RG: most restricted grammar
- is recognized by FSA
CFG- is recognized by PDA

Chapter 1
Turing Machine

- Turing machine - is potentially able to execute any algorithm.
- It is capable of solving every problem that a real computer can do.
- similar to finite automaton but with an unlimited and unrestricted memory.
- It uses infinite tape as its unlimited memory.
- It has a tape head that can read/write symbols and move around the tape.
- Initially the tape contains only the input string and is blank everywhere else. If a machine needs to store info, it may write this info on the tape. To read the info that it has written, the machine can move its head back over it.

- TM is a mathematical model wh consists of an infinite length tape divided into cells on wh input is given (one alphabet at each cell). A state register stores the state of the TM. After reading an input symbol from the input tape, it is replaced with another symbol, its internal state is changed and it moves from one cell to the right or left.

Theory of automata - is a theoretical branch of computer and mathematics. It is the study of abstract machines and the computation problems that can be solved using these machines. The abstract machine is called automata.

→ Initially the tape of TM contains z input strings & read-write tape pts on z 1st alphabet.

Then, if the TM reaches the final state, the input string is accepted, or rejected.

- TM describe a much large class of languages called class of recursively enumerable languages.
- The input tape of TM is of finite length from left but is of infinite length from right.
- TM recognizes - regular language
  - context free lang
  - context sensetive lang
  - recursively enumerable lang
- TM = FSM + tape memory.

D/C b/n FSA and TM

FSA	TM
- FSM describe the class of regular languages	- TM describes a much larger class of languages
- Have lower computational power than TMs	- Have more computational power than FSA
- Input tape is of finite length from both left & right side.	- Input tape is of finite length from left but infinite length from right
- Head can only read the symbols from the tape but not write	- Head is also to read as well as write symbols

Automata - rec
an
fo
m

- Head is able
in right dx
- FSM is just a
states and th
The only mem
is what stat

D/C b/n

PDA - can onl
- is less

TM - can ac

is defined

* TM halts
not define
* In FSA, i
the final
but but i
it halts.

Automata - receive an input string and execute an algorithm to obtain an answer, following a set of rules specific to the machine type.

- Head is able to move in right dxn
- FSM is just a set of states and transitions. The only memory it has is what state it is in

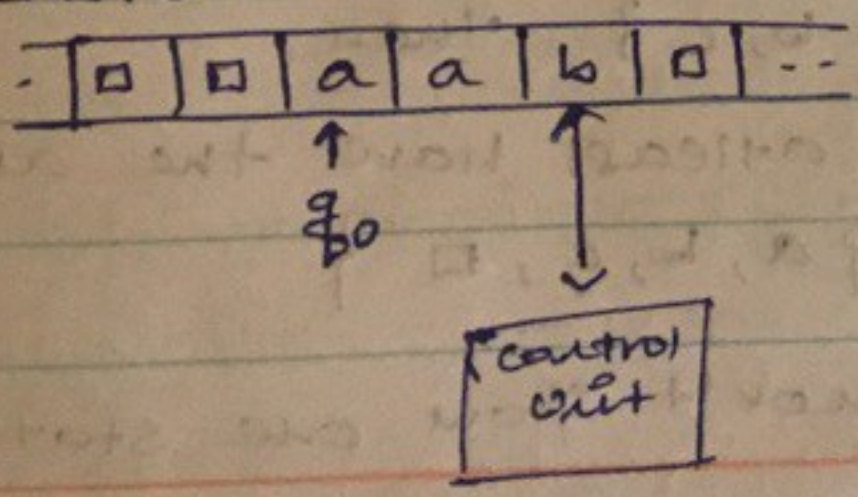
- Head can move in both dxns
- TM is a FSM + tape memory

D/C b/n PDA and TM

PDA - can only access the top of its stack,
- is less computationally powerful

TM - can access any position on an infinite tape.
- is defined for CFL

* TM halts when it reaches final state, since δ is not defined for final state.
* In FSA, inorder to say Accept, the FSA must reach the final state after processing all z string alphabet but in TM when it reaches z final state it halts. (right away accept/reject)



Input string → aab

□ → blank

Accept } halt
reject }

loop → in this case TM doesn't halt

Σ : is given

Control unit : controls the movt b/n states (read/write movt)

: is like transition fn : in FSA in this case addition to transition of state, it also defines read/write movt in a memory.

