Unit 4

Turing Machine

- A turing Machine is an accepting device which accepts the languages (recursively Enumerable languages) generated by type 0 grammar.
- It was invented in 1936 by the Mathematician Alan turing.

Defenition

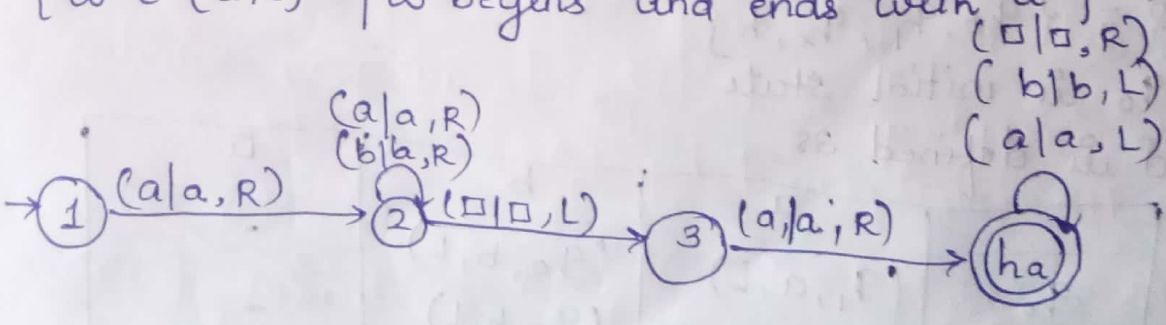
A Turing Machine can be formally described as a 7-tuple $(Q, X, \Sigma, \delta, q_0, B, F)$

Where

Q is a finite set of states.
X is the tape alphabet
Σ is the input alphabet
δ is a transition function : $\delta : Q \times X \longrightarrow Q \times X \times \{L, R\}$
q_0 is the initial state.
B is the Blank symbol.
F is the set of final states.

1) Construct a turing Machine for the language
$L = \{ w \in (a,b)^* \mid w$ begins and ends with a' $\}$

Soln



2) Design a TM that recognizes the language of all strings of even length over alphabet (a, b)

Soln

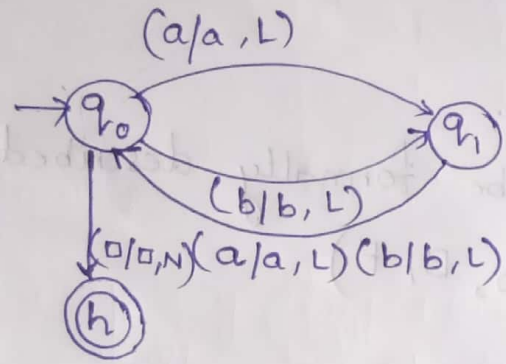
$TM = (Q, \Sigma, T, \delta, q_0, h)$

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

$T = \{a, b, \square\}$

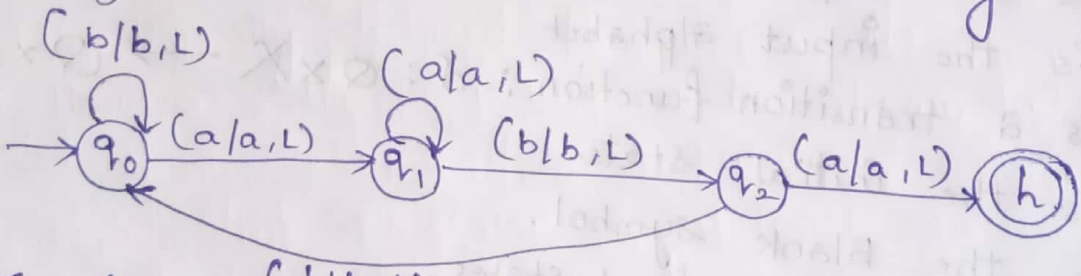
$Q = \{q_0, q_1, h\}$

{aa, ab, ba, aaaa, bbbb, ...}



3) Design a Turing machine that accepts the language of all strings which contain aba as a substring.

Soln



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

$T = \{a, b, \#\}$

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

q_0 is initial state

δ is defined as

	a	b	□
q_0	(q_1, a, L)	(q_0, b, L)	-
q_1	(q_1, a, L)	(q_2, b, L)	-
q_2	(h, a, L)	(q_0, b, L)	$(h, \square, N)$
h	-	-	Accept

4) Design a turing Machine that replace every 0 and 1 with every 1 with 0 in a binary string.

or

Prove that function $f(\omega) = \overline{\omega}$ is computable where $\omega \in \{0,1\}^*$ and $\overline{\omega}$ the one's complement of ω.

Soln

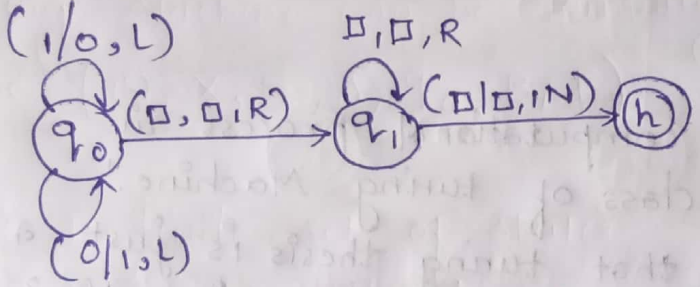
$TM = (Q, \Sigma, T, \delta, q_0, h)$

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

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

$T = \{0,1,\square\}$

q_0 is initial state



N denotes head does not move that is continous scanning same cell.

5) Design a TM that recognizes the set of all strings of 0's and 1's containing at least one 1.

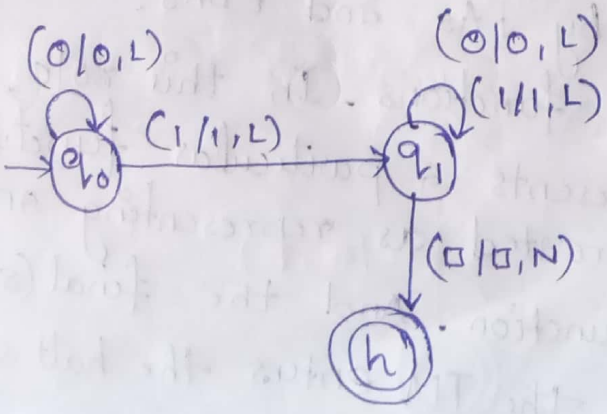
Soln

$TM = (Q, \Sigma, \Gamma, \delta, q_0, h)$

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

$T = \{0,1,\square\}$

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



Here the halt state h is reached only when at least one 1 is encountered and h works as accepted state.

Let us process the string : ⊔⊔00011⊔

$$\sqcup 00011 q_0 \vdash \sqcup 0001 q_1 1 \vdash \sqcup 000 q_1 11 \vdash \sqcup 00 q_1 011 \vdash \sqcup 0 q_1 0011 \vdash \sqcup q_1 00011 \vdash h \sqcup 00011$$

Turing Thesis

"The power of any computational process is captured within the class of turing Machine."

It may be noted that turing thesis is just a conjecture and not a theorem, hence turing thesis cannot be logically deduced from more elementary facts. However, the conjecture can be shown to be false. if a more powerful computational model is proposed that can recognize at least one more language that is not recognized by the TM.

The turing Machine are designed to play atleast the following three different roles:

(i) As accepting device for languages, similar to the role played by FAs and PDAs.

(ii) As a computer of functions. In this role, a Turing Machine represents a particular function Initial input is treated as representing an argument of the function. And the final (string) on the tape when the TM enters the halt state

treated as representative of the value obtained by an application of the function to the argument represented by the initial string.

(iii) As an enumerator of strings of a language, that outputs the strings of a language, one at a time, in some systematic order that is as a list.

Combing a Turing Machine for complicated Task

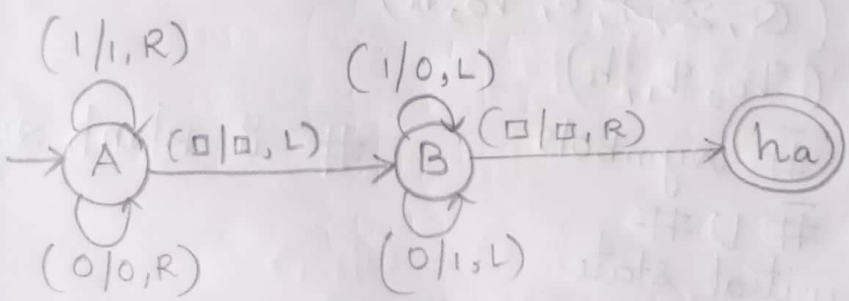
- In this section we present a method to build complex turing machines by combining simpler ones.
- A simple Turing machine can work as "function" for other complex turing machine.

1) construct a turing Machine to compute 1's complement of a given binary numbers.

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

The function $f()$ is defined by

$f(x)$ = 1's complement (x) / x is a binary string.



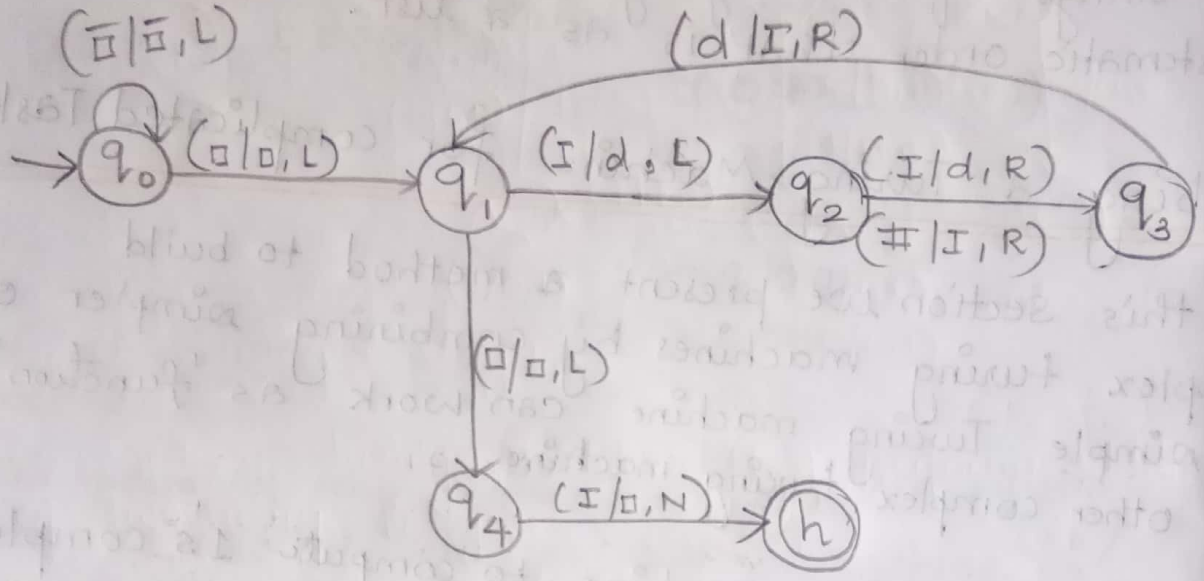
2) Construct a turing Machine that computes the following function

$$f(n,m) = n+m$$

$f(2,3) = 2+3 = 5$

(II) + (III)

= IIIII



3) Design a turing machine which works as eraser.

