St. Joseph's College of Engineering & Technology Palai

Department of Computer Science & Engineering

S4 CS

CS010 406 Theory of Computation

Module 4

Theory of Computation - Module 4

Syllabus

Turing Machines – Formal definition – Language acceptability by TM –TM as acceptors, Transducers - designing of TM- Two way infinite TM- Multi tape TM - Universal Turing Machines-
Church's Thesis-Godelization.- - Time complexity of TM -
Halting Problem - Rice theorem - Post correspondence problem-
Linear Bounded Automata.

Contents

I Turing Machine (TM) **4**

II Designing of Turing Machines **23**

III TM as Transducers **38**

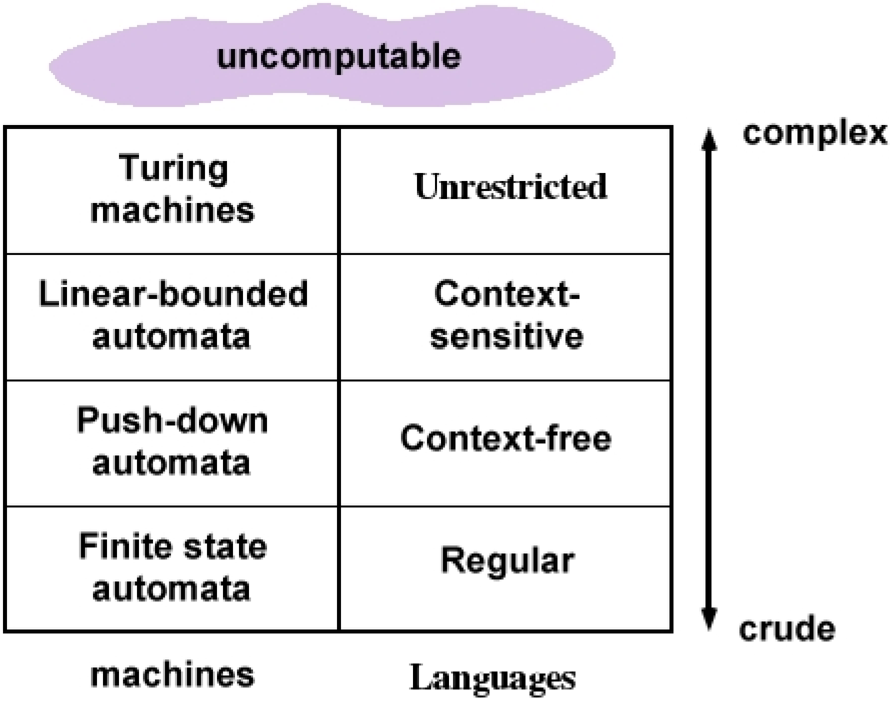
IV Types of TM **46**

V Church's Thesis **51**

VI Godelization **52**

Turing Machines

Consider the following figure:



machines	Languages
Turing machines	Unrestricted
Linear-bounded automata	Context-sensitive
Push-down automata	Context-free
Finite state automata	Regular

From the diagram, unrestricted grammars produce unrestricted languages (Type-0 languages). These Type-0 languages are recognised using Turing machines.

Following is an example for unrestricted grammar:

S ⟶ ACaB

CaBc ⟶ aaC

CB ⟶ DB

aD ⟶ Da

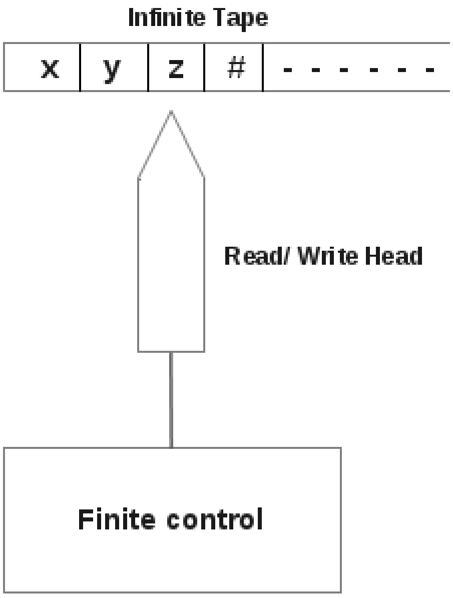
aEC ⟶ Ea

AE ⟶ ε

Here LHS and RHS can contain any set of terminals and non terminals.

Part I. Turing Machine (TM)

A Turing machine is a computing device that can be represented as,



From the above diagram, a TM consists of a,

1. Tape

   It is divided into a number of cells. Tape is infinite. A cell can hold a tape symbol or a blank symbol (#).

2. Finite control

   At a particular instant, finite control is in a state. States are divided into,

   Start state (q_0),

   Halt state (h),

   Intermediate states ($q_1, q_2, ...$).

3. Tape head

   Tape head points to one of the tape cells. It communicates between finite control and tape. It can move left , right or remain stationary.

1 Definition of a TM

A Turing machine is,

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

where

Q is a set of states,

$\sum$ is a set of input symbols,

Γ is a set of tape symbols, including #,

δ is the next move function from

$$(Q \times \Gamma) \longrightarrow (Q \times \Gamma \times \{L, R, N\}),$$

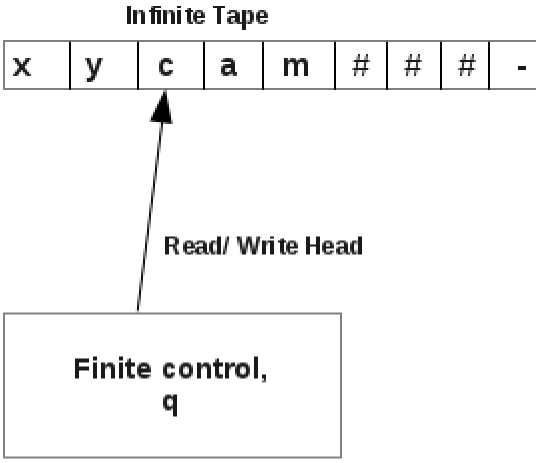
q_0 is the start state,

$\#$ is the blank symbol,

h is the halt state.

Example:

Consider the following TM,

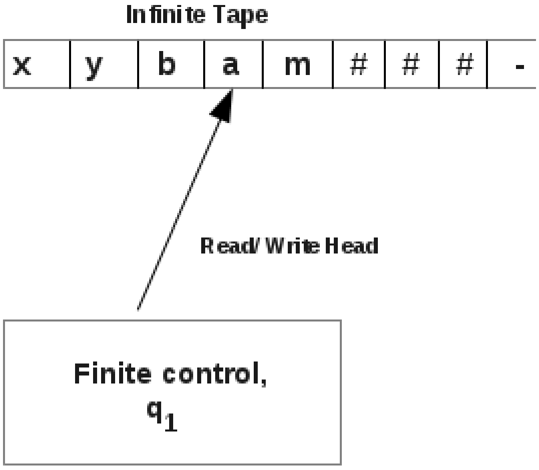


Here TM is in state, q and head points to tape symbol, c.

Let a move is defined as,

$\delta(q, c) = (q_1, b, R)$

This means, if TM is in state q and points to input symbol, c, then it enters state, q_1, replaces symbol, c with b and moves one position towards right (R). Thus the TM now is,

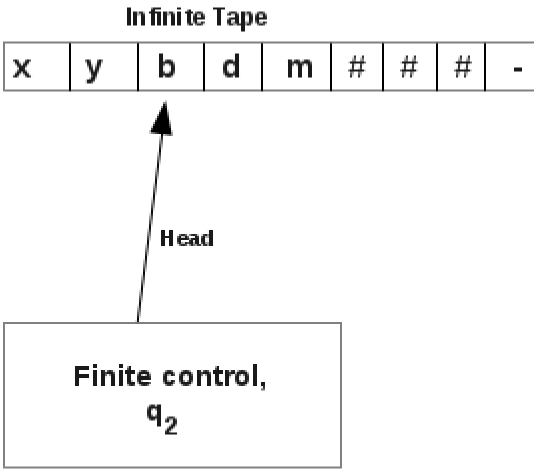


Now TM is in state q_1,and head points to the tape symbol, a.

Let a move of the TM is defined as,

$\delta(q_1, a) = (q_2, d, L)$

This means, if TM is in state q_1 and points to input symbol, a, then it enters state, q_2, replaces symbol, a with d and moves one position towards left (L). Thus the TM now is,

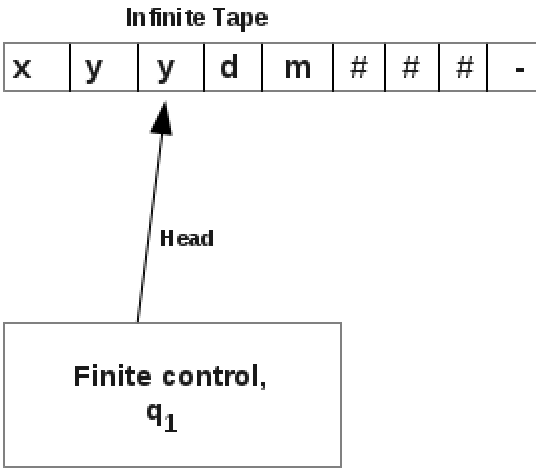


Now TM is in state q_2,and head points to the tape symbol, b.

Let a move of the TM is defined as,

$\delta(q_2, b) = (q_1, y, N)$

This means, if TM is in state q_2 and points to input symbol, b, then it enters state, q_1, replaces symbol, b with y and head remains in the same position (N). Thus the TM now is,



Now TM is in state q_1,and head points to the tape symbol, y.

Example:

Consider a Turing machine,

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

where

$Q = \{q_0, h\}$

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

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

$q_0 = q_0$

$\# = \#$

$h = h$

δ is defined as,

$\delta(q_0, a) = (q_0, \#, L)$

$\delta(q_0, b) = (q_0, \#, L)$

$\delta(q_0, \#) = (h, \#, N)$

$\delta(h, \#) = ACCEPT$

Language Acceptability by TM

A string, w is said to be accepted by a Turing machine, if TM halts in an accepting state. That is,

$$q_0 w \overset{*}{\vdash} \alpha_1 h \alpha_2$$

A string, w is not accepted by a TM, if TM halts in a non-accepting state or does not halt.

Example 1:

Consider a Turing machine,

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

where

$Q = \{q_1, q_2, q_3, q_4, q_5, q_6\}$

$\sum = \{0, 1, x, y\}$

$\Gamma = \{0, 1, x, y, \#\}$

$q_0 = q_1$

$\# = \#$

$h = q_6$

δ is defined as,

$$\delta(q_1, 0) = (q_2, x, R) \qquad \delta(q_1, \#) = (q_5, \#, R)$$

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

$$\delta(q_2, y) = (q_2, y, R) \qquad \delta(q_3, 0) = (q_4, 0, L)$$

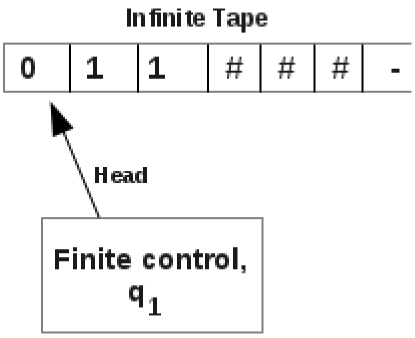
$$\delta(q_3, x) = (q_5, x, R) \qquad \delta(q_3, y) = (q_3, y, L)$$

$$\delta(q_4, 0) = (q_4, 0, L) \qquad \delta(q_4, x) = (q_1, x, R)$$

$$\delta(q_5, y) = (q_5, x, R) \qquad \delta(q_5, \#) = (q_6, \#, R)$$

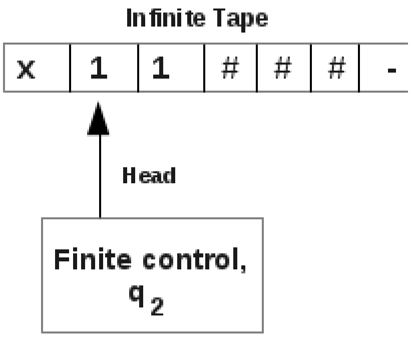
Check whether the string 011 is accepted by the above TM?

Initially, TM is in start state, q_1 and the tape contains the given string 011. Initially, head points to symbol 0 in the input string.



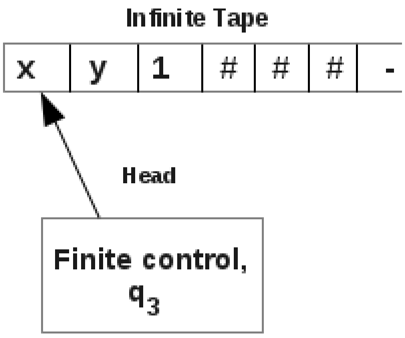
A given transition is, $\delta(q_1, 0) = (q_2, x, R)$

Then, TM becomes,



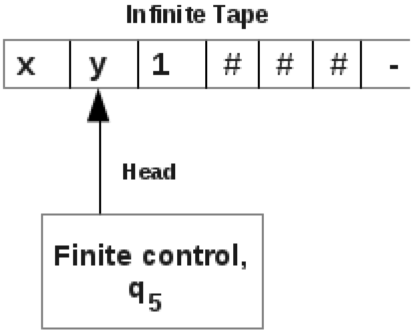
A given transition is, $\delta(q_2, 1) = (q_3, y, L)$

Then, TM becomes,



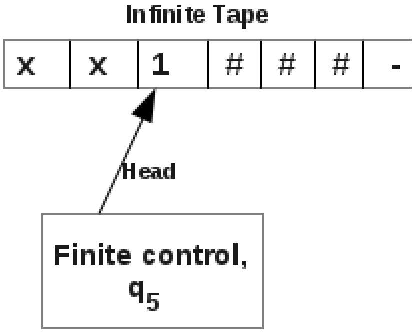
A given transition is, $\delta(q_3, x) = (q_5, x, R)$

Then, TM becomes,



A given transition is, $\delta(q_5, y) = (q_5, x, R)$

Then, TM becomes,



$\delta(q_5, 1)$ is not defined for the TM. So the string 011 is not accepted by the above TM.

Example 2:

Consider a Turing machine,

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

where

$Q = \{q_1, q_2, q_3, q_4, q_5, q_6\}$

$\sum = \{0, 1, x, y\}$

$\Gamma = \{0, 1, x, y, \#\}$

$q_0 = q_1$

$\# = \#$

$h = q_6$

δ is defined as,

$\delta(q_1, 0) = (q_2, x, R)$ $\delta(q_1, \#) = (q_5, \#, R)$

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

$\delta(q_2, y) = (q_2, y, R)$ $\delta(q_3, 0) = (q_4, 0, L)$

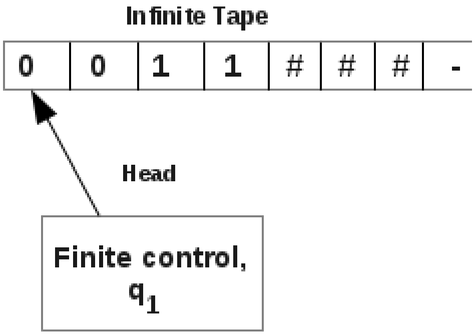
$\delta(q_3, x) = (q_5, x, R)$ $\delta(q_3, y) = (q_3, y, L)$

$\delta(q_4, 0) = (q_4, 0, L)$ $\delta(q_4, x) = (q_1, x, R)$

$\delta(q_5, y) = (q_5, x, R)$ $\delta(q_5, \#) = (q_6, \#, R)$

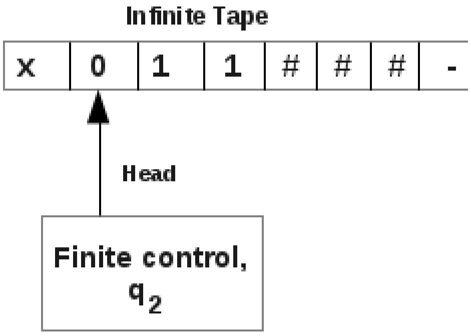
Check whether the string 0011 is accepted by the above TM?

Initially, TM is in start state, q_1 and the tape contains the given string 0011. Initially, head points to symbol 0 in the input string.



