Theory of Computation (CT-502)

Course Instructor
ANUJ GHIMIRE

DISCLAIMER

- *This document does not claim any originality and cannot be used as a substitute for prescribed textbooks.*
- *The information presented here is merely a collection from various sources as well as freely available material from internet and textbooks.*
- *The ownership of the information lies with the respective authors or institutions.*

Chapter 4

Turing Machine

Turing Machine

- We have observed neither finite automata nor pushdown automata can be regarded as truly general models for computers, as they are not capable of recognizing simple languages as $L= \{a^n b^n c^n \mid n > 0\}$.
- Now we take up the study of devices called ***Turing Machine*** that can recognize this and many more complicated languages.
- It is an abstract machine developed by English Mathematician ***Alan Turing*** in 1936 and is the theoretical foundation of modern computer.

Turing Machine

- A Turing machine consists of a ***finite control***, a ***tape*** and a ***tape head*** that can be used for reading or writing on that tape.
- The ***tape*** consists of a infinite long series of "squares" each of which can hold a single symbol. If no symbol then it contains ***blank***.
- The 'tape head' or 'read/write head' can read a symbol from the tape, write a symbol to the tape and move one square in either direction.

Turing Machine

- Tape serves as
  - Input device
  - Memory available for use during computation
  - Output device
- TM is much more accurate model of general purpose computer and can do everything that a real computer can do.

Turing Machine

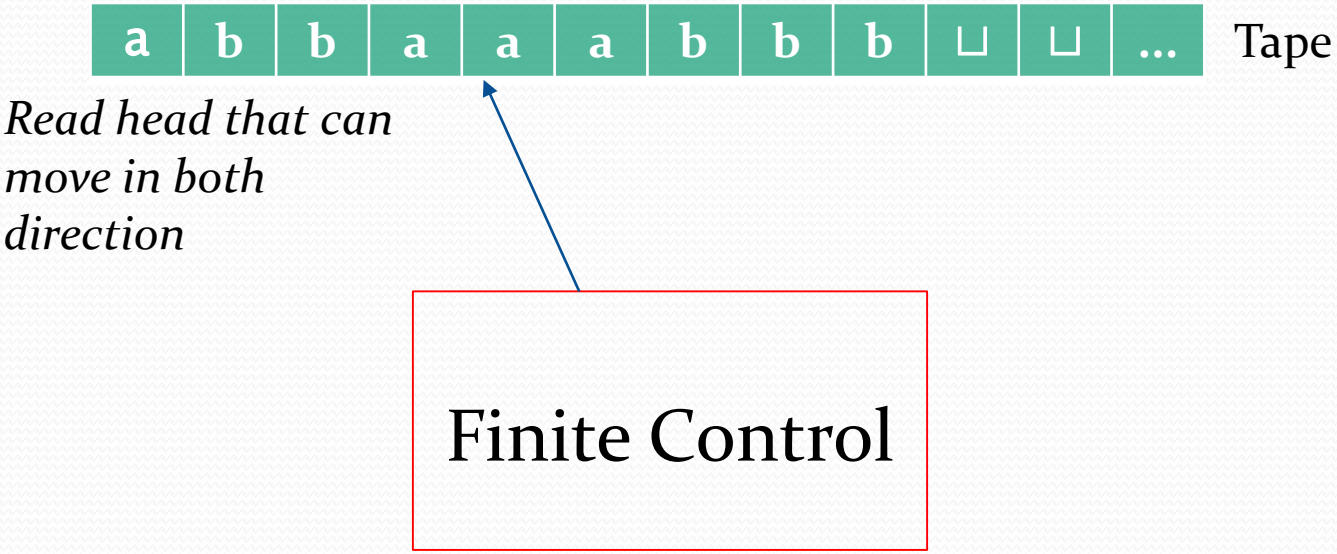


Fig: Block Diagram of Turing Machine

Turing Machine

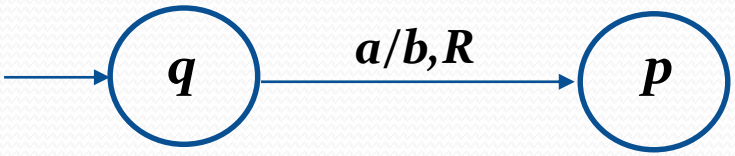
- As we see tape is sequence of infinite symbols over which there is an arrow known as ***tape head***, which is positioned over the symbol on which current control is present.
- Tape head can move either one step left or one step right at a single time of computation.
- Tape, can contain any kind of alphabets like 0,1, a, b, x etc. and it also contains a special symbol i.e. the ***blank symbol (␣).***

Turing Machine

- Blank is a special symbol used to fill the infinite tape.
- ***Read***, ***Write***, ***Move the tape head one step LEFT***, ***Move the tape head one step RIGHT*** are the operations that can be performed on tape.

Turing Machine

Rule of Operation:



Rule 1:

At each step of the computation:

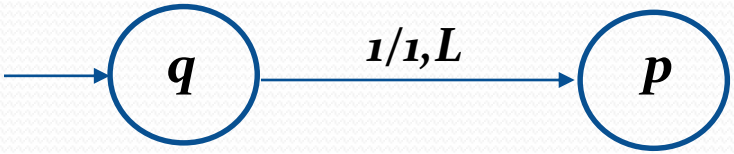
→Read the current symbol

→Write the same cell.

→ Move exactly one cell either LEFT or RIGHT.

Note:

- *If we are at the left end of the tape and trying to move LEFT, then do not move, just stay at left end.*
- *If we don't want to write/update the cell, then just write the same symbol.*



Turing Machine

Rule 2:

→*Initial State*

→ *Final states: (there are two final states)*

The ACCEPT state.

The REJECT state.

→*Computation can either*

HALT and ACCEPT

HALT and REJECT

LOOP (the machine fails to HALT).

Turing Machine

- Formally Turing Machine is defined by 7 tuple:

$$T=(Q, \Sigma, \Gamma, \delta, q_o, B, F)$$

Where,

Q=*Finite set of states*

Σ=*Finite set of input symbols*

Γ=*Finite set of tape symbols*

q_o=*Start state of TM*

B=*Blank symbol*

F=*Set of final states (ACCEPT and REJECT state)*

Turing Machine

Here, δ is the transition function of the form

$$Q \times \Sigma \rightarrow Q \times \Gamma \times (R/L)$$

i.e. when we are on a particular state, on getting a particular symbol, we write something on the tape sequence and we move right or left on the tape and then we go to the next state.

So, the transition function of TM will be written as:

$$\delta(q_0, a) \rightarrow (q_1, y, R)$$

*Initially we are on q_0 state, on getting particular input a, we go to another state q_1 writing the symbol y on tape sequence and move **right** on tape head*

Instantaneous Description of TM

- A configuration of TM is described by Instantaneous description (ID) as in PDA.
- A string $x_1 x_2 x_3 x_4 x_{i-1} q\ x_i x_{i+1}.....x_n$ represents the ID of TM in which:
  - q is the state of TM.
  - the tape head scanning the i^{th} symbol from the left.
  - $x_1\ x_2\ x_3.....x_n$ is the portion of tape between the leftmost and rightmost non-blank.

Instantaneous Description of TM

- Let, $T=(Q, \Sigma, \Gamma, \delta, q_o, B, F)$ be a TM.
  - Define a relation $\vdash$ to define the move of TM
  - Here $\vdash$ sign is called a "turnstile notation" and represent one move and $\vdash^*$ sign represents a sequence of moves of the TM.
  - For $(q, x_i) \rightarrow (p, y, L)$ is the transition i.e. next move is leftward then,

$$x_1 x_2 x_3 x_4 x_{i-1} q x_i x_{i+1} \ldots\ldots x_n \vdash x_1 x_2 x_3 x_4 x_{i-2} p x_{i-1} y x_{i+1} \ldots\ldots x_n$$

  - This move reflects the change of state from q to p and the replacement of symbol x_i, with y and then head is positioned at $i-1$ (i.e. next scan is x_{i-1}).

Instantaneous Description of TM

- For $(q, x_i) \rightarrow (p, y, R)$ is the transition i.e. next move is rightward then,

$$x_1 x_2 x_3 x_4 x_{i-1} q x_i x_{i+1} \ldots\ldots x_n \vdash x_1 x_2 x_3 x_4 x_{i-1} y p x_{i+1} \ldots\ldots x_n$$

- This move reflects the change of state from q to p and the replacement of symbol x_i, with y and then head is positioned at $i+1$ (i.e. next scan is x_{i+1}).

Design of TM_Example(1)

Design a Turing Machine (TM) which recognizes the language L=01*0.