+

The Standard TM

- 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ where

Q, Σ, Γ are all finite sets and :

Q → set of states

Σ → input alphabet $\subseteq \Gamma - \{\square\}$

Γ → the tape alphabet : (contains all symbols that can be written onto the tape)

$\delta : (Q, \Gamma) \rightarrow (Q, \Gamma \times (\{L, R\}))$ ... transition fn

L → left, R → right

$\square \in \Gamma$: is special symbol called the blank

$q_0 \in Q$: is initial state

$F \subseteq Q$: set of final states

→ Γ is derived from Σ

ex 1 : $\Sigma = \{a, b, c\}$ then

Γ will atleast have the alphabet $\{a, b, c, \square\}$

→ δ → decides the movt from one state to another.

(X) At any

by start

Q X Γ

At each

change

to move

Eg1 De

δ :

Eg2 D

δ :

(XI) Tran

where

3 ite

Ex

(*) At any time configuration of a TM is defined by state, alphabet & pos of read/write tape.

$Q \times \Gamma = Q, \Gamma, \{L, R\}$

At each step Q & Γ of the TM may or may not change but it is must for the read/write tape to move either right or left.

Eg1 Design a TM for $L = \{ab^*\}$

→ here there is state change b/c from z initial state (q_0) as it has read a but after that the state fxn differs (it might get b or □). So q_0 & q_1 hold dif state fxn so they should differ

δ: $\delta(q_0, a) = (q_1, a, R)$

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

$\delta(q_1, \square) = (q_f, \square, L)$

Eg2 Design a TM for $L = \{a, b\}^*$

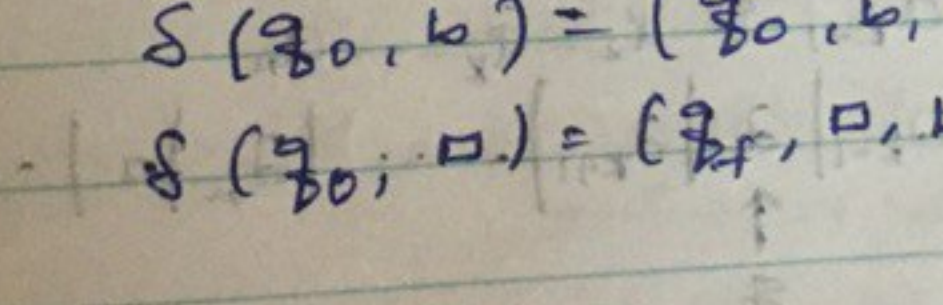
δ: $\delta(q_0, a) = (q_0, a, R)$

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

$\delta(q_0, \square) = (q_f, \square, L)$

(*) Transition graphs can be used to represent TMs where the edges of the graph can be labeled w 3 items.
- the current tape symbol
- the symbol that replaces the current tape symbol
- the dxn fxn w/c the read/write head is to move.

Ex



$Q = \{q_0, q_1\}$
$\Sigma = \{a, b\}$
$\Gamma = \{a, b, \square\}$
$q_0 = \{q_0\}$
$F = \{q_1\}$
$\square = \{\square\}$

δ: $\delta(q_0, a) = (q_0, b, R)$

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

$\delta(q_0, \square) = (q_1, \square, L)$

config of TM - 1 means the state at a/c the TM is at any time.

(X) Two consecutive configurations can not be exactly the same, coz at least pos of z read/write tape changes

Configuration of a TM

- Config of a TM is completely determined by the current state of the control unit, the contents of the tape, and the pos of the read-write head
- we use ID (Instantaneous description) to represent configuration and exhibit a sequence of configs
- ID -> tells us what the config of the TM is in any given instance.

notation : $X_1 q_i X_2$ or $a_1 \cdots a_{k-1} q_i a_k a_{k+1} \cdots a_n$, where
- read/write tape head is on z cell holding a_k
- current state is q_i
- $X_1 = a_1 \cdots a_{k-1}$ & $X_2 = a_k \cdots a_n$

□	a_1	a_2	⋯	a_{k-1}	a_k	a_{k+1}	⋯	a_n	□	⋯
					↑ q_i					

→ TM is said to halt starting from initial config $X_1 q_i X_2$ if $X_1 q_i X_2 \vdash^* Y_1 q_j a Y_2$ for any q_j & for w/c $\delta(q_j, a)$ is undefined.

Computation : sequence of configs leading TM to a halt state, starting from the initial state.

