Introduction to Formal Languages, Automata and Computability

Turing Machine

K. Krithivasan and R. Rama

Introduction

The concept of Turing machine is particularly helpful in offering a clear and realistic demarcation of what constitutes a single step of execution. The problem of how to separate the steps from one another in a step-by-step procedure is clearly specified when it is put into the form of a Turing machine table.

To start with, we may specify an effective procedure as follows :

An effective procedure is a set of rules which tell us, from moment to moment, precisely how to behave. With this in mind, Turing's thesis may be stated as follows: Any process which could naturally be called an effective procedure can be realised by a Turing machine.

contd.

One cannot expect to prove Turing's thesis, since the term 'naturally' relates rather to human dispositions than to any precisely defined quality of a process. Support must come from intuitive arguments. Hence it is called Turing's hypothesis or Church-Turing thesis and not as Turing's theorem.
Till today Turing machine is taken as the model of computation. Whenever a new model of computation (like DNA computing, membrane computing) is defined, it is the practice to show that this new model is as capable as the Turing machine. This proves the power of the new model of computation.

Turing Machine as an Acceptor

The Turing machine can be considered as an accepting device accepting sets of strings. Later we shall see that Turing machines accept the family of languages generated by type 0 grammars. The set accepted by a Turing machine is called a recursively enumerable set. When we consider the Turing machine as an accepting device, we usually consider a one way infinite tape. The Turing machine consists of a one way infinite read-write tape and a finite control.

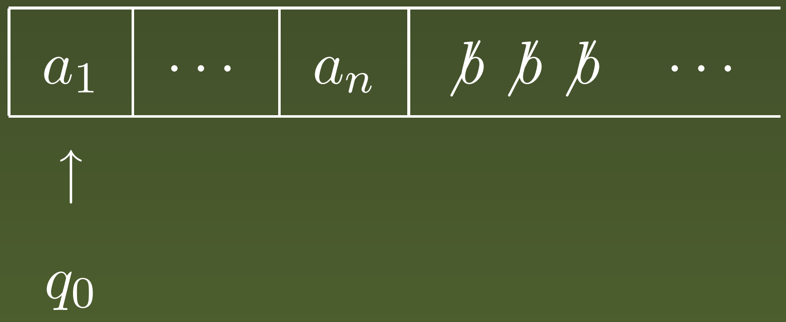


Figure 1: Initial configuration

contd.

The input $a_1 \ldots a_n$ is placed at the left end of the tape. The rest of the cells contain the blank symbol $\not b$. Initially the tape head points to the leftmost cell in the initial state q_0. At any time the tape head will point to a cell and the machine will be in a particular state. Suppose the machine is in state q and pointing to a cell containing the symbol a, then depending upon the δ mapping (transition function) of the TM it will change state to p and write a symbol X replacing a and move its tape head one cell to the left or to the right. The Turing machine is not allowed to move off the left end of the tape. When it reaches a final state it accepts the input. Now we consider the formal definition.

Definition

A Turing machine (TM) $M = (K, \Sigma, \Gamma, \delta, q_0, F)$ is a 6-tuple, where

- K is a finite set of states
- Σ is a finite set of input symbols
- Γ is a finite set of tape symbols, $\Sigma \subseteq \Gamma$, $\not{b} \in \Gamma$ is the blank symbol
- q_0 in K is the initial state
- $F \subseteq K$ is the set of final states
- δ is a mapping from $K \times \Gamma$ into $K \times \Gamma \times \{L, R\}$.

contd.

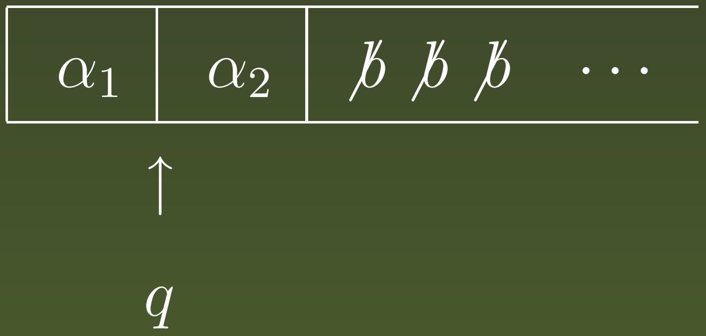
Note

(a) Turing machine mapping is defined in such a way that it is deterministic. One can define nondeterministic TM. Though they have the same power as far as accepting power is concerned, the number of steps may exponentially increase if a deterministic TM tries to simulate a nondeterministic TM.

(b) In some formulations the head remaining stationary is allowed. i.e., $\delta : K \times \Gamma \rightarrow K \times \Gamma \times \{L, S, R\}$. But we shall stick to $\{L, R\}$ as remaining stationary can be achieved by two moves, first moving right and then moving back left.

contd.

An Instantaneous Description (ID) of a Turing machine is a string of the form $\alpha_1 q \alpha_2, \alpha_1, \alpha_2 \in \Gamma^*, q \in K$.
This means that at that particular instance $\alpha_1\alpha_2$ is the content of the tape of the Turing machine. q is the current state and the tape head points to the first symbol of α_2. See



The relationship between IDs can be described as follows:

If $X_1 \ldots X_{i-1} q X_i X_{i+1} \ldots X_m$ is an ID and

contd.

$\delta(q, X_i) = (p, Y, R)$ then the next ID will be

$X_1 \ldots X_{i-1} Y p X_{i+1} \ldots X_m$.
If $\delta(q, X_j) = (p, Y, L)$ then the next ID will be
$X_1 \ldots X_{i-2} p X_{i-1} Y X_{i+1} \ldots X_m$. We denote this as

$$X_1 \ldots X_{i-1} q X_i X_{i+1} \ldots X_m \vdash X_1 \ldots X_{i-2} p X_{i-1} Y X_{i+1} \ldots X_m.$$

$q_0 X_1 \ldots X_m$ is the initial ID. Initially the tape head points to the leftmost cell containing the input. If $q X_1 \ldots X_m$ is an ID and $\delta(q, X_1) = (p, Y, L)$, machine halts. i.e., moving off the left end of the tape is not allowed. If $X_1 \ldots X_m q$ is an ID, q is reading the leftmost blank symbol. If $\delta(q, \not{b}) = (p, Y, R)$ next ID will be $X_1 \ldots X_m Y p$.

contd.

If $\delta(q, \not{b}) = (p, Y, L)$ next ID will be $X_1 \ldots X_{m-1} p X_m Y$. $\overset{*}{\vdash}$ is the reflexive, transitive closure of $\vdash$. i.e., $ID_0 \vdash ID_1 \vdash \cdots \vdash ID_n$ is denoted as $ID_0 \overset{*}{\vdash} ID_n$, $n \geq 0$. An input will be accepted if the TM reaches a final state.

