3) For all $i \geq 0$, uv^iwx^iy ϵ L.

Languages that are not CFL

$\{a^nb^n\ c^n|n\geq1\ \}$ is not CFL.

$\{a^p|p$ is prime$\}$ is not CFL.

Q. Show that L= $\{a^p|p$ is prime$\}$ is not CFL.

Solution:-

Let us consider z=aaaaa

u=a,v=a,w=a,x=a,y=a,

Let us consider i=3

uv^3wx^3y=aaaaaaaaa=a^9 doesn’t belong to L.

Hence L is not context free.

CLOSURE PROPERTIES OF CFL

1. CFLs are closed under union, concatenation and Kleen closure.

2. CFLs are closed under substitution.

3. CFLs are closed under homomorphism.

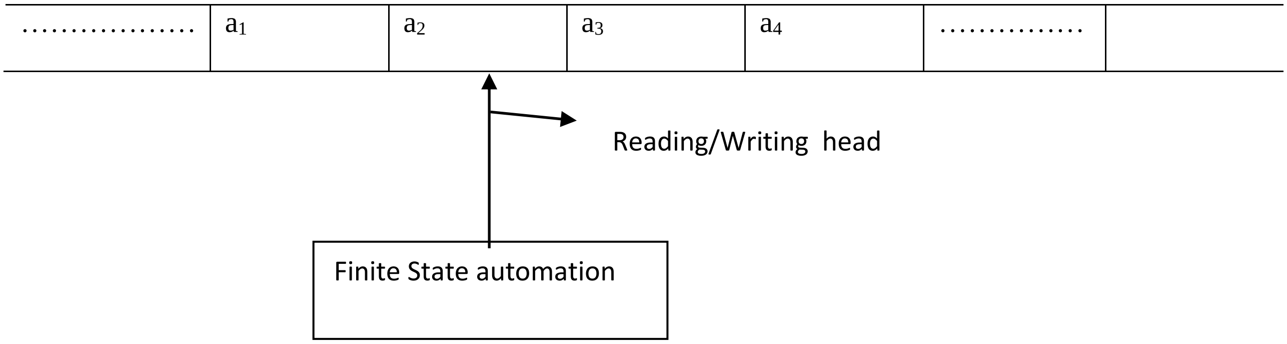
4. CFLs are closed under inverse homomorphism.

5. If L is a CFL and R is a regular set, then L∩R is a CFL.

MODULE 3

TURING MACHINE MODEL

The TM can be thought of as a finite state automaton connected to a R/W head. It has one tape which is divided into number of cells.



Each cell can store only one symbol. R/W examines the tape symbols. In one move, the machine examines the present symbol under R/W head on the tape and the present state of an automation to determine:

1) A new symbol to be written in the tape in the cells under R/W head.

2) A motion of the R/W head along the tape either the head moves one cell left on one cell right.

3) The next state of automation.

The formal notation of TM:

A TM is a 7-tuple notation

$M=(Q,\sum, \Gamma,\delta,q_0,B,F)$ where

Q=Finite set of states.

$\sum$=Finite set of inputs.

Γ=Finite set of tape symbols.

$\delta =Qx \Gamma=Qx \Gamma x\{L,R\}$

q_0=Initial state

B=Blank symbol.

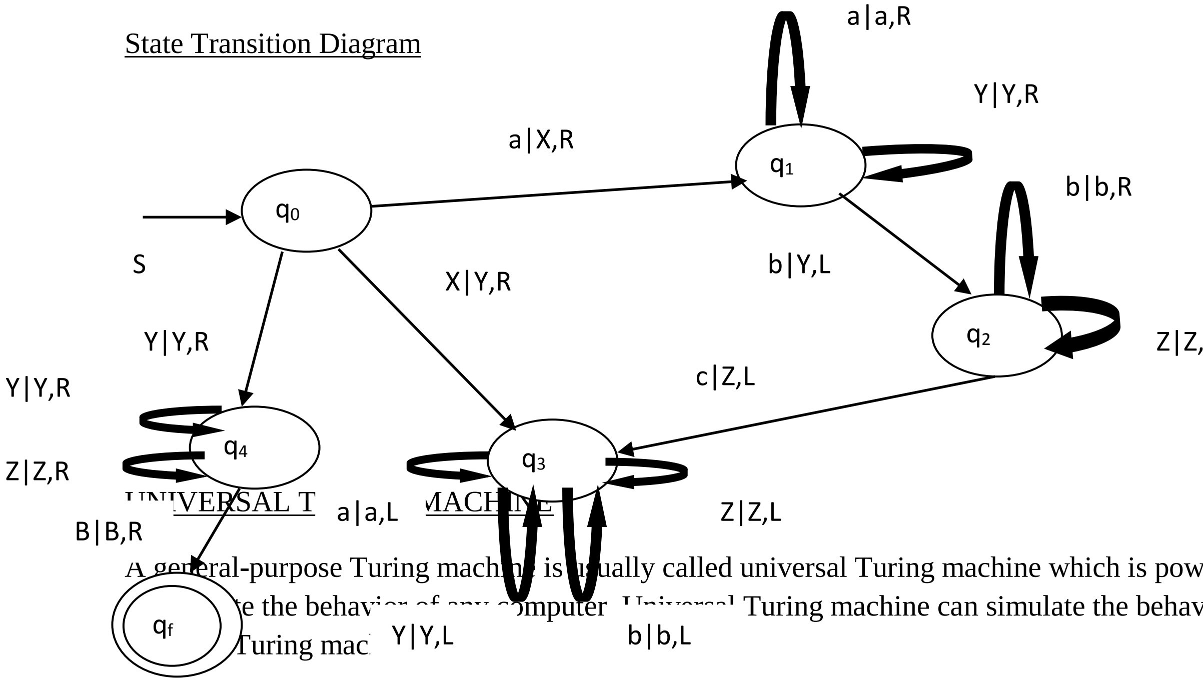
$F\subseteq Q$ is a set of final states.