- If TM goes into a loop and never halts we use special notation $X_1 q_i X_2 \vdash^* \infty$

TM can be r
- grap
- Inst

Eg $\delta(q_0, a$
$\delta(q_0, b$
$\delta(q_0,$

The ID wi
i) q
ii) Initial

ID
□ q_0 b b a
□ b b b $q_$
□ b b b b $q_$

(X) TM can
1) h

2) T

TM
- Input str
out the
- machine
pos on
- If aft

TM can be represented using
- graph
- Instantaneous desc. (ID)

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

i) $w = aa$
ii) $w = bbaabb$

The ID will be :

i) $q_0aa \vdash bq_0a \vdash bbq_0\sqcup \vdash bq_1b$

ii) Initial config =>

$\sqcup$	b	b	a	a	b	$\sqcup$
*	↑					

ID => $\sqcup q_0bbaabb\sqcup$

$\sqcup q_0bbaabb\sqcup \vdash \sqcup bq_0baabb\sqcup \vdash \sqcup bbq_0aabb\sqcup \vdash$
$\sqcup bbbq_0abb\sqcup \vdash \sqcup bbbbq_0bb\sqcup \vdash \sqcup bbbbbbq_0\sqcup \vdash$
$\sqcup bbbbbq_1b\sqcup$ ← final config & TM halts since $\delta(q_1, b)$ is not defined.

(*) TM can be used as
1) Language acceptor : accept or reject any given string by determining whether that string is member of a language.

2) Transducer : is automata that gives output

TM as Language Acceptor

- Input string is written on the tape, w blanks filling out the unused portions.
- Machine starts w initial state q_0 & read/write head pos on the 1st alphabet of the string.
- If after a sequence of moves, TM enters a final

L(M) → lang of any given turing machine
TM enters into final state only for those strings
are member of Σ lang given.

(*) lang acceptor doesn't care about output
TM as

state and halts, then w is accepted.

? - If TM halts on a non final state → w rejected

Example 1: Design a TM that accepts the language denoted by the regular expression

00^* → is regular lang, so it's TM recognizable

$L(M) = \{00^*\}$

$L(M) = \{0, 00, 000, 0000, \dots\}$

$\Sigma = \{0, 1\}$ $\quad \Gamma = \{0, 1, \sqcup\}$

δ: $\delta(q_0, 0) = (q_1, 0, R)$

→ here there is state change b/c q_0 is not a final state and from it there is only 1 possible w/c is 0 but from both q_1 & w/c are 0's

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

$\delta(q_1, \sqcup) = (q_f, \sqcup, L)$

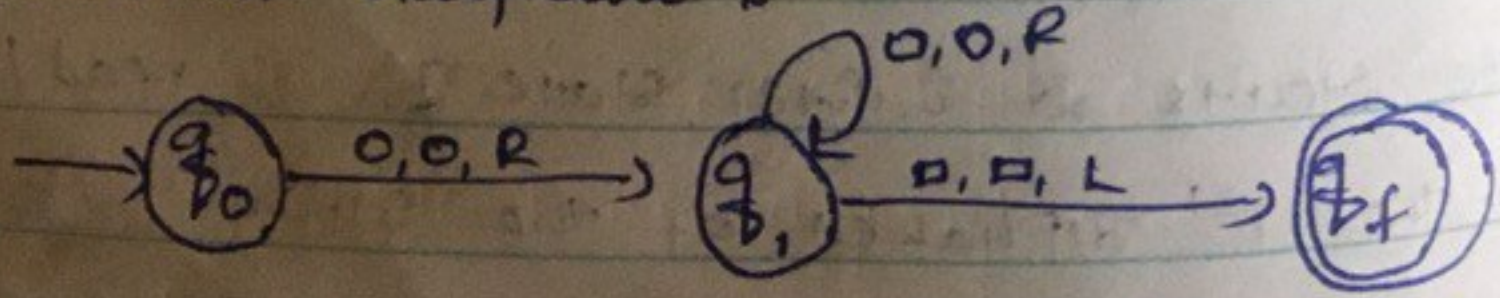
Q: $\{q_0, q_1, q_f\}$

$\sqcup = \{\sqcup\}$

$F = \{q_f\}$

$q_0 = q_0$

→ Transition diagram:



- Every regula
- Every co
- each state
- here the
with an
$\Sigma = \{a,$
$\Gamma = \{a,$
δ:
$\delta(q_0$
$\delta(q$
$\delta(q$
$\delta(q$
$\delta($
- Transi