Definition A string w is accepted by the TM, $M = (K, \Sigma, \Gamma, \delta, q_0, F)$ if $q_0 w \overset{*}{\vdash} \alpha_1 q_f \alpha_2$ for some $\alpha_1, \alpha_2 \in \Gamma^*$, $q_f \in F$. The language accepted by the TM M is denoted as

$$T(M) = \{w | w \in \Sigma^*, q_0 w \overset{*}{\vdash} \alpha_1 q_f \alpha_2 \text{ for some } \alpha_1, \alpha_2 \in \Gamma^*, q_f \in F\}$$

Note

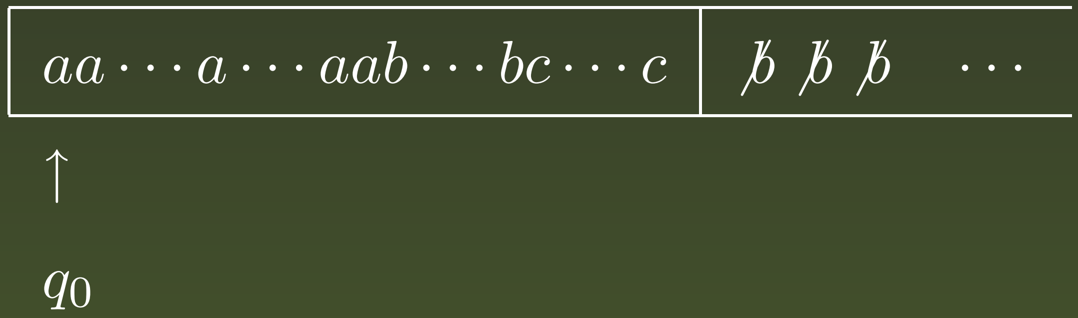
(a) It should be noted that by definition, it is not necessary for the TM to read the whole input. If $w_1 w_2$ is the input and the TM reaches a final state after reading w_1, $w_1 w_2$ will be accepted; for that matter any string $w_1 w_j$ will be accepted. Usually while constructing a TM we make sure that the whole of the input is read.

(b) Usually we assume that after going to a final state, the TM halts, i.e., it makes no more moves.

(c) A string w will not be accepted by the TM, if it reaches an ID $\eta_1 r \eta_2$ from which it cannot make a next move; $\eta_1 \eta_2 \in \Gamma^*$, $r \in K$ and r is not a final state or while reading w, the TM gets into a loop and is never able to halt.

Example

Let us consider a TM for accepting $\{a^i b^j c^k | i, j, k \geq 1, i = j + k\}$. The informal description of the TM is as follows. Consider the figure which shows the initial ID.


$aa \cdots a \cdots aab \cdots bc \cdots c$
$\not b \; \not b \; \not b \quad \cdots$
$\uparrow$
q_0

The machine starts reading a 'a' and changing it to a X; it moves right; when it sees a 'b', it converts it into a Y and then starts moving left. It matches a's and b's. After that, it matches a's with c's. The machine accepts when the

contd.

number of a's is equal to the sum of the number of b's and the number of c's.

Formally $M = (K, \Sigma, \Gamma, \delta, q_0, F)$

$K = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8\}$ $\quad F = \{q_8\}$

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

$\Gamma = \{a, b, c, X, Y, Z, \not{b}\}$

δ is defined as follows:

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

In state q_0, it reads a ‘a’ and changes it to X and moves right in q_1.

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

In state q_1 it moves right through the ‘a’s.

contd.

$\delta(q_1, b) = (q_2, Y, L)$
When it sees a 'b' it changes it into a Y.
$\delta(q_2, a) = (q_2, a, L)$
$\delta(q_2, Y) = (q_2, Y, L)$
In state q_2 it moves left through the 'a's and Ys.
$\delta(q_2, X) = (q_0, X, R)$
When it sees a X it moves right in q_0 and the process repeats.
$\delta(q_1, Y) = (q_3, Y, R)$
$\delta(q_3, Y) = (q_3, Y, R)$
$\delta(q_3, b) = (q_2, Y, L)$

After scanning the 'a's it moves through the Ys still it

contd.

sees a ‘b’, then it converts it into a Y and moves left.
$\delta(q_3, c) = (q_4, Z, L)$
When no more ‘b’s remain it sees a ‘c’ in state q_3, changes that it into Z and starts moving left in state q_4. The process repeats. After matching ‘a’s and ‘b’s, the TM tries to match ‘a’s and ‘c’s.
$\delta(q_4, Y) = (q_4, Y, L)$
$\delta(q_4, a) = (q_4, a, L)$
$\delta(q_4, X) = (q_0, X, R)$
$\delta(q_3, Z) = (q_5, Z, R)$
$\delta(q_5, c) = (q_4, Z, L)$
$\delta(q_5, Z) = (q_5, Z, R)$
$\delta(q_4, Z) = (q_4, Z, L)$

contd.

When no more ‘a’s remain it sees a Y in state q_0 checks that all ‘b’s and ‘c’s have been matched and reaches the final state q_8.

$\delta(q_0, Y) = (q_6, Y, R)$
$\delta(q_6, Y) = (q_6, Y, R)$
$\delta(q_6, Z) = (q_7, Z, R)$
$\delta(q_7, Z) = (q_7, Z, R)$
$\delta(q_7, \not{b}) = (q_8, \not{b}, halt)$

contd.

Let us see how a string *aaabbcc* will be rejected. Let us trace the sequence of IDs on *aaabbcc*

$$\begin{aligned}
q_0aaabbcc &\vdash Xq_1aabbcc \vdash Xaq_1abbcc \vdash Xaaq_1bbcc \vdash Xaq_2aYbcc \\
&\vdash Xq_2aaYbcc \vdash q_2XaaYbcc \vdash Xq_0aaYbcc \vdash XXq_1aYbcc \\
&\vdash XXaq_1Ybcc \vdash XXaYq_3bcc \vdash XXaYYcc \vdash XXq_2aYYcc \\
&\vdash Xq_2XaYYcc \vdash XXq_0aYYcc \vdash XXXq_1YYcc \vdash XXXYq_3Ycc \\
&\vdash XXXYYq_3cc \vdash XXXYq_4YZc \vdash XXXq_4YYZc \vdash XXq_4XYYZc \\
&\vdash XXXq_0YYZc \vdash XXXYq_6YZc \vdash XXXYYq_6Zc \vdash XXXYYZq_7c
\end{aligned}$$

contd.

