Shraddha Patel
CO23

# Assignment - 1

Q1) Ans: **Decidabimerablty :-**

A problem is said to decidable if we can always construct a corresponding algorithm that can answer the problem correctly.
A problem is said to be decidable problem if there exist a corresponding Turning Machine which halts on every input with an answer yes or no.

**Undecidabimerablty :-**

The problem for which we can't construct an algorithm that can answer the problem correctly in finite time are termed as Undecidable problems. These problems may be Partially decidable Problems but their will never be decidable that is there will always be a condition that will lead the turning machine into an infinite loop without providing an answer at all.

eg:- ① Wheather a CFG generates all the strings or not.
② Ambiguity of CFG?

Q2) Ans: **Recursive Enumerable :-**

RE languages or types 0 languages are generated by type - 0 grammer an RE language can be accept or recognized by Turing machine which means it will enter into final state for the strings or may be or may not enter into rejecting state strings of language. It means Turning machine can loop the strings which are not part of the language.

## Recursive Language :-

A recursive language can be decided by machine which means it will enter into finite state for the strings of language and real state for the strings which are not part language.

eg:- $L = \{a^n b^n c^n \mid n \geq 1\}$ is recursive because construct a turing machine which move to state if the strings that Turing Machine will also halt in this case. RE language are also as Turing decidable languages.

Q3) Ans = Rice Theorem :-

- If P is non-trivial property & the language holding the property, $L_P$ is recognized by Turing machine $M_1$, then $L_P = \{<M> \mid L(M_0) \in P\}$ is undecidable.

- Properties :-

① There exist turing machine $M_1$ & $M_2$ that recognize the same language .i.e, either $(<m_1>, <m_2> \in L)$ or $(<m_1>, <m_2> \notin L)$.

② There exist turing machine $M_1$ & $M_2$ where $M_1$ recognize the language which $M_2$ does not i.e, $<m_1> \in L$ & $<m_2> \notin L$.

Q4) Ans = Post Correspondence Problem :-

The PCP introduced by Emil Post in 1946, is an undecidable decision

Problem. The PCP problem over an alphabet $\Sigma$ is state as follows :-
Given the following two lists M and N of non-empty strings over $\Sigma$ -

$$M = (x_1, x_2, x_3, ----, x_n)$$

$$N = (y_1, y_2, y_3, ----, y_n).$$

We can say that there is a Post correspondence Solution, if for some $i_1, i_2, ---, i_k$, where $1 \le i_j \le n$, the condition $x_{i_1}, ---, x_{i_k} = y_{i_1} --- y_{i_k}$ satisfies.

eg:- Find wheather the list $M = (abb, aa, aaa)$ & $N = (bba, aaa, aa)$ have a PCS?

Solution:-

| | $x_1$ | $x_2$ | $x_3$ |
|---|---|---|---|
| M | abb | aa | aaa |
| N | bba | aaa | aa |

Here,

$x_2 x_1 x_3$ = 'aaabbaaa'

and $y_2 y_1 y_3$ = 'aaabbaaa'

We can see that

$$x_2 x_1 x_3 = y_2 y_1 y_3$$

Hence, the solution is $i = 2$, $j = 1$ and $k = 3$.

# Assignment - 2

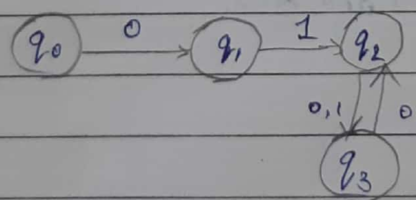
Q1) Ans: **Structural Induction :-**

Many structures in CS are recursively defined i.e, Parts of them exhibit the Same characteristic and have the same properties as the whole Structural Induction is a proof method that works in the domain of recursively defined structures. Structural Induction is used to prove that Same proposition, P(x) holds for all x of some most of recursively defined structures such as formula, list or error.

**Mutual Induction :-**

Mutual induction is the most acurate way for providing L = L(A) being the language and L(n) being the language of a given automation. This method of proof clearly Contains that a certain languages is accepted by by the set, proving the set of states will mutually lead to a final State by a given string.

Q2) Ans: $L = \{w \mid w$ is of even length and begin with $01\}$. DFA for language L :-




DFA State Transition Table

| Q \ Σ | 0 | 1 |
|---|---|---|
| $q_0$ | $q_1$ | - |
| $q_1$ | - | $q_2$ |
| $q_2$ | $q_3$ | $q_3$ |
| $q_3$ | $q_2$ | $q_2$ |

Q3) Ans = $R = \{(1,2), (2,3), (3,4), (5,4)\}$

$\therefore$ Transitive Closure of R.