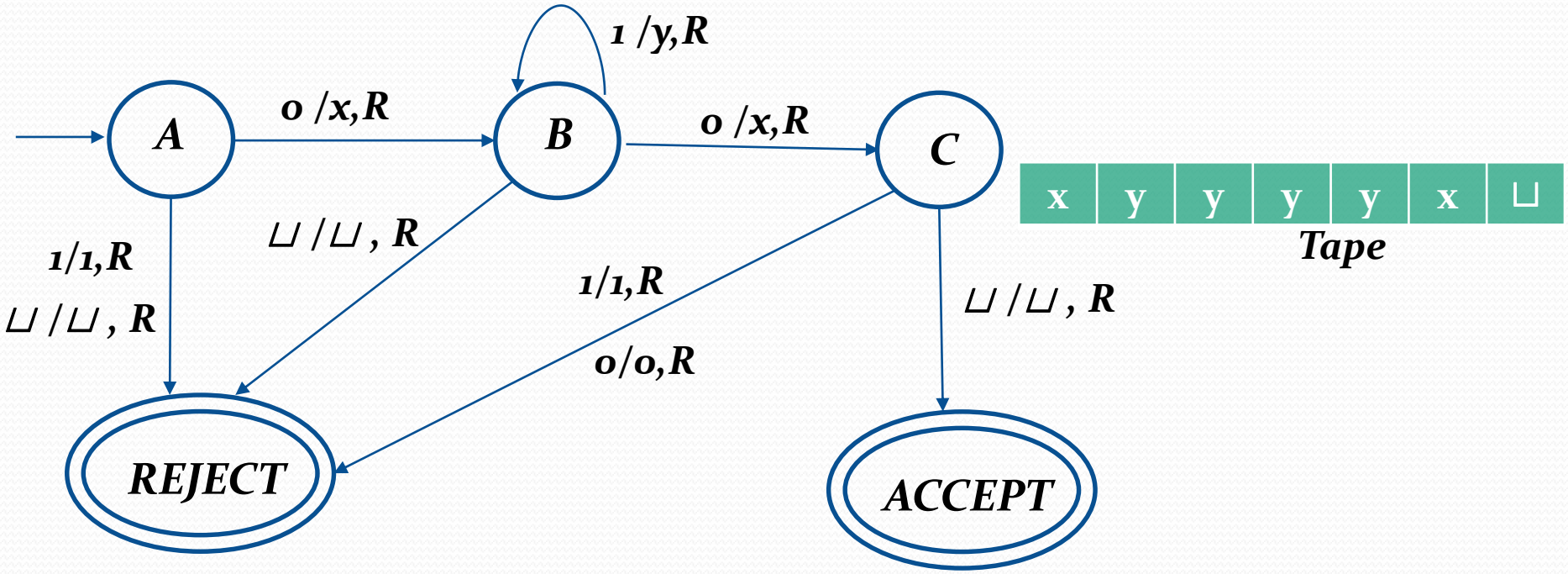
The set of string generated by the given language are

{00, 010, 0110, 01110, 011110,.....}

- *TM should transit from start state to next state on getting* ***0*** *by moving tape arrow* ***right*** *and writing* ***x*** *on tape.*
- *On getting any number of* ***1's*** *TM should stay in that state writing* ***y*** *on tape each time and moving tape* ***right*** *each time.*

Design of TM_Example(1)

- *Finally the TM goes to **ACCEPT** state by reading **blank(␣)** symbol and writing the same **blank (␣)** symbol and moving to **right**.*



Design of TM_Example(1)

- *As we know that TM is deterministic i.e. each state should be defined for each input symbol (0, 1) and blank symbol (␣ as well).*
- *Sometimes we will see REJECT state and transitions to it may be missing, then do not think it as wrong.*
- *It is right because, in any TM if transition is missing then it goes to REJECT state by default.*

Design of TM_Example(1)

- *Finally*

$T=(Q, \sum, \Gamma, \delta, q_o, B, F)$

Where,

Q=*Finite set of states* = **{A,B,C,ACCEPT, REJECT}**

$\sum$=*Finite set of input symbols* =**{0,1}**

Γ=*Finite set of tape symbols* **{0,1,x,y,B}**

q_o=*Start state of TM*=**{A}**

B=Blank symbol

F=Set of final states =**{ACCEPT}**

Design of TM_Example(1)

The transition function ***(δ)*** *are as follows:*

1. $\delta(A, 0) \rightarrow (B, x, R)$
2. $\delta(A, 1) \rightarrow (REJECT, 1, R)$
3. $\delta(A, \sqcup) \rightarrow (REJECT, \sqcup, R)$
4. $\delta(B, 0) \rightarrow (C, \sqcup, R)$
5. $\delta(B, 1) \rightarrow (B, y, R)$
6. $\delta(B, \sqcup) \rightarrow (REJECT, \sqcup, R)$
7. $\delta(C, 0) \rightarrow (REJECT, 0, R)$
8. $\delta(C, 1) \rightarrow (REJECT, 1, R)$
9. $\delta(C, \sqcup) \rightarrow (ACCEPT, \sqcup, R)$

Design of TM_Example(1)

The transition table is:

	0	1	B (⊔)
A	(B, x, R)	(REJECT, 1, R)	(REJECT , ⊔, R)
B	(C , ⊔, R)	(B , y, R)	(REJECT , ⊔, R)
C	(REJECT , 0, R)	(REJECT , 1, R)	(ACCEPT , ⊔, R)
ACCEPT			
REJECT			

Design of TM_Example(2)

Design a Turing Machine (TM) which recognizes the language $L=a^nb^n$ for n>=o.

The set of string generated by the given language are

{ ε, ab, aabb, aaabbb, aaaabbbb,.....}

*Here , the idea is that the number of **a's** and **b's** in the string must be equal and the occurrence of **b** will begin when the occurrence of **a** stops.*

*Furthermore, when the occurrence of **b** starts the occurrence of **a** is not possible.*

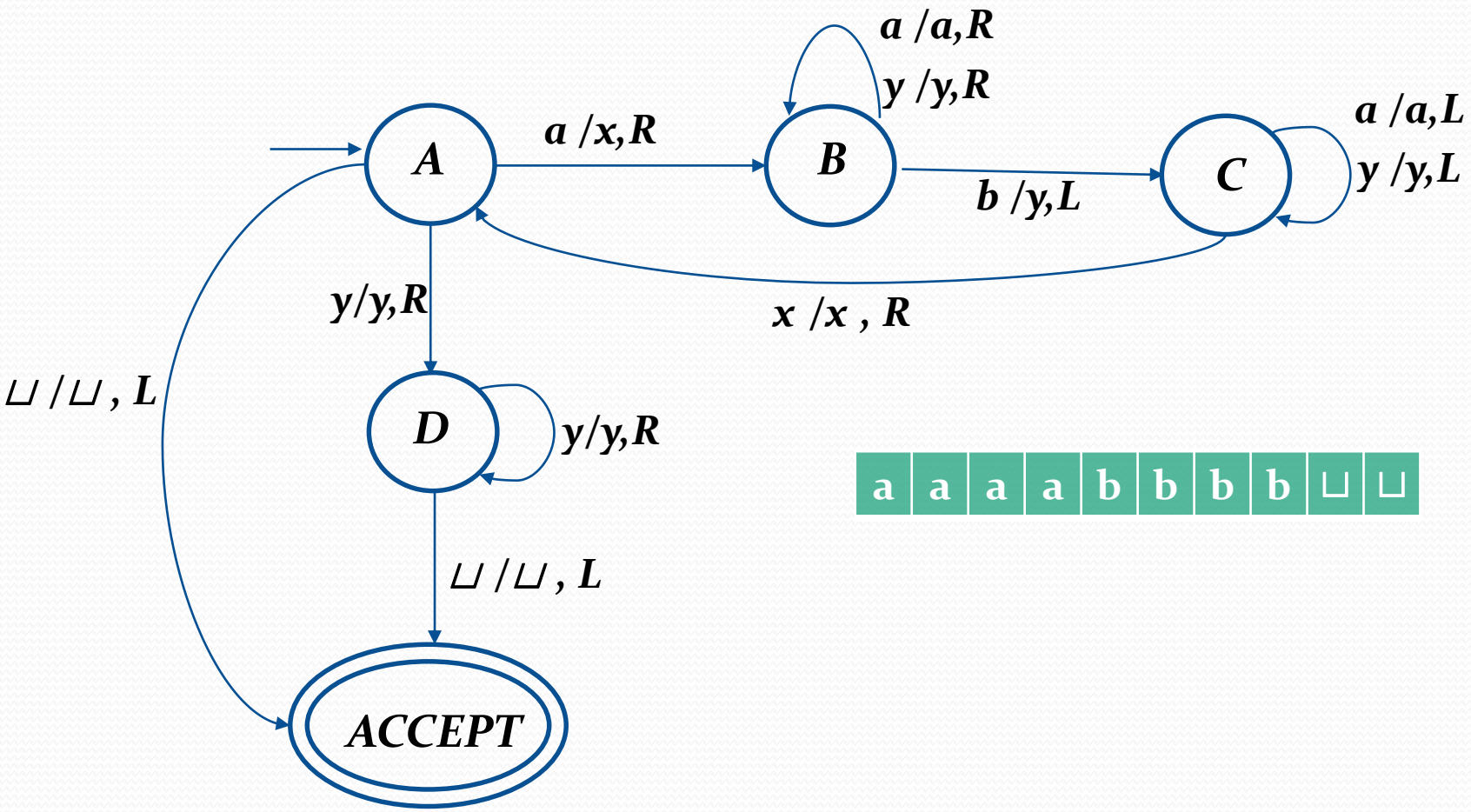
Design of TM_Example(2)