The machine halts without accepting as there is no move for (q_7, c).
Let us see the sequence of moves for $aaaabc$

$$
\begin{aligned}
q_0aaaabc &\vdash Xq_1aaabc \vdash Xaq_1aabc \vdash Xaaq_1abc \vdash Xaaaq_1bc \\
&\vdash Xaaq_2aYc \vdash Xaq_2aaYc \vdash Xq_2aaaYc \vdash q_2XaaaYc \\
&\vdash Xq_0aaaYc \vdash XXq_1aaYc \vdash XXaq_1aYc \vdash XXaaq_1Yc \\
&\vdash XXaaYq_3c \vdash XXaaq_4YZ \vdash XXaq_4aYZ \vdash XXq_4aaYZ \\
&\vdash Xq_4XaaYZ \vdash XXq_0aaYZ \vdash XXXq_1aYZ \vdash XXXaq_1YZ \\
&\vdash XXXaYq_3Z \vdash XXXaYZq_5
\end{aligned}
$$

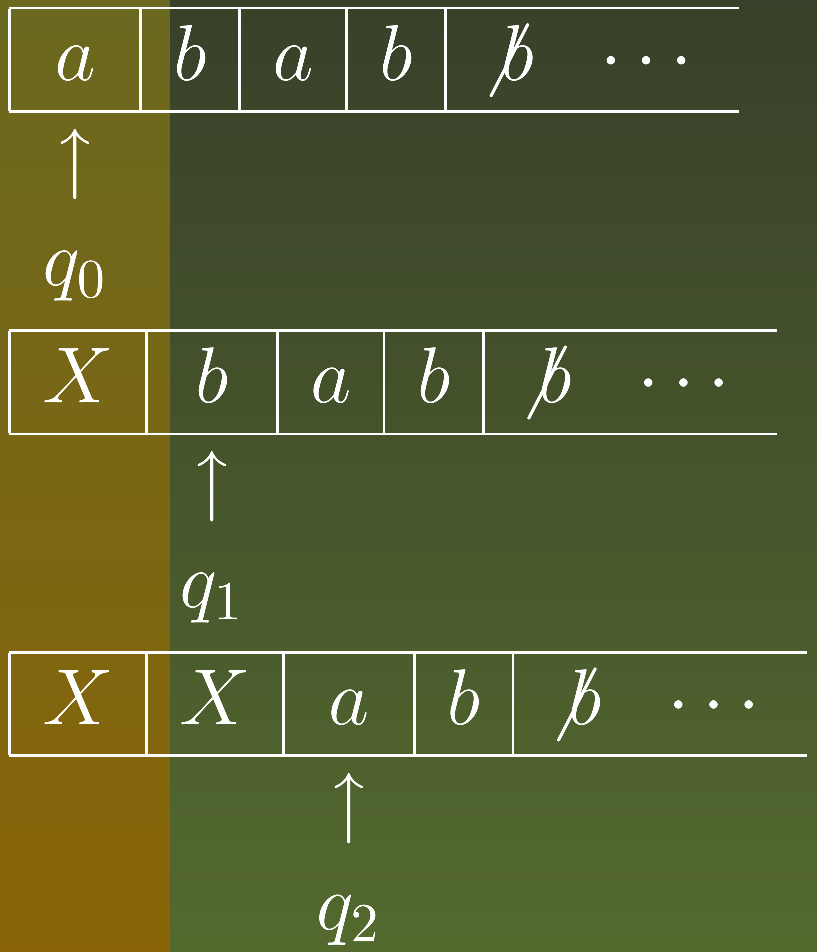
The machine halts without accepting as there is no further move possible and q_5 is not an accepting state.
It can be seen that only strings of the form $a^{i+j}b^ic^j$ will be accepted.

Example

Construct a Turing machine which will accept the set of strings over $\Sigma = \{a, b\}$ beginning with a 'a' and ending with a 'b'.
Though this set can be accepted by a FSA, we shall give a TM accepting it.
$M = (K, \Sigma, \Gamma, \delta, q_0, F)$ where
$K = \{q_0, q_1, q_2, q_3\}$ $\quad F = \{q_3\}$
$\Sigma = \{a, b\}$ $\quad \Gamma = \{a, b, X, \not{b}\}$
δ is defined as follows:
$\delta(q_0, a) = (q_1, X, R)$
$\delta(q_1, a) = (q_1, X, R)$
$\delta(q_1, b) = (q_2, X, R)$
$\delta(q_2, a) = (q_1, X, R)$

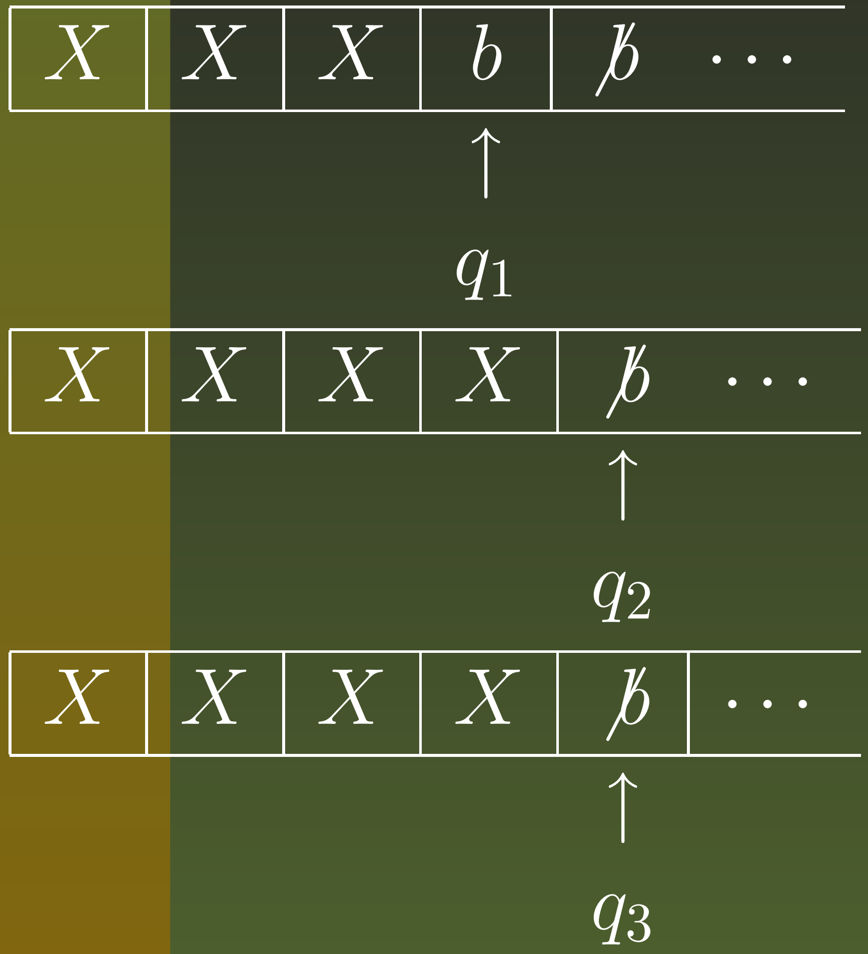
contd.