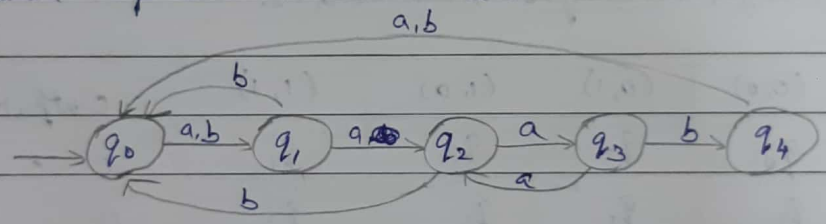
$\{(1,2), (2,3), (1,3), (3,4), (2,4), (5,4)\}$

$\therefore$ Symmetric Closure of R.

$\{(1,2), (2,1), (2,3), (3,2), (3,4), (4,3), (5,4), (4,5)\}$.

Q4) Ans = $L = \{x \mid x$ is made up of $(a,b)$ and ends with 'aab'$\}$.

Regular expression = $(a+b)^*.aab$.




Transition Table.

| Q\E | a | b |
|---|---|---|
| $q_0$ | $q_1$ | $q_1$ |
| $q_1$ | $q_2$ | $q_0$ |
| $q_2$ | $q_3$ | $q_0$ |
| $q_3$ | $q_2$ | $q_4$ |
| $q_4$ | $q_0$ | $q_0$ |

Q5) Ans = DFA minimization using Myhill - Nerode Theorem :-

- Algo 1 :-

Input :- DFA

Output :- Minimized DFA.

Step-1 :- Draw a table for all pairs of states ($Q_i$, $Q_j$), not necessarily connected directly.

Step-2 :- Consider every state ($Q_i$, $Q_j$) in the DFA where $Q_i \in F$ and $Q_j \notin F$ or vice versa and mark them.

Step-3 :- Repeat this step until we cannot mark anymore states - If there is any unmarked pair ($Q_i$, $Q_j$), mark it if the pair $\{\delta(Q_i, A), \delta(Q_i, A)\}$ is marked for some input alphabet.

Step-4 :- Combine all the unmarked pair ($Q_i$, $Q_j$) and make them a single state in the reduced DFA.

- Algo 2 :-

Step-1&2 :- We draw a table for all pair of state & mark the state pairs

| | a | b | c | d | e | f |
|---|---|---|---|---|---|---|
| a | | | | | | |
| b | | | | | | |
| c | ✓ | ✓ | | | | |
| d | ✓ | ✓ | | | | |
| e | ✓ | ✓ | | | | |
| f | | | ✓ | ✓ | ✓ | |

Step-3 :- We input 1 to state 'a' and 'T', it will go to state 'c' and 'f' respectively. (c, f) is already marked, hence we will mark pair (a, f).

Now we input 1 to state 'b' and 'f'; it will go to state 'd' and 'f' respectively. (d,f) is already marked, hence we will mark pair (b,f).

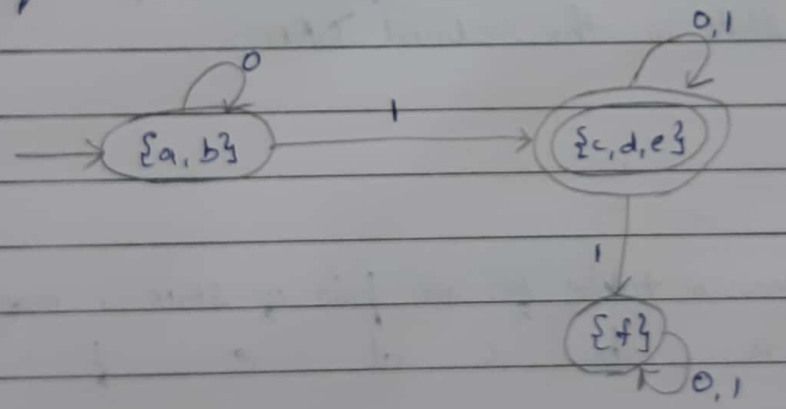
| | a | b | c | d | e | f |
|---|---|---|---|---|---|---|
| a | | | | | | |
| b | | | | | | |
| c | ✓ | ✓ | | | | |
| d | ✓ | ✓ | | | | |
| e | ✓ | ✓ | | | | |
| f | ✓ | ✓ | ✓ | ✓ | ✓ | |

After step-3, we have got state combinations {a, b} {c, d} {c, e} {d, e} that are unmarked.

