UNIT-V UNSOLVABLE PROBLEMS AND COMPUTABLE FUNCTION

Unsolvable Problems and Computable Functions – Primitive recursive functions – Recursive and recursively enumerable languages – Universal Turing machine. MEASURING AND CLASSIFYING COMPLEXITY: Tractable and Intractable problems- Tractable and possibly intractable problems - P and NP completeness - Polynomial time reductions.

PART-A

1. What is recursive enumerable and recursive set?

Recursive Enumerable set:

The languages accepted by Turing Machines are called as recursive enumerable languages. Recursively enumerable set includes regular languages and CFL's. some languages in recursive enumerable set are not accepted by TM.

Recursive set:

The subset of recursive enumerable set is called as recursive set. All the languages in the recursive set are accepted by TM.

2. What is decidable problem(or) decidability?

Decidable problem:

*A problem whose language is recursive is said to be decidable.

*A problem is decidable if there is an algorithm that takes as input an instance of the problem and determines whether the answer to that instance is yes or no.

3. What is undecidable problem(or) undecidability?

Undecidable problem:

*A problem whose language is recursively is said to be undecidable.

*A problem is undecidable if there is no algorithm that takes as input an instance the problem and determines whether the answer to that instance is yes or no.

4. What are the closure properties of recursive & recursively enumerable languages?

- The complement of recursive language is recursive.
- The union of two recursive language is recursive.

- The union of two recursively enumerable language is recursively enumerable.
- If a language L and its complement of are recursively enumerable the L and then L is recursive.

5. What is universal turing machine?

Universal Turing machine:

If the problem is undecidable.

Does the turing machine M accept input w? here both M and w are the parameters of the problem. In formalizing a problem as a language. The input w is restricted to be over {0,1} and M to have tape alphabet {0,1,B}. This Turing Machine is referred as Universal Turing Machine.

6. What is diagonalization language?

In the table if the entry is

0-> means wi is not in L(Mj)

1-> means wi is in L(Ml)

We can construct a language Ld from the table by using the diagonal entries of the table to determine membershin in L1.

7. What is Post correspondence Problem(PCP)?

- An instance of post correspondence problem CP(P) consist of two lists.
  $A = w_1, \ldots, w_k$ and
  $B = x_1, \ldots, x_k$

of strings over some alphabet Σ

- This instance of PCP has a solution if there any sequence of integers
  $i_1, i_2, \ldots, i_m$, with $M \geq 1$ such that
  $w_{i1}, w_{i2}, \ldots, w_{im} = x_{i1}\, x_{i2} \ldots x_{im}$
- The sequence $i_1, \ldots i_m$ is a solution to this PCP

Example :

Let Σ={O,1} Let A and B be list of three strings.

	List A	List B
i	w_i	x_i
1	1	111
2	10111	10
3	10	0

Solution :

- This PCP has a solution
  M = 4
  $i_1 = 2$, $i_2 = 1$, $i_3 = 1$, and $i_4 = 3$,
  $w_{i1}\ w_{i2}\ w_{i3}\ w_{i4} = x_{i1}\ x_{i2}\ x_{i3}\ x_{i4}$
  101111110 = 101111110
  The Solution

i_1	=	2
i_2	=	1
i_3	=	1
i_4	=	3

8. What is MPCP?

The modified version of PCP is the following given lists A and B of K strings each from Σ* say

A = w1, w2,….wk

B = x1,x2,….xk

This problem has a solution if there exists sequence of integers

I1,i2,…ir such that

W1,wi1,…wir = x1 xi1xi2….xir

9. What is primitive recursive function?

The primitive recursive function is defined as follows

All initial functions are elements of PR

For any k>= 0 and m>= 0 if N k – N ang g1,g2,…gk : Nm -> N are elements of PR , then f(g1,g2,…gk) obtained from f and g1, g2,…. Gk by composition is an element of PR.

For any n>=0 any function g:N n+1 -> N in PR and any function.

H: N n+2 -> N in PR.

No other functions are in the set PR.

No other functions are in the set PR.

10. What are the initial functions?

a) constant functions

b) The successor function

c) The projection function

11. What are P problems?

The problems that can be solved in a polynomial time by deterministic turing machine. Eg) Krushals Algorithm b) Decidable Problem

12. What are NP problems?

The problems that can be solved in a non-deterministic polynomial time by non deterministic turing machine.