Soln

In the above problem we want to design a turing machine which works as a eraser, that is if we pass input I□abb□ then output should be □□, that is null string.

$TM = (Q, \Sigma, T, \delta, q_0, h)$

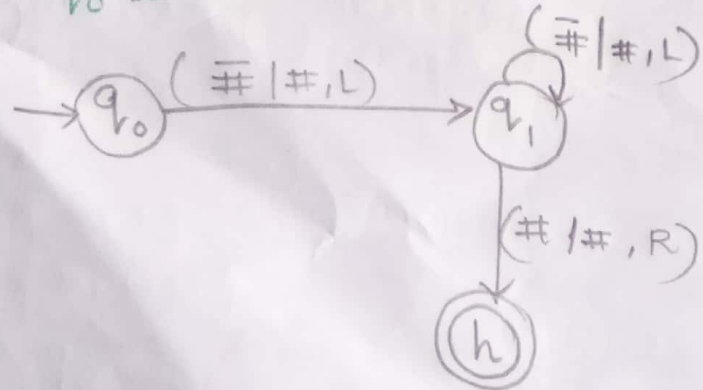
$Q = (q_0, q_1, h)$

$\Sigma =$ # symbols

$T = \overline{\#} \cup \#$

is Blank symbol.

q_0 is initial state



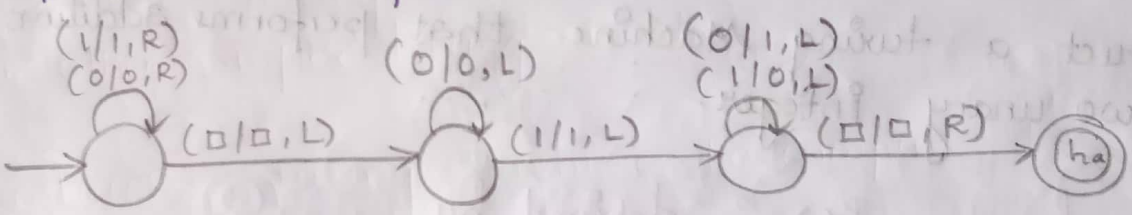
Let us process the string $ababaq_0 \vdash \#ababq_1\# \vdash \#abaq_1\#\# \vdash \#abq_1\#\#\# \vdash \#aq_1\#\# \vdash \#q_1\#\# \vdash \#\#h$

4) Compute the turing Machine to compute 2's complement.

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

The function $f()$ is defined by

$f(x)$ = 2's complement of the number.

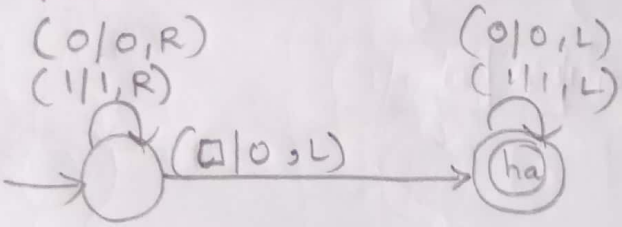


5) Construct a turing Machine to double the binary number.

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

The function $f(x)$ is defined by

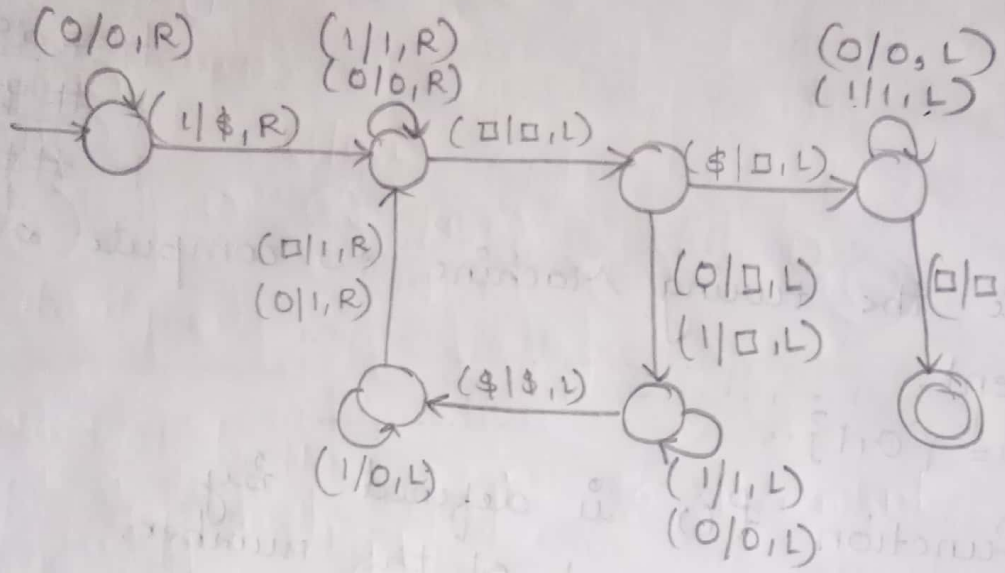
$f(x)$ = twice times the binary numbers.



6) Construct a turing machine to compute $\log_2 x$.

Soln.

$\log_2(1) = 1$
$\log_2(4) = 2$
$\log_2(16) = 4$
$\log_2(8) = 3$



7) Construct a turing Machine that performs addition of two unary Integer.

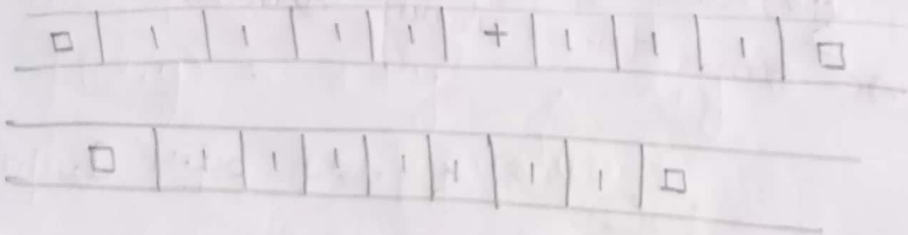
Soln

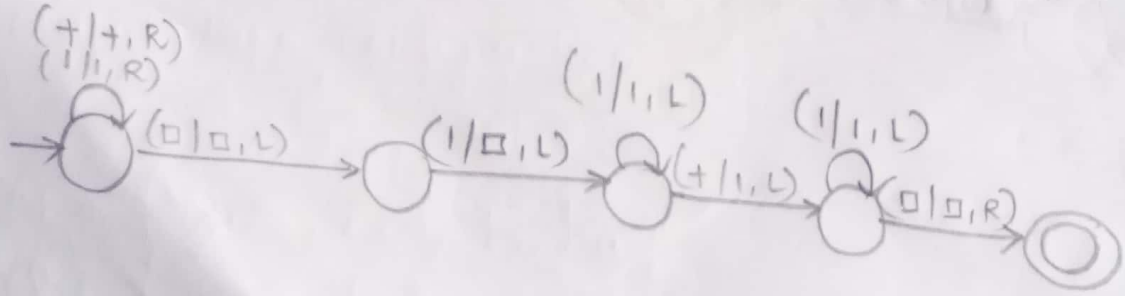
The function f() is defined by

$f(x,y) = x+y \mid x,y \in 1^*$

$x = 1^m 1^m \ldots$

$y = 1^n 1^n \ldots$



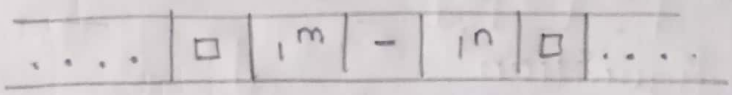


8) Construct a turing machine that performs subtraction of two unary integer.