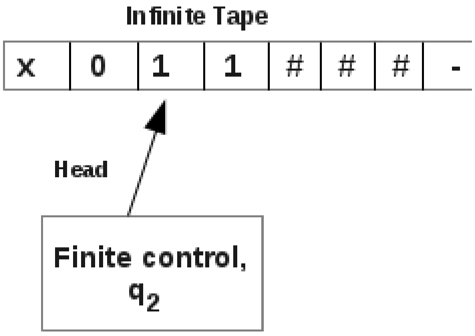
A given transition is, $\delta(q_1, 0) = (q_2, x, R)$

Then, TM becomes,



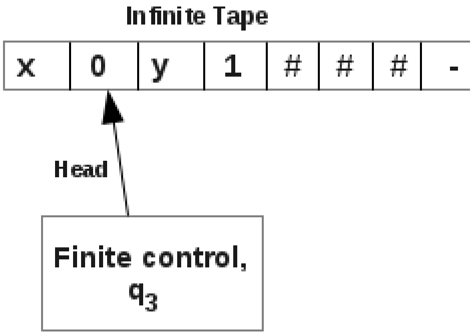
A given transition is, $\delta(q_2, 0) = (q_2, 0, R)$

Then, TM becomes,



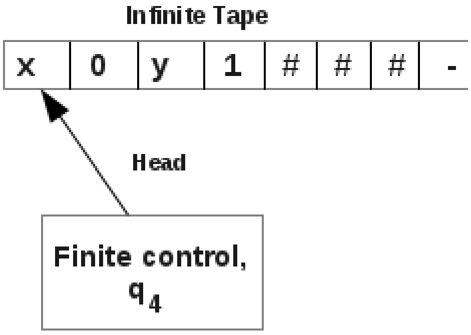
A given transition is, $\delta(q_2, 1) = (q_3, y, L)$

Then, TM becomes,



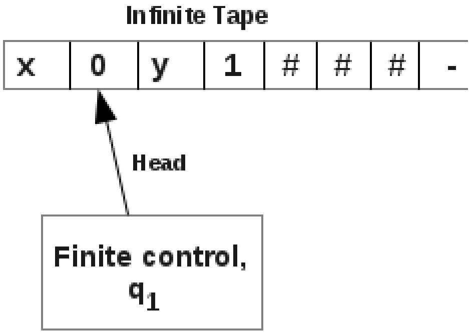
A given transition is, $\delta(q_3, 0) = (q_4, 0, L)$

Then, TM becomes,



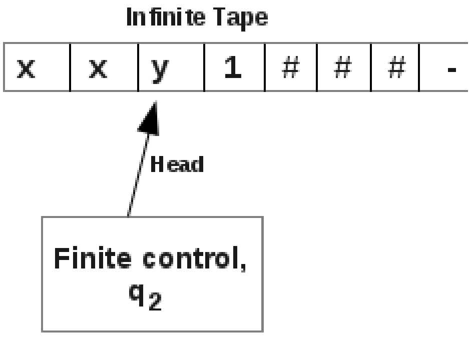
A given transition is, $\delta(q_4, x) = (q_1, x, R)$

Then, TM becomes,



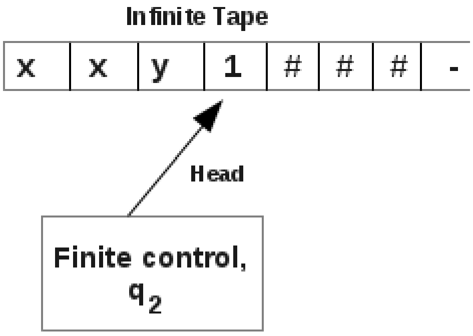
A given transition is, $\delta(q_1, 0) = (q_2, x, R)$

Then, TM becomes,



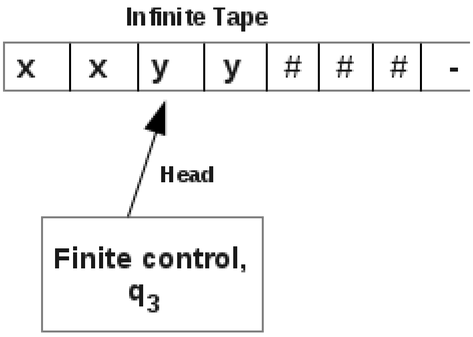
A given transition is, $\delta(q_2, y) = (q_2, y, R)$

Then, TM becomes,



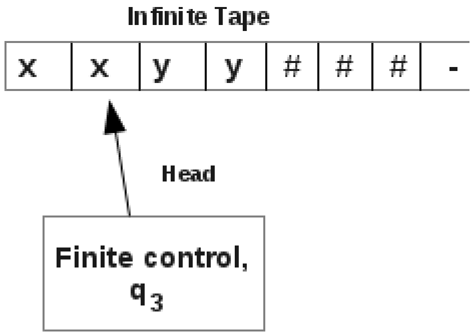
A given transition is, $\delta(q_2, 1) = (q_3, y, L)$

Then, TM becomes,



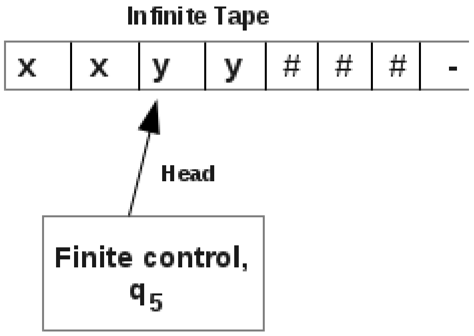
A given transition is, $\delta(q_3, y) = (q_3, y, L)$

Then, TM becomes,



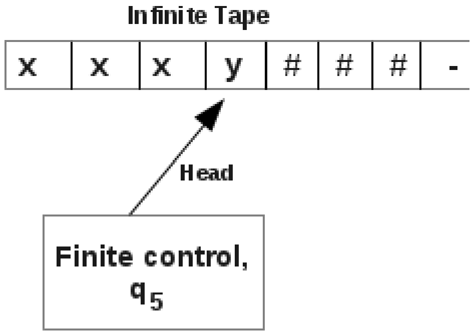
A given transition is, $\delta(q_3, x) = (q_5, x, R)$

Then, TM becomes,



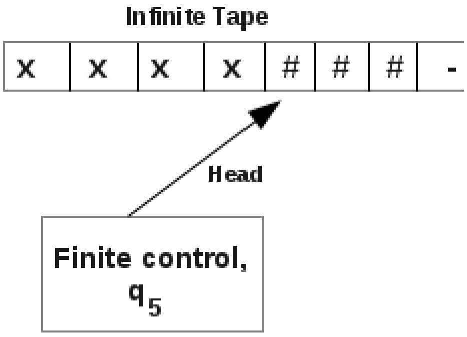
A given transition is, $\delta(q_5, y) = (q_5, x, R)$

Then, TM becomes,



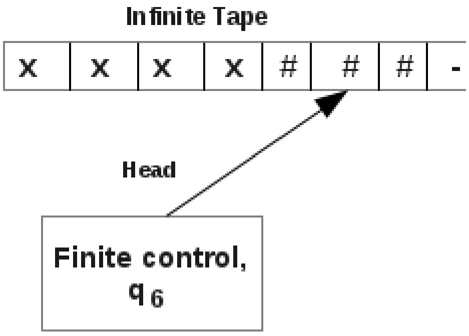
A given transition is, $\delta(q_5, y) = (q_5, x, R)$

Then, TM becomes,



A given transition is, $\delta(q_5, \#) = (q_6, \#, R)$

Then, TM becomes,



q_6 is the halt state or accepting state of the TM. So the string 0011 is accepted by the above TM.

Example 3:

Consider a Turing machine,

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

where

$Q = \{q_1, q_2, q_3, q_4, q_5, q_6\}$

$\sum = \{0, 1, x, y\}$

$\Gamma = \{0, 1, x, y, \#\}$

$q_0 = q_1$

$\# = \#$

$h = q_6$

δ is defined as,

$\delta(q_1, 0) = (q_2, x, R)$ $\delta(q_1, \#) = (q_5, \#, R)$

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

$\delta(q_2, y) = (q_2, y, R)$ $\delta(q_3, 0) = (q_4, 0, L)$

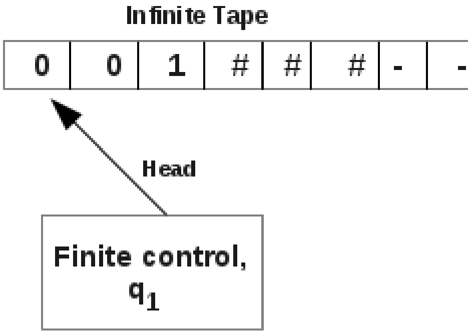
$\delta(q_3, x) = (q_5, x, R)$ $\delta(q_3, y) = (q_3, y, L)$

$\delta(q_4, 0) = (q_4, 0, L)$ $\delta(q_4, x) = (q_1, x, R)$

$\delta(q_5, y) = (q_5, x, R)$ $\delta(q_5, \#) = (q_6, \#, R)$

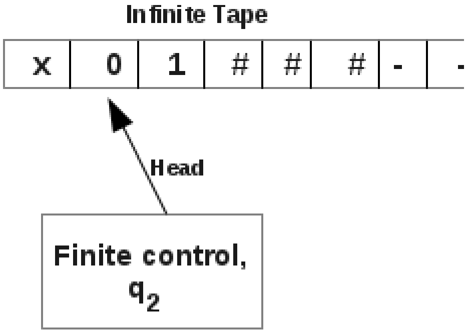
Check whether the string 001 is accepted by the above TM?

Initially, TM is in start state, q_1 and the tape contains the given string 001. Initially, head points to symbol 0 in the input string.



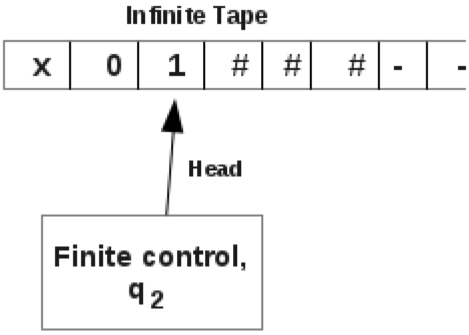
A given transition is, $\delta(q_1, 0) = (q_2, x, R)$

Then, TM becomes,



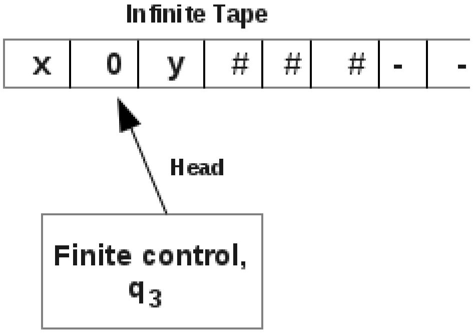
A given transition is, $\delta(q_2, 0) = (q_2, 0, R)$

Then, TM becomes,



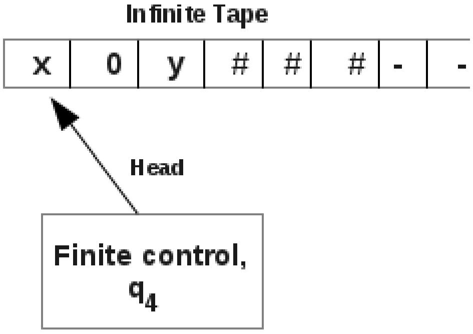
A given transition is, $\delta(q_2, 1) = (q_3, y, L)$

Then, TM becomes,



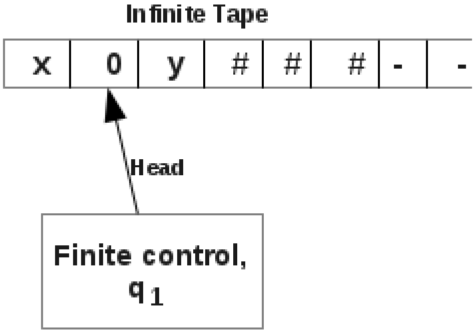
A given transition is, $\delta(q_3, 0) = (q_4, 0, L)$

Then, TM becomes,



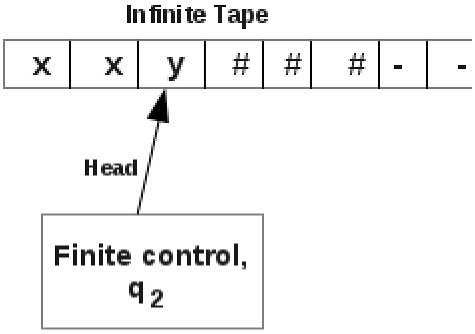
A given transition is, $\delta(q_4, x) = (q_1, x, R)$

Then, TM becomes,



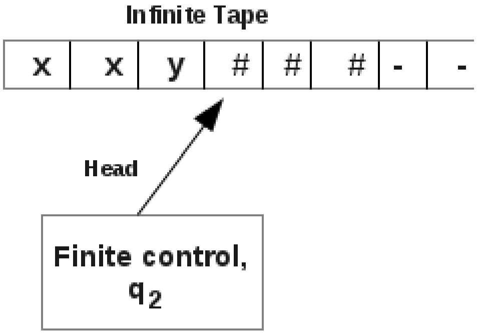
A given transition is, $\delta(q_1, 0) = (q_2, x, R)$

Then, TM becomes,



A given transition is, $\delta(q_2, y) = (q_2, y, R)$

Then, TM becomes,



Now TM halts because no δ is defined for q_2 and #. But q_2 is not the halt state of TM. So the string 001 is not accepted by the above TM.

2 Representation of Turing Machines

We can represent a TM in different ways. They are,

Instantaneous descriptions,

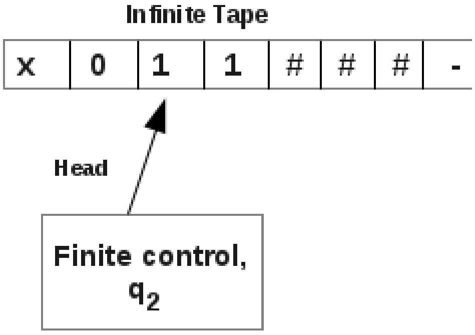
Transition tables,

Transition diagrams.

Representation of TM by Instantaneous Descriptions

In all the above examples, Turing machines shown in the pictures were instaneous descriptions. This means an instant of a TM is shown.

Following shows an instantaneous description of a TM,



Currently, symbol under the head is 1.

Current state is q_2.

Symbols on the left of 1 are x and 0.

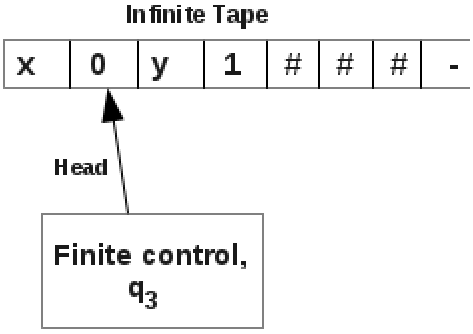
Symbnols on the right of 1 are 1, #, #, ...

This instant of TM can be written as,

$x\,0\,q_2\,1\,1\,\#\,\#\,\#$.

Let a transition is defines as, $\delta(q_2, 1) = (q_3, y, L)$

Then TM moves to,



That is,

$x\,q_2\,0\,y\,1\,\#\,\#\,\#$.

This transition from one state to other can be written as,

$x\,0\,q_2\,1\,1\,\#\,\#\,\# \vdash x\,q_2\,0\,y\,1\,\#\,\#\,\#$

Representation of TM by Transition Table

Let a Turing machine is defined as,

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

where

$Q = \{q_1, q_2, q_3, q_4, q_5, q_6\}$

$\sum = \{0, 1, x, y\}$

$\Gamma = \{0, 1, x, y, \#\}$

$q_0 = q_1$

$\# = \#$

$h = q_6$

δ is defined as,

$\delta(q_1, 0) = (q_2, x, R)$ $\delta(q_1, \#) = (q_5, \#, R)$

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

$\delta(q_2, y) = (q_2, y, R)$ $\delta(q_3, 0) = (q_4, 0, L)$

$\delta(q_3, x) = (q_5, x, R)$ $\delta(q_3, y) = (q_3, y, L)$

$\delta(q_4, 0) = (q_4, 0, L)$ $\delta(q_4, x) = (q_1, x, R)$

$\delta(q_5, y) = (q_5, x, R)$ $\delta(q_5, \#) = (q_6, \#, R)$

This TM can be represented as a Transition table.