We can recombine {c, d} {c, e} {d, e} into {c, d, e}

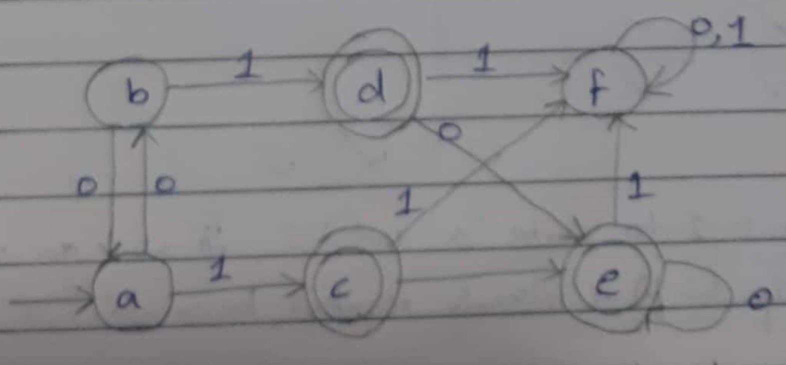
Hence, we got two combined states as - {a, b} and {c, d, e}.

So the final minimized DFA will contain 3 states {f}, {a, b} & {c, d, e}.




- DFA Minimization Using Equivalence Theorem :-

DFA :-



Algorithm :-

Step-1 :- All the states Q are divided in two partitions - final states and non-final states and are denoted by $P_0$. All the states in a partition are $0^{th}$ equivalent. Take a counter k and initialize it with 0.

Step-2 :- Increment k by 1. For each partition in $P_K$, divide the states in $P_K$ into two partitions if they are k-distinguishable. Two states within this partition X and Y are k-distinguishable if there is an input S such that $\delta(X, S)$ and $\delta(Y, S)$ are $(K-1)$ distinguishable

Step-3 :- If $P_K \neq P_{K-1}$, repeat step 2, otherwise go to step-4.

Step-4 :- Combine $K^{th}$ equivalent sets and make them the new states of the reduced DFA.

Scanned with CamScanner

| q | $\delta(q,0)$ | $\delta(q,1)$ |
|---|---|---|
| a | b | c |
| b | a | d |
| c | e | f |
| d | e | f |
| e | e | f |
| f | f | f |

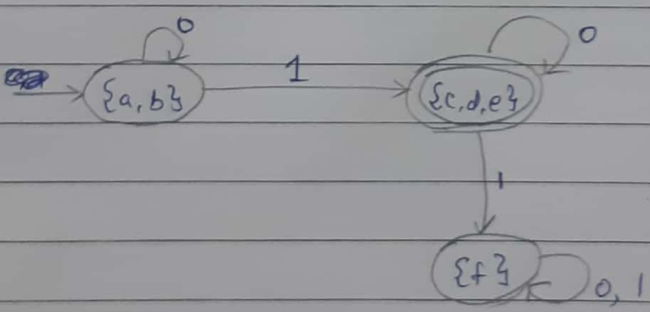
$\therefore P_0 = \{(c,d,e), (a,b,f)\}$

$P_1 = \{(c,d,e), (a,b), (f)\}$

$P_2 = \{(c,d,e), (a,b), (f)\}$

Hence, $P_1 = P_2$

There are three states in the reduced DFA. The reduced DFA is as follows :-




| Q | $\delta(q,0)$ | $\delta(q,1)$ |
|---|---|---|
| {a,b} | {a,b} | {c,d,e} |
| {c,d,e} | {c,d,e} | {f} |
| {f} | {f} | {f}. |

# Assignment - 03

Q1) Ans: **For a Binary Adder :-**

a) Draw FSM :-

Input consist of two symbol 0 or 1.

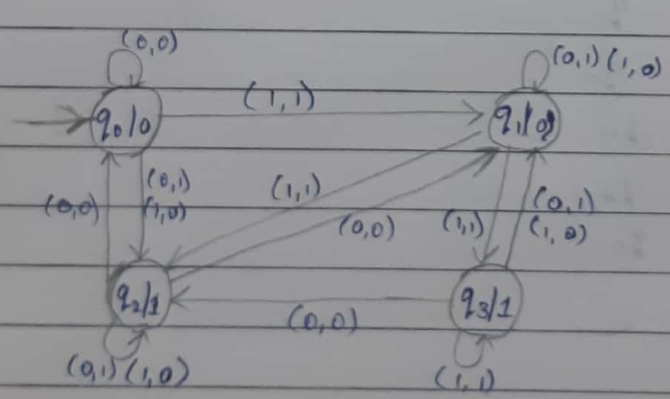
I = {(0,0), (0,1), (1,1)}

where I is a Combination of input symbol.