Soln

$$f(x,y) = \begin{cases} x-y & ; \text{when } x>y \\ 0 & ; \text{otherwise} \end{cases}$$

$= \{|x-y| \text{ , } x \& y \text{ are unary integer}$



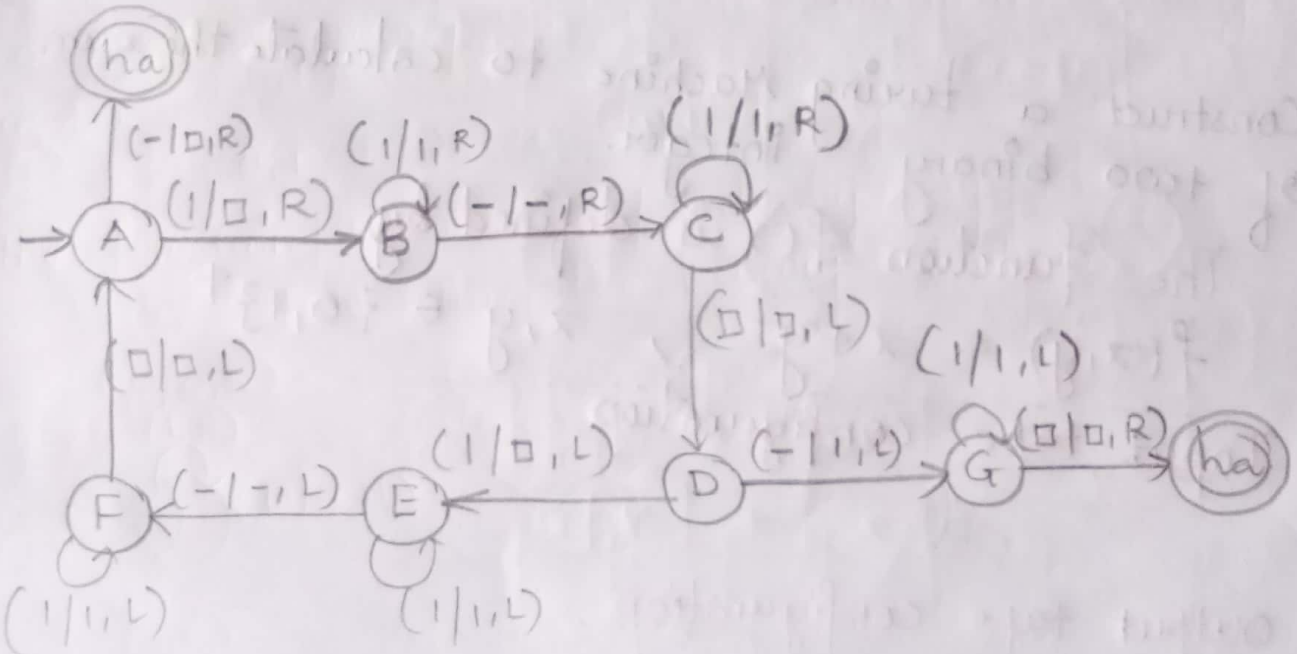
The function $f()$ is defined by

$f(x,y) = \{ |x-y| \mid x, y \in \{1\}^+ \}$

$M = \{Q, \Sigma, T, q_0, \delta, \square, ha\}$

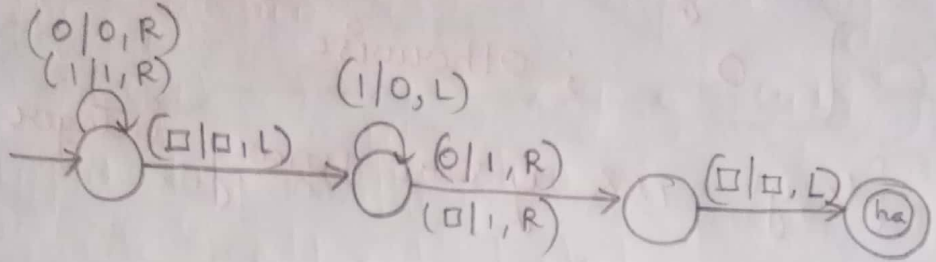
$\Sigma = \{1\}$

$T = \{1, 0, \square\}$



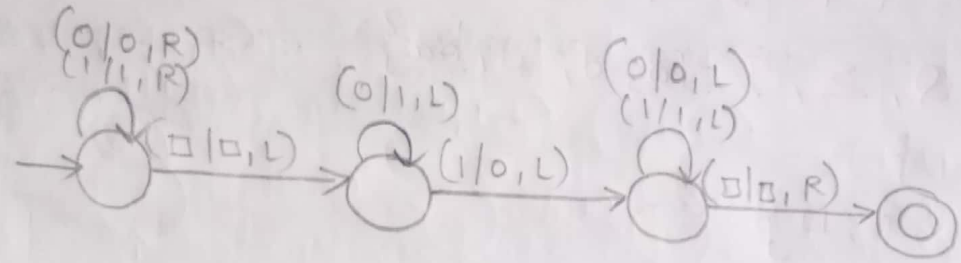
9) Construct a turing Machine to increment given binary integer by 1.

Soln



10) Construct a turing machine to decrement given integer by 1.

Soln

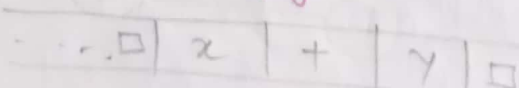


11) Construct a turing Machine to calculate the sum of two binary integer.

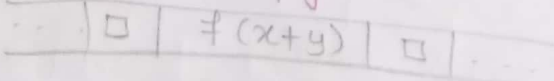
Soln The function $f()$ is defined by

$$f(x,y) = x+y \quad , \quad x,y \in \{0,1\}^+$$

Input tape configuration



Output tape configuration



$M = \{Q, \Sigma, T, \delta, q_0, h_a\}$

$Q = \{\ldots, h_a\}$

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

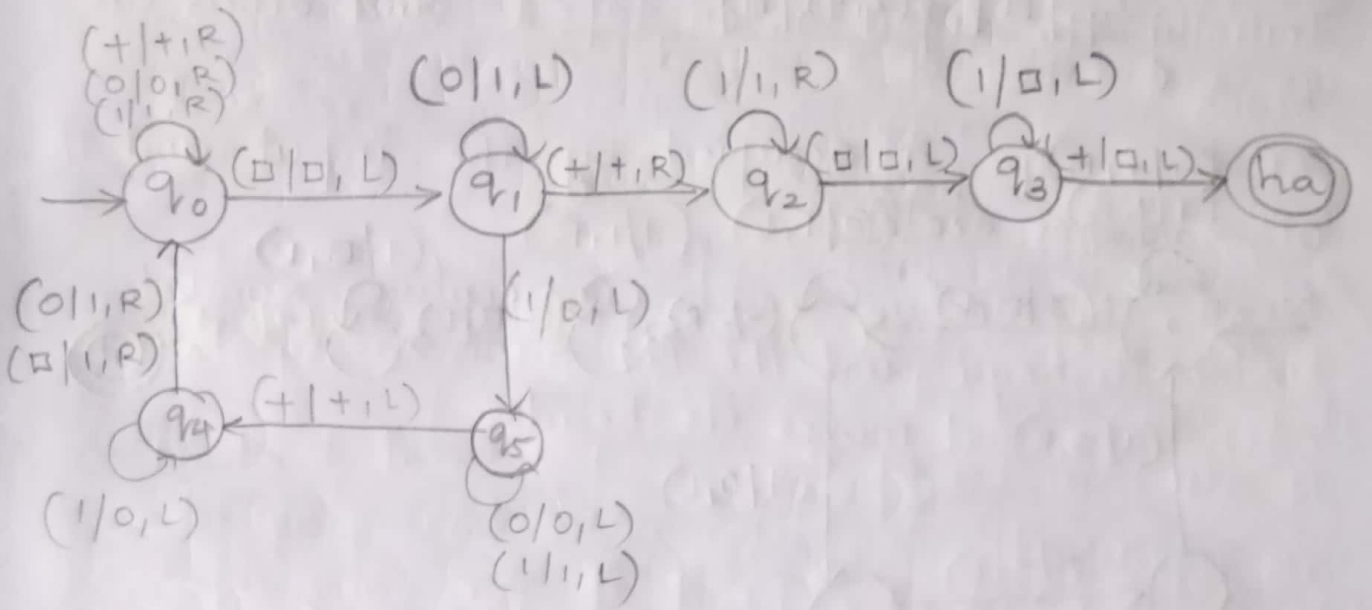
$T = \{0, 1, \square\}$

$q_0 = q_0$

Logics:

Increment by 1
- Look for the first 0 from LSB
  - replace by 1
- Replace every 1 by 0

Decrement by 1
- Look for the first 1 from LSB
- replace by 0.



12) Construct a turing machine to calculate difference of two binary numbers.

Sol^n.

The function $f()$ is defined by

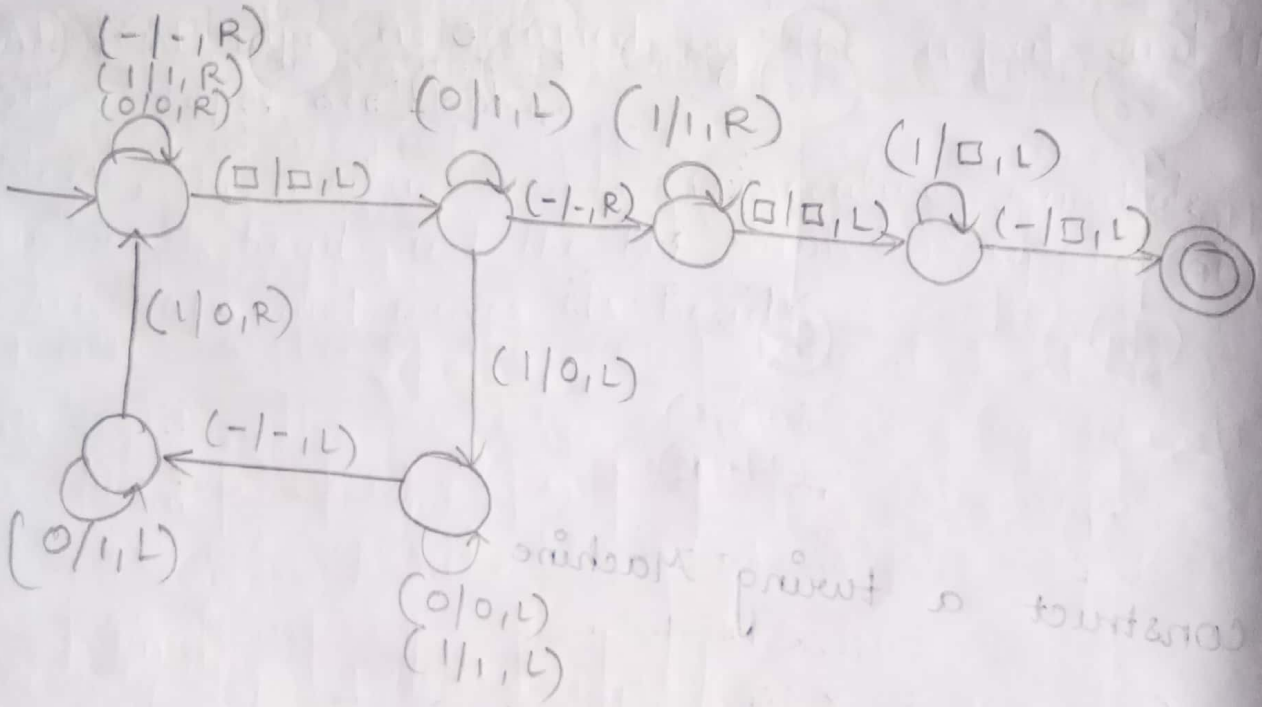
$$f(x,y) = \{x-y \;,\; x,y \in \{0,1\}^+\}$$

Input tape configuration

...	□	x	-	y	□	...