Transition table corresponding to the above TM is,

	Tape Symbol				
Current State	#	x	y	0	1
$\longrightarrow q_1$	$(q_5, \#, R)$			(q_2, x, R)	
q_2			(q_2, y, R)	$(q_2, 0, R)$	(q_3, y, L)
q_3		(q_5, x, R)	(q_3, y, L)	$(q_4, 0, L)$	
q_4		(q_1, x, R)		$(q_4, 0, L)$	
$*q_5$	$(q_6, \#, R)$		(q_5, x, R)		

Above is the transition table representation of a TM.

In the above transition table, the entry, (q_2, y, R) corresponding to row q_2 and column y denotes the transition,

$\delta(q_2, y) = (q_2, y, R)$

Representation of TM by Transition Diagram

Let a Turing machine is defined as,

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

where

$Q = \{q_1, q_2, q_3, q_4, q_5, q_6\}$

$\sum = \{0, 1, x, y\}$

$\Gamma = \{0, 1, x, y, \#\}$

$q_0 = q_1$

$\# = \#$

$h = q_6$

δ is defined as,

$\delta(q_1, 0) = (q_2, x, R)$ $\delta(q_1, \#) = (q_5, \#, R)$

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

$$\delta(q_2, y) = (q_2, y, R) \qquad \delta(q_3, 0) = (q_4, 0, L)$$

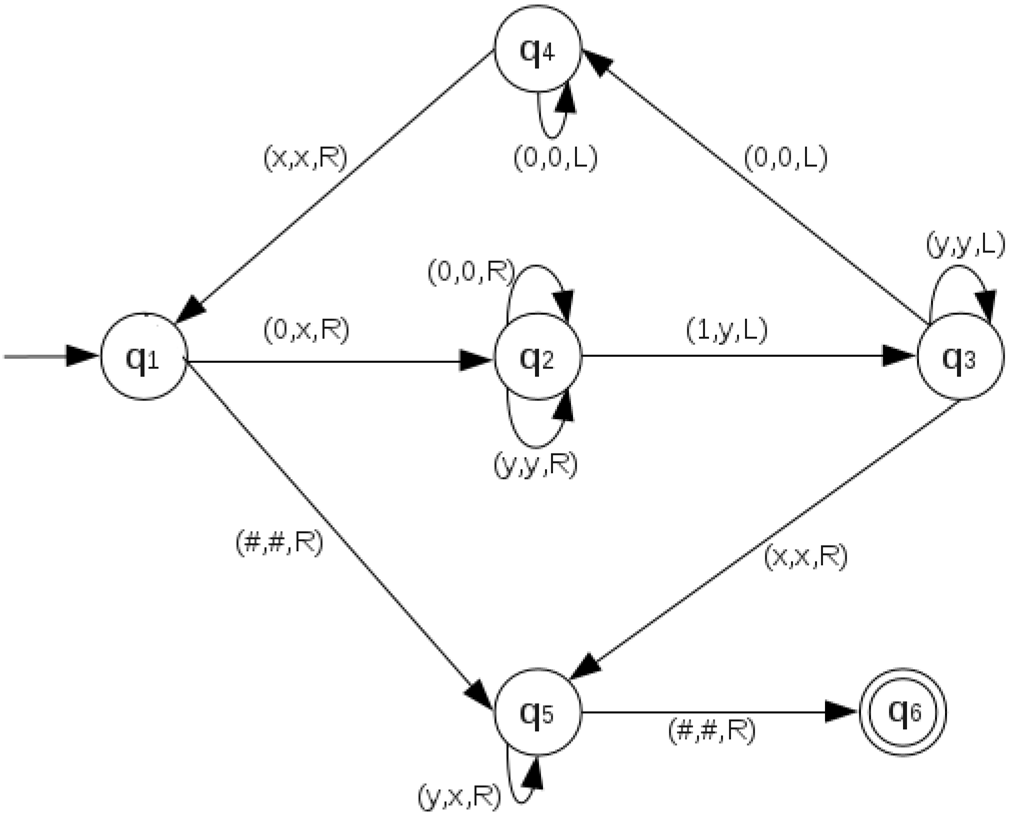
$$\delta(q_3, x) = (q_5, x, R) \qquad \delta(q_3, y) = (q_3, y, L)$$

$$\delta(q_4, 0) = (q_4, 0, L) \qquad \delta(q_4, x) = (q_1, x, R)$$

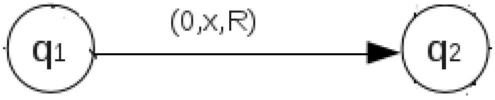
$$\delta(q_5, y) = (q_5, x, R) \qquad \delta(q_5, \#) = (q_6, \#, R)$$

This TM can be represented as a Transition diagram.

Transition diagram corresponding to the above TM is,



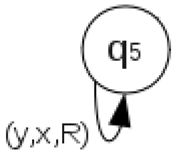
In the above transition diagram, the part of th picture,



denotes the transition,

$$\delta(q_1, 0) = (q_2, x, R)$$

Also the part of the picture,



denotes the transition,

$$\delta(q_5, y) = (q_5, x, R)$$

Exercises:

1. Consider the TM given in the following transition table:

Current State	Tape symbol		
	#	0	1
$\longrightarrow q_1$	$(q_2, 1, L)$	$(q_1, 0, R)$	
q_2	$(q_3, \#, R)$	$(q_2, 0, L)$	$(q_2, 1, L)$
q_3		$(q_4, \#, R)$	$(q_5, \#, R)$
q_4	$(q_5, 0, R)$	$(q_4, 0, R)$	$(q_4, 1, R)$
$*q_5$	$(q_2, 0, L)$		

Check whether the string 00 is accepted by the above TM.

2. Consider the TM given in the following transition table:

Current State	Tape symbol				
	#	0	1	x	y
$\longrightarrow q_1$	$(q_6, \#, R)$	$(q_2, 0, R)$			
q_2		$(q_2, 0, R)$	(q_2, y, L)		(q_2, y, R)
q_3		$(q_4, 0, L)$		(q_5, x, R)	(q_3, y, L)
q_4		$(q_4, 0, L)$		(q_1, x, R)	
q_5	$(q_6, \#, R)$				(q_5, y, R)
$*q_6$					

Check whether the string 0011 is accepted by the above TM.

Part II. Designing of Turing Machines

3 Designing of TM

Following are the basic guidelines for designing a TM.

1. The fundamental objective of scanning a symbol in the tape is to know what to do in the future. TM must remember the past symbols scanned. TM can remember this by going to the next unique state.

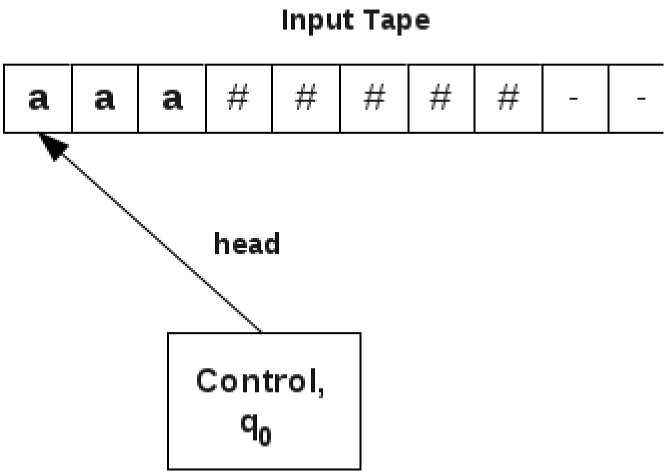
2. The number of states must be minimised. This is achieved by changing the states only when there is a change in the written symbol or when there is a change in the movement of the head.

Following examples show designing of TMs.

Example 1:

Design a turing machine over $\{a\}$ to accept the language L=$\{a^n|n$ is odd$\}$.

The initial instant of the TM is shown below. The input tape contains aaa followed by blanks. The TM is initially in state q_0 and head points to the first a of the input string w.



The working of the TM is as follows:

In state q_0, head reads symbol a from the current cell and

1. keeps the contents of the current cell as a,

2. changes state of the TM to q_1, and

3. moves to the next cell on the right side.

The corresponding transition function is,

$$\delta(q_0, a) = (q_1, a, R)$$

In state q_1, head reads symbol a from the current cell and

1. keeps the contents of the current cell as a,

2. changes the state of the TM to q_0, and

3. moves to the next cell on the right side.

The corresponding transition function is,

$$\delta(q_1, a) = (q_0, a, R)$$

Thsi process continues till a blank symbol, # is reached.

If the blank symbol is reached in state q_1, it indicates that input string contains an odd number of a's.

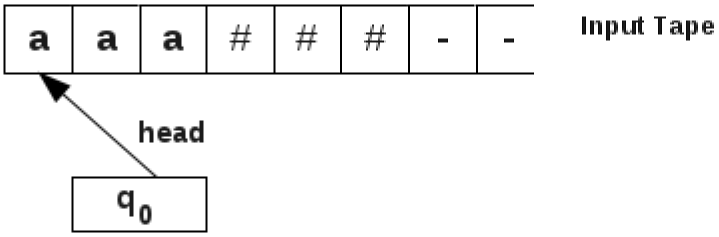
If the blank symbol is reached in state q_0, it indicates that input string contains an even number of a's.

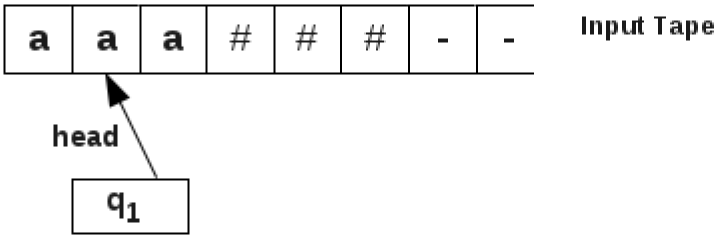
Hence, $q_1$is the final state.

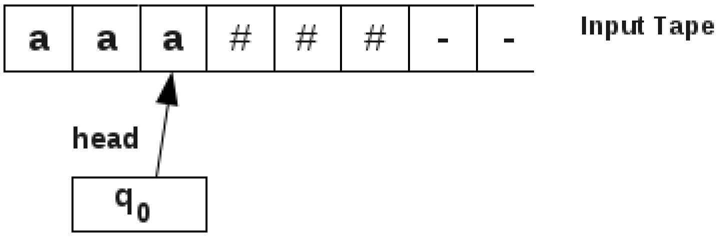
Transition table for the TM is,

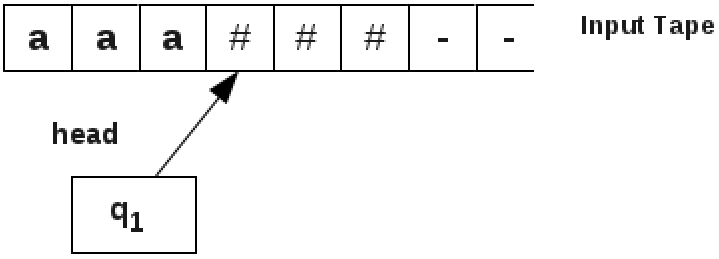
	Input	Symbol
Current State	a	#
$\longrightarrow q_0$	(q_1, a, R)	-
$*q_1$	(q_0, a, R)	-

Consider the processing of string, aaa using the above TM,



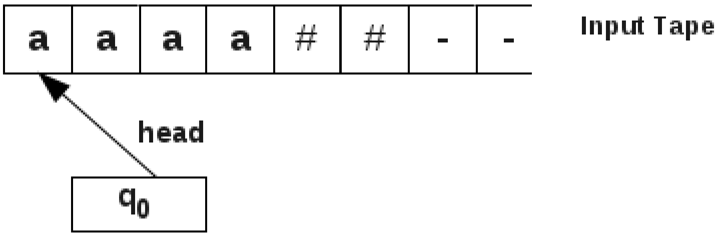


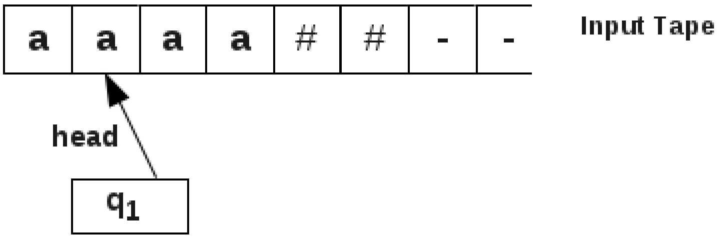


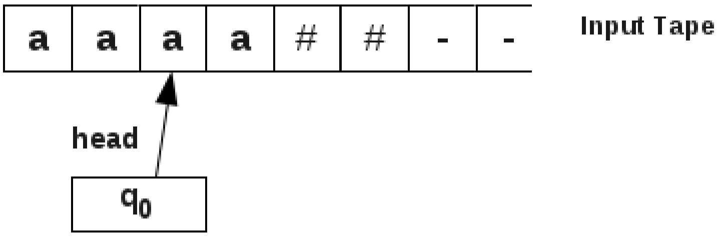


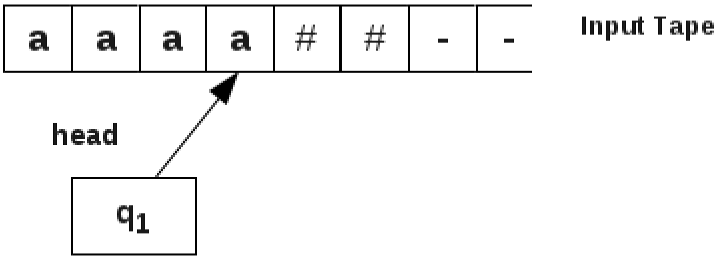
The TM halts at state q_1. $q_1$is a final state. So the string aaa is accepted.

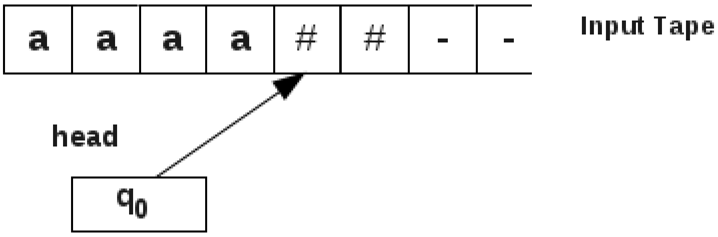
Consider the processing of string, aaaa using the above TM,









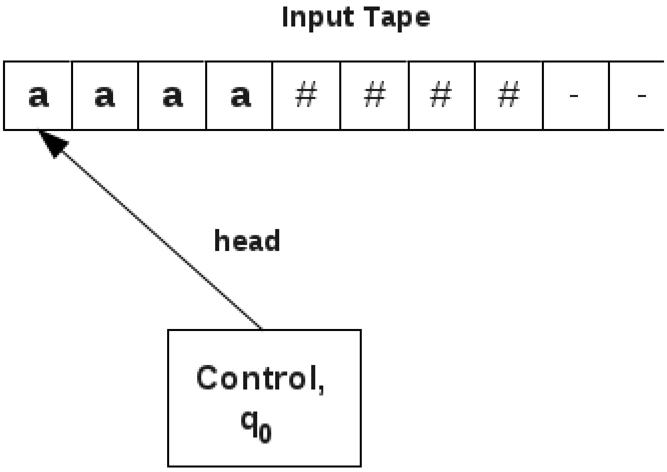


The TM halts at state q_0. $q_0$is not a final state. So the string aaaa is not accepted.

Example 2:

Design a turing machine over $\{a\}$ to accept the language L=$\{a^n|n$ is even$\}$.

The initial instant of the TM is shown below. The input tape contains $aaaa$ followed by blanks. The TM is initially in state q_0 and head points to the first a of the input string w.



The working of the TM is as follows:

In state q_0, head reads symbol a from the current cell and

1. keeps the contents of the current cell as a,
2. changes state of the TM to q_1, and
3. moves to the next cell on the right side.

The corresponding transition function is,

$$\delta(q_0, a) = (q_1, a, R)$$

In state q_1, head reads symbol a from the current cell and

1. keeps the contents of the current cell as a,
2. changes the state of the TM to q_0, and
3. moves to the next cell on the right side.