$\delta(q_2, b) = (q_2, X, R)$
$\delta(q_2, \not b) = (q_3, halt)$
Let us see how the machine accepts *abab*.



a b a b $\not b$ $\cdots$
q_0
X b a b $\not b$ $\cdots$
q_1
X X a b $\not b$ $\cdots$
q_2

contd.


X X X b $\not{b}$ $\cdots$
q_1
X X X X $\not{b}$ $\cdots$
q_2
X X X X $\not{b}$ $\cdots$
q_3

It can be seen that after initially reading 'a', the machine goes to state q_1.

contd.

Afterwards if it sees a 'a' it goes to state q_1; if it sees a 'b' it goes to q_2. Hence when it sees the leftmost blank symbol, if it is in state q_2 it accepts as this means that the last symbol read is a 'b'.

Turing Machine as a Computing Device

We looked at the Turing machine as an acceptor. In this section, we consider the Turing machine as a computing device. It computes functions which are known as partial recursive functions.

Example[Unary to binary converter] The input is a string of a's which is taken as the unary representation of an integer; a^i represents integer i. The output is of the form b_iX^i where b_i is a binary string which is the binary representation of integer i. The mapping for the Turing machine which does this is given below. The tape symbols are $\{ \not{b}, a, X, 0, 1\}$. The machine has two states q_0 and q_1. q_0 is the initial state and a right moving state. q_1 is a left moving state.

contd.

$\delta(q_0, a) = (q_1, X, L)$
$\delta(q_1, X) = (q_1, X, L)$
$\delta(q_1, \not{b}) = (q_0, 1, R)$
$\delta(q_1, 0) = (q_0, 1, R)$
$\delta(q_1, 1) = (q_1, 0, L)$
$\delta(q_0, X) = (q_0, X, R)$
$\delta(q_0, \not{b}) = (q_2, \not{b}, halt)$

The machine works like a binary counter. When it has converted j 'a's into X's, it prints binary number j to the left of the position where it started. On input $aaaaa$ the TM should output $101XXXXX$.

Copy machine

Given an input $\#w\#$, where w is a string of a's and b's, the machine makes a copy of w and halts with $\#w\#w\#$. The machine starts in state q_0, the initial state on the leftmost symbol of w.

It reads a 'a' changes that into a X and moves right in state q_a. When it sees the first blank symbol, it prints a 'a' and moves left in state q_1. If it sees a 'b' in q_0, it changes that into a Y and moves right in state q_b. When it sees the first blank symbol, it prints a 'b' and moves left in state q_1. In state q_1 it moves left till it sees a 'X' or a 'Y' and the process repeats. When no more 'a' or 'b' remains to be copied, the machine goes to q_2, prints a # after the copy

contd.

it has made and moves left in q_3. In q_3 it moves left till the # symbol. Then moving left it converts the 'X's and 'Y's into 'a's and 'b's respectively and halts when it sees the leftmost # symbol. q_a and q_b are used to remember the symbol the machine has read.

The state set is $\{q_0, q_a, q_b, q_1, q_2, q_3, q_4, q_5\}$.

Tape symbols are $\{\#, a, b, X, Y, \not b\}$

δ mappings are given by :

$\delta(q_0, a) = (q_a, X, R)$

$\delta(q_a, a) = (q_a, a, R)$

$\delta(q_0, b) = (q_b, Y, R)$

$\delta(q_b, a) = (q_b, a, R)$

$\delta(q_a, b) = (q_a, b, R)$

contd.

$$
\begin{aligned}
&\delta(q_a, \#) = (q_a, \#, R)\\
&\delta(q_a, \not b) = (q_1, a, L)\\
&\delta(q_b, b) = (q_b, b, R)\\
&\delta(q_b, \#) = (q_b, \#, R)\\
&\delta(q_b, \not b) = (q_1, b, L)\\
&\delta(q_1, a) = (q_1, a, L)\\
&\delta(q_1, b) = (q_1, b, L)\\
&\delta(q_1, \#) = (q_1, \#, L)\\
&\delta(q_1, X) = (q_0, X, R)\\
&\delta(q_1, Y) = (q_0, Y, R)\\
&\delta(q_0, \#) = (q_2, \#, R)\\
&\delta(q_2, a) = (q_2, a, R)\\
&\delta(q_2, b) = (q_2, b, R)
\end{aligned}
$$

contd.

$$\delta(q_2, \not{b}) = (q_3, \#, L)$$
$$\delta(q_3, a) = (q_3, a, L)$$
$$\delta(q_3, b) = (q_3, b, L)$$
$$\delta(q_3, \#) = (q_4, \#, L)$$
$$\delta(q_4, X) = (q_4, a, L)$$
$$\delta(q_4, Y) = (q_4, b, L)$$
$$\delta(q_4, \#) = (q_5, \#, halt)$$

contd.

The sequence of moves in input $\#abb\#$ can be described as follows :

$$
\begin{aligned}
&\#q_0abb\# \vdash \#Xq_abb\# \overset{*}{\vdash} \#Xbb\#q_a \vdash \#Xbbq_1\#a \overset{*}{\vdash} \#q_1Xbb\#a \\
&\vdash \#Xq_0bb\#a \vdash \#XYq_bb\#a \overset{*}{\vdash} \#XYb\#aq_b \vdash \#XYb\#q_1ab \\
&\overset{*}{\vdash} \#XYq_0b\#ab \vdash \#XYYq_b\#ab \overset{*}{\vdash} \#XYY\#abq_b \overset{*}{\vdash} \#XYY\#aq_1bb \\
&\overset{*}{\vdash} \#XYYq_0\#abb \vdash \#XYY\#q_2abb \overset{*}{\vdash} \#XYY\#abbq_2 \vdash \#XYY\#abq_3b\# \\
&\overset{*}{\vdash} \#XYYq_3\#abb\# \vdash \#XYq_4Y\#abb\# \overset{*}{\vdash} q_4\#abb\#abb\# \vdash q_5\#abb\#abb\#
\end{aligned}
$$

Example

Given two integers i and j, $i > j$, to compute the quotient and reminder when i is divided by j.