Example: Undecidable problem

13. What is NP complete problems?

NP complete problems are NP problems.

The NP completeness problem in which ever language in NP reduces to P in polynomial time.

PART-B

1. RECURSIVE AND RECURSIVELY ENUMERABLE LANGUAGES

- **State and prove the closure Properties of recursive and recursively enumerable languages. [Nov/Dec 2012]**

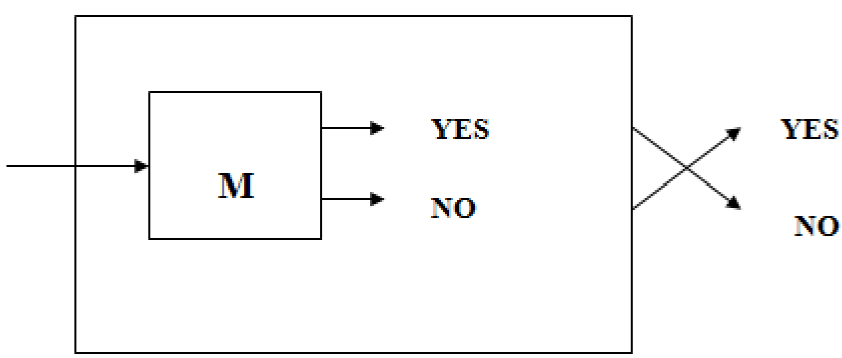
Closure properties of recursive and recursively enumerable languages :

Theorem 1 :

The complement of a recursive language is recursive.

Proof :

- Let L be a recursive language and M a tuning machine that halts on all inputs accepts L.
- Construct M’ from M so that if M enters final state on input w then m’ halts with accepting.
- If M halts without accepting, M’ enters final state.
- Since one of these two events occurs M’ an algorithm.
- Clearly L (M) is the complement of L and thus the complement of L is a recursive language.



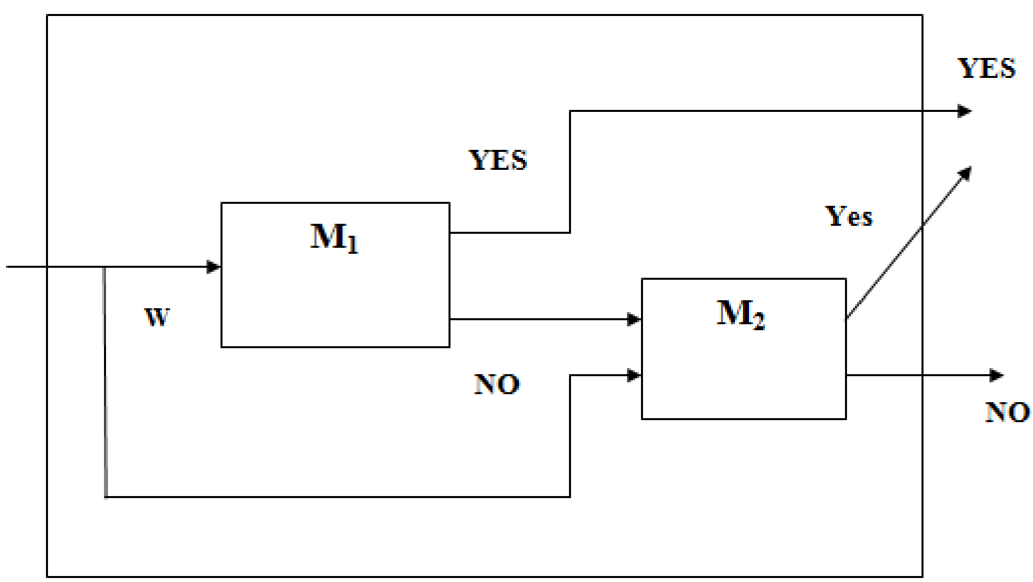
Theorem 2 :

- The union of two recursive language is recursive.
- The union of two recursively enumerable language is recursively enumerable.