The corresponding transition function is,

$$\delta(q_1, a) = (q_0, a, R)$$

Thsi process continues till a blank symbol, # is reached.

If the blank symbol is reached in state q_0, it indicates that input string contains an even number of a's.

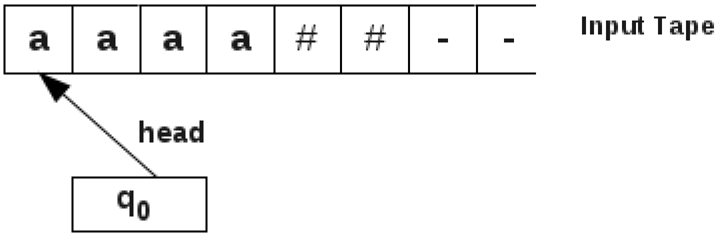
If the blank symbol is reached in state q_1, it indicates that input string contains an odd number of a's.

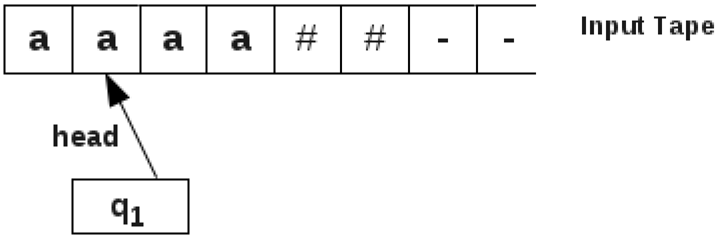
Hence, $q_1$is the final state.

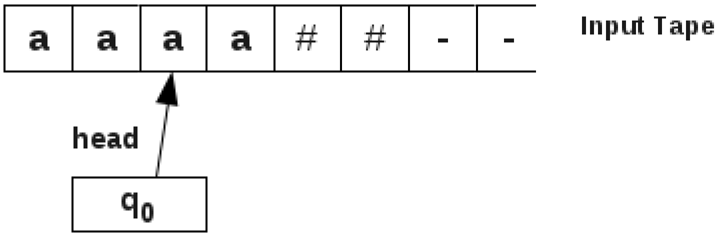
Transition table for the TM is,

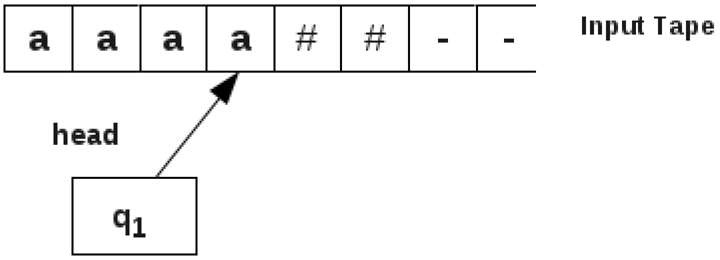
	Input	Symbol
Current State	a	#
$\longrightarrow *q_0$	(q_1, a, R)	-
q_1	(q_0, a, R)	-

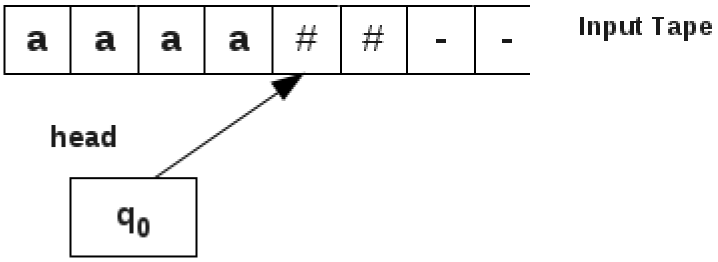
Consider the processing of string, aaaa using the above TM,











The TM halts at state q_0. $q_0$is a final state. So the string aaaa is accepted.

Example 3:

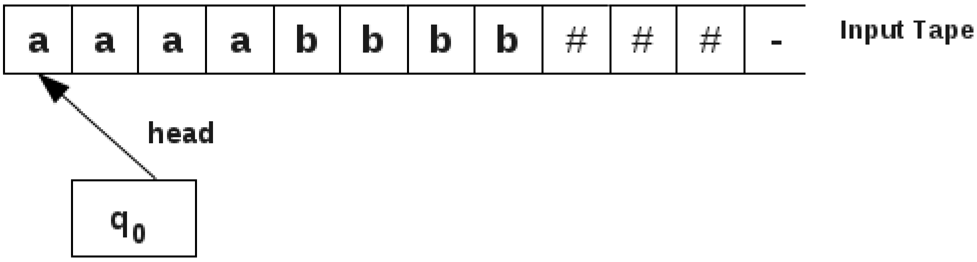
Design a Turing machine over $\{a, b\}$ to accept the language L=$\{a^n b^n | n \geq 1$ }.

Example strings in this language are ab, aabb, aaabbb, aaaabbbb,.........

Let the string be aaaabbbb.

String recognition process of the Turing machine is as follows:

Initially, TM is in state q_0 and head points to first a in the string.



1. Read the first symbol a of the string and convert this a to blank (#).

2. The head keeps moving towards right till the input string is crossed and the first blank symbol after the string is reached.

3. From this #, turn one cell left and reach the last symbol of the input string. If it is 1, convert it to #.

After performing the first three steps, input tape with aaaabbbb###, will contain #aaabbb####. That is, leftmost a is cancelled with rightmost b.

4. Now after converting the last b to #, the head keeps moving towards left till a # is reached, turn towards right and converts the first available a to #, and keeps moving towards right till a # is reached. It then turns left and converts the first available b to # and keeps moving towards left.

After step 4, input tape will contain ##aabb#####. This marks the cancellation of the next leftmost a by the next rightmost b.

5. The head keeps repeating step 4 till all the a's are cancelled by teh corresponding b's.

6. If the cancellation process is successfully completed, then the string is accepted.

7. If there are remaining uncancelled a's or b's, then the input string is not accepted.

Transition table corresponding to the TM is given below:

	Input	Symbol	
Current State	a	b	#
$\longrightarrow q_0$	$(q_1, \#, R)$	—	—
q_1	(q_0, a, R)	(q_2, b, R)	—
q_2	—	(q_2, b, R)	$(q_3, \#, L)$
q_3	—	$(q_4, \#, L)$	—
q_4	(q_5, a, L)	(q_4, b, L)	$(q_6, \#, R)$
q_5	(q_5, a, L)	—	$(q_0, \#, R)$
q_6	—	—	$(q_f, \#, R)$
$*q_f$	—	—	—

The working of the above TM is explained below:

1. The head reads the first symbol of the input string in state q_0 and does the following:

a. If the symbol is # or b, then TM halts. If the first symbol is #, then it is a null string and does not belong to the language. If the symbol is 'b', then it is also an invalid string. Thus, q_0 is a non-final state.

b. If the first symbol is 'a', then head converts this input symbol to #, changes state to q_1, and moves towards right. $\delta(q_0, a) = (q_1, \#, R)$

2. In state q_1, head does the following:

a. If the head is pointing to input symbol a, then symbol 'a' remains unchanged, state q_1 remains unchanged, and head moves to next cell on the right side. $\delta(q_1, a) = (q_1, a, R)$

b. If the head is pointing to input symbol b, then symbol 'b' remains unchanged, state q_1 changes to state q_2, and head moves to next cell on the right side. $\delta(q_1, b) = (q_2, b, R)$

In state q_1, head simply passes over all the 0s of the input string from left to right and changes to state q_2 as soon as symbol b is received.

3. In state q_2, head does the following:

a. If the head is pointing to input symbol b, then symbol 'b' remains unchanged, state q_2 remains unchanged, and head moves to next cell on the right side. $\delta(q_2, b) = (q_2, b, R)$

b. If the head is pointing to symbol #, then symbol '#' remains unchanged, state q_2 changes to state q_3, and head moves to cell on the left side. $\delta(q_2, \#) = (q_3, \#, L)$

In state q_2, head simply passes over all the b's of the input string from left to right and changes to state q_3 as soon as symbol # is received and turns towards left.

4. In state q_3, head does the following:

a. If the head is pointing to input symbol b, then symbol 'b' is replaced with #, state q_3 changes to q_4, and head moves to next cell on the left side. $\delta(q_3, b) = (q_4, \#, L)$

In state q_3, symbol 'b' is replaced with # in response to the 'a' symbol replaced with # in state q_0.

5. In state q_4, head does the following:

a. If the head is pointing to input symbol b, then symbol 'b' remains unchanged, state q_4 remains unchanged, and head moves to next cell on the left side. $\delta(q_4, b) = (q_4, b, L)$

b. If the head is pointing to symbol a, then symbol 'a' remains unchanged, state q_4 changes to state q_5, and head moves to cell on the left side. $\delta(q_4, a) = (q_5, a, L)$

c. If the head is pointing to symbol #, then symbol '#' remains unchanged, state q_4 changes to state q_6, and head moves to cell on the right side. $\delta(q_4, \#) = (q_6, \#, R)$. This indicates that all a's have been cancelled.

In state q_4, head simply passes over all the b's of the input string from right to left and changes to state q_5 as soon as symbol a is received.

6. In state q_5, head does the following:

a. If the head is pointing to input symbol a, then symbol 'a' remains unchanged, state q_5 remains unchanged, and head moves to next cell on the left side. $\delta(q_5, a) = (q_5, a, L)$

b. If the head is pointing to symbol #, then symbol '#' remains unchanged, state q_5 changes to state q_0, and head moves to cell on the right side. $\delta(q_5, \#) = (q_0, \#, R)$

In state q_5, head simply passes over all the a's of the input string from right to left and changes to state q_0 as soon as symbol # is received.

7. In steps 1-6, leftmost a and rightmost b cancel out each other. This is done by converting them to blanks. This is repeated till all the a's are cancelled by the corresponding b's.

When all a's are cancelled by the respective b's, head will point ot a # symbol in state q_6 and moves to final state where it halts and it indicates that the string is accepted.

8. In state q_6, head does the following:

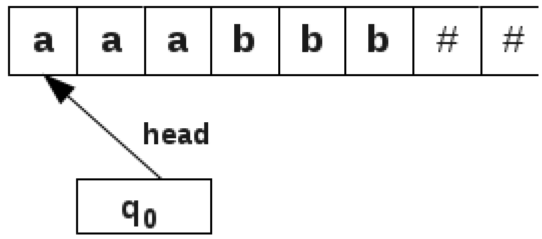
a. If the head is pointing to input symbol #, then it indicates that all the a's have been successfully matched with the corresponding b's. Hence the TM changes state to q_f and halts. It indicates taht the string is accepted. $\delta(q_6, \#) = (q_f, \#, R)$

b. If the head is pointing to symbol b, this shows that the number of b's is more than the number of a's and hence the TM halts in state q_6 to indicate the non-acceptance of the string.

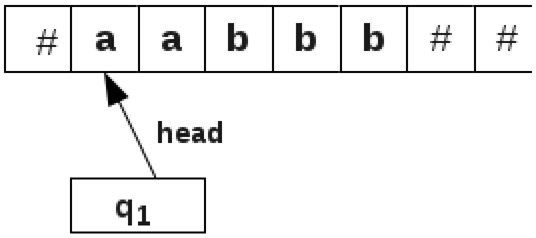
Let the input string is aaabbb.

TM processes the string as follows:

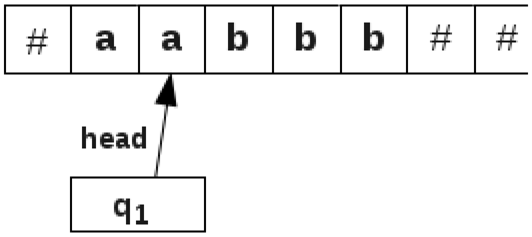
1



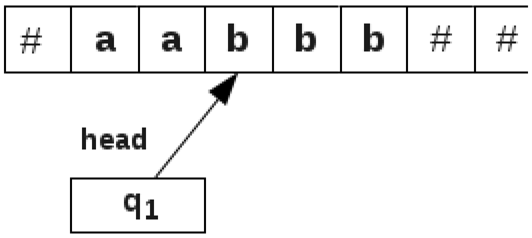
2



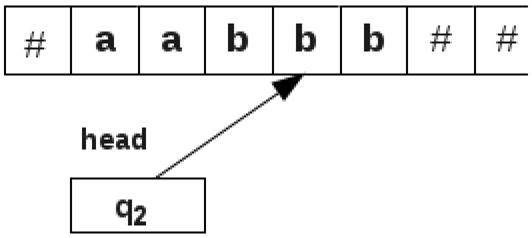
3



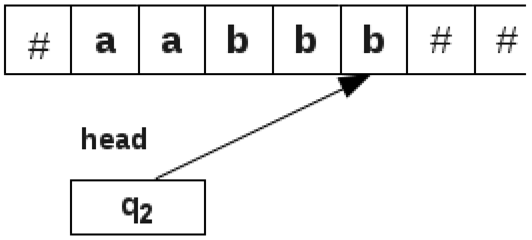
4



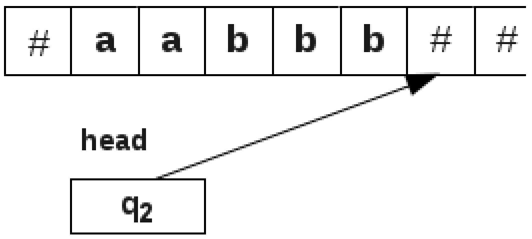
5



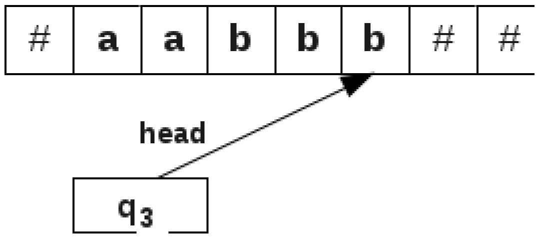
6



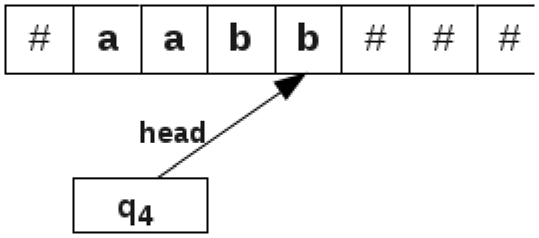
7



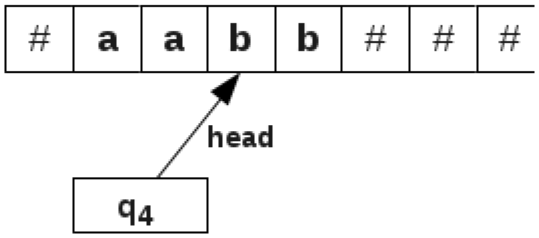
8



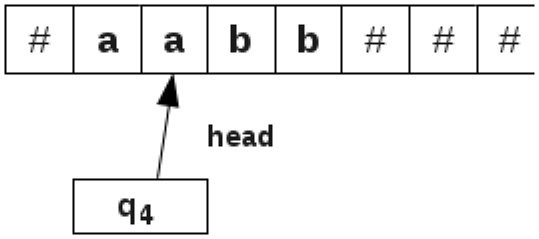
9



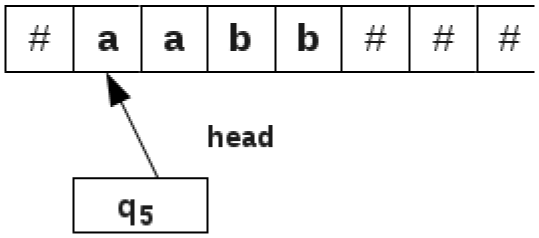
10



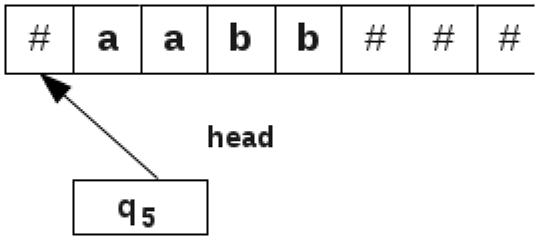
11



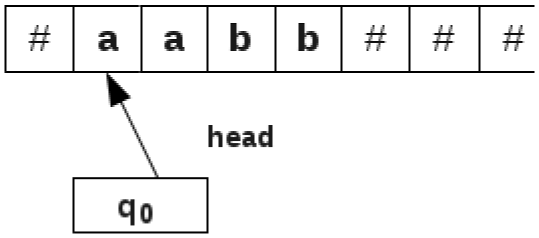
12



13



14



After the 14th step leftmost a is cancelled with the rightmost b. This process is repeated and finally we get all #s and TM will reach state q_f. Then the string aaabbb is accepted by the above TM.