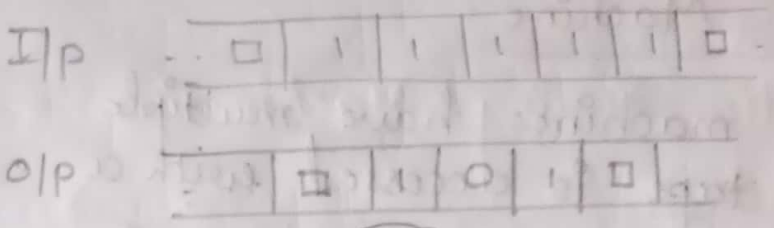
Output tape configuration

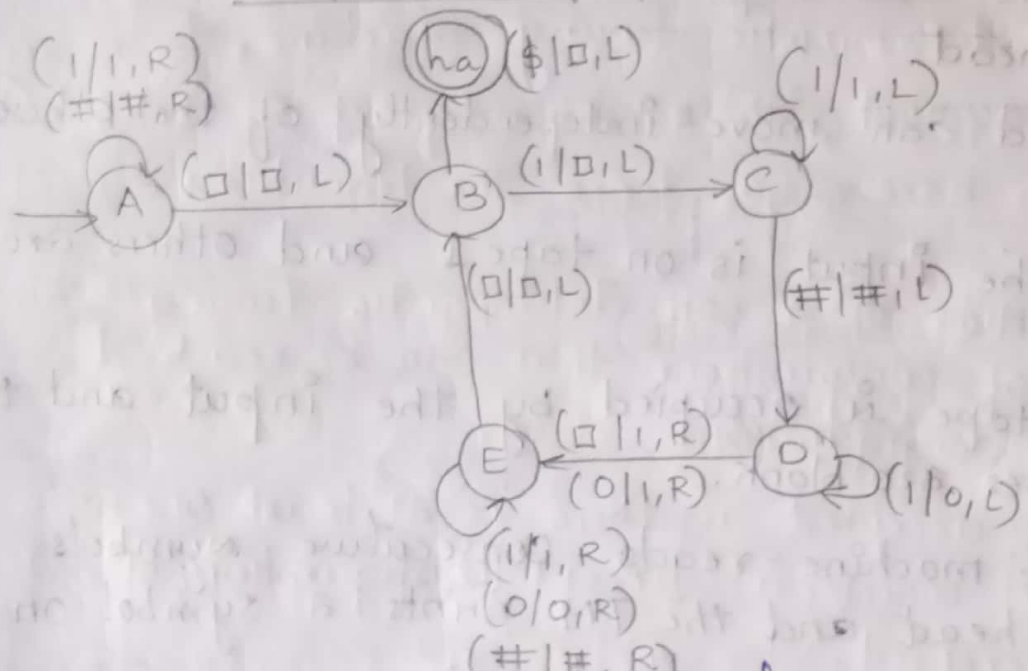
...	□	f(x,y)	□	...



13) $f(x_1) = \{ x_2, x_1$ is an unary integer and x_2 is a binary integer$\}$ (unary to binary)

Sol:





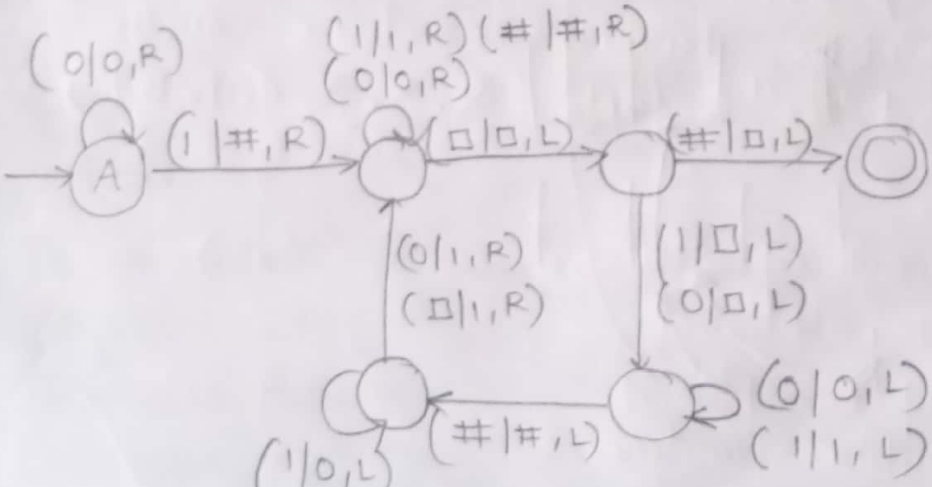
14) Construct a turing Machine for $f(x) = \{ \log_2 x \mid x$ is a binary integer $\}$

$\log_2(1) = 0 \mid 2^0$

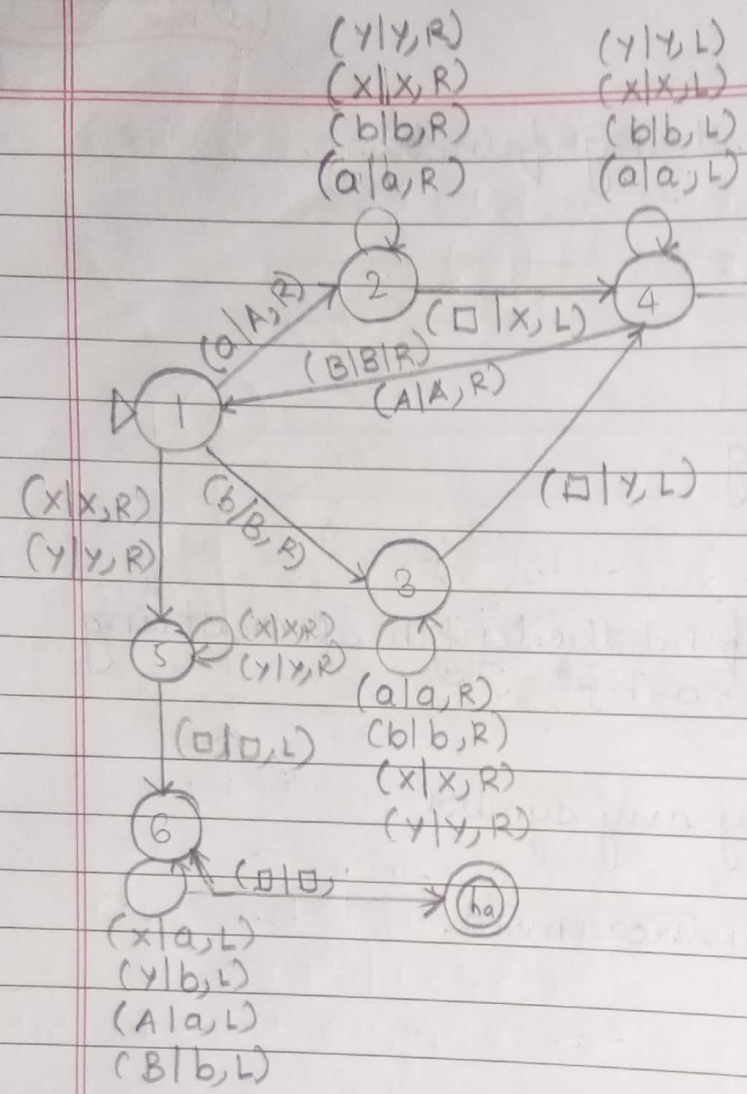
$\log_2(2) = 1 \mid 2^1$

1 1 0
2

1 0 □
1 # 1 □ □
1 0 # □ □ □



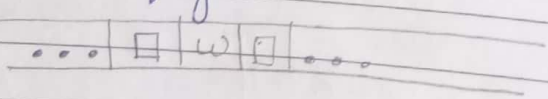
Q. Turing Machine for $L = \{ww \mid w \in \{a,b\}^*\}$