Q. Construct a Turing Machine that accepts {a^n b^n c^n|n≥1}

Instanteneous Description

$q_0$aaabbbccc|-- X $q_1$aabbbccc |-- Xa $q_1$abbbccc|-- Xaa$q_1$bbbccc|-- XaaY$q_2$bbccc|-- XaaYb$q_2$bccc|-- XaaYbb$q_2$ccc|-- XaaYbq_3bZcc|-- XaaYq_3 bbZcc|-- Xaaq_3YbbZcc|--Xaq_3 aYbbZcc|-- Xq_3 aaYbbZcc|-- q_3XaaYbbZcc|--X $q_0$aaYbbZcc|-- X Xq_1aYbbZcc|-- X Xaq_1YbbZcc|-- X XaY$q_1$bbZcc|-- X XaYYq_2bZcc|-- X XaYYbq_2Zcc|-- X XaYYbZ$q_2$cc|-- X XaYYbq_3 ZZc|-- X XaYYq_3 b ZZc|-- X XaYq_3Y b ZZc|-- X Xaq_3 YY b ZZc|-- X Xq_3 aYY b ZZc|-- X q_3X aYY b ZZc|--X Xq_0 aYY b ZZc|-- X XXq_1 YY b ZZc|-- X XXYq_1 Y b ZZc|-- X XXYYq_1 b ZZc|-- X XXY Y Y q_2 ZZc|-- X XXY Y Y Z q_2 Zc|-- X XXY Y Y Z Z q_2c|-- X XXY Y Y Z q_3 Z Z|-- X XXY Y Y q_3Z Z Z|-- X XXY Y q_3 YZ Z Z|-- X XXY q_3Y Y Z Z Z|-- X XX q_3Y Y YZ Z Z|-- X X q_3X Y Y YZ Z Z|--X X X q_0 Y Y YZ Z Z|-- X X X Y q_4 Y YZ Z Z|-- X X X Y Y q_4 YZ Z Z|-- X X X Y Y Y q_4 Z Z Z|-- X X X Y Y Y Z q_4 Z Z|-- X X X Y Y Y Z Z q_4 Z|-- X X X Y Y Y Z Z Z q_4|-- X X X Y Y Y Z Z ZB q_fB

State Transition Diagram



UNIVERSAL T[illegible] MACHINE

A general-purpose Turing machine is usually called universal Turing machine which is powerful enough [illegible] the behavior of any computer. Universal Turing machine can simulate the behavior of an [illegible] Turing mac[illegible]

Deterministic Turing Machine

A Turing machine is called deterministic if at most one description can be possible from any description.

OR

There exists at most one entry in any cell of state transition table representation of Turing machine.

Nondeterministic Turing Machine

A Turing machine is called nondeterministic if more than one description can be possible from at least one description.

OR

There exists more than one entry in at least one cell of state transition table representation of Turing machine.

RECURSIVE LANGUAGE & RECURSIVELY ENUMERABLE SET

A language L over the alphabet ∑ is called recursive if there is a TM say T, that accepts every word in L and rejects every word in L´, that is

Accept(T)=L, Reject(T)= L´, Loop(T)=φ

A language L over the alphabet ∑ is called recursively enumerable if there is a TM say T, that accepts any word in L and either reject or loops forever for every word in L´, that is

Accept(T)=L

Reject(T) +Loop(T)= L´

LINEAR BOUNDED AUTOMATA

This model accepts the set of context-sensitive languages, the length of the tape is restricted by a linear function.

A LBA is a nondeterministic Turing machine which has a single tape whose length is not infinite but bounded by a linear function.

The model can be formally defined as M(Q,∑, ┌,δ,q$_0$,B,$,§,F)

Q=Finite set of states.

∑=Finite set of inputs.