Proof :

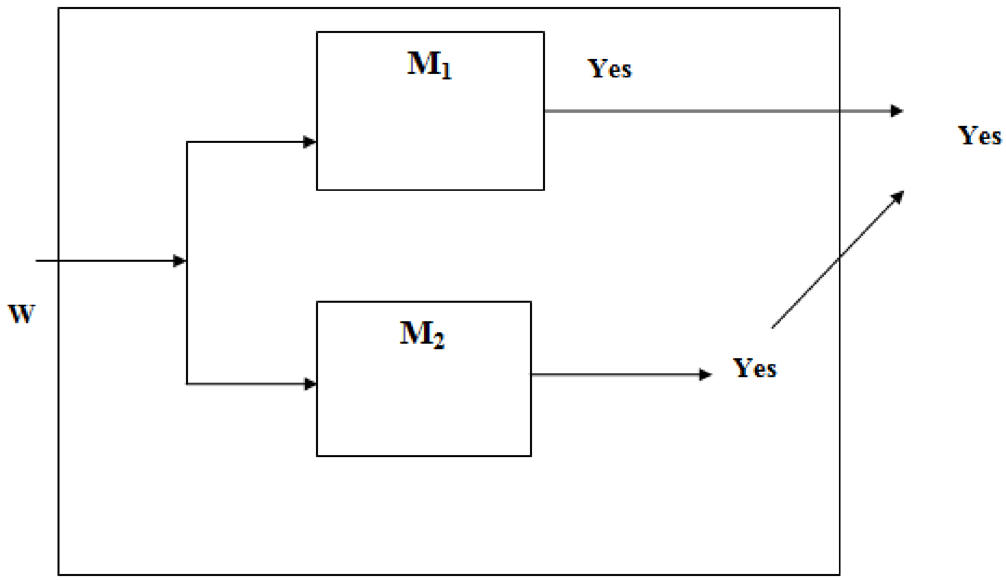
Let L_1 and L_2 be recursive language accepted by algorithms M_1 and M_2

- Let L_1 and L_2 be recursive language accepted by algorithms M_1 and M_2
- We construct M1, which first simulates M_1
- If M_1 accepts then M accepts.
- If M1 rejects then M simulates M_2 and accepts iff M_2 accepts.
- Since both M1 and M2 are algorithms, M is generated to halt.
- Clearly M accepts L_1 U L_2



- For recursively enumerable languages the above construction does not work.
- Since M_1 may not halt.
- Instead M can simultaneously simulate M_1 and M_2 on separate tapes.

- ✶ If either accepts, then M accepts.

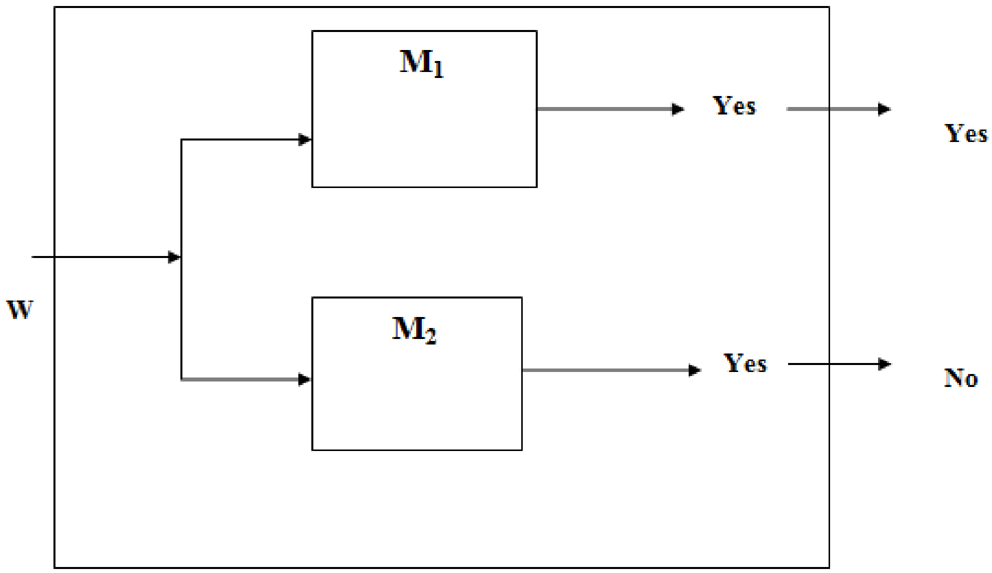


Theorem 2 :

If a language L and its complement $\overline{L}$ (and hence $\overline{L}$) is recursive.