This problem can be solved using the TM using the following algorithm:

Algorithm

- ***Change "a" to "x"***
- ***Move RIGHT to First "b"***
  - ***If None: REJECT***
- ***Change "b" to "y"***
- ***Move LEFT to Leftmost "a"***
- ***Repeat the above steps until no more "a"s***
- ***Make sure no more "b" s remain***

Design of TM_Example(2)



Transition Diagram of TM for $L=a^n b^n$

Design of TM_Example(2)

- *Finally this Turing Machine can be represented as:*

$T=(Q, \sum, \Gamma, \delta, q_o, B, F)$

Where,

Q=*Finite set of states* = $\{A,B,C,D, ACCEPT\}$

$\sum$=*Finite set of input symbols* =$\{a,b\}$

Γ=*Finite set of tape symbols* $\{a,b,x,y,B\}$

q_o=*Start state of TM*=$\{A\}$

B=*Blank symbol*

F=*Set of final states* =$\{ACCEPT\}$

Design of TM_Example(2)

The transition function ***(δ)*** *are as follows:*

1. $\delta(A, a) \rightarrow (B, x, R)$
2. $\delta(A, y) \rightarrow (D, y, R)$
3. $\delta(A, \sqcup) \rightarrow (ACCEPT, \sqcup, L)$
4. $\delta(B, a) \rightarrow (B, a, R)$
5. $\delta(B, b) \rightarrow (C, y, L)$
6. $\delta(B, y) \rightarrow (B, y, R)$
7. $\delta(C, a) \rightarrow (C, a, L)$
8. $\delta(C, x) \rightarrow (A, x, R)$
9. $\delta(C, y) \rightarrow (C, y, L)$
10. $\delta(D, y) \rightarrow (D, y, R)$
11. $\delta(D, \sqcup) \rightarrow (ACCEPT, \sqcup, R)$

Design of TM_Example(2)

The transition table is:

	a	b	x	y	B (⊔)
A	(B, x, R)			(D , y, R)	(ACCEPT , ⊔, L)
B	(B ,a, R)	(C , y, L)		(B , y, R)	
C	(C ,a, L)		(A , x, R)	(C , y, L)	
D				(D , y, R)	(ACCEPT , ⊔, R)
ACCEPT					

Design of TM_Example(2)

Now check for the string ***w=aaabbb***

$A\ \mathbf{a}aabbb \vdash x\mathbf{B}aabbb$
$\vdash xaa\mathbf{B}bbb$
$\vdash xaa\mathbf{C}ybb$
$\vdash x\mathbf{C}aaybb$
$\vdash x\mathbf{A}aaybb$
$\vdash xx\mathbf{B}aybb$
$\vdash xxay\mathbf{B}bb$
$\vdash xxay\mathbf{C}yb$
$\vdash xx\mathbf{C}ayyb$

$\vdash xx\mathbf{A}ayyb$
$\vdash xxx\mathbf{B}yyb$
$\vdash xxxyy\mathbf{B}b$
$\vdash xxxyy\mathbf{C}y$
$\vdash xxx\mathbf{A}yyy$
$\vdash xxxy\mathbf{D}yy$
$\vdash xxxyyy\mathbf{D}\sqcup$
$\vdash xxxyyy\sqcup$ **Accept** $\sqcup$

Hence Halt and Accept the string

Design of TM_Example(2)

Similarly check for the string ***w=aabbba***

$A\ \mathbf{a}abbba \vdash \mathbf{x}\mathbf{B}abbba$

$\vdash \mathbf{x}a\mathbf{B}bbba$

$\vdash \mathbf{x}a\mathbf{C}ybba$

$\vdash \mathbf{x}\mathbf{C}aybba$

$\vdash \mathbf{x}\mathbf{A}aybba$

$\vdash \mathbf{x}\mathbf{x}\mathbf{B}ybba$

$\vdash \mathbf{x}\mathbf{x}\mathbf{y}\mathbf{B}bba$

$\vdash \mathbf{x}\mathbf{x}\mathbf{y}\mathbf{C}yba$

$\vdash \mathbf{x}\mathbf{x}\mathbf{C}\mathbf{y}\mathbf{y}ba$

$\vdash \mathbf{x}\mathbf{x}\mathbf{C}\mathbf{y}\mathbf{y}ba$

$\vdash \mathbf{x}\mathbf{C}\mathbf{x}\mathbf{y}\mathbf{y}ba$

$\vdash \mathbf{x}\mathbf{x}\mathbf{C}\mathbf{y}\mathbf{y}ba$

Here the Turing Machine fails to Halt and enters in the LOOP and cannot accept the string ***w=aabbba***

Design of TM_Example(3)

Design a Turing Machine (TM) which recognizes the even palindrome string over {a,b}*.

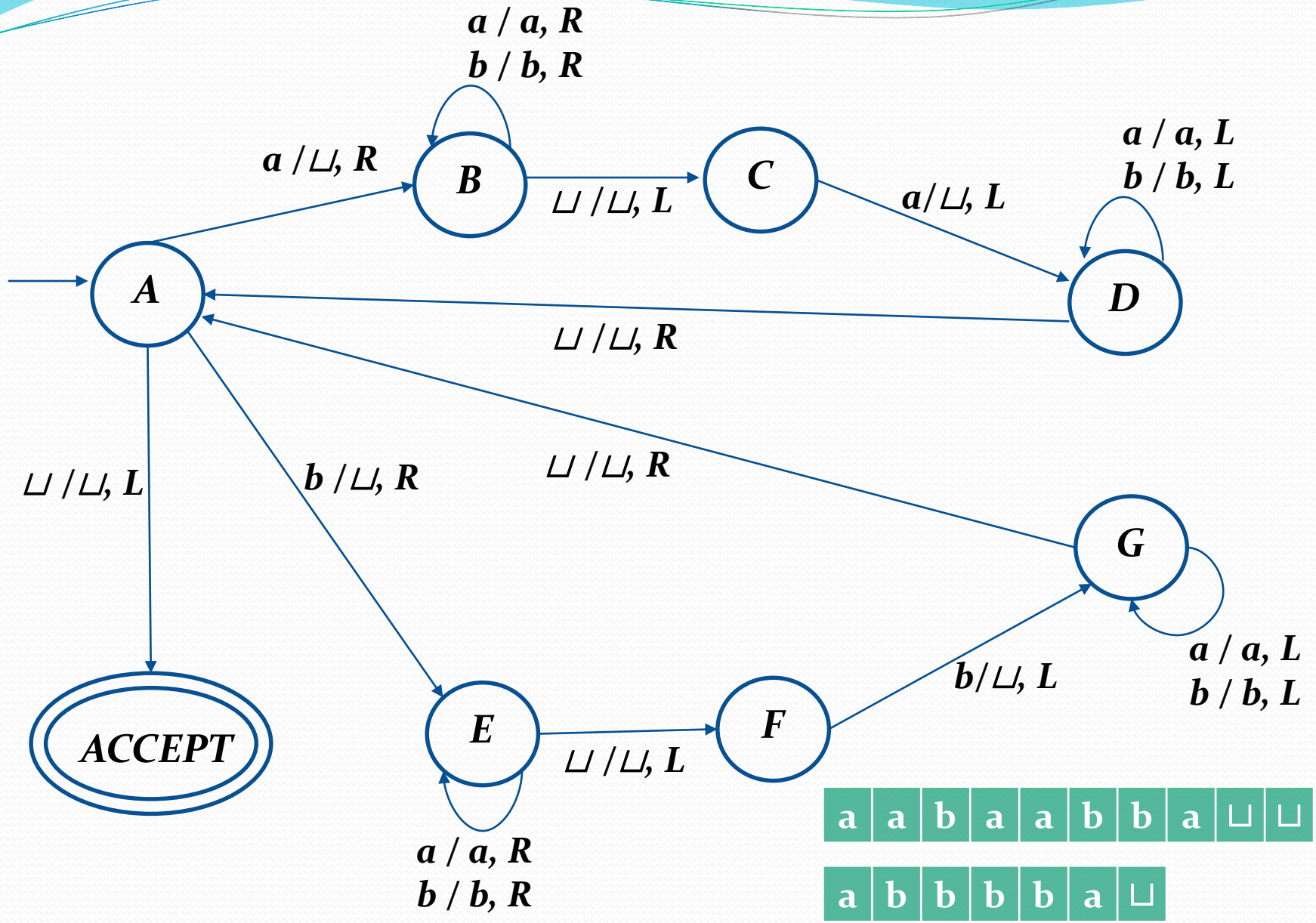
The set of string generated by the given language are

{ ε, abba, aabbaa, abaaba, bbaabb,.....}

- *Initially tape consists of input string **w**, tape head is on leftmost symbol of **w** and TM is in start state A.*
- *The tape head reads leftmost symbol of **w**, replace it by ⊔.*
- *Then tape head moves to the rightmost symbol and test whether it is equal to leftmost symbol*

Design of TM_Example(3)