The input is

$\cdots$	$\#$	a^i	$\#$	b^j	$\#$	$\cdots$
				$\uparrow$		

with the tape head positioned on the leftmost 'b' in the initial state q_0.

The output is

$\cdots$	$\#$	X^i	$\#$	b^j	$\#$	c^k	$\#$	d^ℓ	$\#$	$\cdots$

where k is the quotient when i is divided by j and l is the remainder. The TM which does this is described as follows :

The TM converts the b's into Y's and a's into X's one by one.

contd.

When it sees no more b's it prints a 'c' after the # meaning j has been subtracted from i once

$\dots$	#	$X^j \ \ a^{i-j}$	#	b^j	#	c	$\dots$

This repeats as many times as possible. Each time a 'c' is printed. Finally when the number of a's which have to be converted to X's is less than j, the TM while trying to convert a 'a' into a 'X', will not find a 'a'. At this stage it would have converted (i mod $j + 1$) b's into Y's. The TM prints a # after c's and prints i mod $j + 1$ d's. It does this by changing a Y into a 'b' and printing a 'd' after rightmost # and d's. When all the Y's have been converted into

contd.

b's, we have i mod $j+1$ d's after the rightmost #. The TM erases the last d and prints a # and halts. The set of states are $\{q_0, \ldots, q_{21}\}$. The tape symbols are $\{\not b, \#, a, b, c, d, X, Y\}$
The mappings are given by
$\delta(q_0, b) = (q_1, Y, L)$
changes 'b' to Y and moves left.
$\delta(q_1, Y) = (q_1, Y, L)$
$\delta(q_1, \#) = (q_2, \#, L)$
$\delta(q_2, a) = (q_2, a, L)$
moves left.
$\delta(q_2, \#) = (q_3, \#, R)$
$\delta(q_2, X) = (q_3, X, R)$

contd.

when the leftmost $\#$ or an X is seen the head starts moving right

$\delta(q_3, a) = (q_4, X, R)$

one ‘a’ is changed into X

$\delta(q_4, a) = (q_4, a, R)$

$\delta(q_4, \#) = (q_5, \#, R)$

$\delta(q_5, Y) = (q_5, Y, R)$

moves right

$\delta(q_5, b) = (q_1, Y, L)$

process starts repeating

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

all ‘b’s have been converted to Ys

$\delta(q_6, c) = (q_6, c, R)$

$\delta(q_6, \not{b}) = (q_7, c, L)$

contd.

one ‘c’ is printed
$\delta(q_7, c) = (q_7, c, L)$
$\delta(q_7, \#) = (q_8, \#, L)$
moves left
$\delta(q_8, Y) = (q_8, b, L)$
Ys are changed back to ‘b’s
$\delta(q_8, \#) = (q_0, \#, R)$
process starts repeating
$\delta(q_3, \#) = (q_9, \#, R)$
all ‘a’s have been changed. Now the number of ‘c’s represents the quotient. Ys represented the remainder
$\delta(q_9, Y) = (q_9, Y, R)$
$\delta(q_9, b) = (q_9, b, R)$

contd.

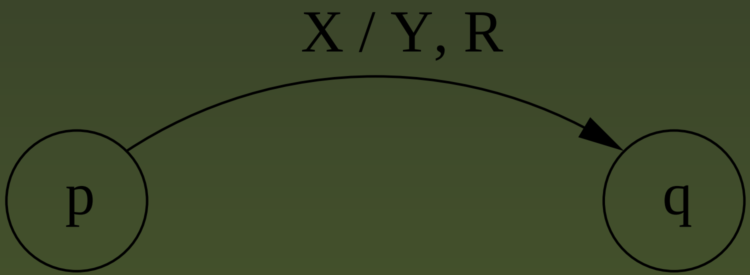
$\delta(q_9, \#) = (q_{10}, \#, R)$
$\delta(q_{10}, c) = (q_{10}, c, R)$
moves right
$\delta(q_{10}, \not{b}) = (q_{11}, \#, L)$
# is printed after the ‘c’s
$\delta(q_{11}, c) = (q_{11}, c, L)$
$\delta(q_{11}, \#) = (q_{12}, \#, L)$
$\delta(q_{12}, b) = (q_{12}, b, L)$
$\delta(q_{12}, Y) = (q_{13}, b, R)$
$\delta(q_{13}, b) = (q_{13}, b, R)$
$\delta(q_{13}, \#) = (q_{14}, \#, R)$
$\delta(q_{14}, c) = (q_{14}, c, R)$
$\delta(q_{14}, \#) = (q_{15}, \#, R)$

contd.

$\delta(q_{15}, d) = (q_{15}, d, R)$
$\delta(q_{15}, \not{b}) = (q_{16}, d, L)$
$\delta(q_{16}, d) = (q_{16}, d, L)$
$\delta(q_{16}, \#) = (q_{11}, \#, L)$
Ys are copied as ‘d’s
$\delta(q_{12}, \#) = (q_{17}, \#, R)$
after all Ys have been copied as ‘d’s the process starts finishing
$\delta(q_{17}, b) = (q_{17}, b, R)$
$\delta(q_{17}, \#) = (q_{18}, \#, R)$
$\delta(q_{18}, c) = (q_{18}, c, R)$
$\delta(q_{18}, \#) = (q_{19}, \#, R)$
$\delta(q_{19}, d) = (q_{19}, d, R)$

contd.