Example 4:

Design a Turing machine over $\{a, b\}$ to accept the language L=$\{ww^R | w \in (a, b)^+\}$.

Example strings in this language are aa, abba, bb, baab, ababbbbaba, ...

String recognition process of the Turing machine is as follows:

1. Initially, TM is in state q_0 and head points to first symbol in the string.

2. The symbol may be a or b. Aftyer reading this symbol, this symbol is replaced with a # and keeps moving towards right till the whole of the string is crossed and the first # after the string is reached.

3. From this blank, the head turns one cell left and reaches the last symbol of the input string. If this symbol matches with the first symbol of the input string, it converts it into a #.

4. Steps 1-3 match the first and last symbols of the input string. If this matching is successful, the head traverses back and crosses the string to reach the first #. From this #, it turns towards right and matches the leftmost and rightmost symbols of the remaining string. If the matching is successful, then the symbols are converted to #s.

5. The process repeats till the symbols are matched.

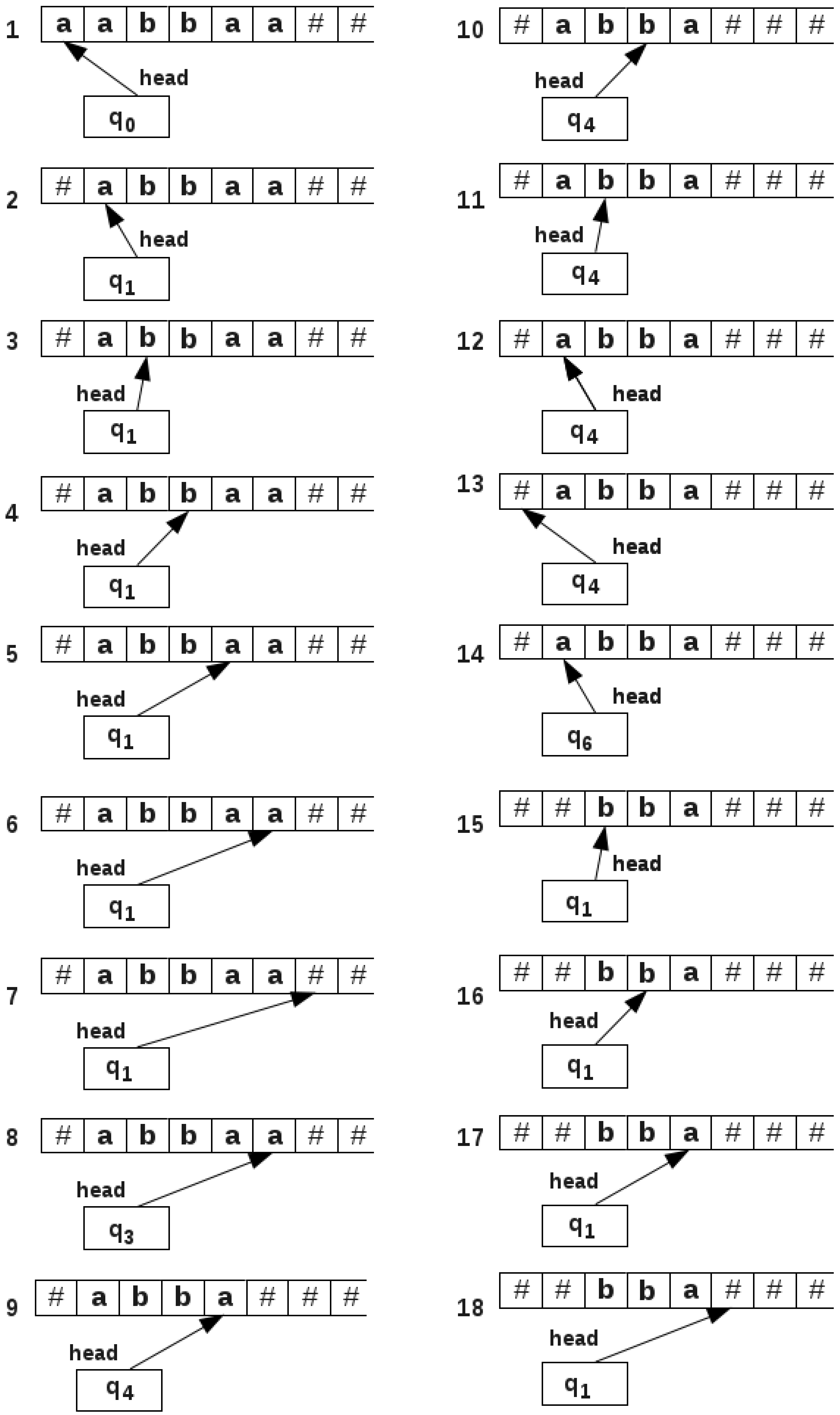
6. To keep track of the symbol to be matched, if symbol 'a' is received in state q_0, then head enters state q_1, and if symbol 'b' is received, then head enters state q_2.

Transition table corresponding to the Turing machien is given below:

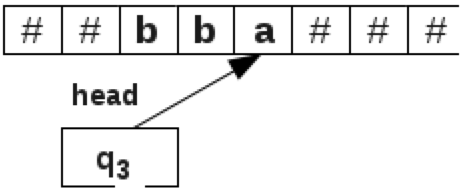
	Input	Symbol	
Current State	a	b	#
$\longrightarrow q_0$	$(q_1, \#, R)$	$(q_2, \#, R)$	—
q_1	(q_1, a, R)	(q_1, b, R)	$(q_3, \#, L)$
q_2	(q_2, a, R)	(q_2, b, R)	$(q_5, \#, L)$
q_3	$(q_4, \#, L)$	—	—
q_4	(q_4, a, L)	(q_4, b, L)	$(q_6, \#, R)$
q_5	—	$(q_4, \#, L)$	—
q_6	$(q_1, \#, R)$	$(q_2, \#, R)$	$(q_f, \#, R)$
$*q_f$	—	—	—

Let the input string is abaaba.

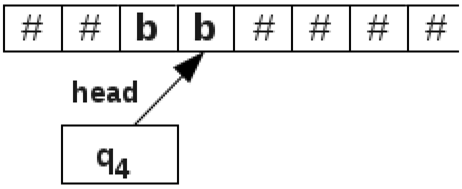
TM processes the string as follows:


1
a a b b a a # #
head
q_0
2
# a b b a a # #
head
q_1
3
# a b b a a # #
head
q_1
4
# a b b a a # #
head
q_1
5
# a b b a a # #
head
q_1
6
# a b b a a # #
head
q_1
7
# a b b a a # #
head
q_1
8
# a b b a a # #
head
q_3
9
# a b b a # # #
head
q_4
10
# a b b a # # #
head
q_4
11
# a b b a # # #
head
q_4
12
# a b b a # # #
head
q_4
13
# a b b a # # #
head
q_4
14
# a b b a # # #
head
q_6
15
# b b a # #
head
q_1
16
# b b a # #
head
q_1
17
# b b a # #
head
q_1
18
# b b a # #
head
q_1

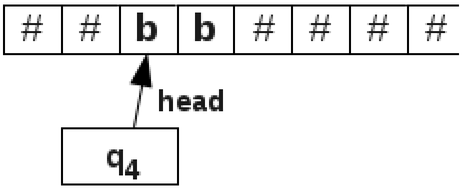
19



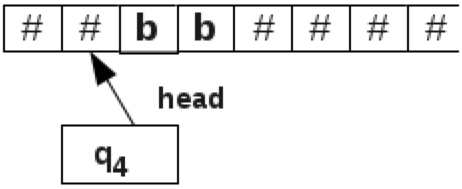
20



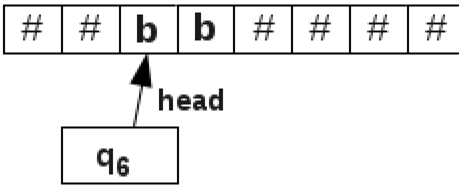
21



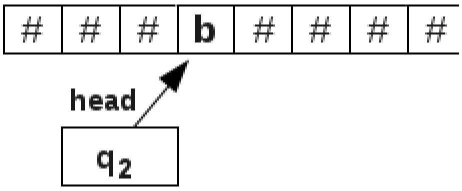
22



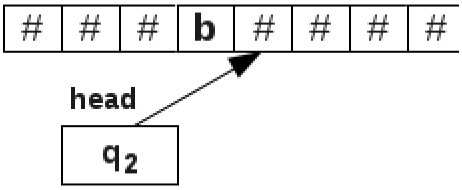
23



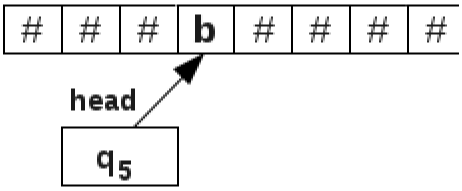
24



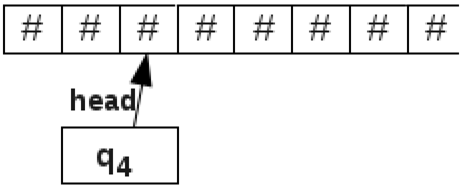
25



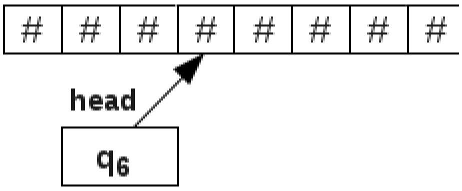
26



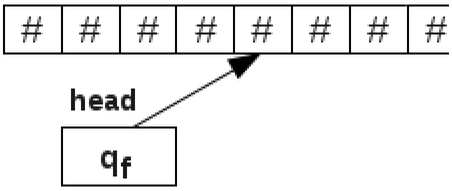
27



28



29



Example 5:

Design a Turing machine over $\{a, b, c\}$ to accept the language L={$a^nb^nc^n|n \geq 1$ }.

Example strings in this language are abc, aabbcc, aaabbbccc, aaaabbbbcccc, ...

String recognition process of the Turing machine is as follows:

1. Initially, TM is in state q_0 and head points to first symbol, a in the string. a is replaced with a #, changes state to q_1, and moves towards right.

2. In state q_1, the head continues to move towards right till all the a's are passed over and the first 'b' is obtained. Now 'b' is replaced with 'x', changes state to q_2, and again moves towards right.

3. In state q_2, the head continues to move towards right till all the b's are passed over and the first 'c' is obtained. Now 'c' is replaced with 'y', changes state to q_3, and turns back towards left.

4. In steps 1-3, each symbol 'a' is matched with the corresponding symbols 'b' and 'c'. In state q_3, head continues to move towards left and passes over the whole string to reach the blank symbol, #. At #, it turns right and changes state to q_4.

5. In state q_4, head reads symbol 'a', replaces it with a #, changes to state $q_1$and moves towards right.

6. Now steps 1-3 are repeated to match the symbols a, b and c. this process continues till the symbol 'x' is obtained in state q_4. This shows that all a's were replaced with #'s. Now if the string is valid, no 'b' should be available in state $q_{4.}$

7. In state q_4,head continues to move towards right till all the x's are passed over and no symbol, 'b' is obtained. If 'b' is found, it indicates a mismatch in the number of 'a' and 'b' symbols. TM halts in state q_4 to indicate the rejection of the string.

If no 'b' is encountered, and if symbol 'y' is obtained, then it changes state to q_5, and moves towards right.

8. In state q_5,head continues to move towards right till all the y's are passed over and no symbol, 'c' is obtained. If 'c' is found, it indicates a mismatch in the number of 'a' and 'c' symbols. TM halts in state q_5 to indicate the rejection of the string.

If no 'c' is encountered, and if symbol '#' is obtained, then it changes state to q_f, and moves towards left.

9. In state q_f, TM halts to indicate the acceptance of the string.

Transition table for the TM is shown below:

	Input Symbol					
Current State	a	b	c	x	y	#
$\longrightarrow q_0$	$(q_1, \#, R)$	—	—	—	—	—
q_1	(q_1, a, R)	(q_2, x, R)	—	(q_1, x, R)	—	—
q_2	—	(q_2, b, R)	(q_3, y, L)	—	(q_2, y, R)	—
q_3	(q_3, a, L)	(q_3, b, L)	—	(q_3, x, L)	(q_3, y, L)	$(q_4, \#, R)$
q_4	$(q_1, \#, R)$	—	—	(q_4, x, R)	(q_5, y, R)	—
q_5	—	—	—	—	(q_5, y, R)	$(q_f, \#, L)$
$*q_f$	—	—	—		—	—

Exercises:

1. Design a TM that accepts the language $L = \{0^n | n$ is a multiple of 3}.
2. Design a TM that accepts the language $L = \{a^n b^{2n} | n > 0\}$.
3. Design a TM over {a,b,c} to accept the language $L = \{wcw | w \in (a|b)^*\}$.
4. Design a TM over {a,b} to accept the language $L = \{ww | w \in (a|b)^*\}$.

Part III. TM as Transducers

4 TM as Transducers

In all the above TMs, TM either accepts or rejects a string. That is, TM acts as a language acceptor.

A TM can function as a transducer. That is, a string is given as input to TM, it produces some output.

We can view a Turing machine as a transducer.

Input to the computation is a set of symbols on the tape.

At the end of computation, whatever remains on the tape is the output.

A TM can be viewed as a transducer for the implementation of function f defined as,

$$\widehat{w} = f(w).$$

A function, f is said to be computable or Turing computable, if there exists a TM $M = (Q, \sum, \Gamma, \delta, q_0, \#, q_f)$ such that

$$q_0 w \overset{*}{\vdash} q_f \, f(w)$$

where w is in the domain of f.

We can see that all common mathematical functions are turing computable.

This means, basic operations such as addition, subtraction, multiplication, division can be performed on it.

This means that a TM is an abstract model of our modern computer system.

Example 1:

Design a Turing machine that computes the following function,

f(m, n) = m + n.

A positive integer on a Turing tape can be represented by an equal number of 0s.

For example,

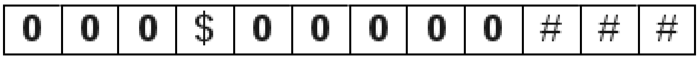
integer 5 is represented as 00000, and

integer 8 is represented as 00000000.

In the tape two integers are separated using the symbol, $.

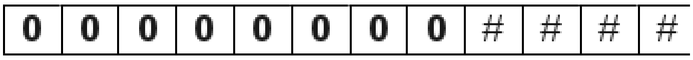
Let us assume that m=3 and n=5.

Then the input tape of the Turing machine will be,



0	0	0	$	0	0	0	0	0	#	#	#

After addition, tape will contain the results of addition as shown below:



0	0	0	0	0	0	0	0	#	#	#	#

TM for addition works as follows:

Head reads the 0s of the first number, m and reaches the separator symbol, $. Symbol, $ is replaced with 0 and head moves towards right.

It continues to move towards right till the second number, n is passed over and # is reached.

At #, it turns left and replaces rightmost 0 with #.

Now the tape contains the sum of m and n, and TM halts.

Conversion of the separator symbol, $ to 0, helps TM to make the single number on the tape. The conversion of symbol, 0 to # removes the additional 0 that was generated from the conversion of $ to 0.

Transition table for the TM is shown below:

	Input	Symbol	
Current State	0	$	#
$\longrightarrow q_0$	$(q_0, 0, R)$	$(q_1, 0, R)$	—
q_1	$(q_1, 0, R)$	—	$(q_2, \#, L)$
q_2	$(q_f, \#, L)$	—	—
$*q_f$	—	—	—

Above table is describes as follows:

1. Initially, TM is in state q_0. Head reads the first symbol, 0 and moves towards right without changing the state.

2. Head continues to move towards right till the separator symbol, $ is reached. It replaces $ with 0, changes state to q_1, and continues to move towards right.