Proof :

- ✶ Let M_1 and M_2 accept L and τ respectively.
- ✶ Construct M to simulate simultaneously M and M_2
- ✶ M accepts w if M_1 accepts w and rejects if M_2 accepts w
- ✶ Since w is in either L or τ we know that exactly one of M_1 or M_2 will accept.
- ✶ Thus M will always say either 'yes' or 'no' but will never say both.
- ✶ Since M is an algorithm that accepts L, it follows that L is recursive.



2.A.UNIVERSAL TURING MACHINE

❖ **Define Universal Turing machine and discuss about the Universal codes.**

Universal turning Machine :

- Diagnolization is used to show the problem is undecidable.
- The problem is :
  "Does the turning machine M accept w" ?
- Here both M and w are the parameters of the problem.
- In formalizing a problem as a language the input w is restricted to be over {0, 1} and M to have tape alphabet {0, 1, B}

Turning Machine codes :

- We encode turning machines with restricted alphabets as strings over {0, 1}
- Let
  $M = (Q,\{0,1\},\{0,1,B\},\delta,q_1,B,\{q_2\})$ be a turning and Blank as the additional tape

  symbol.
- We assume that
  $Q = \{q_1, q_2, \dots q_n\}$
  $F = \{q_2\}$

- ✶ There is no need for more than one final state in any TM, since once it accepts it may as well halt.
- ✶ The symbols 0, 1, B are used with the synonyms x_1, x_2, x_3 respectively.
- ✶ Directions L and R used with the synonyms D_1 and D_2.
- ✶ The move $\delta(q_i, x_j) = q_k, x_l, D_m)$ is encods by the binary string.

  $O^i \mid O^j \mid O^k \mid O^l \mid O^m$
- ✶ A binary code for turning machine M is

  ||| Code, || $Code_2$ || … || $Code_r$ |||.

  Where each $code_i$is of the form $0^i10^j10^k10^l10^m$
- ✶ Each move of M is encoded by one of the $code_i$'s
- ✶ The moves need not be in any particulars order
- ✶ The pair <M, w> is represented by the string of the form

  |||$Code_1$ ||$Code_2$ || … $Code_r$ ||| followed by w
- ✶ Any such string will be denoted <m, w>

Example :

- ✶ Encode the following TM, M <M, 1011> in binary

  M = $(\{q_1, q_2, q_3\}, \{0, 1\}, \{0, 1, B\}, \delta, q_1, B_1, \{q_2\})$

  Where δ is

  $\delta(q_1, 1) = (q_3, 0, R)$

  $\delta(q_3, 0) = (q_1, 1, R)$

  $\delta(q_3, 1) = (q_2, 0, R)$

  $\delta(q_3, B) = (q_3, 1, L)$

Solution :

1) $\delta(q_1, 1) = (q_3, 0, R)$

   $\delta(q_1, x_2) = (q_3, x_1, D_2)$

   Code1 = $0^1 10^2 10^3 10^1 10^2$
2) $\delta(q_3, 0) = (q_1, 1, R)$

   $\delta(q_3, x_1) = (q_1, x_2, D_2)$

   $Code_2 = 0^3 10^1 10^1 10^2 10^2$
3) $\delta(q_3, 1) = (q_2, 0, R)$

   $\delta(q_3, x_2) = (q_2, x_1, D_2)$

   $Code_3 = 0^3 10^2 10^2 10^1 10^2$

4) $\delta(q_3, B) = (q_3, 1, L)$

$\delta(q_3, x_3) = (q_3, x_2, D_1)$

$Code_4 = 0^3 1 0^3 1 0^3 1 0^2 1 0^1$

Thus the string denoted by <M, 1011> is

$|||0^1|0^2|0^3|0^1|0^2|0^3|0^1|0^1|0^2|0^2||0^3|0^2|0^2|0^1|0^2|0^3|0^3|0^3|0^2|0^1|||1011$

1110100100010100110001010100100110001001001010011000100010001001011 11011

2.B.A NON RECURSIVELY ENUMERABLE LANGUAGE

❖ **Prove that L_d is not recursively enumerable**

A non – recursively enumerable language :
Lemma :

Ld is not recursively enumerable.