- *If they are equal, then rightmost symbol is replaced by ⊔ and the tape head moves to new leftmost symbol and whole process is repeated.*
- *If they are not equal, TM enters the reject state and computation terminates.*
- *TM enters accept state as soon as string currently stored on tape is empty.*


a / a, R
b / b, R
a /␣, R
B
␣ /␣, L
C
a/␣, L
a / a, L
b / b, L
A
D
␣ /␣, R
␣ /␣, L
b /␣, R
␣ /␣, R
G
a / a, L
b / b, L
b/␣, L
ACCEPT
E
F
␣ /␣, L
a / a, R
b / b, R
a a b a a b b a ␣ ␣
a b b b b a ␣

Design of TM_Example(3)

- *Finally this Turing Machine can be represented as:*

$T=(Q, \sum, \Gamma, \delta, q_o, B, F)$

Where,

Q=*Finite set of states =* $\{A,B,C,D,E,F,G\ ACCEPT\}$

$\sum$=*Finite set of input symbols =*$\{a,b\}$

Γ=*Finite set of tape symbols* $\{a,b,B\}$

q_o=*Start state of TM=*$\{A\}$

B=*Blank symbol*

F=*Set of final states =*$\{ACCEPT\}$

Design of TM_Example(3)

The transition function ***(δ)*** *are as follows:*

1. $\delta(A, a) \rightarrow (B, \sqcup, R)$
2. $\delta(A, b) \rightarrow (E, \sqcup, R)$
3. $\delta(A, \sqcup) \rightarrow (ACCEPT, \sqcup, L)$
4. $\delta(B, a) \rightarrow (B, a, R)$
5. $\delta(B, b) \rightarrow (B, b, R)$
6. $\delta(B, \sqcup) \rightarrow (C, \sqcup, L)$
7. $\delta(C, a) \rightarrow (D, \sqcup, L)$
8. $\delta(D, a) \rightarrow (D, a, L)$
9. $\delta(D, b) \rightarrow (D, b, L)$
10. $\delta(D, \sqcup) \rightarrow (A, \sqcup, R)$
11. $\delta(E, a) \rightarrow (E, a, R)$
12. $\delta(E, b) \rightarrow (E, b, R)$
13. $\delta(E, \sqcup) \rightarrow (F, \sqcup, L)$
14. $\delta(F, b) \rightarrow (G, \sqcup, L)$
15. $\delta(G, a) \rightarrow (G, a, L)$
16. $\delta(G, b) \rightarrow (G, b, L)$
17. $\delta(G, \sqcup) \rightarrow (A, \sqcup, R)$

Design of TM_Example(3)

The transition table is:

	a	b	B (␣)
A	***(B, ␣, R)***	***(E, ␣, R)***	(ACCEPT , ␣, L)
B	***(B,a, R)***	***(B,b, R)***	***(C, ␣, L)***
C	***(D, ␣, L)***		
D	***(D, a, L)***	***(D, b, L)***	***(A, ␣, R)***
E	***(E,a, R)***	***(E,b, R)***	***(F, ␣, L)***
F		***(F, ␣, L)***	
G	***(G, a, L)***	***(G, b, L)***	***(A, ␣, R)***
ACCEPT			

Now check for the string ***w=baabbaab***

Design of TM_Example(4)

Design a Turing Machine (TM) which recognizes the language $L=a^n b^n c^n \mid n>=1$.

The set of string generated by the given language are

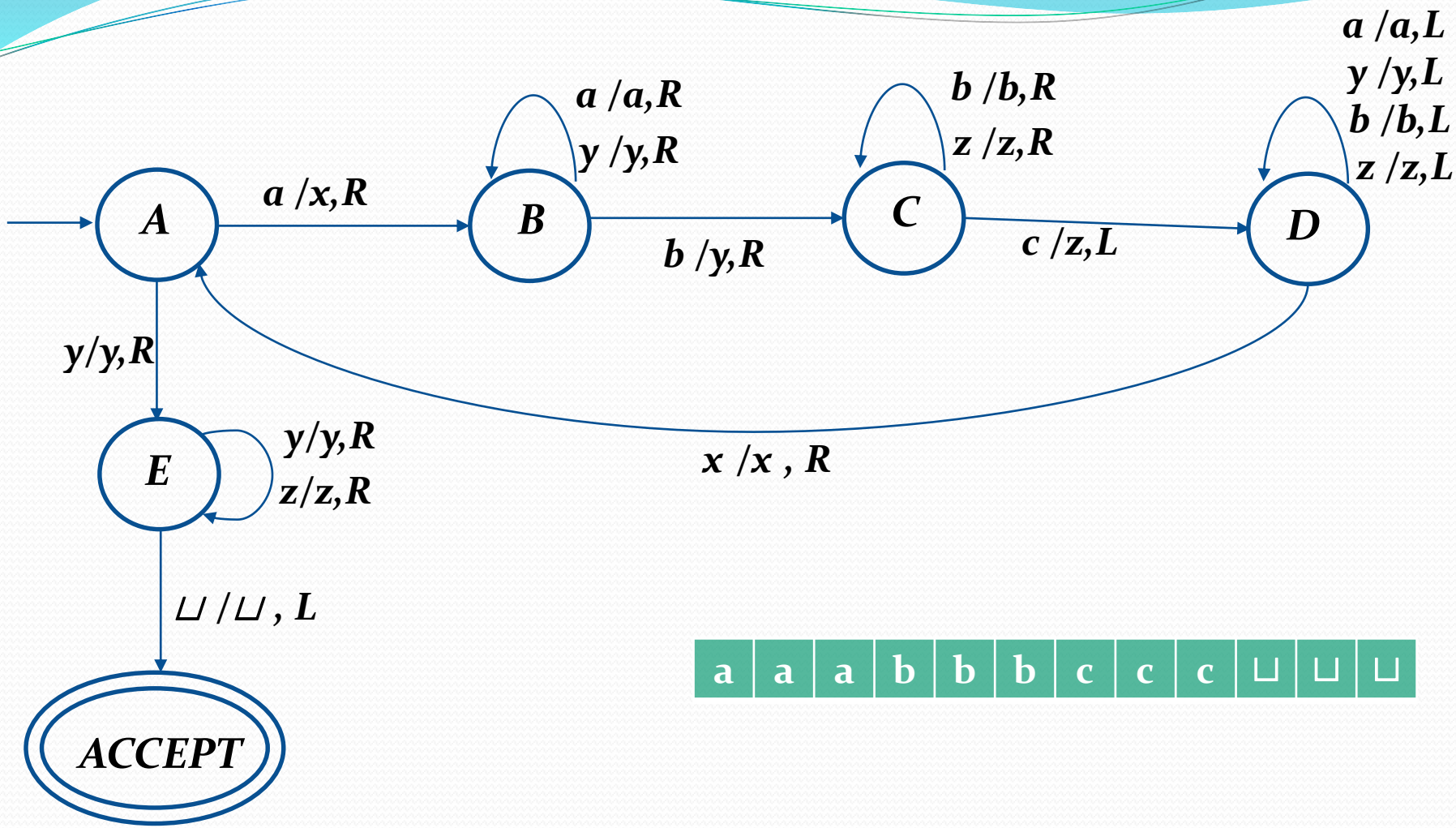
{ abc, aabbcc, aaabbbccc, ……..}

Design of TM_Example(4)

This problem can be solved using the TM using the following algorithm:

Algorithm

- *Change "a" to "x"*
- *Move RIGHT to First "b"*
  - *If None: REJECT*
- *Change "b" to "y"*
- *Move RIGHT to First "c"*
  - *If None: REJECT*
- *Change "c" to "z"*
- *Move LEFT to Leftmost "a"*
- *Repeat the above steps until no more "a"s*
- *Make sure no more "b" s and "c"s remain*



Transition Diagram of TM for $L=a^n b^n c^n$

Design of TM_Example(4)

- *Finally this Turing Machine can be represented as:*

$T=(Q, \sum, \Gamma, \delta, q_o, B, F)$

Where,

Q=*Finite set of states* = $\{A,B,C,D,E, ACCEPT\}$

$\sum$=*Finite set of input symbols* =$\{a,b\}$

Γ=*Finite set of tape symbols* $\{a,b,c,x,y,z,B\}$

q_o=*Start state of TM*=$\{A\}$

B=*Blank symbol*

F=*Set of final states* =$\{ACCEPT\}$

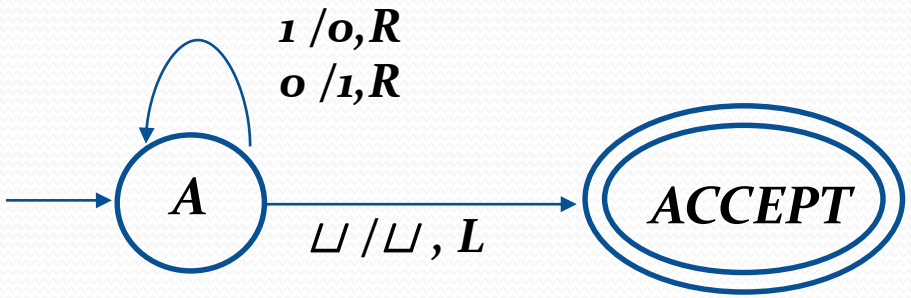
Design of TM_Example(4)

The transition function (δ) are as follows:

1. $\delta(A, a) \rightarrow (B, x, R)$
2. $\delta(A, y) \rightarrow (E, y, R)$
3. $\delta(B, a) \rightarrow (B, a, R)$
4. $\delta(B, b) \rightarrow (C, y, L)$
5. $\delta(B, y) \rightarrow (B, y, R)$
6. $\delta(C, b) \rightarrow (C, b, R)$
7. $\delta(C, z) \rightarrow (C, z, R)$
8. $\delta(C, c) \rightarrow (D, z, L)$
9. $\delta(D, a) \rightarrow (D, a, L)$
10. $\delta(D, b) \rightarrow (D, b, L)$
11. $\delta(D, y) \rightarrow (D, y, L)$
12. $\delta(D, z) \rightarrow (D, z, L)$
13. $\delta(D, x) \rightarrow (A, x, R)$
14. $\delta(E, y) \rightarrow (E, y, R)$
15. $\delta(E, z) \rightarrow (E, z, R)$
16. $\delta(E, \sqcup) \rightarrow (ACCEPT, \sqcup, L)$

Design of TM_Example(5)

Design a Turing Machine (TM) whose output is 1's complement of binary string.