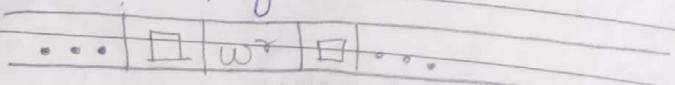
* String Reverse

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

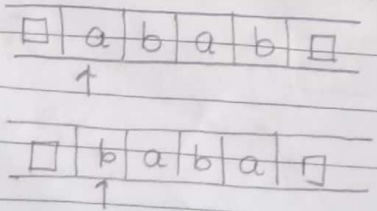
I/p tape configuration



O/p tape configuration



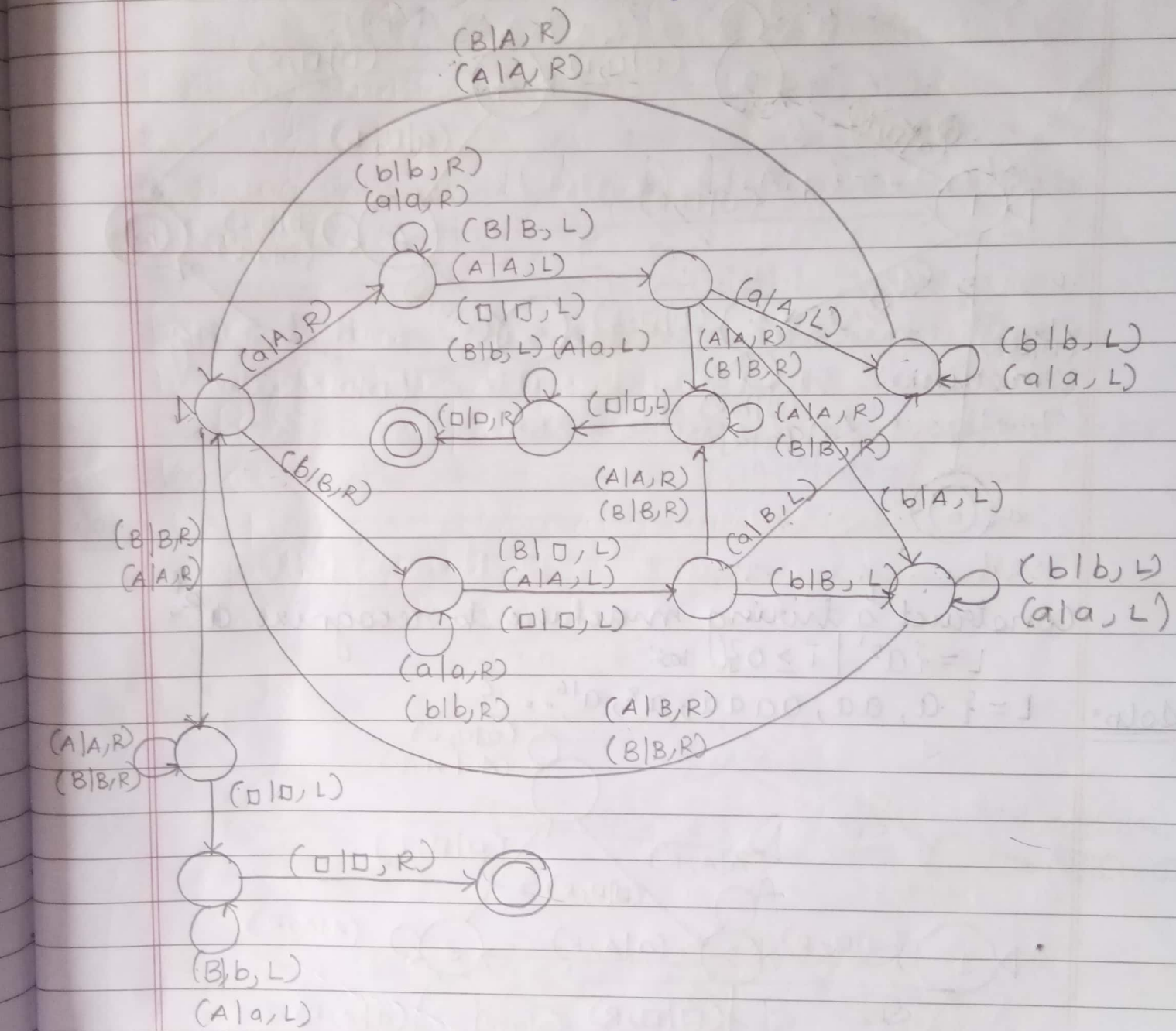
Example:

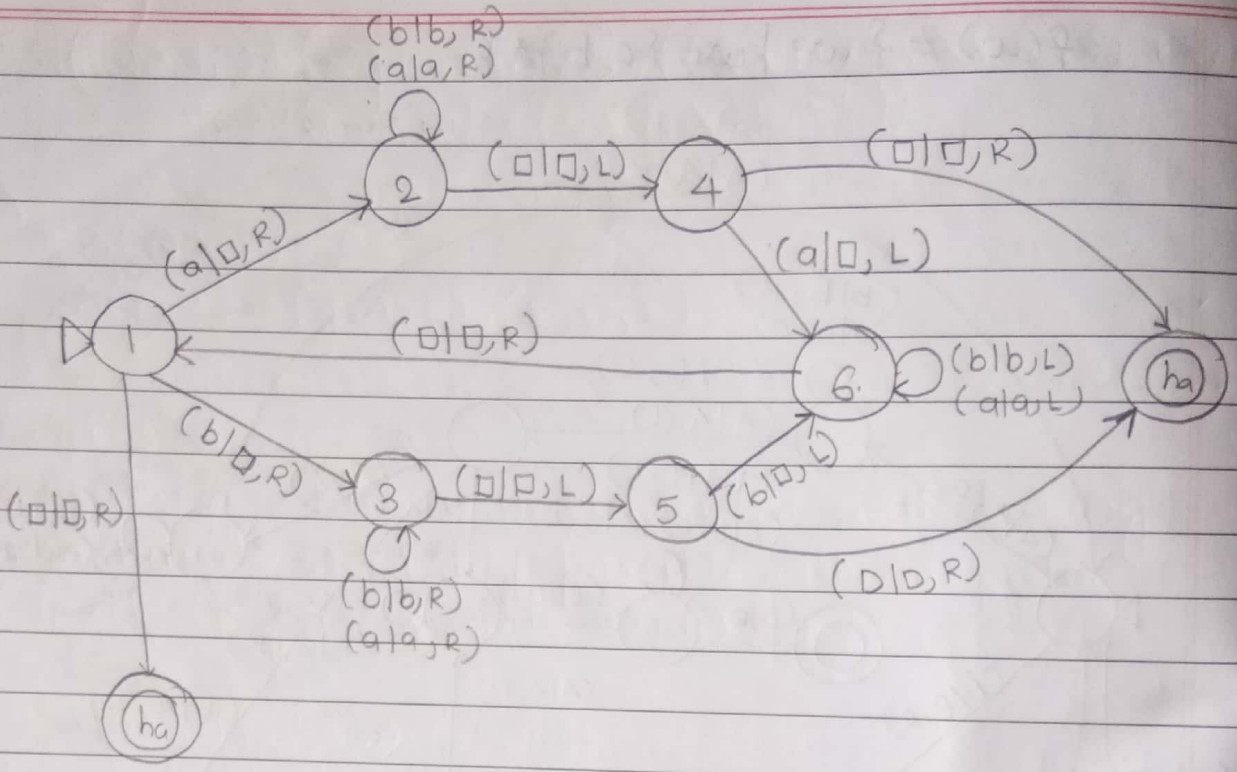


The function .f() is defined by

classmate
Date
Page

$f(w) = \{ w^* \mid w \in \{a,b\}^* \}$

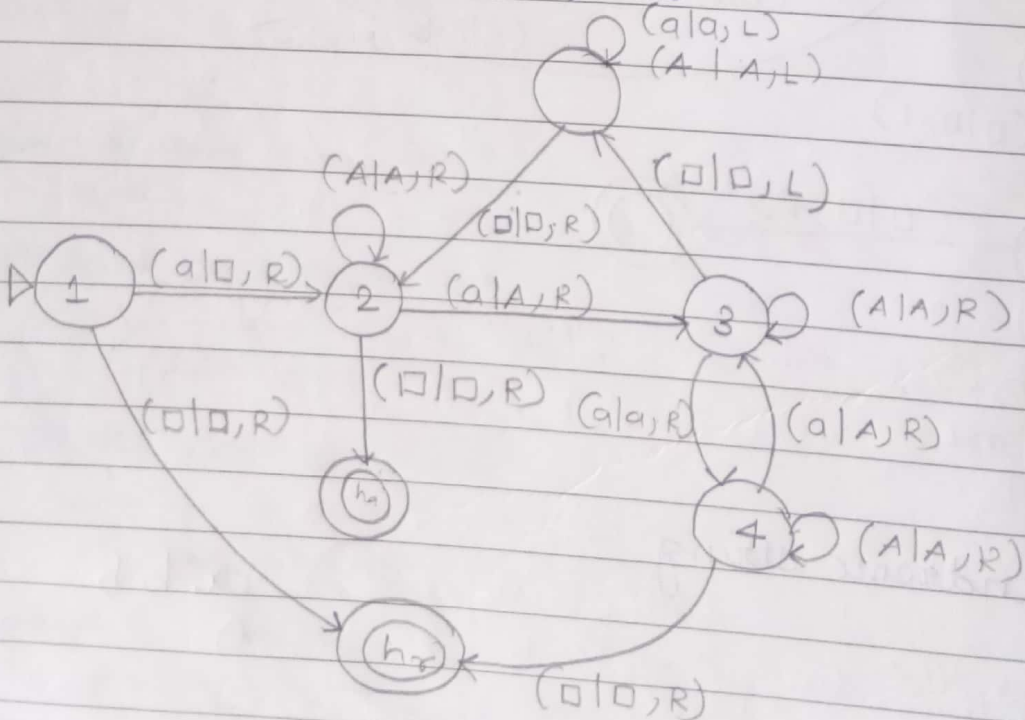




Construct a turing machine to recognize a^{2^i}.