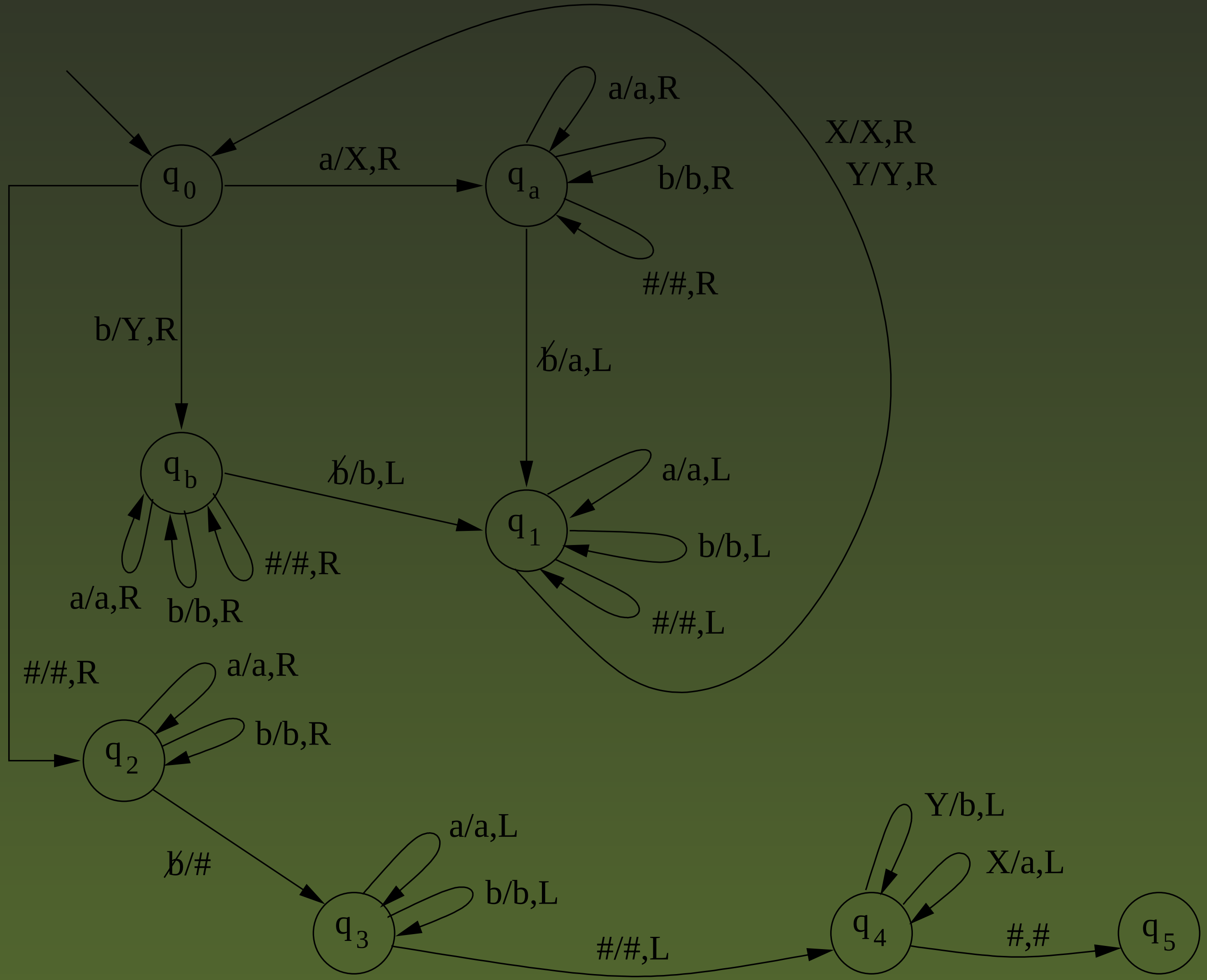
$\delta(q_{19}, \not{b}) = (q_{20}, \not{b}, L)$
$\delta(q_{20}, d) = (q_{21}, \#, halt)$
The move of a TM can be represented as a state diagram



means the TM when in state p and reading X, prints a Y over X, goes to state q and moves right.

The state diagram for previous example can be represented as

contd.


q0
a/X,R
qa
a/a,R
b/b,R
#/#,R
X/X,R
Y/Y,R
b/Y,R
b̸/a,L
qb
b̸/b,L
q1
a/a,L
b/b,L
#/#,L
a/a,R
b/b,R
#/#,R
#/#,R
q2
a/a,R
b/b,R
b̸/#
q3
a/a,L
b/b,L
#/#,L
q4
Y/b,L
X/a,L
#,#
q5

Techniques for Turing Machine Construction

Designing a Turing machine to solve a problem is an interesting task. It is somewhat similar to programming. Given a problem, different Turing machines can be constructed to solve it. But we would like to have a Turing machine which does it in a simple and efficient manner. Like we learn some techniques of programming to deal with alternatives, loops etc, it is helpful to understand some techniques in Turing machine construction, which will help in designing simple and efficient Turing machines. It should be noted that we are using the word ‘efficient’ in an intuitive manner here.

contd.

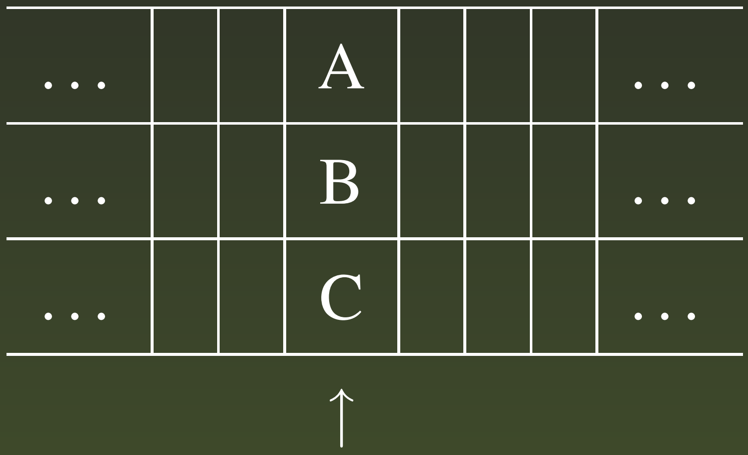
1. Considering the state as a tuple.

In an earlier example we considered a Turing machine which makes a copy of a given string over $\Sigma = \{a, b\}$. After reading a 'a', the machine remembers it by going to q_a and after reading a 'b', it goes to q_b. In general we can represent the state as $[q, x]$ where $x \in \Sigma$ denoting that it has read a 'x'.

2. Considering the tape symbol as a tuple.

Sometimes we may want to mark some symbols without destroying them or do some computation without destroying the input. In such cases it is advisable to have multiple tracks on the tape. This is equivalent to considering the tape symbol as a tuple.

contd.



There is only one tape head. In the above figure there are three tracks. The head is pointing to a cell which contains A on the first track, B on the second track and C on the third track. The tape symbol is taken a 3-tuple $[A, B, C]$. Some computation can be done in one track by manipulating the respective component of the tape symbol. This is very useful in checking off symbols.

contd.

3. Checking off symbols.

We use one track of the tape to mark that some symbols have been read without changing them.

Example

Consider a Turing machine for accepting

$$w\#w\#w, w \in \{a, b\}^*$$

A tape having two tracks is considered.

...	a	b	b	#	a	b	b	#	a	b	b	...
...												...

contd.