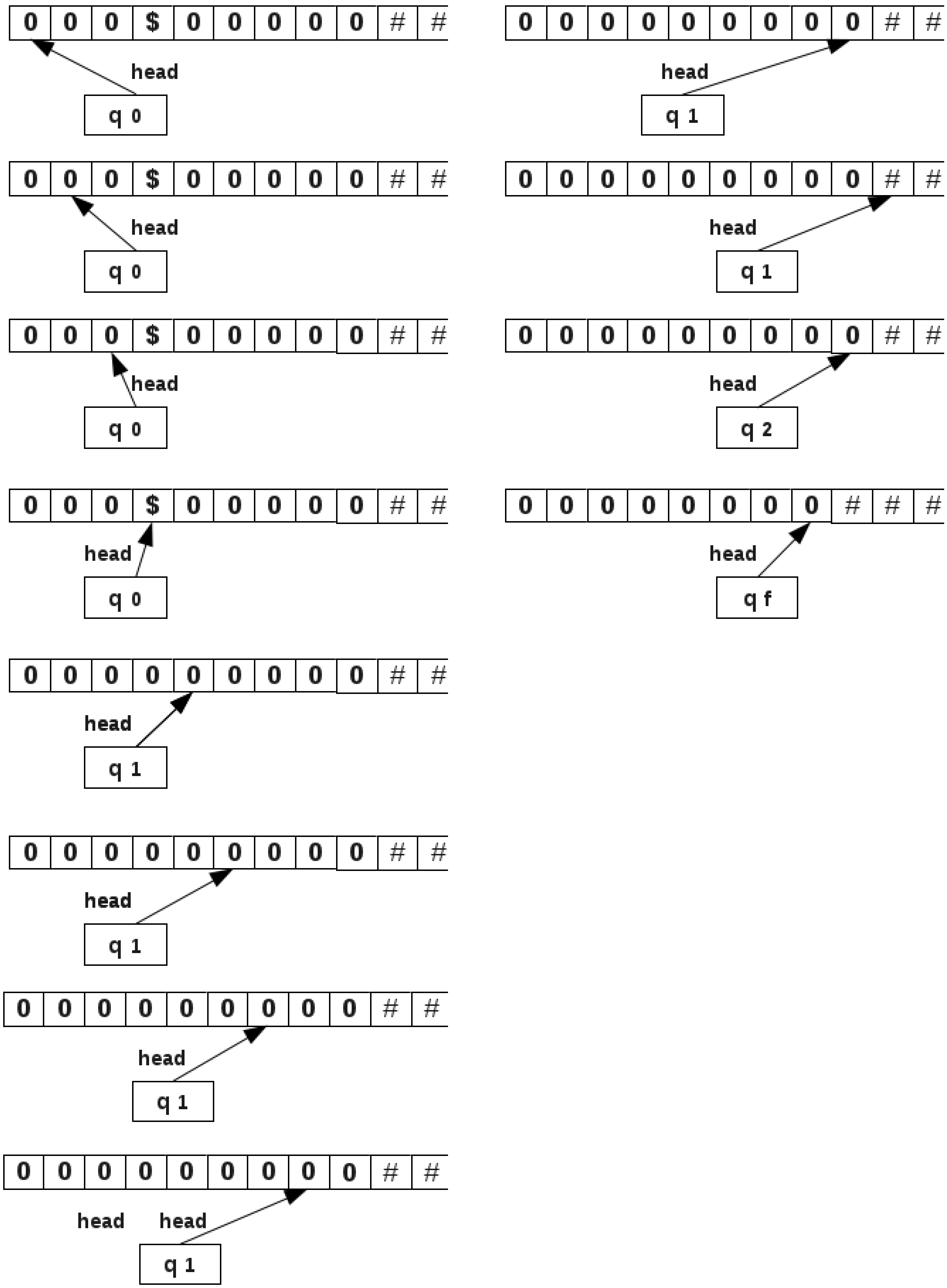
3. Head continues to move towards right and passes over the second number to reach #.

4. On reaching the #, head changes state to q_2 and turns left.

5. In state q_2, it replaces symbol 0 with #, and changes state to q_f.

6. In the final state, TM halts to indicate the completion of the process.

Addition of 3 and 5 is shown below:


0 0 0 $ 0 0 0 0 0 # #
head
q 0
0 0 0 $ 0 0 0 0 0 # #
head
q 0
0 0 0 $ 0 0 0 0 0 # #
head
q 0
0 0 0 $ 0 0 0 0 0 # #
head
q 0
0 0 0 0 0 0 0 0 0 # #
head
q 1
0 0 0 0 0 0 0 0 0 # #
head
q 1
0 0 0 0 0 0 0 0 0 # #
head
q 1
0 0 0 0 0 0 0 0 0 # #
head head
q 1
0 0 0 0 0 0 0 0 0 # #
head
q 1
0 0 0 0 0 0 0 0 0 # #
head
q 1
0 0 0 0 0 0 0 0 0 # #
head
q 2
0 0 0 0 0 0 0 0 # # #
head
q f

Example 2:

Design a Turing machine that can multiply two numbers.

f(m, n) = m x n.

Let m=2, n=4

Tape contains two integers, both containing a termination symbol $, at the end as shown below:

0	0	$	0	0	0	0	$	#	#	#	#

After the multiplication operation, tape contains the result as shown below:

0	0	$	0	0	0	0	$	0	0	0	0	0	0	0	0	#	#

TM for multiplication works as follows:

Leftmost 0 is replaced with symbol x and head moves towards right.

Head moves towards right till the remaining 0s are passed over and the separator symbol $ is reached. It crosses $ and reaches the leftmost 0 of the number n. It replaces this 0 with y and moves towards right.

Head moves towards right and reaches and crosses the second separator symbol, $.

After crossing $, it replaces # with 0.

Now head turns towards left and replaces second leftmost 0 of n to y.

Head then turns right and keeps moving towards right till a # is reached. It replaces # with 0 and turns towards left.

This process is repeated till all the 0's in the number n are converted to y and their copy is made after the second $.

This completes one cycle of copying.

After one cycle of copying, head turns left and converts all y's to 0. It moves towards left and converts the second leftmost 0 of m to x and makes a copy of n after the second $.

The cycle of copying n after the second $ is repeated m times and the number of such cycles is tracked with the help of the number of x symbols in the number m.

With m cycles of copying n after the second $, tape contains mxn numbers of 0s, which is the product.

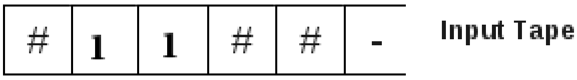
Transition table for the TM is shown below:

	Symbol				
Current State	0	$	x	y	#
$\longrightarrow q_0$	(q_1, x, R)	$(q_9, \$, L)$	—	—	—
q_1	$(q_1, 0, R)$	$(q_2, \$, R)$		—	—
q_2	(q_3, y, R)	$(q_7, \$, L)$	—	—	—
q_3	$(q_3, 0, R)$	$(q_4, \$, R)$	—	—	—
q_4	$(q_4, 0, R)$	—	—	—	$(q_5, 0, L)$
q_5	$(q_5, 0, L)$	$(q_6, \$, L)$	—	—	—
q_6	$(q_6, 0, L)$	—	—	(q_2, y, R)	—
q_7	—	$(q_8, \$, L)$	—	$(q_7, 0, L)$	—
q_8	$(q_8, 0, L)$	—	(q_0, x, R)	—	—
q_9	—	—	$(q_9, 0, L)$	—	$(q_f, \#, R)$
$*q_f$	—	—		—	—

Example 3:

Design a TM that copies strings of 1's.

For this problem, let the given string is 11 as shown below:

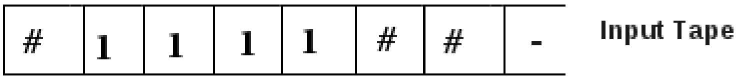


#	1	1	#	#	-

Input Tape

Note that we store a # before the string in the tape.

The output from TM should be as follows:



#	1	1	1	1	#	#	-

Input Tape

Following is the transition table for this TM:

	Input Symbol		
Current State	1	#	x
$\longrightarrow q_0$	(q_0, x, R)	$(q_1, \#, L)$	—
q_1	$(q_1, 1, L)$	$(q_3, \#, R)$	$(q_2, 1, R)$
q_2	$(q_2, 1, R)$	$(q_1, 1, L)$	—
$*q_3$	—	—	—

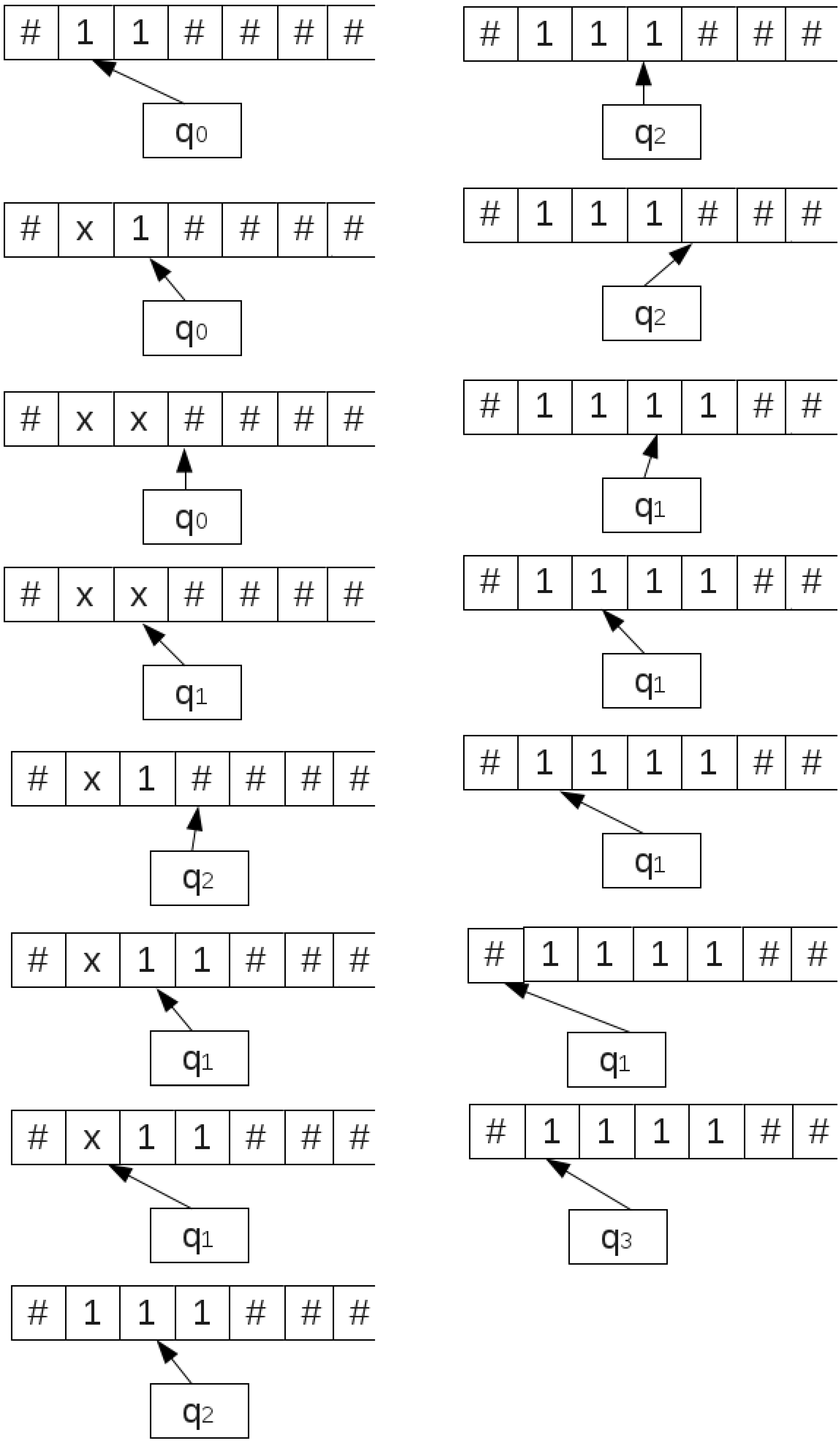
Here the TM replaces every 1 by symbol x. Then TM replaces rightmost x by 1. It goes to the right end of the string and writes a 1 there. Thus TM has added a 1 for the rightmost 1 in the input string. This process is repeated.

TM reaches q_1 after replacing all 1's by x's and reading the # symbol at the end of the input string. After replacing x by 1, TM reaches q_2. TM reaches q_3 at the end of the process and halts.

If the string is 11, we get 1111 at the end of computation.

If the string is 111, we get 111111 at the end of computation.

Consider the processing of the string 111:



# 1 1 # # # #
q_0
# x 1 # # # #
q_0
# x x # # # #
q_0
# x x # # # #
q_1
# x 1 # # # #
q_2
# x 1 1 # # #
q_1
# x 1 1 # # #
q_1
# 1 1 1 # # #
q_2
# 1 1 1 # # #
q_2
# 1 1 1 # # #
q_2
# 1 1 1 1 # #
q_1
# 1 1 1 1 # #
q_1
# 1 1 1 1 # #
q_1
# 1 1 1 1 # #
q_1
# 1 1 1 1 # #
q_3

TM as a Computer

Thus a TM can perform the basic operations such as addition, multiplication, subtraction and division. That means it can act as a computer. TM is a simple mathematical model of a computer. TM can do everything a computer can do. If TM cannot solve certain problems, then these problems are beyond the theoretical limits of computation.

Exercises:

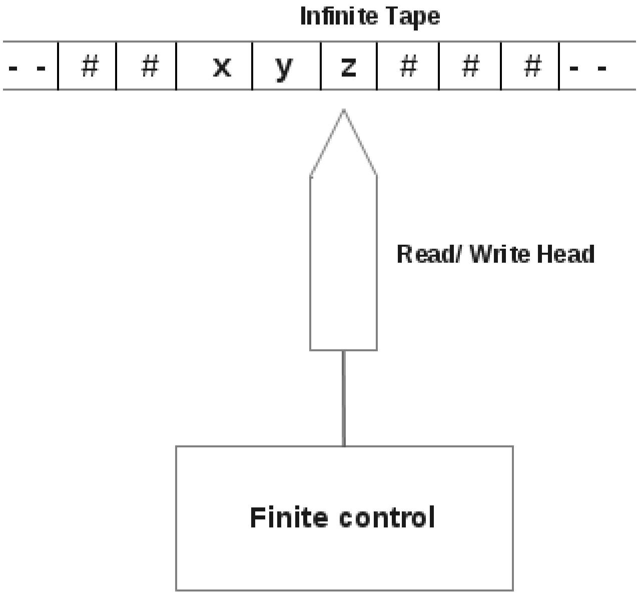
1. Design a Turing machine that computes m - n where m and n are positive integers and m>n.
2. Design a TM to divide m by 3 and to compute the quotient and the remainder.

Part IV. Types of TM

5 Two Way Infinite Turing Machines

The TM we discussed so far had a tape that was finite on the left end and infinite on the right end.

Two way infinite TM is an extension to this such that tape is infinite at both ends. This is shown below:



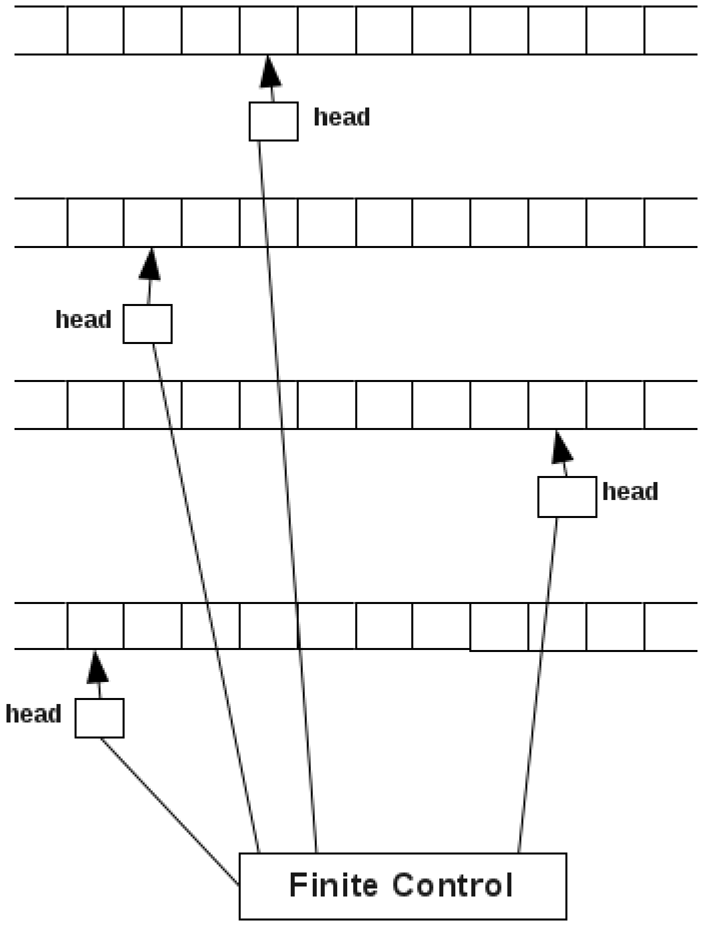
Thus on both sides of the tape, there is an infinite sequence of blank symbols (#).