- Suppose we have a list of (0+1)* in canonical order where

$w_i \rightarrow$ is the ith word

$M_j \rightarrow$ is the TM whose code is the integer j written in binary

- We can construct an infinite table that tells for all I and j whether w_i is in L (M_j)

- In the table, if the entry is

0 – means w_i is not in $L(M_j)$
1 – means w_i is in L (M_j)

- We can construct a language Ld from the table by using the diagonal entires of the table to determine membership in L_d
- To guarantee that no TM accepts L_d. we insist that w1 is in L diff the the (i, i) entry is o, that is if M_i does not accept w_1.
- Suppose that some TM Mj, accepted L_d then the following contradictions may occur.
  1) If w_j is in L_d then (j, j) entry o, implying that w_j is not in $L(m_j)$ and contradiciting $L_d = L(M_j)$
  2) If w_j is not in L_d, then the (j, j) entry is 1, implying that w_j is in $L(M_j)$ which contradicts $L_d = L(M_j)$

- ✶ As wj is either in or not in L_d, we conclude that our assumption $L_d = L(M_j)$ is false.
- ✶ Thus no TM in the list accepts L_d. And no turning machine accepts L_d.

Lemma :

Ld is not recusively enumerable.

3. THE UNIVERSAL LANGUAGE

The Universal Language :

- ✶ The universal language Lu is defined to be

{<M, w> | M accepts w}

- ✶ We say Lu, universal, since the question of whether any particular string w in (0 +1)* is a accepted by any particulars TM M is equivalent to the question of whether. <M^1, w> is Lu, where M1 is the TM with the tape alphabet {0, 1, B} equivalent.

Theorem 1 :

Lu is recursively enumerable.

Proof :

- ✶ We shall exhibit a three tape TM M_1 accepting L_u
- ✶ The first tape of M1 is the input tape and the input head on that tape is used to look up moves of the TM M when given code <M, w> as input.
- ✶ The second tape of M_1 will simulate of M.
- ✶ The alphabet of M is {0, 1, B}, so each, of M's tape can be held in one tape cell of M^1's second tape.
- ✶ The third type holds the state of M with q_i represented by O^i
- ✶ The behavior of M_1 is as follows
  1) Check the format of tape 1, to see that it has a prefix of the form 111Code, 11 code2 || … || coder ||| and that there no two codes that begin with o^i | o^j | for the same i and j for the same i and j Also check that if o^i | o^j | o^k | o^l | o^m is a code then $i \leq j \leq 3, l \leq l \leq 3$ and
  2) Initialise tape 2 contain w the portion of the input beyond the second block of three 1's. Initialise tape 3 to hold a single O. representing q_1. All three tape heads are positioned on the left most symbols.
  3) If tape e holds OO. the code for the final state, halt and accept.
  4) Let x_j be the symbol currently scanned by tape head2 and let O^i be the current content of tape 3.

Scan tape 1 from the left end to the second 111 looking for a substring beginning 110^i | O^j11. If no such string is found, halt and reject. M has no next move and has not accepted. If such a code is found, let it be o^i | o^j | o^k | o^l | o^m Then put O^k on tape 3 print x_2 on the tape cell scanned by tape head 2 and move that head in Direction D_m.

1) to step (3)

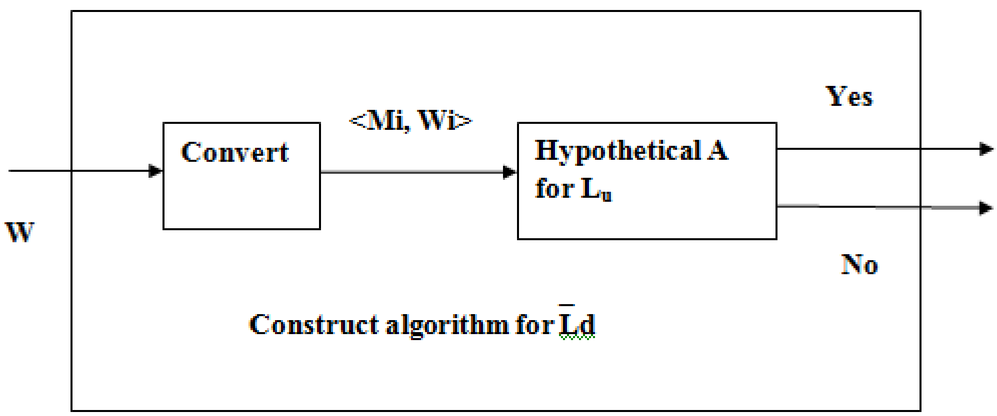
- □ $_1$ accepts < M, W > if, M accepts w. It is also true that if m run forever on w, M_1 will run forever on <m, w>.
- □ f m halts on w without accepting, M_1 does the same on < m_1, w >.
- □ hus lu is recursively enumerable.