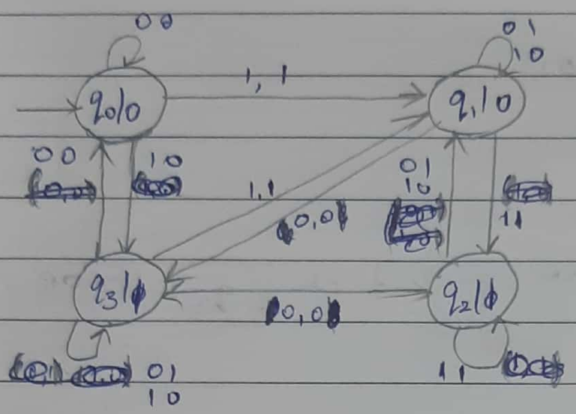
finite set of state, S

⇒ S = {(carry), (no carry)} ; O = {0, 1}.

| S\I | (0,0) | (0,1) | (1,0) | (1,1) | Output |
|---|---|---|---|---|---|
| $q_0$ | $q_0$ | $q_2$ | $q_2$ | $q_1$ | 0 ← carry |
| $q_1$ | $q_2$ | $q_1$ | $q_1$ | $q_3$ | 0 |
| $q_2$ | $q_0$ | $q_2$ | $q_2$ | $q_1$ | 1 ← carry |
| $q_3$ | $q_2$ | $q_1$ | $q_1$ | $q_3$ | 1 |

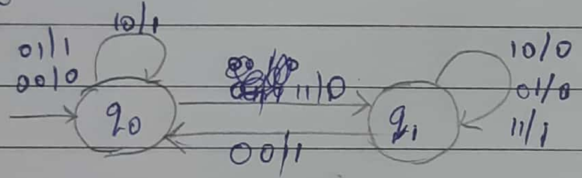


b) Moore Machine :-




| Present State | Next State 00 | 01 | 10 | 11 | Output |
|---|---|---|---|---|---|
| $q_0$ | $q_0$ | $q_3$ | $q_3$ | $q_1$ | 0 |
| $q_1$ | $q_3$ | $q_1$ | $q_1$ | $q_2$ | 1 |
| $q_2$ | $q_3$ | $q_1$ | $q_1$ | $q_2$ | 0 |
| $q_3$ | $q_0$ | $q_3$ | $q_3$ | $q_1$ | 1 |

c) Mealy Machine :-




State Transition Table :-

| | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| $q_0$ | $q_0, 0$ | $q_0, 1$ | $q_0, 1$ | $q_1, 0$ |
| $q_1$ | $q_0, 1$ | $q_1, 0$ | $q_1, 0$ | $q_1, \phi$ |

| Input | Output |
|---|---|
| 00 | 0 |
| 01, 10 | 1 |
| 11 | 0 |
| 01, 10 | 0 |
| 11 | 1 |
| 00 | 1 |

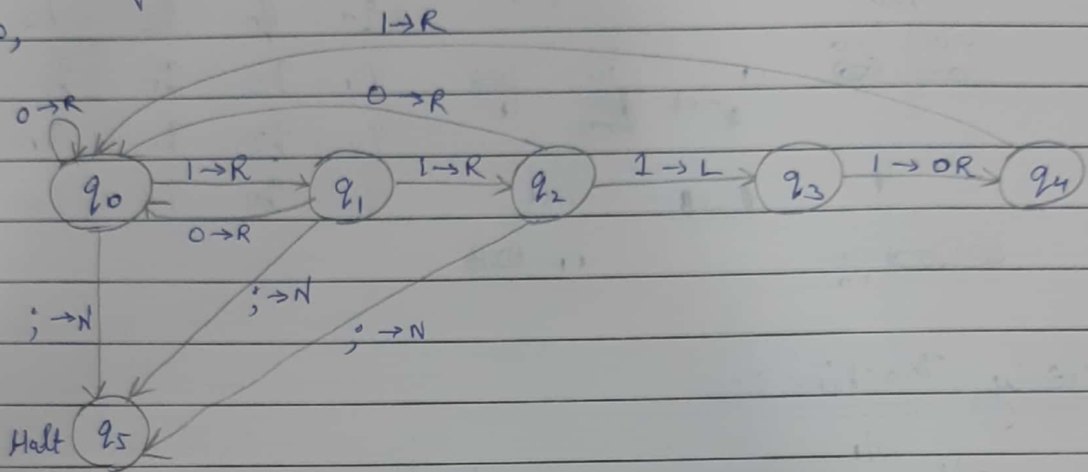
Scanned with CamScanner

# Assignment - 04

Q1) Ans = TM replaces all occurances of '111' with '101' from sequence of 0's & 1's.