This model does not provide any additional computational capability.

6 Multitape Turing Machines

A multitape TM consists of multiple tapes. Each tape has a separate head.

This is shown below:



There are n tapes, each divided into cells. The first tape holds the input string. Initially, all the other tapes hold the blank symbol, #.

A head has three possible options:

to move towards left (L),

to move towards right (R), or

to remain stationary (N).

All the heads are connected to a finite control. Finite control is in a state at an instant.

1. Initially, finite control is in state q_0.
2. Head in the lowermost tape points to the cell containing leftmost symbol of the input string.
3. All the cells in the upper tapes contain # symbol.

When a transition occurs,

1. Finite control may change its state.
2. Head reads the symbol from the current cell and writes a symbol on it.
3. Each head can move towards left (L), right (R) or stay stationary (N).

An example transition function for a 4 tape TM is given below:

$\delta(q_1, [a_1, a_2, a_3, a_4]) = (q_2, [b_1, b_2, b_3, b_4], [L, R, R, N])$

Here the finite control is in state q_1, It reads the symbol $a_1$from the uppermost tape, $a_2$from the next uppermost tape and so on.

After reading, finite control changes its state to q_2, and replaces the symbol $a_1$by b_1, $a_2$by b_2, $a_3$by b_3, $a_4$by b_4.

After this head in the upper most tape turns left. Head in the 2nd tape turns right. Head in the 3rd tape turns right. Head in the 4th tape remains stationary.

The advantage of using multi tape TM can be seen from the following example.

Example:

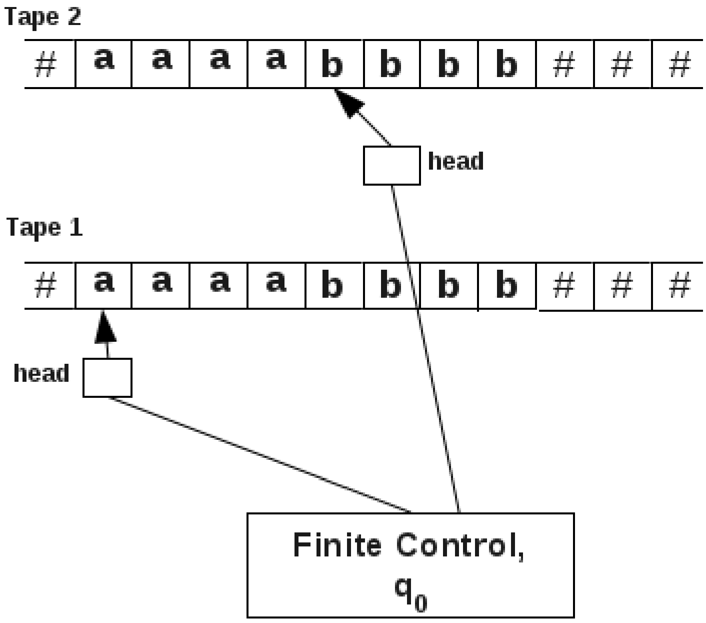
Consider the language $L = \{a^n b^n | n \geq 1\}$.

In a normal TM, head has to move back and forth to match each pair of symbols a and b. On a multitape TM, no such movements are needed.

This is done by making a copy of the input string in another tape.

Let the input string be aaaabbbb.

Let the input srtring be in tape 1. Make a copy of this string in tape 2.



The head of tape 1 is positioned on the first a of the input string. Head of tape 2 is positioned on the first b of the input string.

Now the heads advance on both tapes simultaneously towards right and the string is accepted if there are equal number of a's and b's in the string.

This will happen if the head on tape 1 encounters the first b and the head on tape 2 encounters the first # simultaneously.

7 Universal Turing Machines

In the previous sections, a separate TM was designed for each language.

For example, for the language, $L = \{a^n b^n | n \geq 1\}$, we designed a TM.

For the language, $L = \{a^n b^n c^n | n \geq 1\}$, we designed another TM and so on.

A Universal turing Machine (UTM) takes the code of a normal TM and a string w, and checks whether w is recognised by TM.

This is done as follows:

Consider a TM defined as,

$$TM = (Q, \sum, \Gamma, \delta, q_0, \#, q_f)$$

where

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

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

$\Gamma = \{a, b, x, y, \#\}$

δ is defined as,

$\delta(q_0, a) = (q_1, b, R)$

$\delta(q_0, x) = (q_1, y, L)$

$\delta(q_1, x) = (q_2, y, R)$

$\delta(q_2, y) = (q_f, \#, R)$

$\delta(q_f, x) = (q_0, y, L)$

The code for this Tm is to be made.

First we encode all the components of this TM using binary coding. In binary coding only symbols 0 and 1 are available.

Here 0 is used to code all the transition functions and 1 is used as a separator.

States of the TM are,

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

The states can be coded as,

$q_0 = 0$

$q_1 = 00$

$q_2 = 000$

$q_f = 0000$

Tape symbols are,

$\Gamma = \{a, b, x, y, \#\}$

Tape symbols can be coded as,

$a = 0$

$b = 00$

$x = 000$

$y = 0000$

$\# = 00000$

Direction of motion is coded as follows:

$L = 0$

$R = 00$

Now the transition functions are coded as follows:

$\delta(q_0, a) = (q_1, b, R) \quad \Longrightarrow 010100100100$

$$\delta(q_0, x) = (q_1, y, L) \qquad \Longrightarrow 010001001000010$$

$$\delta(q_1, x) = (q_2, y, R) \qquad \Longrightarrow 00100010001000010 0$$

$$\delta(q_2, y) = (q_f, \#, R) \qquad \Longrightarrow 0001000001000010000010 0$$

$$\delta(q_f, x) = (q_0, y, L) \qquad \Longrightarrow 00001000101000010$$

In the coding of TM, coding ends with 111. Transition functions are separated using 11.

Total coding of the TM involves coding of the transition functions.

Total coding of the TM is,

$0101001001001101000100100000101100100010001000001001100010000100001000000100110000100010100000101111$

Let w=abbb be the string to be checked on the Turing machine, M. The input to UTM will be,

$0101001001001101000100100000101100100010001000001001100010000100001000000100110000100010100000101111abbb$

UTM uses the binary code of the turing machine, M on string abbb and will check if abbb is recognised by M. If true UTM, will halt to say yes and if false UTM will stop to say no.

UTM and Modern Computer

Thus UTM is a generic machine that is capable of implementing the code of any arbitrary TM. This concept of UTM led to the modern day computer.

The concept of UTM is similar to the behaviour of modern day computer. Our modern computer system can choose a program for a problem and solve it on the required data.

Part V. Church’s Thesis

8 Church’s Thesis

It is also called Church- Turing thesis.

Church’s thesis states that

A function is said to be computable if it can be computed by a Turing machine.

This thesis talks about mechanical computation devices and the kind of computations they can perform.

Some more definitions of this thesis are given below:

1. Any mechanical computation can be performed by a Turing machine.

2. For every computational problem, there is a corresponding Turing machine.

3. Turing machines can be used to model any mechanical computer.

4. The set of languages that can be decided by a TM is the same as that which can be decided by any mechanical computing machine.

5. If there is no TM that decides problem P, there is no algorithm that can solve problem P.

Part VI. Godelization

9 Godelization

Godelization is an encoding technique which encodes a string as a number. This is called as Godel numbering.

Godel numbering is based on the concept that every positive integer can be factored into a unique set of prime factors. For example,

6 = 2 x 3

8 = 2 x 2 x 2

9 = 3 x 3

10 = 2 x 5

20 = 2 x 2 x 5

30 = 2 x 3 x 5

50 = 2 x 5 x 5

100 = 2 x 2 x 5 x 5

5,71,725 = 3 x 3 x 3 x 5 x 5 x 7 x 11 x 11

Also, it is possible to assign a serial number to each prime number, as

1 to 2

2 to 3

3 to 5

4 to 7

5 to 11, and so on.

Now the number,

5,71,725 = 3 x 3 x 3 x 5 x 5 x 7 x 11 x 11 = $2^0 \times 3^3 \times 5^2 \times 7^1 \times 11^2$

can be represented in the form of a sequence (0, 3, 2, 1, 2).

In terms of Godel numbering, we say that the Godel number associated with the sequence (0, 3, 2, 1, 2) is 5,71,725.

Thus the formal definition of Godel numbering is,

For any finite sequence $(x_0, x_1, x_2, ...x_n)$ the associated Godel number, G_n is,

$$G_n(x_0, x_1, x_2, ...x_n) = 2^{x_0}3^{x_1}5^{x_2}7^{x_3}.........P_n^{x_n},$$

where P_n is the n^{th} prime number.

The importance of Godel numbering lies in the fact that each Godel number encodes a unique sequence and a sequence encodes a unique Godel number.

This allows us to represent different configurations of a multi parameter dependent object in the form of a unique number.

A TM at any instant, can exist in different configurations, with each configuration being defined by the current state and position of the head. The position of the head can be defined by a cell number. It is possible to number each state and each cell, and to encode every configuration of a TM in the form of a Godel number. Also, it is possible to define every transition function of a TM in the form of a Godel number.

Part VII. Time Complexity of Turing Machine

10 Time Complexity of Turing Machine

A Turing machine takes a string w, it gives the output in the form yes or no. Here it is possible to exactly measure the number of moves made by the head of the TM during the processing of the string.

Although the number of moves made by the TM to process the string depend on the specific string, it is possible to make an estimate of the maximum possible number of moves that TM will make to accept or reject a string of length of n.

Time complexity of a Turing machine, M is written as,

$\tau_M(n)$

where τ is the time complexity,

n is a natural number.

Time complexity of a TM is the maximum number of moves made by the head to accept or reject a string of length n in the worst case scenario.

Example 1:

Consider the TM to accept the language, $L = \{ww^R | w \in (a, b)^+\}$

Find the time complexity of the TM.

Let the string is of length n.

Here the TM works as follows:

It checks the first symbol of the input string, converts it to #, and travels the whole of the string and reaches the # just following the string. It turns back and matches the last symbol of the string with the first one. If the match occurs, this last symbol is replaced with a #. Next the head travels back to the second symbol of the input string.

Here the number of moves required is 2n + 1.

On reaching the second symbol, it finds its corresponding match.

Here the number of moves required is 2n - 3.

When al lthe pairs are successfully matched, TM just makes one move and halts.

The worst case is one in which the input string is valid and the TM has to match all the corresponding pairs. Also, with every match, number of moves required to match the next symbol decreases. the process stops when all the symbols are successfully matched and no move is required. Now the TM halts.

Total number of moves made by the TM is,

(2n+1) + (2n-3) + (2n-7) + ... + 1

= ((2n+1) + 1) /2 x (n/2 +1)

= (n+1) x (n/2 +1)

=$n^2/2 + 3n/2 + 1$

Thus the time complexity of TM is $\mathcal{O}(n^2)$

$\tau(n) = \mathcal{O}(n^2)$

Example 2:

Consider the TM to accept the language, $L = \{a^n b^n | n \geq 1\}$

Find the time complexity of the TM.

Here head replaces leftmost 0 by x and leftmost 1 by y. Then head reaches back to the second symbol. Here 2n moves are needed.

Thus at most 2n moves are needed to match a 0 with a 1.

For matching all 0s with 1s, this process is to be repeated n/2 times.

Then the total number of moves needed is around 2n x n/2. That is $O(n^2)$

Time complexity of this TM is,

$$\tau(n) = \mathcal{O}(n^2)$$

Part VIII. Halting Problem of TM

Here we use a process called reduction.

Let A is the problem of finding some root of $x^4 - 3x^2 + 2 = 0$, and

B is the problem of finding some root of $x^2 - 2 = 0$.

Here $x^2 - 2$ is a factor of $x^4 - 3x^2 + 2 = 0$.

Thus a root of $x^2 - 2 = 0$ will also be a root of $x^4 - 3x^2 + 2 = 0$.

Then we can say that A is reducible to B.

Thus a problem A is reducible to problem B if a solution to problem B can be used to solve problem A.

Then if A is reducible to B and B is decidable, then A is decidable.

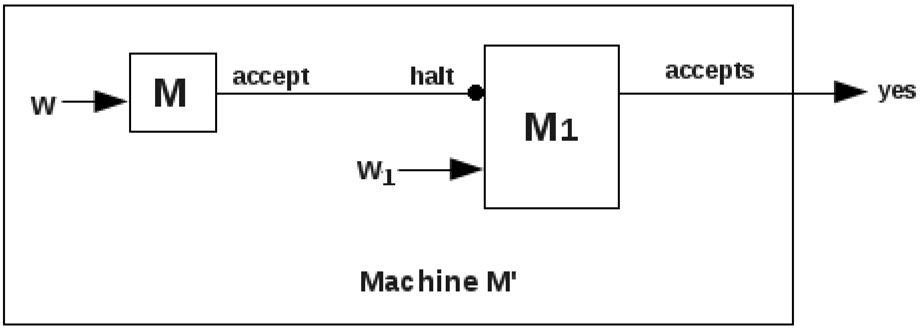
If A is reducible to B and A is undecidable, then B is undecidable.

11 Halting Problem of TM

This problem states that halting problem of a Turing machine is undecidable.

Proof

To show this, we reduce problem of halting to a problem of acceptance. Consider the following TM,



Here we take an instance (M, w) and construct another instance (M_1, w_1).

It is taken such that M_1 halts on input w_1, iff M accepts w.

The machine M' stops when M_1 halts.

Initially, the string w is fed to the TM, M and w_1 is fed to the TM, M_1.

If M accepts w, then it sends a halt signal to M_1. Then TM, M_1 halts on input w_1.

If M rejects w, then M_1 does not halt on w_1.

Thus halting of M_1 depends on the acceptance behaviour of M. Acceptance behaviour of a TM is undecidable, halting of M_1 or M' is undecidable.

Part IX. Rice Theorem

Recursive and Non-Recursive Languages

Recursive Languages

A language, L is recursive, if

it is possible to design a TM that halts in the final state to say yes if w∈L, and

halts in the non-final state to say no if w $\notin$ L.

Recursively Enumerable Languages

A language, L is recursively enumerable, if

it is possible to design a TM that halts in the final state to say yes if w∈L, and

halting cannot be guranteed if w $\notin$ L.

Languages that are neither Recursive nor Recursively Enumerable

A language, L is neither recursive nor recursively enumerable, if

the structure of the language is such that no TM which recognises w can be designed.

12 Rice Theorem

Rice's theorem states that

every non-trivial property of a recursively enumerable language is undecidable.

Proof

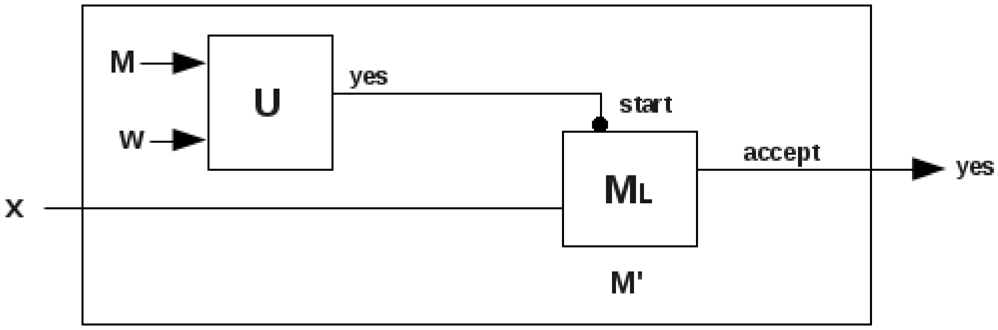
A non-trivial property is one that is possessed by some objects of a class, but not all.