Theorem 2

Lu is not recursive

Proof :

- ✶ Suppose A is an algorithm, recognizing lu.
- ✶ Then we can recognize L_d as follows.
- ✶ Given string w in (o + 1)*, determine for I w = w_i
- ✶ Integer I in binary is the code for some TM M_i Feed < M_i, w_i> to algorithm A accept w iff M_i accepts w_i



- ✶ Thus we have an algorithm for Ld. since no such algorithm exists, we know our assumption, that algorithm A for Lu exists is false.
- ✶ Hence Lu is recursively enumerable, but not recursive.

4. POST CORRESPONDANCE PROBLEM

❖ **Explain about PCP** **[Nov/Dec 2013]**

- **Explain PCP and decidable and undecidable problems with example.**

[Apr/May 2015]

1. Post correspondence Problem : CP(P)

- An instance of post correspondence problem CP(P) consist of two lists.
  $A = w_1, \ldots, w_k$ and
  $B = x_1, \ldots, x_k$

of strings over some alphabet Σ

- This instance of PCP has a solution if there any sequence of integers
  $i_1, i_2, \ldots, i_m$, with $M \geq 1$ such that
  $w_{i1}, w_{i2}, \ldots, w_{im} = x_{i1}\, x_{i2} \ldots x_{im}$
- The sequence $i_1, \ldots i_m$ is a solution to this PCP

Example :

Let $\Sigma = \{O,1\}$ Let A and B be list of three strings.

	List A	List B
i	w_i	x_i
1	1	111
2	10111	10
3	10	0

Solution :

- This PCP has a solution
  M = 4
  $i_1 = 2$, $i_2 = 1$, $i_3 = 1$, and $i_4 = 3$,
  $w_{i1}\, w_{i2}\, w_{i3}\, w_{i4} = x_{i1}\, x_{i2}\, x_{i3}\, x_{i4}$
  101111110 = 101111110
  The Solution

$\boxed{i_1 = 2}$

i_2	=	1
i_3	=	1
i_4	=	3

A modified version of PCP [MPCP]

- The Modified version of PCP (MPCP) is the following.
  Given list A and B, of K strings each from $\Sigma *$ say
  $A = w_1, w_2 \ldots, w_k$
  $B = x_1, x_2 \ldots, x_k$
- This problem has a solution if there exists sequence of integers
  $i_1, i_2, \ldots, i_r$, such that
  $w_1 w_{i1} w_{i2} \ldots w_{ir} = x_{i1} x_{i2} \ldots x_{ir}$
- The difference between the MPCP and PCP is that in the MPCP a solution required to start with the firs string on each list.

Lemma

If PCP were decidable, then MPCP would be decidable. That is,

MPCP reduces to PCP

Proof :

Let

$A = w_1, w_2 \ldots, w_k$ and
$B = x_1, x_2 \ldots, x_k$
be an instance of the MPCP

- We convert this instance of MPCP to an instance of PCP that has a solution, iff our MPCP instance has a solution.
- If PCP were decidable then MPCP would be decidable.
- Let $\Sigma \rightarrow$ is the alphabet containing all the symbols in List A and B
- Let ϕ and δ not be in Σ
- Let $y_i \rightarrow$ be obtained from w_i by inserting the symbol $\not\subset$ after each character of w_i
- Let $Z_i \rightarrow$ be obtained from x_i by inserting the symbol $\not\subset$ ahead of each character of x_i
- Great new words

$y_0 = \not\subset y_1$ $\quad\quad Z_0 = Z_1$

$y_{k+1} = \$$ $\quad\quad Z_{k+1} = \not\subset \$$

Example :