$L = \{a^{2^i} \mid i \geq 0\}$

Soln: $L = \{a, aa, aaaa, a^8, a^{16}, \ldots\}$

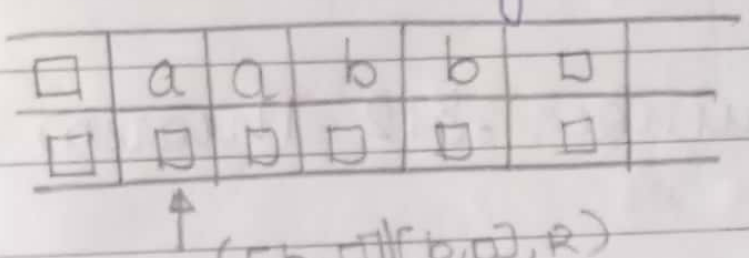


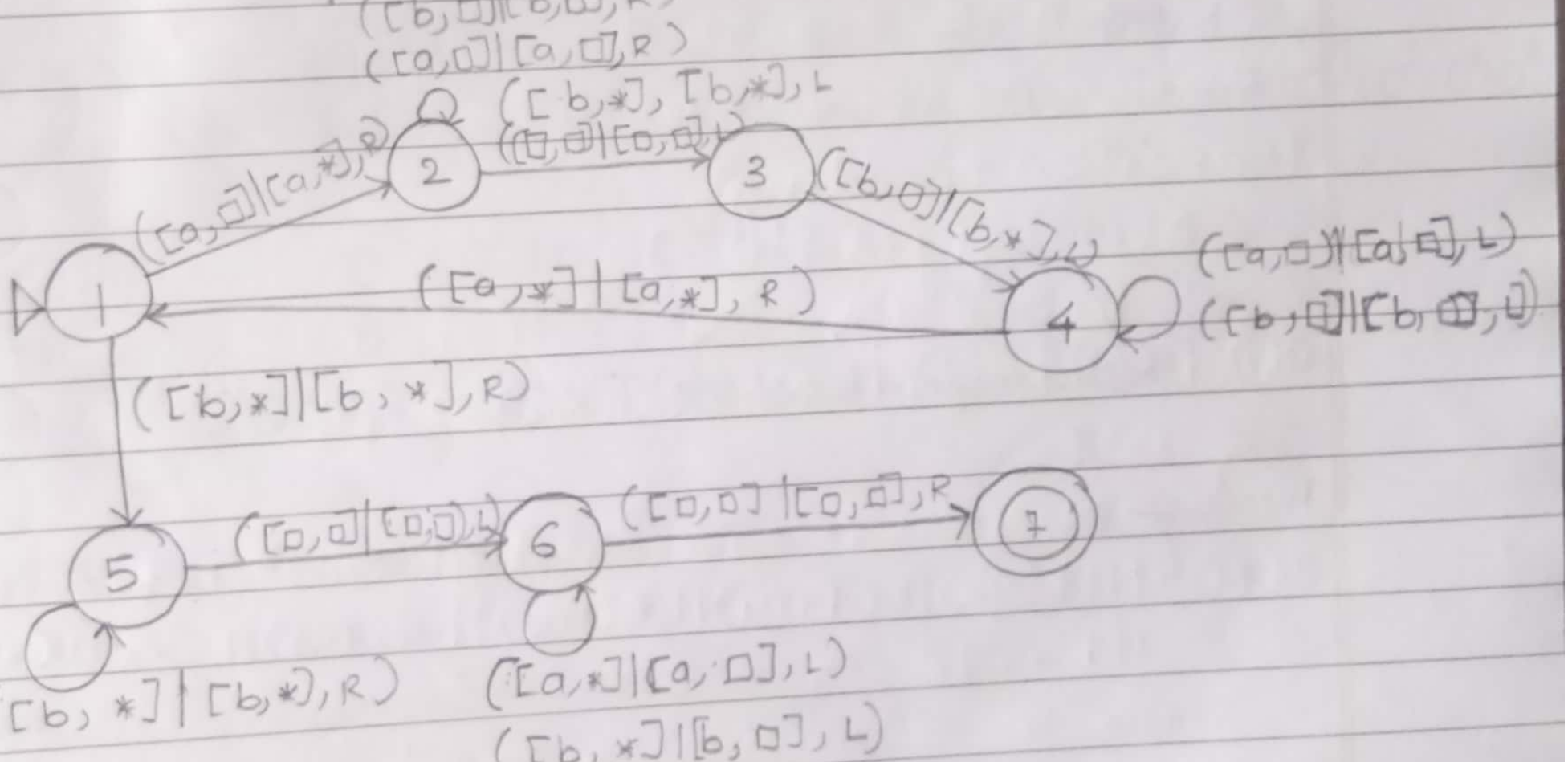
□ a a a a □

□ □ A A

1. Design a turing machine with 2 dimensional tape to determine the string $L = \{a^n b^n \mid n > 0\}$

Ans:

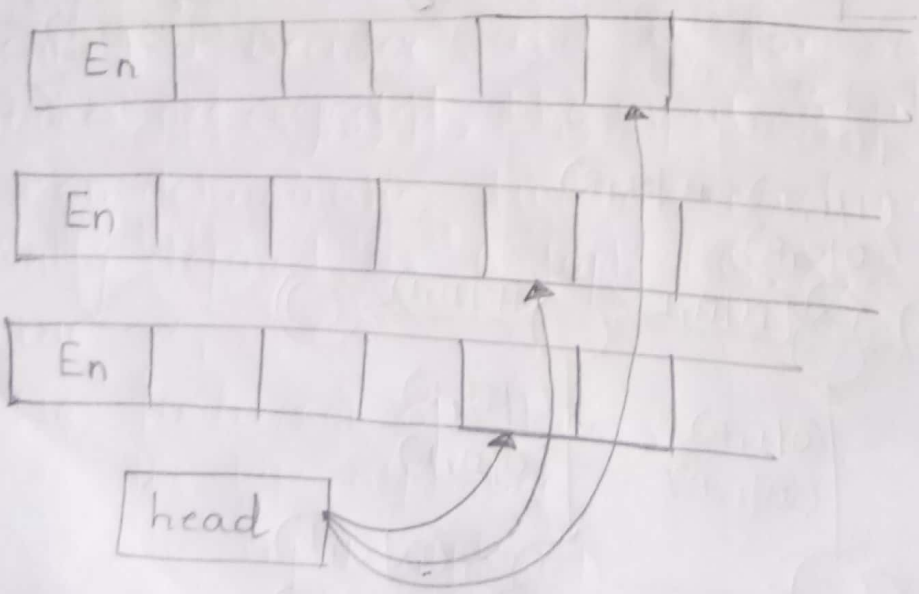




Turing Machine with More complex storage / Variants of Turing Machine

Multitape Turing Machine

- Multi tape turing machines have multiple tapes where each tape is accessed with a separate head.
- Each head can move independently of the other heads.
- Initially the input is on tape 1 and others are blank.
- At first tape is occupied by the input and the other tapes are blank.
- Next, the machine reads consecutive symbols under its head and the TM prints a symbol on each tape and moves its heads.



A multitape turing Machine can be formally described as a 6-tuple $(Q, X, B, \delta, q_0, F)$ where

Q is a finite set of states.
X is the tape alphabet
B is the Blank symbol
δ is a relation on states and symbols where

$$\delta : Q \times X^K \longrightarrow Q \times (X \times \{\text{left\_shift}, \text{Right\_shift}, \text{No\_shift}\})^K$$

where there is K number of tapes.
q_0 is the initial state
F is the set of final states

* Every multitape turing machine has an equivalent single tape Turing Machine.

- Multidimensional Turing Machine

A multidimensional turing Machine is one in which the tape can be viewed as extending infinitely in more than one direction.

$$\delta : Q \times T \longrightarrow Q \times \Gamma \times \{L, R, U, D\}$$

where U and D specify movement of the read write head up and down, resp.

- To simulate this machine on a standard turing machine, we can use the two-track model.
- The two track tape of the simulating machine will use one track to store cell contents and the other one to keep the associated address.
- The cell address can involve arbitarily large integers, so the address track cannot use a fixed size field to store addresses. Instead

we must use a variable field size arrangement, using some special symbols to delimit the fields.

Non-Deterministic Turing Machine

- In a non-deterministic turing machine, for every state and symbol, there are a group of actions the TM can have. So, here the transitions are not deterministic.
- The computation of a non-deterministic Turing Machine is a tree of configurations that can be reached from start configuration.
- An input is accepted if there is at least one node of the tree which is accept configuration, otherwise it is not accepted.
- If all branches of the computational tree halt on all inputs, the non-deterministic turing machine is called a Decider and if for some input, all branches are rejected, the input is also rejected.
- A non-deterministic turing machine can be formally defined as 6-tuple $(Q, X, \Sigma, \delta, q_0, B, F)$ where

  Q is finite set of states
  X is the tape alphabet
  Σ is the input alphabet
  δ is a transition function

  $$\delta : Q \times X \longrightarrow P(Q \times X \times \{\text{left\_shift}, \text{Right\_shift}\})$$

  q_0 is initial state
  B is Blank Symbol
  F is the set of final states.

A universal Turing Machine

- Universal turing machine is a reprogrammable machine which can be used to simulate other turing machine including itself.
- It is denoted by 'U'

Input of universal turing machine
1. Description of transition of M
2. Input string of M.

Encoding

The encoding of turing machine is a binary string which begins with control sequence 111 and ends with it.
It consist of block separated by sequence 11 and each block is of the form $0^i 1 0^j 1 0^k 1 0^l 1 0^r$

Steps

1. Alphabet Encoding
2. State Encoding
3. Head Movement Encoding
4. Transition Encoding

Alphabet Encoding

$\Sigma = \{a_1, a_2 \dots a_n\} \cup T$

encoding $(a_1) = 0^1$

encoding $(a_2) = 0^2$

⋮

State encoding

Let $Q = \{q_1, q_2 \dots q_n\}$

$e(q_1) = 0^1$

$e(q_2) = 0^2$

Head Movement Encoding

encoding (L) = 0^1

encoding (R) = 0^2

encoding (S) = 0^3

Tape symbol encoding

$T = \{\square, x, y\}$

encoding ($\square$) = 0^1

encoding (x) = 0^2

⋮

Transition Encoding

$\delta(i, a) = (j, A, \{hm\})$ — head movement

$e(i) \mid e(a) \mid e(j) \mid e(A) \mid e(hm)$

Turing Machine with initial state q_0,

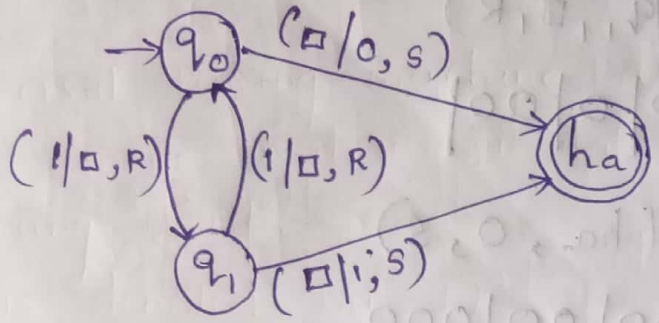
$e(M) = 111\, e(q_0) \,11\, e(m_1) \,11\, e(m_2) \,11\, e(m_3) \,11 \ldots 11\, e(m_n) 111$