Language of Turing Machine

- If M = (Q , ∑ , Γ , δ , qo, B, F) is a TM, then language of TM, L(M) is the set of strings w in ∑* such that:

  $(q_o, w) \vdash^* (\alpha, p, \beta)$ where p ϵ F and α and β are any tape strings.

- ***Recursively Enumerable Language***
  - The set of Recursively Enumerable languages are precisely those language that can be accepted by TM.
  - Strings that are not in the language may be rejected or may cause TM to go into infinite loop.

Language of Turing Machine

- ***Recursive Language***
  - A language is recursive if there exists a TM that accept every string of the language and rejects strings that are not in language.
  - Recursive language always halts TM.
- ***Turing Recognizable Language***
  - A language L is Turing Recognizable if there exists a TM such that for all strings ***w***:
  - If w $\in$ L , eventually TM enter ***ACCEPT*** state
  - If w $\notin$ L , either TM enters ***REJECT*** or TM never terminates.

Language of Turing Machine

- ***Turing Decidable Language***
  - A language L is Turing Decidable if there exists a TM such that for all strings ***w***:
  - If w $\in$ L , TM enter ***ACCEPT*** state
  - If w $\notin$ L , TM enters ***REJECT*** state
- Hence, decidable language always terminates while recognizable language can run forever without deciding.

Roles of Turing Machine

- The Turing Machine is designed to perform at least the following three roles:
  - *As a language recognizer.*
  - *As a computing of function.*
  - *As an enumerator of string of a language.*

Turing Machine *(As a Computing Function)*

- A TM can be used to compute a function.
- For such TM, we adopt following policy to input string to TM which is input to computation function:
  - String ***w*** is presented into form of ***BwB***, where ***B*** is blank symbol , and placed on the tape.
  - The head is positioned at a blank symbol which immediately follows the string ***w***.
  - The TM is said to halt on input ***w*** if we can reach to halting state after performing some operation.

Turing Machine *(As a Computing Function)*

Formal Definition:

- *A function* $\boldsymbol{f(x) = y}$ *is said to be computable by TM defined as* $(Q, \Sigma, \Gamma, \delta, q_o, B, F)$ *if*

$$(\boldsymbol{q_o}, \boldsymbol{BxB}) \vdash^* (\boldsymbol{q_f}, \boldsymbol{ByB})$$

- It means that if we give input ***x*** to TM, it gives output ***y*** as string if it computes the function f(x) = y.

Turing Machine *(As a Computing Function)*

Design a TM which computes the function f(x) = x+1 for each x belonging to set of natural numbers.

Solution

Given function f(x) = x+1,

We represent x by 1's on tape .

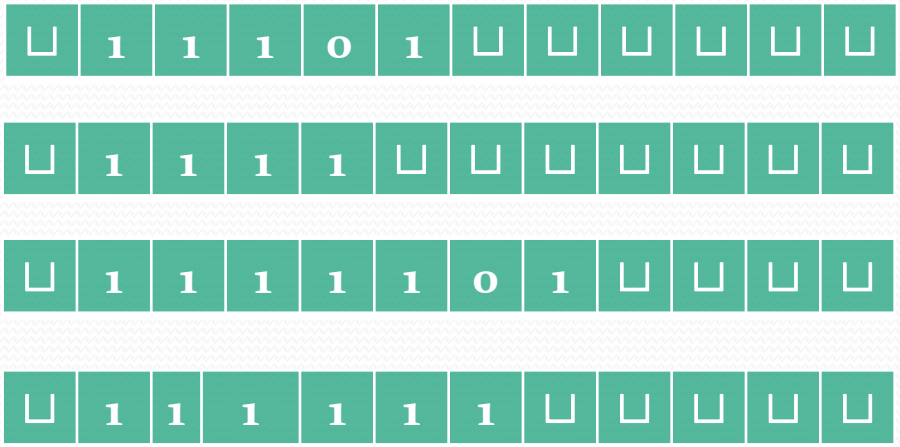
i.e. for x =1, input tape will be ***B1B***

for x = 2, input tape will be ***B11B*** *and so on....*

Similarly , output can be seen by number of 1's on tape when machine halts.

Turing Machine *(As a Computing Function)*

Here, let TM =(Q , ∑ , Γ , δ , qo, B, F) which computes the function f(x) = x+1.

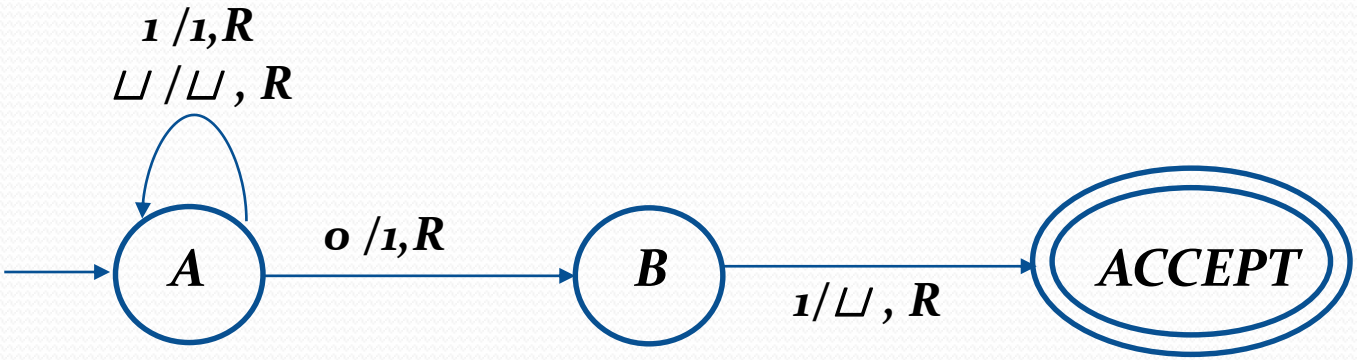


Simply to compute the function f(x) = x+1

we store the value of occurrence of x in tape by 1 and to perform f(x) = x+1 operation we simply increases the number of 1 in tape by one.

Turing Machine *(As a Computing Function)*

Its transition diagram can be drawn as:



Turing Machine *(As a Computing Function)*

- *Finally this Turing Machine can be represented as:*

$T=(Q, \Sigma, \Gamma, \delta, q_o, B, F)$

Where,

Q=*Finite set of states* = ***{A,B, ACCEPT}***

Σ=*Finite set of input symbols* =***{0,1}***

Γ=*Finite set of tape symbols* ***{0,1,B}***

q_o=*Start state of TM*=***{A}***

B=*Blank symbol*

F=*Set of final states* =***{ACCEPT}***

.

Turing Machine *(As a Computing Function)*

The transition function (δ) are as follows:

1. $\delta(A, 1) \rightarrow (A, 1, R)$
2. $\delta(A, \sqcup) \rightarrow (A, \sqcup, R)$
3. $\delta(A, 0) \rightarrow (B, 1, R)$
4. $\delta(B, 1) \rightarrow (ACCEPT, \sqcup, R)$

Turing Machine *(As a Computing Function)*

Design a TM which computes the function f(x) = x+y

Solution

Given function f(x) = x+y,

We represent x and y by 1's on tape .

i.e. for x =1 and y=1, input tape will be ***B101B***

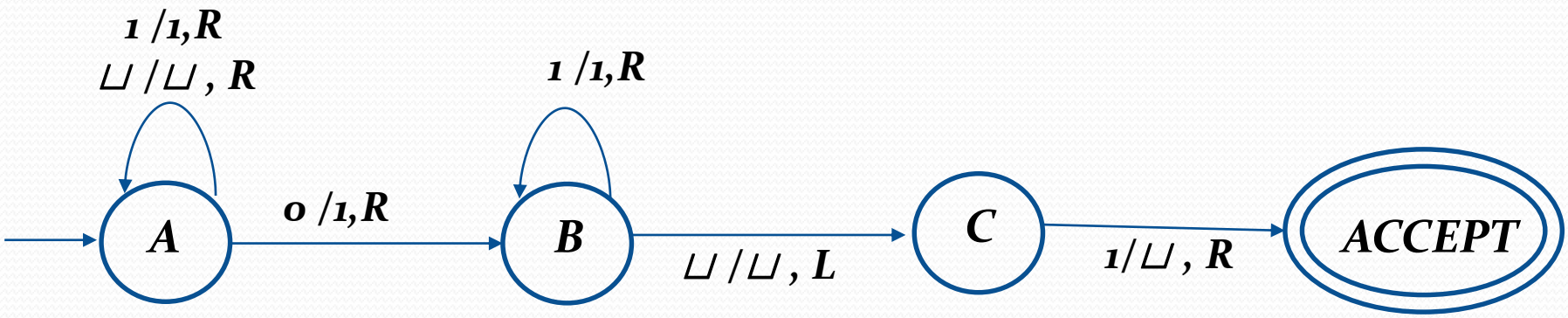
for x = 2, and y=3 input tape will be ***B110111B***

for x=3 and y=1 input tape will be ***B11101B*** *and so on....*

Similarly , output can be seen by number of 1's on tape when machine halts.