- Every regular lang is context-free
  - Every context-free-lang → is decidable by TM (so recognized by TM)
- each state goes wrong if a valid string ends so we can go back and try new from $q_2 \to q_0$

Eg 2 : for $\Sigma = \{a, b\}$, design a TM that accepts

$$L = \{a,b\}^*(ab) = \{(a/b)^*ab\}$$

- here the lang. accepted is any combination of a's & b's with any length but ends in ab

$$L(M) = \{ab, aab, bab, abab, bbab, \dots\}$$

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

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

δ:

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

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

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

$\delta(q_1, b) = (q_2, b, R)$

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

$\delta(q_2, b) = (q_0, b, R)$

$\delta(q_2, \square) = (q_f, \square, L)$

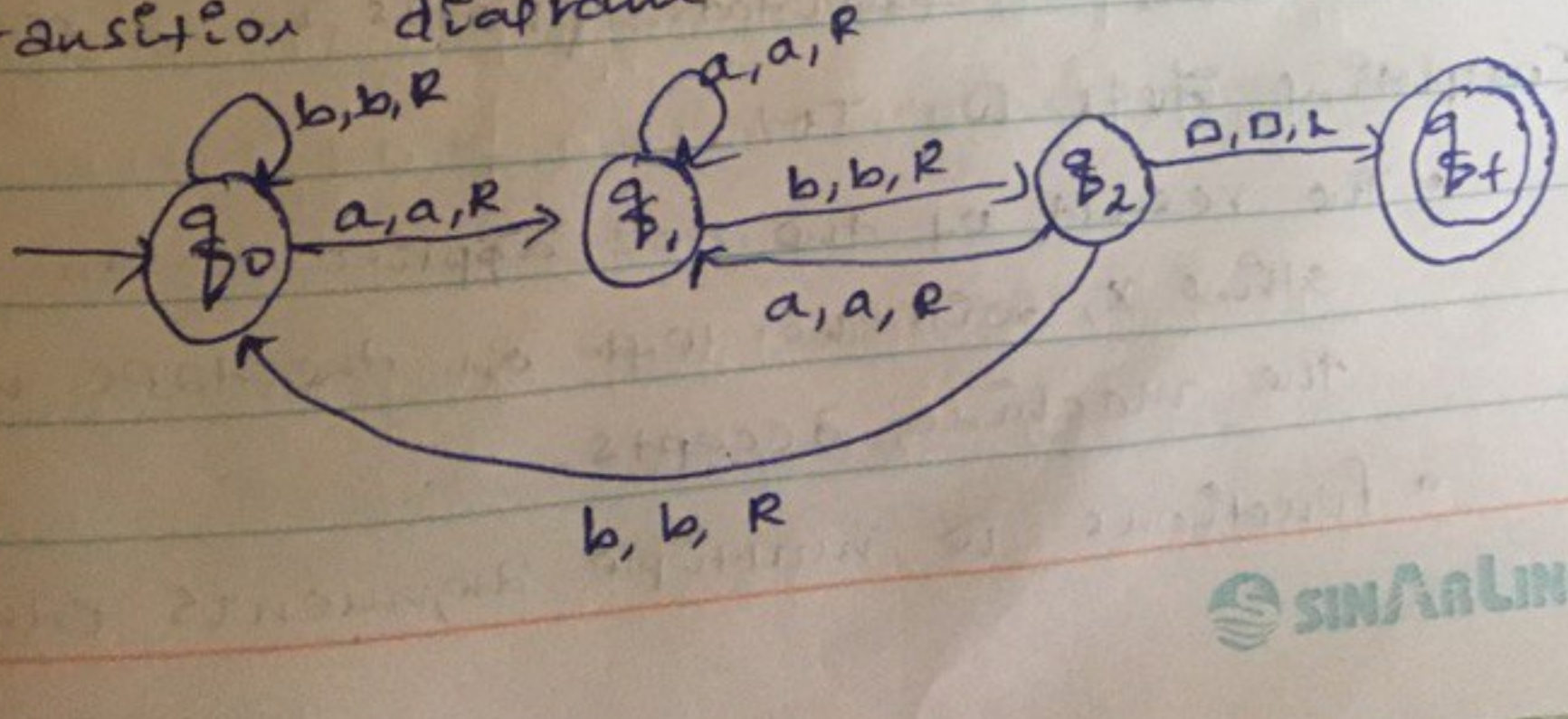
- changes state coz when it gets a we need a state (q_1) to store the info that it has got a & needs only a single b to be a valid string.