Reduce the following MPCP into PCP

	List A	List B
	w_i	x_i
1	1	111
2	10111	10
3	10	0

Solution :

- The lists C and D are constructed from the lists A and B
- Let

  $C = y_0, y_1, \ldots. y_{k+1}$

  $D = Z_0, Z_1, \ldots. Z_{k+1}$

	List C	List D
	y_i	z_i
0	$\not\subset 1 \not\subset 1$	$\not\subset 1 \not\subset 1 \not\subset 1$
1	$1 \not\subset$	$1 \not\subset 1 \not\subset 1 \not\subset$
2	$1 \not\subset 0 \not\subset 1 \not\subset 1 \not\subset 1 \not\subset$	$\not\subset 1 \not\subset 0$
3	$1 \not\subset 0 \not\subset$	$\not\subset 0$
4	$\$$	$\not\subset \$$

- ✶ Lists C and D represents an instance of PCP
- ✶ This instance of PCP has a solution iff the instance of MPCP represented by lists A and list B has a solution.

Undecidability of PCP

Theorem :

PCP is undecidable

Proof :

- ✶ It is sufficient to show that if MPCP were decidable, then PCP would be decidable whether a TM accepts a given word.
- ✶ We reduce Lu to MPCP, which again can be reduced to PCP.
- ✶ For each M and w1 we construct an instance of MPCP that if it has a solution has one starts with

  $\#q_0w\# \alpha_1q_1\beta_1 \#...\#\alpha_k q_k \beta_k$

  where
- ✶ Strings between sucresive #'s are successive ID's in a computation of M with input w.
- ✶ q_x is the final state.
- ✶ Formally the pairs of strings forming lists A and B of the instance of MPCP are given below.
- ✶ The first pair is

List A	List B
#	$\#q_0w\#$

- ✶ The remaining pairs are grouped as

Group 1 :

List A	List B
x	x
#	# for each x in Γ

Group II :

For each q in Q-F, p in Q and x, y and z in Γ

List A	List B	
9 x	y p	if $\delta(q, x) = (p, y, R)$
Zqx	pZy	if $\delta(q, x) = (p, y, L)$
q#	pzy#	if $\delta(q, B) = (p, y, L)$
Zq#	pzY#	if $\delta(q, B) = (p, y, L)$

Group III :

For each q in F, and x and Y in Γ

List A	List B
XqY	q
Xq	q
qY	q

Group IV :

List A	List B	
q##	#	for each q in F

(*x*, y) is a partial solution to MPCP with lists A and B

→if *x* is a prefix of y.

→*x* and y are the concatenation of corresponding strings of lists A and B respectively.

- ✶ If *x*y = y then
  Z is the remainder of (*x*, y)
- ✶ (x, y) =

$$(\#q_0 w\#\alpha_1 q_1\beta_1 \#\alpha_2 q_2 \beta_2 \ldots\#\alpha_{k-1} q_{k-1}\beta_{k-1},$$
$$\#q_0 w\#\alpha_1 q_1\beta_1 \#\ldots\ldots\#\alpha_k q_k \beta_k \#)$$

Example :

$M = (\{q_1, q_2, q_3\}, (0, 1, B\}, \{0, 1\}, \delta, q_1, B, \{q_3\})$

and δ is given as

	0	1	B
q_1	$(q_2, 1, k)$	$(q_2, 0, L)$	$(q_2, 1, L)$
q_2	$(q_3, 0, L)$	$(q_1, 0, R)$	$(q_2, 0, R)$
q_3	-	-	-

Construct an instance of MPCP with lists A and B

w = 01

Solution :

- The first pair is

List A	List B
#	#q, 01#

- The remaining pairs are

Group I :

List A	List B
0	D
1	1
#	#

Group II

List A	List B	
q_1, 0	1 q_2	from $\delta(q_1, 0) = (q_2, 1, R)$
$0q_1$, 1	q_200	

$1q_1, 1$	$q_2\ 10$	from $\delta(q_1, 1) = (q_2, 0, L)$
$0q_1, \#$	$q_2\ 01\ \#$	
$1q_1\ \#$	$q_2 11\#$	from $\delta(q_1, B) = (q_2, 1, L)$
$0q_2\ 0$	$q_3\ 00$	
$1q_2, 0$	$q_0\ 10$	from $\delta(q_{12}, 0) = (q_3, 0, L)$
$q_2\ 1$	$0q_1$	from $\delta(q_2, 1) = (q_1, 0, L)$
$q_2\ \#$	$0q_2\#$	from $\delta(q_2, B) = (q_2, 0, R)$