Turing Machine *(As a Computing Function)*

Its transition diagram can be drawn as:



Turing Machine *(As a Computing Function)*

The transition function (δ) are as follows:

1. $\delta(A, 1) \rightarrow (A, 1, R)$
2. $\delta(A, \sqcup) \rightarrow (A, \sqcup, R)$
3. $\delta(A, 0) \rightarrow (B, 1, R)$
4. $\delta(B, 1) \rightarrow (B, 1, R)$
5. $\delta(B, \sqcup) \rightarrow (C, \sqcup, L)$
6. $\delta(C, 1) \rightarrow (ACCEPT, \sqcup, R)$

Turing Machine *(As a Computing Function)*

- *Finally this Turing Machine can be represented as:*

$T=(Q, \Sigma, \Gamma, \delta, q_o, B, F)$

Where,

Q=*Finite set of states* = ***{A,B,C, ACCEPT}***

Σ=*Finite set of input symbols* =***{0,1}***

Γ=*Finite set of tape symbols* ***{0,1,B}***

q_o=*Start state of TM*=***{A}***

B=*Blank symbol*

F=*Set of final states* =***{ACCEPT}***

.

Turing Machine *(As a Computing Function)*

Design a TM which computes the function f(x) = 2x

Solution

Given function f(x) = 2x

We represent the value of x by 1's on tape .

i.e. for x =1 input tape will be ***B1B***

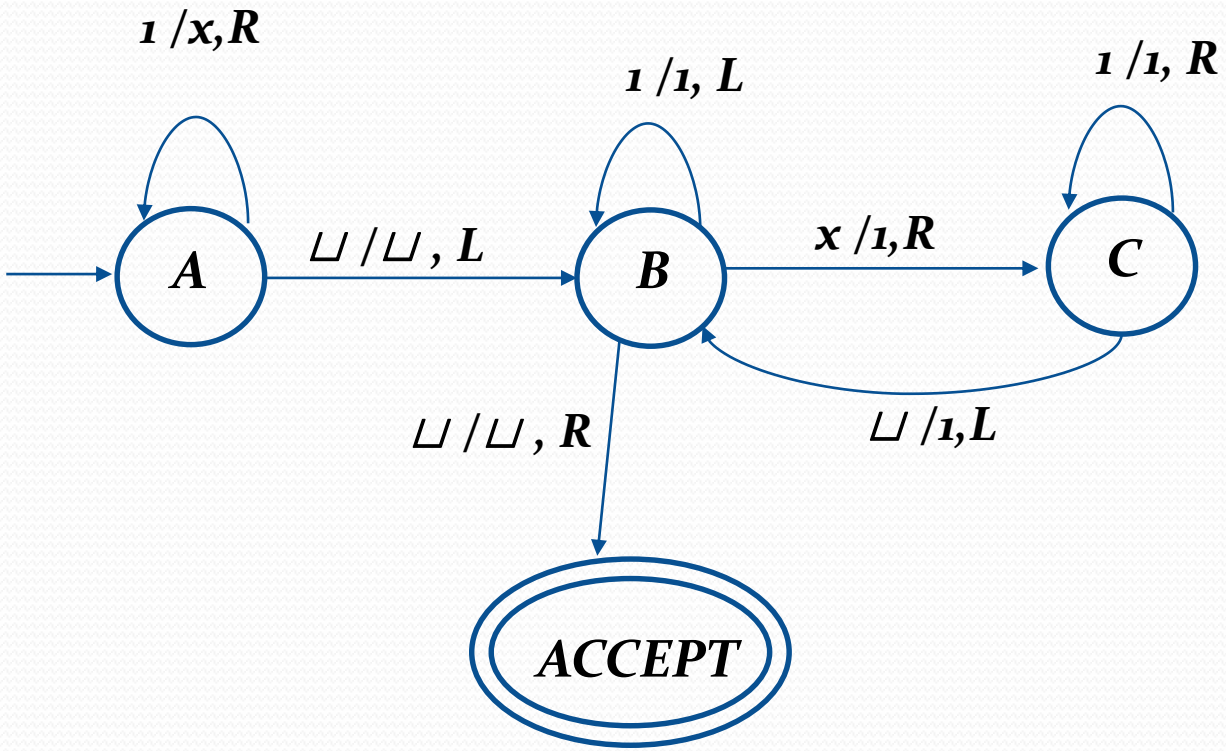
for x =1 output tape must be ***B11B***

for x = 2, input tape will be ***B11B***

for x =2 output tape must be ***B1111B***.

Turing Machine *(As a Computing Function)*

Its transition diagram can be drawn as:



Turing Machine *(As a Computing Function)*

- *Finally this Turing Machine can be represented as:*

$T=(Q, \Sigma, \Gamma, \delta, q_o, B, F)$

Where,

Q=*Finite set of states* = ***{A,B,C, ACCEPT}***

Σ=*Finite set of input symbols* =***{0,1}***

Γ=*Finite set of tape symbols* ***{1,x,B}***

q_o=*Start state of TM*=***{A}***

B=*Blank symbol*

F=*Set of final states* =***{ACCEPT}***

.

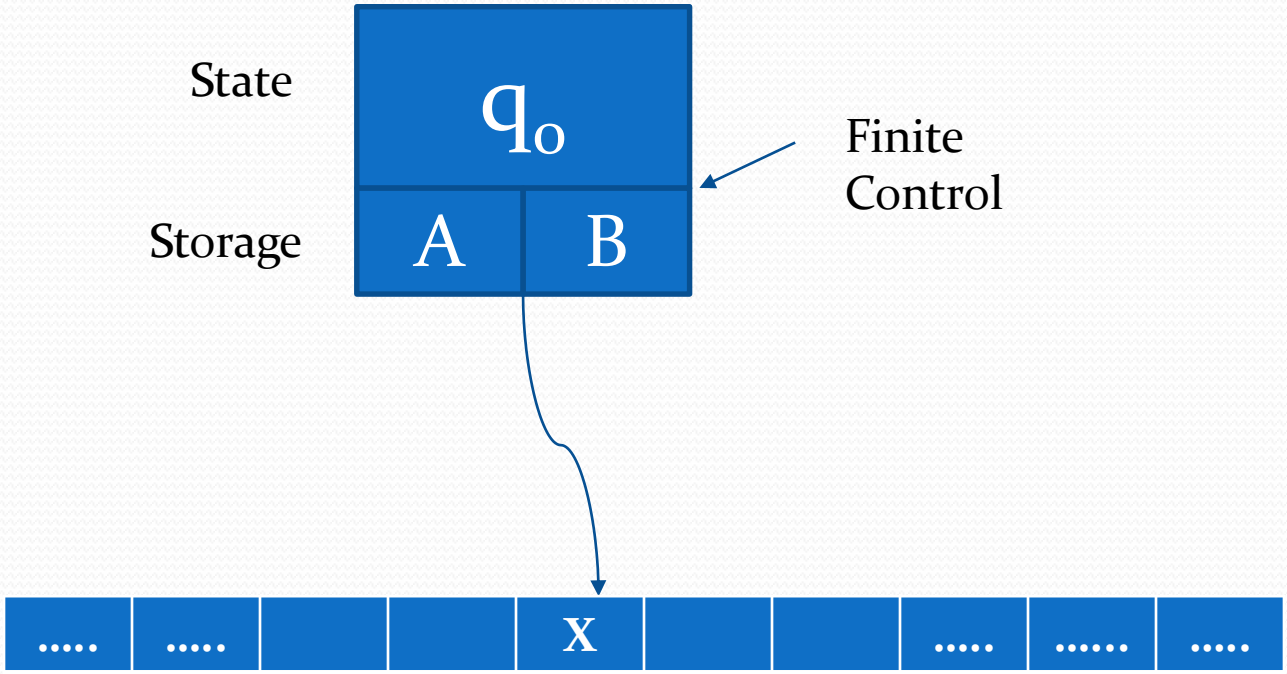
Turing Machine (*With storage in the states*)

- In TM, generally any state represents the position in computation. But, the state can also be used to hold finite amount of data.
- We can use the finite control not only to represent a position in computation but also to hold a finite amount of data.
- Hence , a state is considered as a tuple (state, data)
- In this model of computation, δ is defined by

$$\delta([q, A], X) = ([q_1, X], Y, R)$$

  - It means that ***q*** is the state and data is ***A***.
  - The symbol scanned on the tape is copied to data portion of state and moves right entering state ***q1*** and replacing symbol by ***Y***.