To find the input string

$w = x_1 x_2 x_3 \ldots x_n$

$e(w) = e(x_1) \mid e(x_2) \mid e(x_3) \mid \ldots \ldots e(x_n)$

1) Encode the given turing machine



Soln Given:-

$\Sigma = \{1\}$

$T = \{0, 1, \square\}$

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

$H = \{L, R, S\}$

Alphabet Encoding

$\Sigma = \{1\}$

$e(\square) = 0^1$

$e(0) = 0^2$

$e(1) = 0^3$

State encoding

$e(q_0) = 0^1$

$e(q_1) = 0^2$

$e(ha) = 0^3$

Tape symbol encoding

$e(\square) = 0^1$

$e(0) = 0^2$

$e(1) = 0^3$

Head Movement Encoding

$e(L) = 0^1$

$e(R) = 0^2$

$e(S) = 0^3$

Transition Encoding

$m_1 : \delta(q_0, 1) = (q_1, \square, R)$

$e(m_1) = 0|000|00|0|00$

$m_2 : \delta(q_0, \square) = (ha, 0, S)$

$e(m_2) = 0|0|000|00|000$

$m_3 : \delta(q_1, 1) = (q_0, \square, R)$

$e(m_3) = 00|000|0|0|00$

$m_4 : \delta(q_1, \square) = (ha, 1, S)$

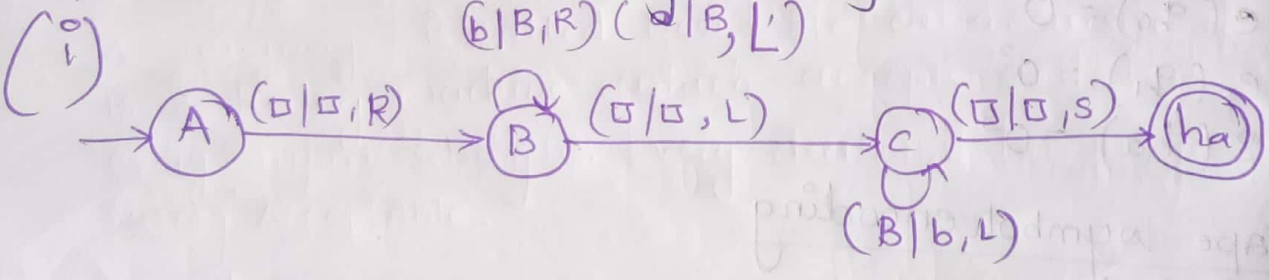
$e(m_4) = 00|0|000|000|000$

Encoding of TM

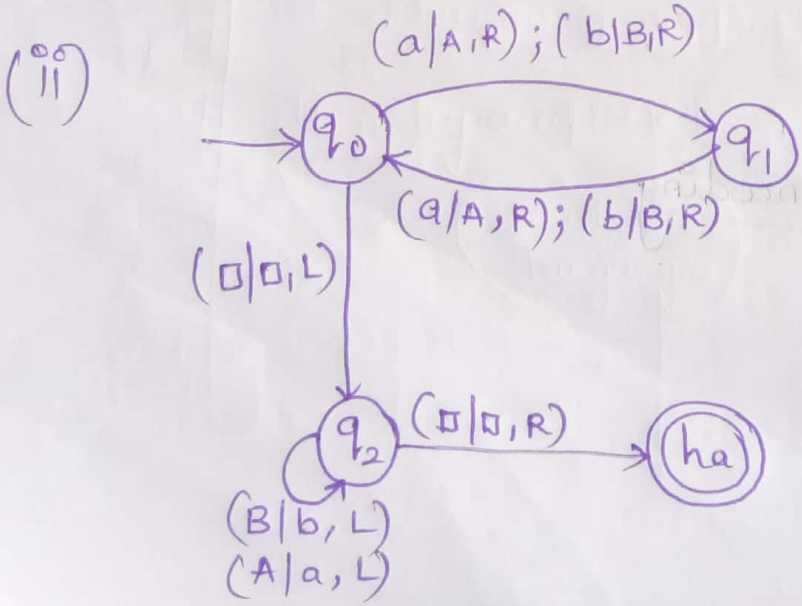
$e(M) = 111\,0\,11\,0|000|00|0|00\,||\,0|0|000|00|000|00|000|0|0|00\,||\,00|0|000|000|000\,|||$

(2) Encode the following Turing Machine.

(i)



(ii)



Linear Bounded Automata

A linear bounded automaton is a multitrack non-deterministic turing machine with a tape of some bounded finite length.

length = function (length of the initial input string, constant c)

Here,

Memory information $\leq c \times$ input information

The computation is restricted to the constant bounded area. The input alphabet contains two special symbols which serve as left end markers and right end markers which mean the transitions neither move to the left of the left end marker nor to the right end marker of the tape

A linear bounded automaton can be defined as an 8-tuple $(Q, X, \Sigma, q_0, M_L, M_R, \delta, F)$

where

Q is a finite set of states

X is the tape alphabet

Σ is the input alphabet

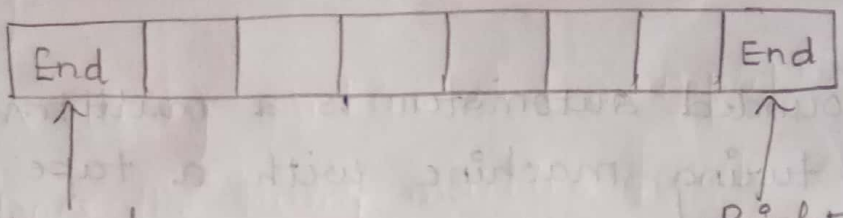
q_0 is the initial state

M_L is the left end marker

M_R is the right end marker where $M_R \neq M_L$.

δ is a transition function which maps each pair (state, tape symbol) to (state, tape symbol, constant 'c') where c can be 0 or +1 or -1.

F is the set of final states.



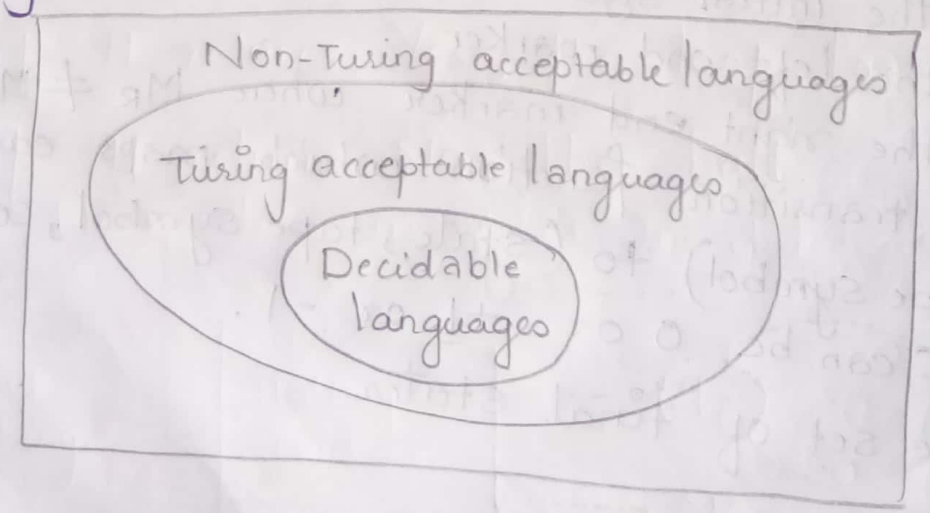
A deterministic linear bounded Automaton is always context-sensitive and the linear bounded automaton with empty language is undecidable.

Computability and Decidability

- A function f on a certain domain is said to be computable if there exists a turing machine that computes the value of f for all argument in its domain.
- A function is uncomputable if no such turing machine exists.

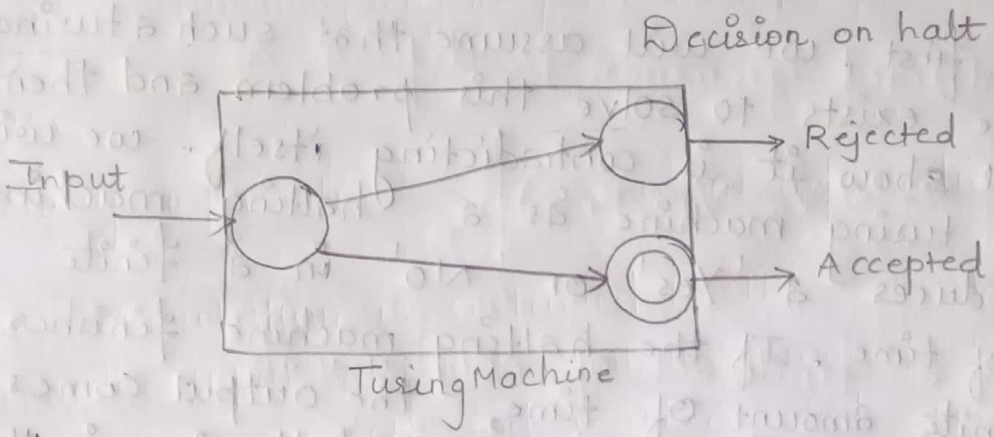
Language Decidability

A language is called Decidable or recursive if there is a turing machine which accepts and halt on every input string w. Every decidable language is turing Acceptable.



A decision problem P is decidable if the language L of all yes instances to P is decidable.

For a decidable language, for each input string, the TM halts either at the accept or the reject state as depicted in the following diagram.

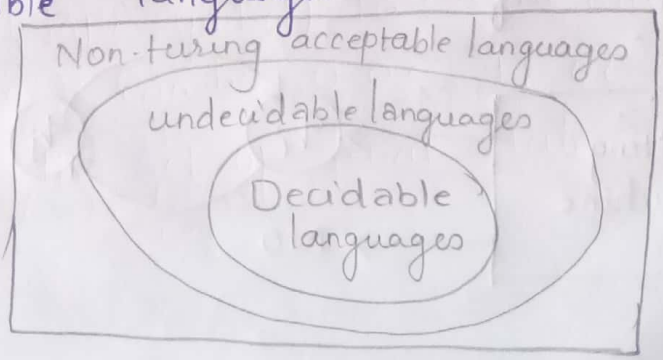


Note

* If a language L is decidable, then its complement L' is also decidable.
* If a language is decidable, then there is an enumerator for it.

Undecidable languages

For an undecidable language, there is no turing machine which accepts the language and makes a decision for every input string w. A decision problem P is called "undecidable" if the language L of all yes instances to P is not decidable. Undecidable languages are not recursive languages, but sometimes, they may be recursively enumerable languages.