Group III

List A	List B
$0q_3 0$	q_3
$0q_3 1$	q_3
$1q_3 0$	q_3
$1q_3 1$	q_3

List A	List B
$0q_3$	q_3
$1q_3$	q_3
$q_3 0$	q_3
$q_3 1$	q_3

Group IV :

List A	List B
$q_3\ \#\ \#$	$\#$

- M accepts input w = 01 by the sequence of Id's
  q, 01, 1 q_2 1, 1 0 q_1, 1 q_2 01, q_3 1 0 1, q_3 01, q_3 1, q_3

5.PRIMITIVE RECURSIVE FUNCTION

- **Discuss about primitive recursive function.**

Primitive Recursive Functions :

- The primitive recursive function is defined as follows.
    1. All initial functions are elements of PR
    2. For any $K \geq 0$ and $M \geq 0$ if N^k – N and g_1, g_2, … g_k; $N^m \to N$ are elements

       of PR, then

f (g_1, g_2, … g_k) obtained from f and g1 of g_2, … g_k, by composition is an element PR

3. For any $n \geq 0$ any function $g: N^{n+1} \to N$ in PR, and any function

$h: N^{n+2} \to N$ in PR

the function $f: N^{n+1} \to N$ obtained from g and h by plimitive recursion is in PR.

4. No other functions are in the set PR.

Initial Function :

- The initial function are the following
  1. Constant functions :

For each $K \geq 0$ and each $a \geq 0$, the constant function

$C_a{}^k = Nk \to N$ for every X E N^k

In the case K = 0 we may identify the function $C_a{}^k$ with the number a

2. **The successor function :**

$N \to N$ is defined by the formula s (x) = x + 1

3. **Project Functions**

For much $K \geq 1$ and each I with $1 \leq i \leq k$

the projection function

$P_1{}^k : N^k \to N$ is defined by the formula

$P_i^{\,k}$ $(x_1, x_2,..., x_i,... x_k) = x_i$

Composition :

- ✶ Suppose f is a partial function from N^k to N and for each I with $1 \leq i \leq k,$ $g_i$is a partial function from N^m to N. The partial function obtained from f and g_1, g_2, …, g_k by composition is the partial function h from Nm to N defined by the formula

$h(x) = (f(g_1(x), g_2(x), \ldots g_k(x))$ (XEN^m)

The Primate recursion operation :

- Suppose $n \geq 0$ and g and h are functions of n and n + 2 variables. respectively The

  function obtained from g and h by the operation of primitive recursion is the function.

$f : N^{n+1} \rightarrow N$ defined by the formula

f (x, 0) = g (x)

$f(x, k+1) = h(x, k, f(x_1 k))$

for XEN^n and every $K \geq 0$

Addition and Multiplication :

- The functions Add and Mult from N x N to N defined by the formulas

Add (x, y) = x + y

Mult (x, y) = x * y

are primitive recursive.

- If Add is obtained from g and h by primitive recursion. g and h must be functions of one of three variables.
- The equations are

Add (x, 0) = g (x)

Add (x, k+1) = h (x, k, Add (x, k))

- Add (x, 0) is x and so g to be the initial functions P_1^1
- h (x, k, Add (x, k) should be S (Add (x, k))
- This means $h(x_1, x_2, x_3)$ should be $s(x_3)$ or $S(P_3^3(x_1, x_2, x_3))$
- Therefore Add can be obtained as

$f_0 = p_1^1$ (an initial function)

$f_1 = S$ (an initial function)

$f_2 = p_3^3$ (an initial function)

$f_3 = S(P_3^3)$ (Obtained from f1 and f_2 by compostion)

f_4 = Add (obtained from f_0 and f_3 by primitive recursion)

- To Obtain Mult

Mult (x, 0) = 0

Mult (x, k+1) = x * (k + 1)

= Add (x * k, x)

$$= \text{Add}\ (x, \text{Mult}\ (x, k))$$