-> now changes state coz q_2 must hold the info that ab is found.

-> changes to state q_1 bc ontop of ab now a is added so we need b for a valid string. ex: aba

-> goes to q_0 b/c now we need complete ab combination for a valid string so goes to start again state w/c is q_0 already

- Transition diagram



ε)
for w = bbaabab, the sequence of moves will be :

$q_0$bbaabab ⊢ b$q_0$baabab ⊢ bb$q_0$aabab ⊢
bba$q_0$abab ⊢ bbaa$q_1$bab ⊢ bbaab$q_2$ab ⊢
bbaabaq_1b ⊢ bbaababq_2□ ⊢ bbaabaq_fb

ii) w = abba

$q_0$abba ⊢ a$q_1$bba ⊢ ab$q_2$ba ⊢ abbq_0a ⊢ abbaq_1□

but (q_1, □) ⇒ is undefined & q_1 is not final state so TM rejects it.

∴ abba ∉ L(M)

TM As Transducers

- Transducer transforms input to output.
- Input → the nonblank symbols on the tape at the initial time.

output → whatever is on the tape after the computation (halts in a final state)

Computing fn's w TM

- The result of the fn applied to an input string x, will be left on the tape when the machine accepts.
- functions w multiple arguments can be

placed on th
by some ch
Formally,
Let M = (Q
be a par
(or f is
where

NOTE: A
sy
s
Example
th
- we use
i.e., a
1's who
eq
comp

Partial fn: a fn which is not defined for some inputs of the right type, that is, for some of a domain.

placed on the input tape w arguments separated by some character.

Formally,

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ be a TM and let f be a partial fn on Σ^*. We say T computes f (or f is Turing-computable) if for every $x \in \Sigma^*$ where f is defined:

$$q_0 x \vdash^* P f(x)$$

where $P \in F$

& for every other x, T rejects on input x.

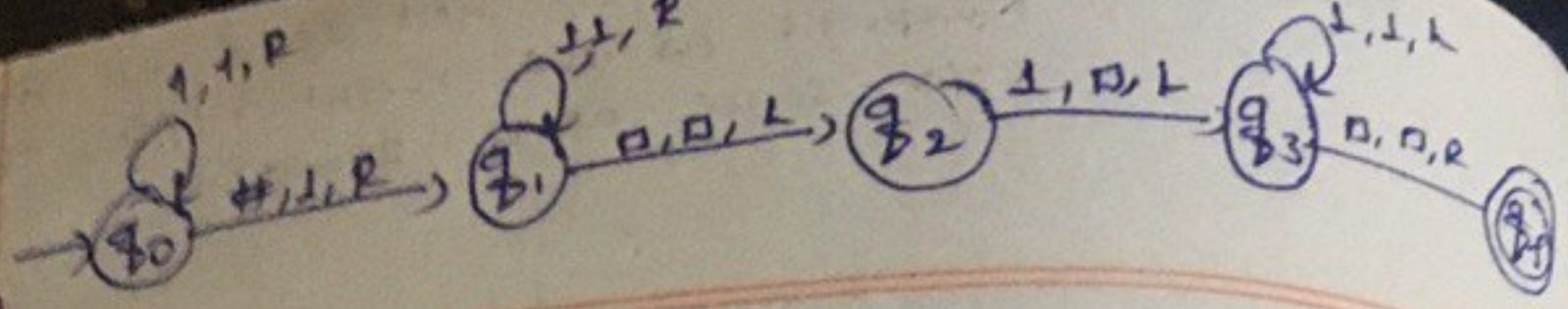
NOTE: A transducer TM must start on the left most symbol of the input and stop on the left most symbol of the output.

Example 1: Given two positive integers, design a TM that computes $x + y$

- we use unary notation to represent the positive integers, i.e., a positive integer is represented by a sequence of 1's whose length is equal to the value of the integer.
  eg 1111 = 4

computation: $q_0 w(x) \# w(y) \vdash^* q_f w(x+y) \#$

=> separates the two nos at initiation and after z result at termination.



Basic idea → is to move the separator, # to the right end.