Turing Machine (*With storage in the states*)



Turing Machine viewed as having finite control and storage

Turing Machine (*With storage in the states*)

- ***Example:*** This model of TM can be used to recognize languages like 01*+10*, where first symbol (0 or 1) that it sees, and checks that it does not appear else where in the input.

Solution,

Here state is thought of a pair with two components

A control portion q_0 *or* q_1 *where* q_0 *indicates that TM has not yet read its first symbol.*

While q_1 *indicates that it has read symbol and is checking that it does not appear elsewhere by moving right and hoping to reach blank cell.*

Turing Machine (*With storage in the states*)

A data portion which remember the first symbol seen, which must be ***0*** *or* ***1*** *,the symbol B in this component means no symbol has been read.*

Let T=(Q, ∑, Γ, δ, q_o, B, F) be the Turing Machine
Where,

Q=Finite set of states
∑=Finite set of input symbols
Γ=Finite set of tape symbols
q_o=Start state of TM
B=Blank symbol
F=Set of final states

Turing Machine (*With storage in the states*)

δ consists of:

a) $\delta([q_0, B], a) \rightarrow ([q_1, a], a, R)$ ***: for a = 0 or 1***

Initially, q_0 *is the control state, and the data portion of the state is* ***B****.*

The symbol scanned is copied into the second component of the state; ***M*** *moves right entering control state* q_1*, as it does so….*

b) $\delta([q_1, a], a') \rightarrow ([q_1, a], a', R)$ ***: for a' = 0 if a = 1***

for a' = 1 if a = 0

In state q_1 *Turing Machine skips over each non blank symbol and continues moving right.*

Turing Machine (*With storage in the states*)

c) $\delta([q_1, a], B) \rightarrow ([q_1, B], B, R)$ **: ACCEPT**

If TM reaches first blank it enters accepting state.

d) $\delta([q_1, a], a) \rightarrow$ **REJECT**

If TM reaches or encounter second occurrence of same symbol it halts without accepting.

Turing Machine (*with storage in the states*)

- *Finally this Turing Machine can be represented as:*

$T=(Q, \sum, \Gamma, \delta, q_o, B, F)$

Where,

Q=*Finite set of states* = $\{q_o, q_1,$ ***ACCEPT, REJECT***$\}$

$\sum$=*Finite set of input symbols* =$\{0,1\}$

Γ=*Finite set of tape symbols* $\{0,1,B\}$

q_o=*Start state of TM*=$\{q_o\}$

B=*Blank symbol*

F=*Set of final states* =$\{$***ACCEPT***$\}$

Extension of Turing Machine

- ***Turing Machine with Multiple Tapes***
  - *Multiple TM consists of a finite control and finite no. of tapes.*
  - *Each tape is divided into cells and each cell can hold any symbol of finite tape alphabet.*
  - *The set of tape symbols include a blank and input symbols.*
  - *In multiple TM initially,*
    - *The input **w** is placed on the first tape.*
    - *All other cells of the tapes hold blank.*
    - *TM in initial state q_0*
    - *The head of first tape is at left end of input.*
    - *All other tape heads are at some arbitrary cells.*

Extension of Turing Machine

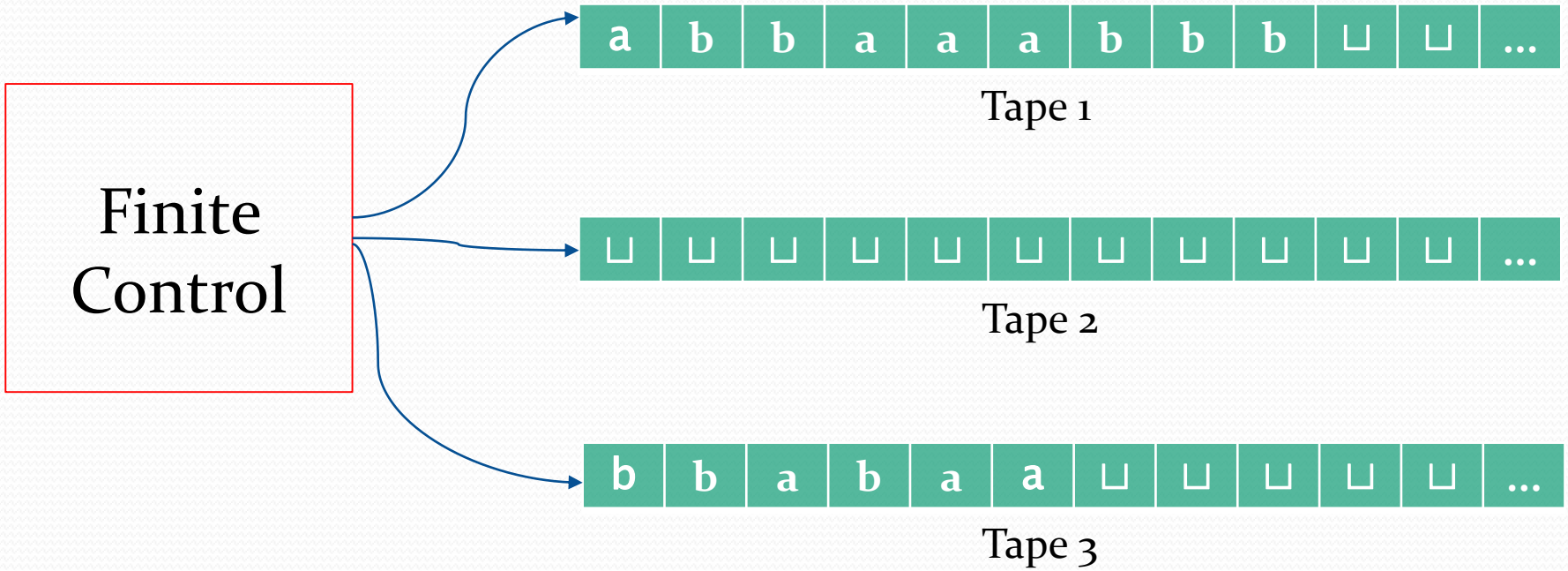


Fig: Block Diagram of Turing Machine

Extension of Turing Machine

- *A move of multiple TM depends on the state and the symbol scanned by each of tape head.*
  - *In one move, the multiple TM does following*
    - *The control enters in a new state, which may be same previous state.*
    - *On each step, a new symbol is written on the cell scanned, these symbols may be same as the symbols previously there.*
    - *Each of the tape head make a move either left or right or remain stationary.*
    - *Different head may have different direction independently i.e. if head of first tape moves leftward, at same time other head can move another directions or remain stationary.*

Extension of Turing Machine

- *For any fixed integer k >=1, a **k-tape** TM is a TM equipped with **k- tapes** and corresponding heads as shown in previous figure.*
- *Formally, k tape TM is a 7 tuple*

$$(Q, \Sigma, \Gamma, \delta, q0, B, F)$$

Here,

Transition function δ is of form:

$$\boldsymbol{Q\ X\ \Gamma^k \rightarrow Q\ X\ \Gamma^k\ X\ \{L,R\}^k}$$

where k is no. of tapes.

Extension of Turing Machine

- *The expression*

$\delta(q_i, a_1, a_2 \ldots\ldots\ldots a_k) \rightarrow (q_j, b_1, b_2 \ldots\ldots\ldots b_k, L, R \ldots\ldots\ldots L)$

means that if the machine is in state q_i *and heads* **1 though k** *are reading symbols* a_1 **through** a_k *,the machine goes to state* q_j *, writes symbol* b_1 **through** b_k *and directs each head to* ***move left or right*** *as specified.*

Equivalence of Single Tape and Multiple Tape TM

Let K>=1 be an integer. Any k tape TM can be converted to an equivalent one tape TM.