┌=Finite set of tape symbols.

δ =Qx ┌=2$^{Qx ┌x\{L,R\}}$

q$_0$=Initial state

B=Blank symbol.

F⊆Q is a set of final states.

$=Left end mark

§=Right end mark

CONTEXT SENSITIVE LANGUAGE

A language L is said to be context sensitive if there exists a context sensitive grammar G, such that L=L(G) or L=L(G)Uϵ. Whereas the context sensitive grammar is defined as grammar G={V,T,P,S}, the products are in the form

X→Y, where X,Y ϵ(VUT)* and |Y|≥|X|

A context sensitive grammar can never generate a language containing the empty string i.e. X→ϵ is not allowed.

Primitive Recursive Functions

Basic Functions

1. Zero Function – Z(x)=0, for all xϵI.
2. Successor Function – S(x)=x+1
3. Projector Function $P_k(x_1,x_2) = x_k$, k=1,2

Composition and Primitive Recursion are used to build more complicated functions from the basic functions.

1. Composition f(x,y)=h(g1(x,y),g2(x,y))
2. Primitive Recursion

$f(x,0)=g_1(x)$
$f(x,y+1)=h(g_2(x,y),f(x,y))$

A function is called Primitive Recursive if and only if it can be constructed from the basic functions Z,S,P_k by successive composition and primitive recursion.

Ackermanns Function

A(0,y)=y+1
A(x,0)=A(x-1,1)
A(x,y)=A(m-1,A(m,n-1))

Addition, Multiplication functions are the Primitive Recursive function but Ackermanns function is not Primitive Recursive.
So for the construction of Ackermanns Function and some other functions µ-recursive function is required.

A function is µ-recursive if it can be constructed from the basic functions by a sequence of µ-operator and the operations of composition and primitive recursion.

Godel Number

In mathematical logic, a Gödel numbering is a function that assigns to each symbol and well-formed formula of some formal language a unique natural number, called its Gödel number.

A Gödel numbering can be interpreted as an encoding in which a number is assigned to each symbol of a mathematical notation, after which a sequence of natural numbers can then represent a sequence of symbols. These sequences of natural numbers can again be represented by single natural numbers, facilitating their manipulation in formal theories of arithmetic.

Gödel used a system based on prime factorization. He first assigned a unique natural number to each basic symbol in the formal language of arithmetic with which he was dealing.

To encode an entire formula, which is a sequence of symbols, Gödel used the following system. Given a sequence $(x_1, x_2, x_3, \ldots, x_n)$of positive integers, the Gödel encoding of the sequence is the product of the first *n* primes raised to their corresponding values in the sequence:

$\mathrm{enc}(x_1 ,x_2 ,x_3 ,\ldots,x_n)=2^{x1}.\ 3^{x2}\ldots p^{xn}$

According to the fundamental theorem of arithmetic, any number obtained in this way can be uniquely factored into prime factors, so it is possible to recover the original sequence from its Gödel number.

Example:-

Suppose the Gödel number for the symbol "&" is 2 and the Gödel number for the symbol "=" is 5. Thus, the Gödel number of the formula "& = " is $2^2.3^5=972$.

Post Correspondence Problem

Halting problem is an undecidable problem. So it is not possible to work with halting problem directly. And it is convenient to establish some intermediate results that bridge the gap between halting problem and other problems.

The intermediate result follow from the undecidability of the halting problem. But much more closer to the problem that we want to study. One such intermediate result is Post Correspondence Problem.

PCP Statement

Given two sequence of n strings on some alphabet ∑, say $A=w_1,w_2,\ldots,w_n$ and $B=v_1,v_2,\ldots\ldots,v_n$ we say there exists a post correspondence solution for pair(A,B)if there exist a nonempty sequence of integers i,j,….,k such that $w_i\ w_j\ldots..w_k=v_i\ v_j\ldots\ldots v_k$.

Example:- Let ∑={0,1} and take A and B as

A→ w_1=00, w_2=101,w_3=110

B→v_1=001,v_2=011,v_3=10

For integer sequence <1,2,3> there exist a PCP solution because w_1 w_2 w_3=v_1 v_2 v_3 .

Suppose

A→ w_1=00, w_2=101,w_3=110

B→v_1=001,v_2=011,v_3=101

No PCP solution because |B|>|A|.

The Post correspondence problem is undecidable.

Decidable Problem

A problem is decidable if an algorithm can be written to solve it.

P,NP,NP-COMPLETE & NP-HARD CLASSES

P is the set of all decision problems solvable by deterministic Turing machine in polynomial time.NP is the set of all decision problems solvable by nondeterministic Turing machine in polynomial time.

A problem L is NP-hard if and only if satisfiability reduces to L. A problem L is NP-complete if and only if L is NP-hard and L϶NP.

Disclaimer

This above material may contain some mistakes. So the reader can read carefully.