Assuming input '111010'

Now,




Hence,

I = {0, 1, ;}

S = {q0, q1, q2, q3, q4, q5 = Halt}

D = {L, R, N}

SFM:-

| S\I | 0 | 1 | ; |
|---|---|---|---|
| q0 | R | q1 R | q5 N |
| q1 | q0 R | q2 R | q5 N |
| q2 | q0 R | q3 L | q5 N |
| q3 | – | 0q4 R | – |
| q4 | – | q0R | – |
| q5 | – | – | – |

Simulation :-

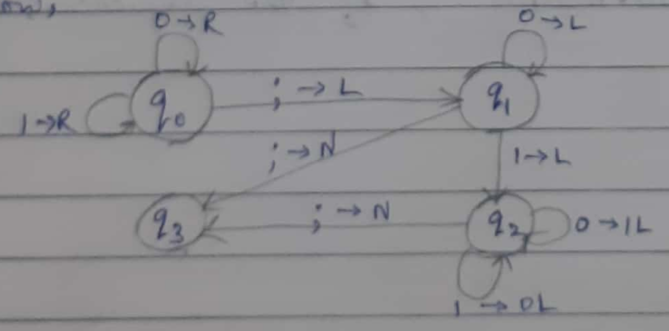
1 1 1 1 0 1 1 1 ; & initial configuration.

11110111; $\delta(q_0, 1) = (q_1 R)$
11110111; $\delta(q_1, 1) = (q_2 R)$
11110111; $\delta(q_2, 1) = (q_3 L)$
10110111; $\delta(q_3, 1) = (0 q_4 R)$
10110111; $\delta(q_4, 1) = (q_0 R)$
10110111; $\delta(q_0, 1) = (q_1 R)$
10110111; $\delta(q_1, 0) = (q_0 R)$
10110111; $\delta(q_0, 1) = (q_1 R)$
10110111; $\delta(q_1, 1) = (q_2 R)$
10110111; $\delta(q_2, 1) = (q_3 L)$
10110101; $\delta(q_3, 1) = (0 q_4 R)$
10110101; $\delta(q_4, 1) = (q_0 R)$
10110101; $\delta(q_3, ;) = (q_5 N)$

Q2) Ans: TM for 2's complement of binary number for this TM.

$\Rightarrow$ $I = \{0, 1, ;\}$
$S = \{q_0, q_1, q_2, q_3 = halt\}$
$D = \{L, R, N\}$

Now,




STM :-

| S\I | 0 | 1 | ; |
|---|---|---|---|
| $q_0$ | R | R | $q_1 L$ |
| $q_1$ | L | $q_2 L$ | $q_3 N$ |
| $q_2$ | 1 L | 0 L | $q_3 N$ |
| $q_3$ | – | – | – |

Simulation :-

; 01010; initial configuration.

; 01010;　　$\delta(q_0, 0) = (R)$

; 01010;　　$\delta(q_0, 1) = (R)$

; 01010;　　$\delta(q_0, 0) = (R)$

; 01010;　　$\delta(q_0, 1) = (R)$

; 01010;　　$\delta(q_0, 0) = (R)$

; 01010;　　$\delta(q_0, ;) = (q_1 L)$

; 01010;　　$\delta(q_1, 0) = (L)$

; 01010;　　$\delta(q_1, 1) = (q_2 L)$

; 01110;　　$\delta(q_2, 0) = (1 L)$

; 00110;　　$\delta(q_2, 1) = (0 L)$

; 10110;　　$\delta(q_2, 0) = (1 L)$

; 10110;　　$\delta(q_2, ;) = (q_3 N)$.

**Algorithm :-**

① Move towards the right till you reach ';' which is the right end-marker of the sequence.

② Start moving towards the left till you reach first 1, then move towards the left and replace each 0 by 1 and each 1 by 0, till you reach the left end-marker ';' of the sequence, then halt.

# Assignment - 5

Q1) Ans: Let us ~~assume~~ prove that the halting problem is unsolvable by contradiction.