For example, being a mathematician is a property that is possessed by some humans but not by all.

Some cats are black but not all. So black colour property cannot be trivially associated with cats.

Let χ be a non-trivial property that is not possessed by all recursively enumerable languages. This problem can be reduced to one consisting of a pair (M, w) such that L possesses χ iff $w \in L(M)$. We take a UTM U that takes a pair (M, w); its output is yes iff χ is possessed by L.

Since L is a recursively enumerable language, there must be a TM, M_L that accepts L. Let x be a string belonging to L. Now, we design a machine M' to decide χ as shown below:



Here U is a UTM.

UTM, U takes the pair (M, w) and checks if $w \in L(M)$. If the output is yes, then the machine $M_L$that accepts the string x starts and the output of the machine M' is yes. Thus the decidability of the problem of possessing the trivial property reduces to the problem of L_u. If the pair $(M, w) \in L_u$, then L possesses χ ; otherwise not. Since L_u is not recursive, possession of χ by L is also not decidable.

Part X. Post Correspondence Problem

13 Post Correspondence Problem

Post correspondence problem (PCP) is a problem formulated by E. Post in 1940.

Let us illustrate this problem by an example:

Example 1:

Let there be two series of strings, say x series and y series as shown below:

i	x_i	y_i
1	10	101
2	01	100
3	0	10
4	100	0
5	1	010

Both X series and Y series contains 5 strings.

Let us call the strings in X series as X substrings, and

the strings in Y series as Y substrings.

If we concatenate X substrings $x_1x_5x_2x_3x_4x_4x_3x_4$, we get 1010101001000100.

If we concatenate Y substrings $y_1y_5y_2y_3y_4y_4y_3y_4,$ we get 1010101001000100.

Let us call these strings as x string and y string.

It can be seen that x string and y string are same.

Here we say that this instance of PCP has a solution in the form 15234434.

If we take 23,

x string is, x_2x_3 =010

y string is, y_2y_3 =10010.

Here x string and ys tring are different. Hence 23 is not a solution to this instance of PCP.

Example 2:

Find the solution to the instance of PCP given in the following table.

i	x_i	y_i
1	0	000
2	01000	01
3	01	1

Solution,

$x_2x_1x_1x_3$= 010000001

$y_2y_1y_1y_3$= 010000001

Hence, 2113 is a solution to this instance of PCP.

Example 3:

Find why the instance of PCP given below cannot have a solution.

i	x_i	y_i
1	0	000
2	010	0100
3	01	100
4	11	110

It can be seen that for every pair, $|x_i| > |y_i|$.

So, in whatever way we concatenate the string, the length of the x string will be longer than the corresponding y string. Thsu there is no solution for this instance of PCP.

Definition of PCP

Let there be two series, x series and y series of size n with same charactere set $\sum$ with their i^{th} element as x_i and y_i, respectively; does there exist a solution tha tforms the same x series and y series?

The generic solution of PCP can be written as,

$$x_{i_1}x_{i_2}x_{i_3}........x_{i_k} = y_{i_1}y_{i_2}y_{i_3}.......y_{i_k}$$

Post Correspondence Problem is unsolvable.

This means it is a non computable function. No turing machine exists for PCP.

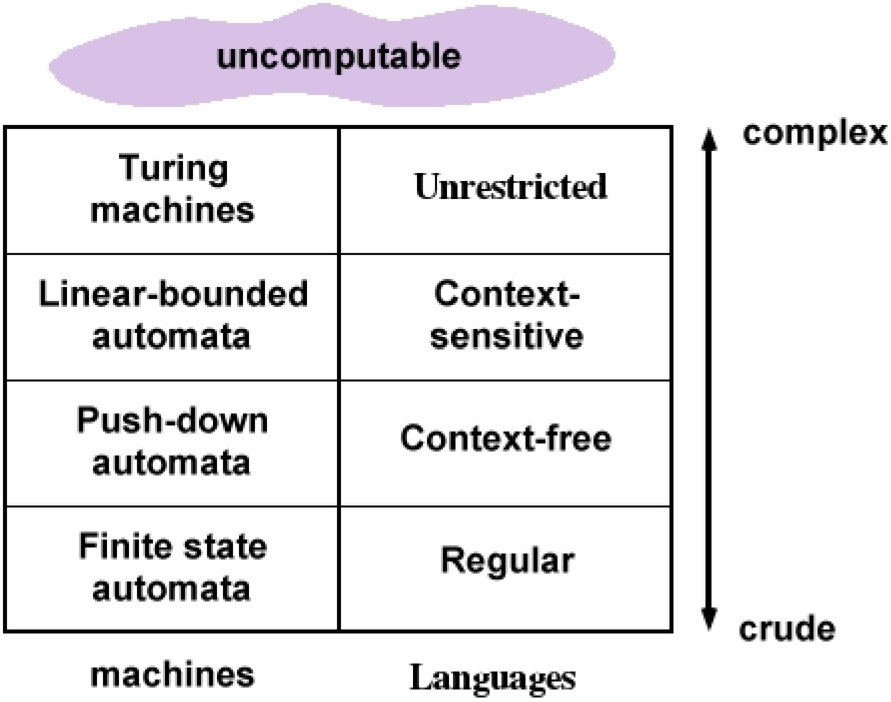
The proof of this is beyond the scope of our study.

Use of PCP

Other problems can be reduced to PCP and we can declare them as unsolvable or undecidable.

Part XI. Linear Bounded Automata

Consider the following figure:



machines	Languages
Turing machines	Unrestricted
Linear-bounded automata	Context-sensitive
Push-down automata	Context-free
Finite state automata	Regular

We learned in Chomsky classification that context sensitive languages (Type-1) are produced from context sensitive grammars.

Context sensitive languages are recognised using linear bounded automata.

The following is an example for a Context Sensitive grammar:

S ⟶ aBCT|aBC

T ⟶ ABCT|ABC

BA ⟶ AB

CA ⟶ AC

CB ⟶ BC

aA ⟶ aa

aB ⟶ ab

bB ⟶ bb

bC ⟶ bc

cC ⟶ cc

in this grammar, LHS of a production is not longer than the RHS.

The language, $L = \{a^n b^n c^n | n > 0\}$ is a context sensitive language.

14 Linear Bounded Automata (LBA)

Context sensitive languages are recognised using linear bounded automata (LBA).

Here input tape is restricted in size. A linear function is used for restricting the length of the input tape.

Many compiler languages lie between context sensitive and context free languages.

A linear bounded automation is a non deterministic Turing machine which has a single tape whose length is not infinite but bounded by a linear function.

Formal Definition

A linear bounded automation is defined as,

$M = (Q, \sum, \Gamma, \delta, q_0, \#,$ <,>,F),

where

Q is a set of states,

$\sum$ is a set of input symbols,

Γ is a set of tape symbols, including #,

δ is the set of transition functions from

$$(Q \times \Gamma) \longrightarrow (Q \times \Gamma \times \{L, R, N\}),$$

q_0 is the start state,

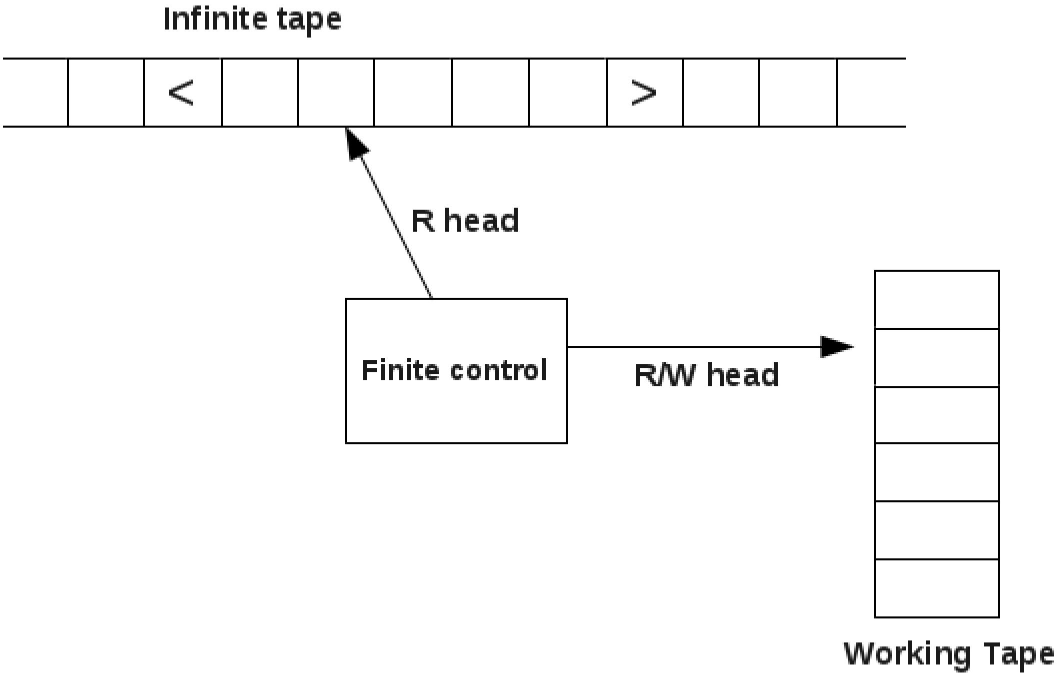
$\#$ is the blank symbol,

< is the left end marker in the input tape,

$>$ is the right end marker in the input tape,

F is the final state.

Following diagram shows a linear bounded automation,



When an input string is processed using an LBA, input string is enclosed between < and > end markers.

The end marker < prevents the R head from getting off the left end of the tape. The right end marker > prevents the R head from getting off the right end of the tape.

On the input tape, head does not write. Also on the input tape, head does not move left. All the computation is to be done between end markers < and >.

On the working tape head can read and write without any restriction.

Thus LBA is same as a Turing machine except that head can move only within the end markers.

Example:

Consider an LBA defined as,

Q= $\{q_0, q_1, q_2, q_3, q_4\}$

$\sum$= {a, b, c}

Γ= {a, b, c, #}

q_0= s

$\delta(q_0$, [) = (q_1, [, R) $\delta(q_1$,]) = (q_1,],Y) $\delta(q_1$, #) = (q_1, #, R)

$\delta(q_1$, a) = (q_2, #, R) $\delta(q_2$, a) = (q_2, a, R) $\delta(q_2$, #) = (q_2, #, R)

$\delta(q_2$, b) = (q_3, #, R) $\delta(q_3$, b) = (q_3, b, R) $\delta(q_3$, #) = (q_3, #, R)

$\delta(q_3$, c) = (q_4, #, L) $\delta(q_4$, c) = (q_4, c, L) $\delta(q_4$, b) = (q_4, b, L)

$\delta(q_4$, a) = (q_4, a, L) $\delta(q_4$, #) = (q_4, #, L) $\delta(q_4$, [) = (q_1, [, R)

In the above, the transition $\delta(q_1, a) = (q_2, \#, R)$ means,

LBA on state q_3, head points to symbol b, remains in state q_3, replaces b with b and head turns towards right by one cell.

Questions (from Old syllabus of S7CS TOC)

MGU/Nov2011

1. Explain the instantaneous description of a Turing machine (4marks).

2. What is a universal turing machine (4marks)?

3a. Design a Turing machine that computes a function f(m,n)=m+n in addition of two integers.

OR

b. Explain the halting problem of Turing machine. Prove that it is undecidable (12marks).

MGU/April2011

1. What are the languages accepted by a Turing machine (4marks)?

2. Define Godelization (4marks).

3a.

OR

b. Design a Turing machine with the initial tape as 01110111110... and the output pattern 01111101110... (12marks).

4a. Show that the halting problem of a Turing machine is undecidable. Explain Godelization with example.

OR

b. Design a m-tape Turing machine that works as a copying machine (12marks).

MGU/Nov2010

1. What is meant by halting of a TM (4marks).

2a. Design a Turing machine to compute a function f where, $f : \sum_0^* \longrightarrow \sum_0^*, \sum_0 = \sum_1 = \{a, b\}, f(w) = \overline{w}, \overline{w}\, si$ the result of replacing an occurrence of a in wbyb and vice versa.

OR

b. Explain Church's thesis and its application. Also explain Godelization (12marks).

3a. Show that halting problem of a Turing machine is not NP-complete (12marks).

OR

b.

MGU/May2010

1a. What is configuration of a Turing machine ?

b. When do we say that a function is Turing computable (4marks)?

2. Explain Church's thesis (4marks).

3a.

Or

b. i. Construct a Turing machine to do the multiplication.

ii. Design a Turing machine to compute n mod 2. (12marks).

MGU/Nov2009

1. a.

b. State Post correspondence problem (4marks).

2. Describe the method of Godelization (4marks).

3a.

OR

b. Prove the equivalence of two way infinite tape with standard turing machine (12marks).

4a.

OR

b. i. Describe the action of a turing machine.

ii. Explain briefly how to enumerate all possible turing machines computations, so that a given computation can be characterised by a single natural number code C. (12marks).

5a. Show that it is not possible to compute the maximum distance travelled by the turing machine head from its initial position during halting computations as a function of code C. Any results that you use should be stated clearly (12marks).

Or

b.

MGU/Nov2008

1. What is Turing machine (4marks)?

2a. Design a Turing machine that recognise the language $L = \{ww^R | w$ is in $(a + b)^*\}$.

OR

b. What do you mean by a universal turing machine? Explain its applications (12marks).

MGU/May2008

1. Explain Church's thesis (4marks).

2. What is multi-head turing machine (4marks)?

3a. Construct a turing machine that accepts the language given by $\{ww^R | w$ is in (0+1)}.

OR

b. Explain universal turing machine and explain its applications (12marks).

MGU/Dec2007

1. What is halting problem of turing machine (4marks)?

2. What is turing computability (4marks)?

3a. Design a turing machine that recognises the language {w| w is in $(a + b)^*\}$.

OR

b. What is church's thesis and Godelization (12marks)?

MGU/July2007

1. Define turing machine (4marks).

2. Briefly explain church's thesis (4marks).

3a. i. Construct a turing machine that increments a binary number.

ii. Write short notes on any two variants of turing machines (12marks).

OR

b. i. Construct a Turing machine that decrements a binary number.

ii. Construct a turing machine that adds two unary numbers (12marks).

4a. State turing machine halting problem. Show that it is undecidable (12marks).

OR

b.

MGU/Jan2007

1. What is a universal turing machine (4marks)?

2. Construct a turing machine which computes the function f(n)=n+2 over unary numbers (4marks).

3a. i. Construct a turing machine that decides the language $L = \{a^n b^n | n \geq 0\}$.

ii. Construct a turing machine that shifts the input string one position to the left.

OR

b. i.

ii. Construct a turing machine that accepts the language $L = \{w \in (a, b)^* | w$ has equal number of a's and b's} (12marks).

MGU/July2006

1. Write short notes on halting problem of turing machine (4marks).

2. Explain church's hypothesis (4marks).

3a. Design a turing machine to recognise the language $L = \{0^n 1^n 0^n | n \geq 1\}$.

OR

b. i. Construct a turing machine that will compute f(x,y)=x+y.

ii. Write a note on universal turing machines (12marks).

MGU/Nov2005

1. What is Church's hypothesis? Explain (4marks).

2. Define a turing machine (4marks).

3a. Prove that a language L is recognised by a Turing machine with a two way infinite tape iff it is recognised by a turing machine with a one way infinite tape.

OR

b. Design a turing machine M to recognise the language $L = \{ww^R | w$ is in $(a + b)^*$ (12marks).

References

.

Nagpal, C, K (2011). Formal Languages and Automata Theory. Oxford University Press.

Pandey, A, K (2006). An Introduction to automata Theory and Formal Languages. Kataria & Sons.

Mishra, K, L, P; Chandrasekaran, N (2009). Theory of Computer Science. PHI.

Linz, P (2006). An Introduction to Formal Languages and Automata. Narosa.

Hopcroft, J, E; Motwani, J; Ullman, J, D (2002). Introduction to Automata Theory, Languages and Computation. Pearson Education.

website: http://sites.google.com/site/sjcetcssz