The first track contains the input. When the TM reads the first a in state q_0, it stores it in its memory (by taking the state as a pair), checks off 'a' by printing a $\surd$ in the second track below a, moves right and after the # symbol checks whether the symbol is a 'a'. If so marks it by putting a $\surd$ in the second track, moves right and again checks the first symbol after # is a 'a' and if so it marks also. It then moves left and repeats the process with each unmarked leftmost symbols in each block. When all the symbols in the first block match with the second and third blocks, the machine halts accepting the string.

contd.

The mappings can be defined as follows:
$\delta(q_0, [a, \not{b}]) = ([q, a], [a, \surd], R)$
$\delta(q_0, [b, \not{b}]) = ([q, b], [b, \surd], R)$
The machine reads the leftmost symbol marks it and remembers whether it is a ‘a’ or ‘b’ by storing it in the state $\delta([q, a], [a, \not{b}]) = ([q, a], [a, \not{b}], R)$
$\delta([q, a], [b, \not{b}]) = ([q, a], [b, \not{b}], R)$
$\delta([q, b], [a, \not{b}]) = ([q, b], [a, \not{b}], R)$
$\delta([q, b], [b, \not{b}]) = ([q, b], [b, \not{b}], R)$
The head passes through symbols in the first block to the right
$\delta([q, a], [\#, \not{b}]) = ([p, a], [\#, \not{b}], R)$
$\delta([q, b], [\#, \not{b}]) = ([p, b], [\#, b], R)$

contd.

When the head encounters a # in the first track, the first component of the state is changed to p.

$\delta([p, a], [a, \surd]) = ([p, a], [a, \surd], R)$
$\delta([p, a], [b, \surd]) = ([p, a], [b, \surd], R)$
$\delta([p, b], [a, \surd]) = ([p, b], [a, \surd], R)$
$\delta([p, b], [b, \surd]) = ([p, b], [b, \surd], R)$
$\delta([p, a], [a, \not{b}]) = ([r, a], [a, \surd], R)$
$\delta([p, b], [b, \not{b}]) = ([r, b], [b, \surd], R)$

contd.

When it encounters a first unchecked symbol it marks it by putting a $\surd$ in the second track and changes the first component of the state to r.

$\delta([r,a],[a,\not{b}]) = ([r,a],[a,\not{b}],R)$
$\delta([r,b],[a,\not{b}]) = ([r,b],[a,\not{b}],R)$
$\delta([r,a],[b,\not{b}]) = ([r,a],[b,\not{b}],R)$
$\delta([r,b],[b,\not{b}]) = ([r,b],[b,\not{b}],R)$

The head moves through the second block without changing symbols, when the first component of the state is r

contd.

$\delta([r,a],[\#,\not b]) = ([s,a],[\#,\not b],R)$
$\delta([r,b],[\#,\not b]) = ([s,b],[\#,\not b],R)$
When it encounters a # in the first track it moves right into the third block changing the first component of the state to s
$\delta([s,a],[a,\surd]) = ([s,a],[a,\surd],R)$
$\delta([s,a],[b,\surd]) = ([s,a],[b,\surd],R)$
$\delta([s,b],[a,\surd]) = ([s,b],[a,\surd],R)$
$\delta([s,b],[b,\surd]) = ([s,b],[b,\surd],R)$
It moves right looking for the unchecked symbol
$\delta([s,b],[b,\not b]) = (t,[b,\surd],L)$
$\delta([s,a],[a,\not b]) = (t,[a,\surd],L)$

When it encounters an unchecked symbol in the third

contd.

it marks it by putting a $\surd$ in the second track and starts moving left.

$\delta(t,[a,\surd]) = (t,[a,\surd],L)$
$\delta(t,[b,\surd]) = (t,[b,\surd],L)$
$\delta(t,[\#,\surd]) = (t',[\#,\surd],L)$

It moves into the second block in state t'

$\delta(t',[a,\not b]) = (t',[a,\not b],L)$
$\delta(t',[b,\not b]) = (t',[b,\not b],L)$
$\delta(t',[a,\surd]) = (t',[a,\surd],L)$
$\delta(t',[b,\surd]) = (t',[b,\surd],L)$

It moves left in the second block.

$\delta(t',[\#,\not b]) = (t'',[\#,b],L)$

It moves left into the first block in state t''.

contd.

$\delta(t'', [a, \not{b}]) = (t'', [a, \not{b}], L)$
$\delta(t'', [b, \not{b}]) = (t'', [b, \not{b}], L)$
It moves left in the first block through unchecked symbols.
When it encounters a checked symbol it moves right in state q_0 and the whole process repeats
$\delta(t'', [a, \surd]) = (q_0, [a, \surd], R)$
$\delta(t'', [b, \surd]) = (q_0, [b, \surd], R)$
This way the machine checks for same symbols in the first, second and third blocks.
Finishing
When the machine encounters a # in the first track in state q_0, it means it has checked all symbols in the first block. Now it has to check that there are no more

contd.

symbols in the second and third block.

$\delta(q_0, [\#, \not{b}]) = (q_1, [\#, \not{b}], R)$

$\delta(q_1, [a, \surd]) = (q_1, [a, \surd], R)$

$\delta(q_1, [b, \surd]) = (q_1, [b, \surd], R)$

If it encounters an unchecked symbol, it halts without accepting

$\delta(q_1, [a, \not{b}]) = (q_n, [a, \not{b}], R)$

$\delta(q_1, [b, \not{b}]) = (q_n, [b, \not{b}], R)$

If it finds all symbols are checked in the second block, it moves to the third block in state q_2

$\delta(q_1, [\#, \not{b}]) = (q_2, [\#, \not{b}], R)$

In the third block it checks whether all symbols have already been checked. If so, it halts in accepting state q_y.

contd.

Otherwise halts in nonaccepting state q_n

$\delta(q_2, [a, \surd]) = (q_2, [a, \surd], R)$

$\delta(q_2, [b, \surd]) = (q_2, [b, \surd], R)$

$\delta(q_2, [a, \not b]) = (q_n, [a, \not b], R)$

$\delta(q_2, [b, \not b]) = (q_n, [b, \not b], R)$

$\delta(q_2, [\not b, \not b]) = (q_y, [\not b, \not b], R)$

If the input has more symbols in the first block (than second block) it moves in the second block in state $[p, a]$ or $[p, b]$ and encounters $[\#, \not b]$. Then it halts rejecting the input

$\delta([p, a], [\#, \not b]) = (q_n, [\#, \not b], R)$

$\delta([p, b], [\#, \not b]) = (q_n, [\#, \not b], R)$

If the input has equal symbols in the first and second

contd.

block but less symbols in the third block, the machine encounters $[\not b, \not b]$ an state $[s, b]$ or $[s, a]$ and halts without accepting

$\delta([s, a], [\not b, \not b]) = (q_n, [\not b, \not b], R)$

$\delta([s, b], [\not b, \not b]) = (q_n, [\not b, \not b], R)$

Thus we find that having two tracks and using the second track to check off symbols is a useful technique.