Here,

*We need to convert a multi-tape TM say **M** to an equivalent single tape TM say **S**.*

*The key idea is to show how to simulate **M** with **S**.*

*Let **M** has **k-tapes**, then **S** simulates the effect of **k-tapes** by storing their information on its single tape.*

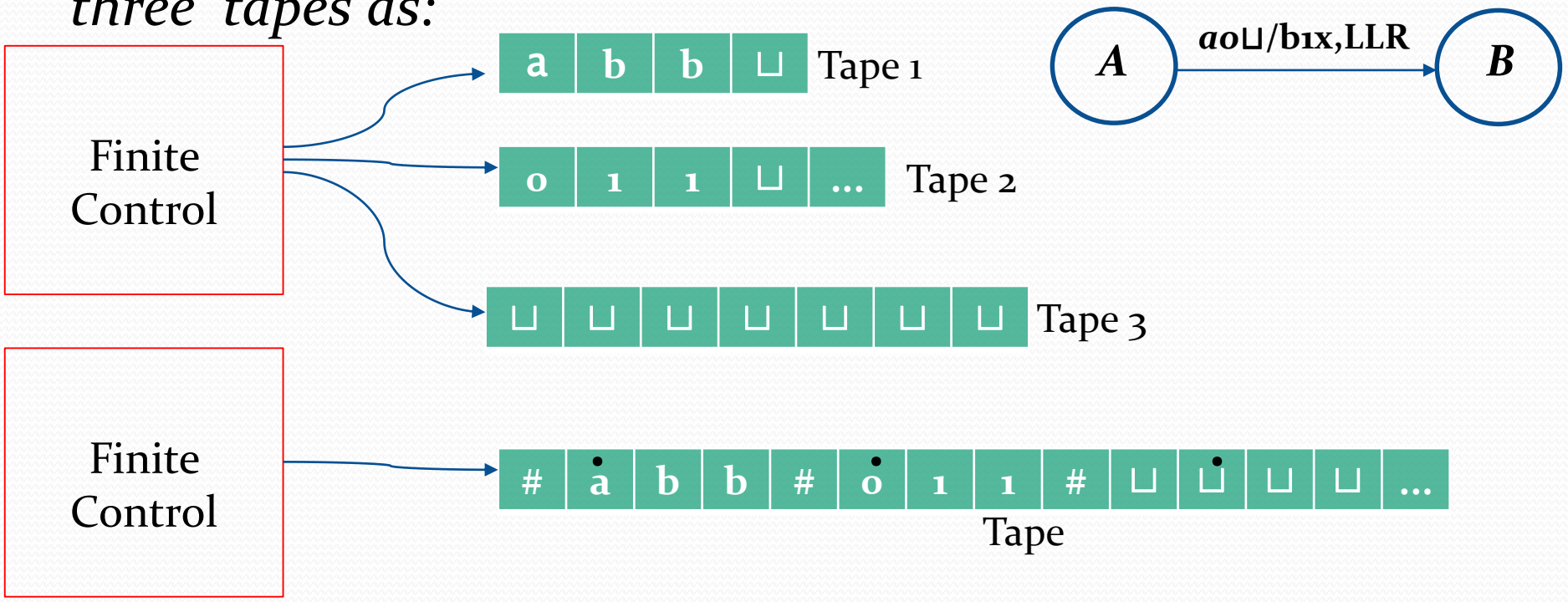
It uses the new symbol # as a delimiter to separate the contents of the different tapes.

Equivalence of Single Tape and Multiple Tape TM

In addition to the contents of those tapes, ***S*** *must keep track of the locations of the heads by writing a tape symbol with a dot(.) above it to mark the place where the head on that tape would be.*

Equivalence of Single Tape and Multiple Tape TM

We can illustrates how one tape can be used to represent three tapes as:



Representing 3 Tape with 1 Tape

Equivalence of Single Tape and Multiple Tape TM

- *On input* $w = w_1, w_2 \ldots\ldots\ldots\ldots w_n$
- *First* ***S*** *puts its tape into the format that represent all k tapes of M.*
- *The formatted tape contains:* $\#w_1 \#w_2\# \ldots . w_n\#B\#B\#\ldots\ldots\ldots\#$
- *To simulate a single move,* ***S*** *scans its tape from the first #, which makes left-hand end, to the (k+1) #, which makes right-hand end, in order to determine the symbols under virtual heads.*
- *Then S makes a second pass to update the tape according to way that* ***M's*** *transition function dictate.*

Equivalence of Single Tape and Multiple Tape TM

- *If at any point **S** moves one of the virtual head to the right onto a #, this action signifies that M has moved the corresponding head onto a previously unread blank portion of that tape.*
- *So **S** write a blank symbol on this tape cell and shifts the tape contents from this cell until rightmost # one unit to the right.*
- *Then it continues the simulation as before.*

Non- Deterministic TM

- Is the one which at any point in a computation may proceed according to several possibilities.
- The transition function δ are subsets rather than single element of set

$$Q \; X \; \Gamma \; X \; \{L,R,N\}$$

- Here transition function δ is such that for each state q and tape symbol x,

  $\delta\ (q,\ x)$ *is set of triples* $\{(q_1,r_1,D_1),\ (q_2,r_2,D_2)\ldots\ldots\ldots\ (q_k,r_k,D_k)\}$ *where k is finite integer.*

- The ***NTM*** can choose at each step, any of the triples to be the next move but it can't however pick a state from one, a tape symbol from other and direction from yet another.

Non- Deterministic TM

- The computation of a **NTM** is a tree whose branches correspond to different possibilities for the machine.
- If some branch of the computation leads to the accept state, the machine accepts its input.
- If M_N is a non deterministic Turing Machine, then there is a deterministic Turing Machine M_D such that

$$L(M_N) = L(M_D)$$

Unrestricted Grammar

- If there is no restriction in the production rule of the grammar then such grammar are called unrestricted grammar.
- A grammar G (V, Σ, R, S) is said to be unrestricted grammar if production in the grammar is of the form:

$$\alpha \rightarrow \beta$$

where α **is** $(V \cup \Sigma)^* . V . (V \cup \Sigma)^*$

β **is** $(V \cup \Sigma)^*$

V : Set of Variables

Σ ***: Set of Terminals***

For Example:

$SaA \rightarrow BaS$

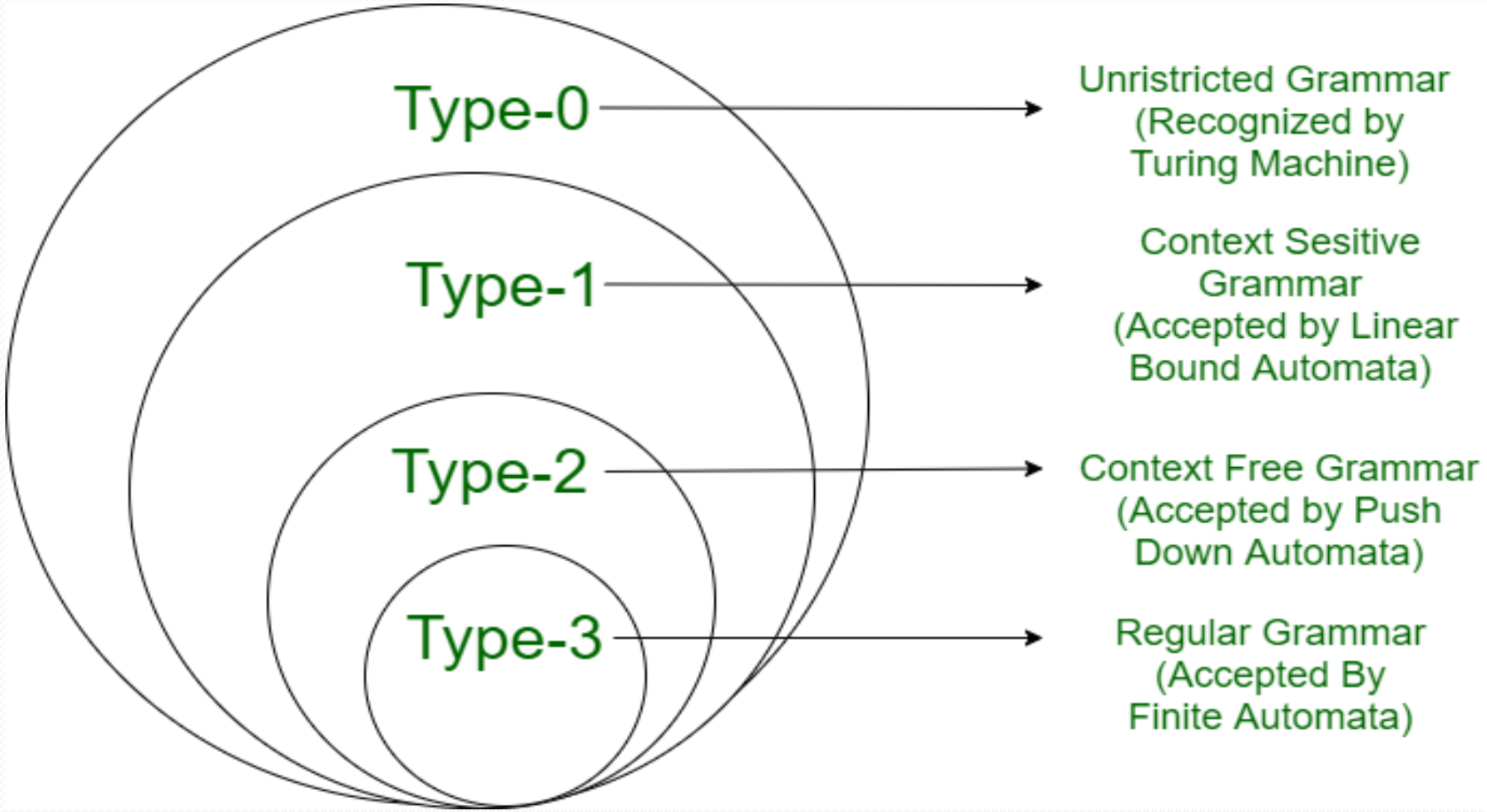
$Bb \rightarrow BaA$

$aA \rightarrow A$

$A \rightarrow a$

Unrestricted Grammar

Chomsky Hierarchy of Grammar



Recursive Function Theory

Explore Yourself !!!