Let us assume that there exists a TM A, which decides wheather or not any computation by an TM T will ever halt, given the description $d_T$ (SFM) of T, and the input tape t of T. Then, for every input $(t, d_T)$ to A, if T halts, then A reaches an 'accept halt' state; else, A reaches a 'reject halt' state. Fig (a) shows a diagrammatic representation of the working of TM A.

We now attempt to construct another TM B, which takes $(t, d_T)$ as the input; it functions as follows:-

First it copies the input & duplicates the same onto its tape. Then it takes this duplicated info. tape as the input to A. Whenever A reaches the 'accept halt' state, B loops forever, and whenever A reaches the 'reject halt' state, B halts. Fig (B) shows a diagrammatic representation of the working of TM B.

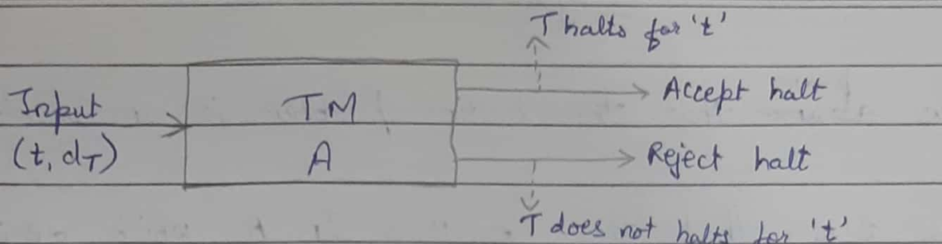
Considering the original behaviour of A, we find that B acts as follows:-

It loops if T halts for input t and halts if T does not halt for the input t. Thus, the working of TM B is exactly opposite to that of TM T.

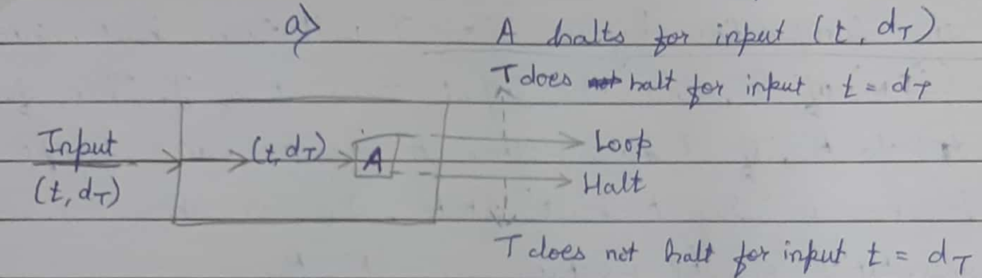
Now, since B itself is a TM, let us set T = B. In this case, B halts for the input if and only if B does not halt for the same input, and loops forever if & only if B halts for the input. Fig (c) shows the working of TM B, which takes itself as input.

This is a contradiction.

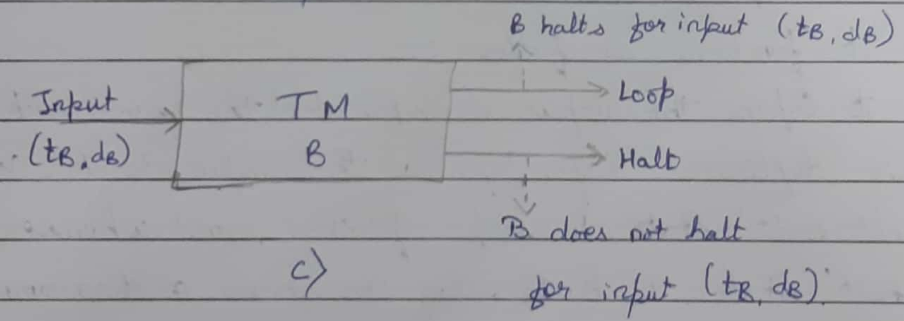
Hence, we conclude that the machine A, which can decide wheather or not any other TM will ever halt cannot exist. Therefore, we conclude that the halting problem is unsolvable.


T halts for 't'
Input
(t, d_T)
TM
A
Accept halt
Reject halt
T does not halts for 't'

a)


A halts for input (t, d_T)
T does ~~not~~ halt for input t = d_T
Input
(t, d_T)
(t, d_T)
A
Loop
Halt
T does not halt for input t = d_T

b)


B halts for input (t_B, d_B)
Input
(t_B, d_B)
TM
B
Loop
Halt
B does not halt
for input (t_B, d_B)

c)