When we consider a single tape multitrack TM, we really take the tape symbol as a tuple. This need not be considered as a variation of Turing machine.

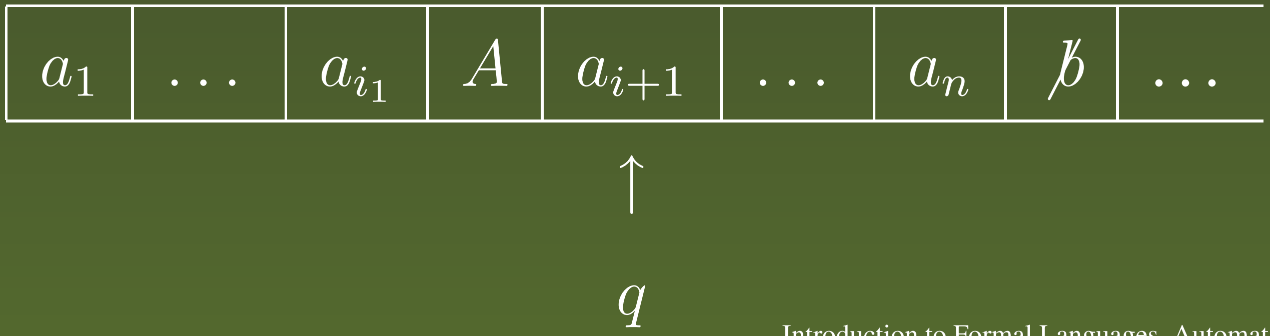
4. Shifting over.

Sometimes we may have to shift symbols on the tape to the right or left to allow for some symbols to be written.

contd.

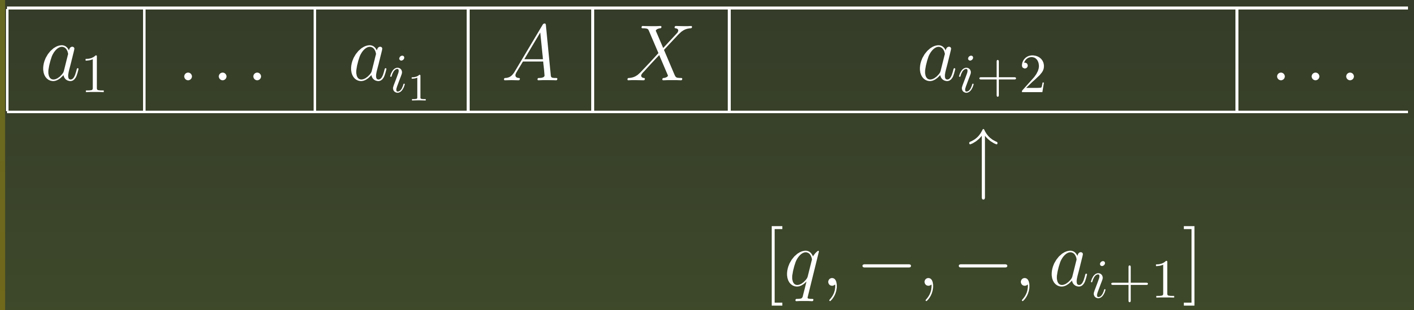
Suppose the contents of the tape are $a_1 \ldots a_{i-1} A a_{i+1} \ldots a_n$ at some instant. A has to be replaced by $abcd$ say. Then $a_{i+1} \ldots a_n$ have to be shifted three cells to the right and then in the space created $abcd$ can be printed. We can use the state as a tuple to store some information and shift symbols. Suppose the head is reading a_{i+1} in state q and the shifting process has to start. Then the TM reads a_{i+1} and goes to a state $[q, -, -, a_{i+1}]$ and prints X over a_i.

The ID



contd.

changes to



Next, the TM reads a_{i+2}, storing it in the fourth component, and shifting a_{i+1} from fourth component to the third component

$\delta([q, -, -, a_{i+1}], a_{i+2}) = ([q, -, a_{i+1}, a_{i+2}], X, R)$

Similarly

$\delta([q, -, a_{i+1}, a_{i+2}], a_{i+3}) = ([q, a_{i+1}, a_{i+2}, a_{i+3}], X, R)$

When it reads a_{i+4}, it deposits a_{i+1} in that cell

contd.

$\delta([q, a_{i+1}, a_{i+2}, a_{i+3}], a_{i+4}) =$
$([q, a_{i+2}, a_{i+3}, a_{i+4}], a_{i+4}, R)$
In general
$\delta([q, a_j, a_{j+1}, a_{j+2}], a_{j+3}) =$
$([q, a_{j+1}, a_{j+2}, a_{j+3}], a_j, R) \quad i+1 \leq j \leq n$
where $a_{n+1}, a_{n+2}, a_{n+3}$ are blank symbol $\not{b}$. Finally it starts moving left
$\delta([q, a_n, \not{b}, \not{b}], \not{b}) = (q', a_n, L)$

In q' it moves left till it finds $AXXX$ and replaces it by $abcd$. A similar method can be used for shifting symbols to the left. Thus storing some information in some components of the state and cyclically moving

contd.

the components helps in the technique of shifting off symbols.

5. Subroutines.

Just as a computer program has a main procedure and subroutines, the TM can also be programmed to have a main TM and TMs which serve as subroutines.

Suppose we have to make n copies of a word w. Input is $\#w\#$ and the output is $\#w\#\underbrace{www\ldots w}_{n\ times}$.

In this case we can write the mappings for a TM M_{sub} which when started on $\#w\#x$ ends up with $\#w\#xw$. The main TM will call this M_{sub} x times. Similarly for multiplying two unary numbers m and n, n has to be

contd.

copied on m times. We can write a sub TM for copying and main TM will call this m times.
In order that a Turing machine M_1 uses another Turing machine M_2 as a subroutine, the states of M_1 and M_2 have to be disjoint. Also when M_1 wants to call M_2, from a state of M_1, the control goes to the initial state of M_2. When the subroutine finishes and returns to M_1, from the halting state of M_2, the machine goes to some state of M_1. Note that a subroutine TM call another TM as its subroutine. This technique helps to construct a TM in a topdown manner dividing the work into tasks and writing a TM for each task and combining them.