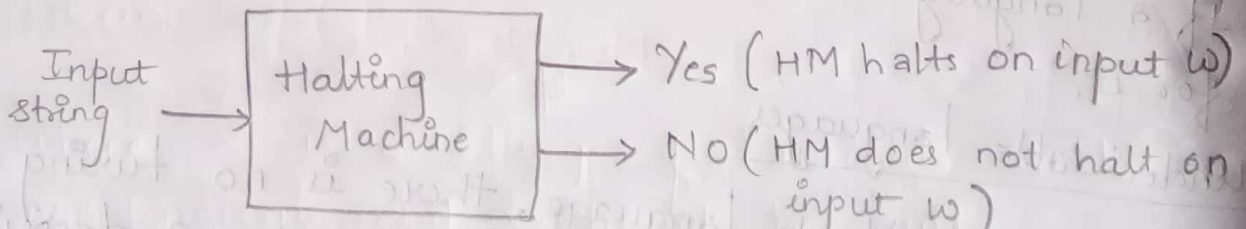


Turing Machine Halting Problem

Input :- A turing machine and an input string w.

Problem – Does the turing machine finish computing of the string w in a finite number of steps? The answer must be either Yes or No.

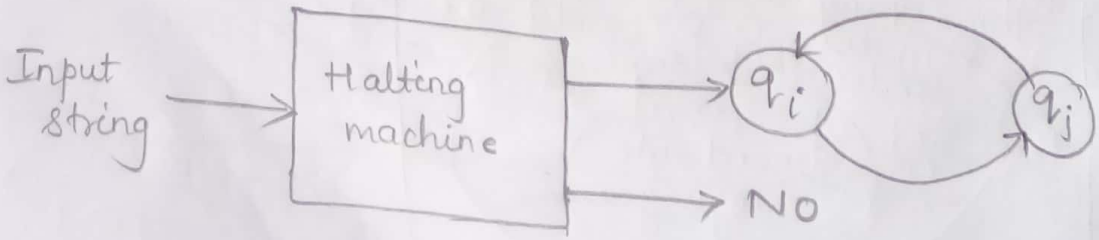
Proof:- At first, we will assume that such a turing machine exists to solve this problem and then we will show it is contradicting itself. we will call this turing machine as a halting machine that produces a 'Yes' or 'No' in a finite amount of time. If the halting machine finishes in a finite amount of time, the output comes as 'Yes', otherwise as 'No'. The following is the block diagram of a halting machine.



Now, we will design an inverted halting machine (HM)' as

- If H returns Yes, then loop forever.
- If H returns No, then halt.

The block diagram of an 'Inverted halting machine'-



Further, a Machine $(HM)_2$ which input itself is constructed as follows -

- If $(HM)_2$ halts on input, loop forever.
- Else, halt

Here, we have got a contradiction. Hence, the halting Problem is undecidable.

Rice Theorem

$L = \{ \langle M \rangle \mid L(M) \in P \}$ is undecidable when P, a non-trivial property of the turing machine, is undecidable.

If the following two properties hold, it is proved as undecidable.

Property 1 If M_1 and M_2 recognize the same language, then either $\langle M1 \rangle \langle M2 \rangle \in L$ or $\langle M1 \rangle \langle M2 \rangle \notin L$.

Let there are two turing machines x_1 and x_2.

Let us assume $\langle x_1 \rangle \in L$ such that

$L(x_1) = \varphi$ and $\langle x_2 \rangle \notin L$.

for an input 'w' in a particular instant, perform the following steps -

- If X accepts w, then simulate x_2 on X
- Run Z on input $\langle w \rangle$
- If Z accepts $\langle w \rangle$, Reject it, and if Z rejects $\langle w \rangle$, accept it.

If X accepts w, then

$L(w) = L(x_2)$ and $\langle w \rangle \notin P$.

If M does not accept w, then

$L(w) = L(X_1) = \phi$ and $\langle w \rangle \in P$

Hence the contradiction arises, hence it is undecidable.

Unrestricted (Type 0 grammar)

The unrestricted grammar is defined as

$$G = (V_n, V_t, P, S)$$

where

V_n is a finite set of non-terminals

V_t is a finite set of terminals

S is starting non terminal $S \in V_n$

and P is set of productions of the following form

$$\alpha \longrightarrow \beta$$

where α is a string of terminals and non terminals and α cannot be null.

β is a string of terminals and non-terminals.

1) construct a Type 0 grammar for the following language

$L = \{a^i \mid i \text{ is a positive power of } 2\}$

Soln :-

$G = (V_n, V_t, P, S)$

$L = \{a^2, a^4, a^6, a^8, \ldots\}$

P is defined as.

$S \longrightarrow ACaB$

$Ca \longrightarrow aaC$ C is a migrating variable. Doubling the number of as again

$CB \longrightarrow DB$

$CB \longrightarrow \varepsilon$

$aD \longrightarrow Da$ D is migrating variable

$AD \longrightarrow AC$

$aE \longrightarrow Ea$

$AE \longrightarrow \varepsilon$

To generate string aa

$\longrightarrow S$

$\longrightarrow ACaB$

$\longrightarrow AaaCB$

$\longrightarrow AaaE$

$\longrightarrow AaEa$

$\longrightarrow AEaa$

$\longrightarrow aa$

$S \longrightarrow ACaB$

$Ca \longrightarrow aaC$

$CB \longrightarrow E$

$aE \longrightarrow Ea$

$aE \longrightarrow Ea$

$AE \longrightarrow \varepsilon$

2) Construct a Type 0 grammar for the language

$L = \{ ww \mid w \in \{a,b\}^* \}$

Soln

$S \longrightarrow FM$

To generate a new symbol

$F \longrightarrow FaA \mid FbB$

for producing the symbol

$Aa \longrightarrow aA$ $\quad Ba \longrightarrow aB$

$Ab \longrightarrow bA$ $\quad Bb \longrightarrow bB$

when the migrating symbol Hits M, creates a copy.

$AM \longrightarrow Ma$

$BM \longrightarrow Mb$

To complete the derivation

$F \longrightarrow \varepsilon$ and $M \longrightarrow \varepsilon$

Thus the final Type 0 grammar

$G = (V_n, V_t, P, S)$

$V_n = \{ S, A, B, F, M \}$

$V_t = \{ a, b \}$

$S = \{ S \}$

P: $S \longrightarrow FM$

$F \longrightarrow FaA \mid FbB$

$Aa \longrightarrow aA$ $\quad Ba \longrightarrow aB$

$Ab \longrightarrow bA$ $\quad Bb \longrightarrow bB$

$AM \longrightarrow Ma$ $\quad BM \longrightarrow Mb$

$F \longrightarrow \varepsilon$

$M \longrightarrow \varepsilon$

Derivation for the string abaaba.

$S \rightarrow FM$
$\rightarrow FaAM$
$\rightarrow FaMa$
$\rightarrow FbBaMa$
$\rightarrow FbaBMa$
$\rightarrow FbaMba$
$\rightarrow FaAbaMba$
$\rightarrow FabAaMba$
$\rightarrow FabaAMba$
$\rightarrow FabaMaba$
$\rightarrow abaMaba$
$\rightarrow abaaba$

$S \rightarrow FM$
$F \rightarrow FaA$
$AM \rightarrow Ma$
$F \rightarrow FbB$
$Ba \rightarrow Ba$
$BM \rightarrow Mb$
$Ab \rightarrow bA$
$Aa \rightarrow aA$
$AM \rightarrow Ma$
$F \rightarrow \varepsilon$
$M \rightarrow \varepsilon$

3) Construct a grammar (Type 0) for the language
$L = \{a^n b^n c^n \mid n \geq 0\}$.

Soln

(i) First we construct grammar for $a^n x^n$

(ii) Then convert x^n in to $b^n c^n$

and P is defined as follows

$S \rightarrow aSBC \mid aBC$
$CB \rightarrow BC$
$aB \rightarrow ab$
$bB \rightarrow bb$
$bC \rightarrow bC$
$cC \rightarrow cc$

string aabbcc

$S \rightarrow aSBC$
$aaBCBC$
$aaBBCC$
$aabBCC$
$aabbCC$
$aabbcc$

Context Sensitive Grammar (Type 1)

Let $G = (V_n, V_t, P, S)$ be a context sensitive grammar

V_n is a finite set of non-terminals

V_t is a finite set of terminals

S is a starting non-terminals $S \in V$

and P is set of production of the following form

$$\alpha A \beta \longrightarrow \alpha \gamma \beta$$

where $A \in N$ and $\alpha, \beta, \gamma \in (T \cup N)^*$

α, β may be empty, but γ must not be non-empty.

1) Construct a context sensitive grammar for the following language $L = \{a^i b^j c^i d^j \mid i, j \geq 1\}$

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

$L = \{abcd, aabccd, \ldots\}$

$S \rightarrow AB$

$A \rightarrow aAX \mid aX$; generate a's and X's for c's.

$B \rightarrow bBd \mid bYd$; Generate equal no. of b's and d's

$Xb \rightarrow bX$; for ordering b's before c

$XY \rightarrow YC$; generate c's with X hits Y

$Y \rightarrow \varepsilon$; Remove the marker Y

String aabbbccddd

→ S	S → AB
→ AB	A → aAX
→ aAXB	A → aX
→ aaXXB	B → bBd
→ aaXXbBd	B → bBd
→ aaXXbbBdd	B → bYd
→ aaXXbbbYddd	Xb → bX
→ aaXbXbbYddd	Xb → bX
→ aaXbbbXYddd	XY → Yc
→ aaXbbbYcddd	Xb → bX
→ aabXbbYcddd	Xb → bX
→ aabbXbYcddd	Xb → bX
→ aabbbXYcddd	XY → Yc
→ aabbbYccddd	Y → ε
→ aabbbccddd	

2) Construct a Type 1 grammar for the language over 0 and 1 having equal no. of 0's and 1's.

$L = \{ w \mid w \in (0,1)^* \text{ and } N_0(w) = N_1(w) \}$

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

$L = \{0011, 01, \varepsilon, 1100, 10, \ldots\}$

$S \longrightarrow 0BS \mid 0B$

$0B \longrightarrow B0$

$B0 \longrightarrow 0B$

$B \longrightarrow 1$

$S \to 0BS$

$\to 0B0B$ $S \to 0B$

$\to 00BB$ $B0 \to 0B$

$\to 0B1B$ $B \to 1$

$\to 0011$ $B \to 1$