To achieve this, we construct $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$

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

$F = \{q_f\}$

δ :

$\delta(q_0, 1) = (q_0, 1, R)$

$\delta(q_0, \#) = (q_1, 1, R)$

$\delta(q_1, 1) = (q_1, 1, R)$

$\delta(q_1, \square) = (q_2, \square, L)$

$\delta(q_2, 1) = (q_3, \square, L)$

$\delta(q_3, 1) = (q_3, 1, L)$

$\delta(q_3, \square) = (q_f, \square, R)$

} to reposition the read-write tape head to point to 2nd alphabet

The sequence of instantaneous description for adding 111 and 1111 is :

$\square q_0 111 \# 1111 \vdash \square 1 q_0 11 \# 1111 \vdash^2 \square 111 q_0 \# 1111 \vdash$

$\square 1111 q_1 1111 \square \vdash^4 \square 11111111 q_1 \square \vdash \square 11111111 q_2 10 \vdash$

$\square 1111111 q_3 1 0 \square \vdash^7 q_3 \square 11111111 \square \vdash \square q_f 11111111 \square$

Eg 2 : Design a TM that copies an input string over the $\Sigma = \{a\}$

$\rightarrow f(x) = x \# x$

Pos of q_0 sh
pts to the 1st
case where the
equation not.

$\underline{\alpha}$ f(aaa)

Initial config:

$Q = \{q_0, q_1,$

$F = \{q_f\}$

δ :

$\delta(q_0, a$

$\delta(q_0, \square$

$\delta(q_1,$

$\delta(q_2,$

$\delta(q_2,$

$\delta(q_2,$

$\delta(q_2$

$\delta(q_2$

$\delta(q_3$

$\delta(q_3$

$\delta(q_4$

$\delta(q_4$

$\delta(q$

$\delta(q$

$\delta(q$

The seq

Pos of q_0 should be positioned in such away that it is to the first alphabet of z output especially in the case where the result of one operation + to operation especially when. If not it doesn't matter.

ex f(aaa) = aaa#aaa (final config)

Initial Config: ⊔|a|a|a|⊔ $\vdash^*$ ⊔|a|a|a|#|a|a|a|⊔

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_f\}$

$F = \{q_f\}$ $q_0 = \{q_0\}$ $\Gamma = \{a, \sqcup, A, \#\}$

δ :

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

$\delta(q_0, \sqcup) = (q_1, \#, L)$ -> to mark a separator

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

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

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

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

$\delta(q_2, \sqcup) = (q_3, a, L)$

$\delta(q_3, a) = (q_3, a, L)$

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

$\delta(q_4, A) = (q_4, A, L)$

$\delta(q_4, a) = (q_2, A, R)$

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

$\delta(q_5, A) = (q_5, a, R)$ -> to change A's back to a

$\delta(q_5, \#) = (q_f, \#, L)$

The sequence of ID for copying : w = aaa

:

$q_0 aaa \vdash^3 aaaq_0\square \vdash aaq_1a\# \vdash aaAq_2\# \vdash$
$aaA\#q_2\square \vdash aaAq_3\#a \vdash aaq_4A\#a \vdash$
$aq_4aA\#a \vdash aAq_2A\#a \vdash^3 aAA\#aq_2\square \vdash aAA\#q_3aa \vdash$
$aAAq_3\#aa \vdash aAq_4A\#aa \vdash^2 \square q_4aAA\#aa \vdash$
$Aq_2AA\#aa \vdash^5 AAA\#aaq_2\square \vdash AAA\#aq_3aa \vdash^2$
$AAAq_3\#aaa \vdash AAq_4A\#aaa \vdash^3 q_4\square AAA\#aaa \vdash$
$\square q_5AAA\#aaa \vdash^3 aaaq_5\#aaa \vdash aaq_1a\#aaa$

Graphs ?

Eg3 Design a TM that copies a string of 1's

→ Computation performed: $q_0w \vdash^* q_fww$ for any $w \in \{1\}^+$

→ to solve the problem, we can implement:

#1 replace every 1 by x

#2 find the rightmost x and replace it w 1.

#3 Travel to the right end of the current nonblank region and create a 1 there.

#4 Repeat step 2 & 3 until there are no more x's.

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

$F = \{q_f\}$ $q_0 = \{q_0\}$ $\Sigma = \{1\}$ $\Gamma = \{1, \square\}$

δ :

$\delta(q_0, 1) = (q_0, X, R)$

$\delta(q_0, \square) = (q_1, \square, L)$

$\delta(q_1, X) = (q_2, 1, R)$

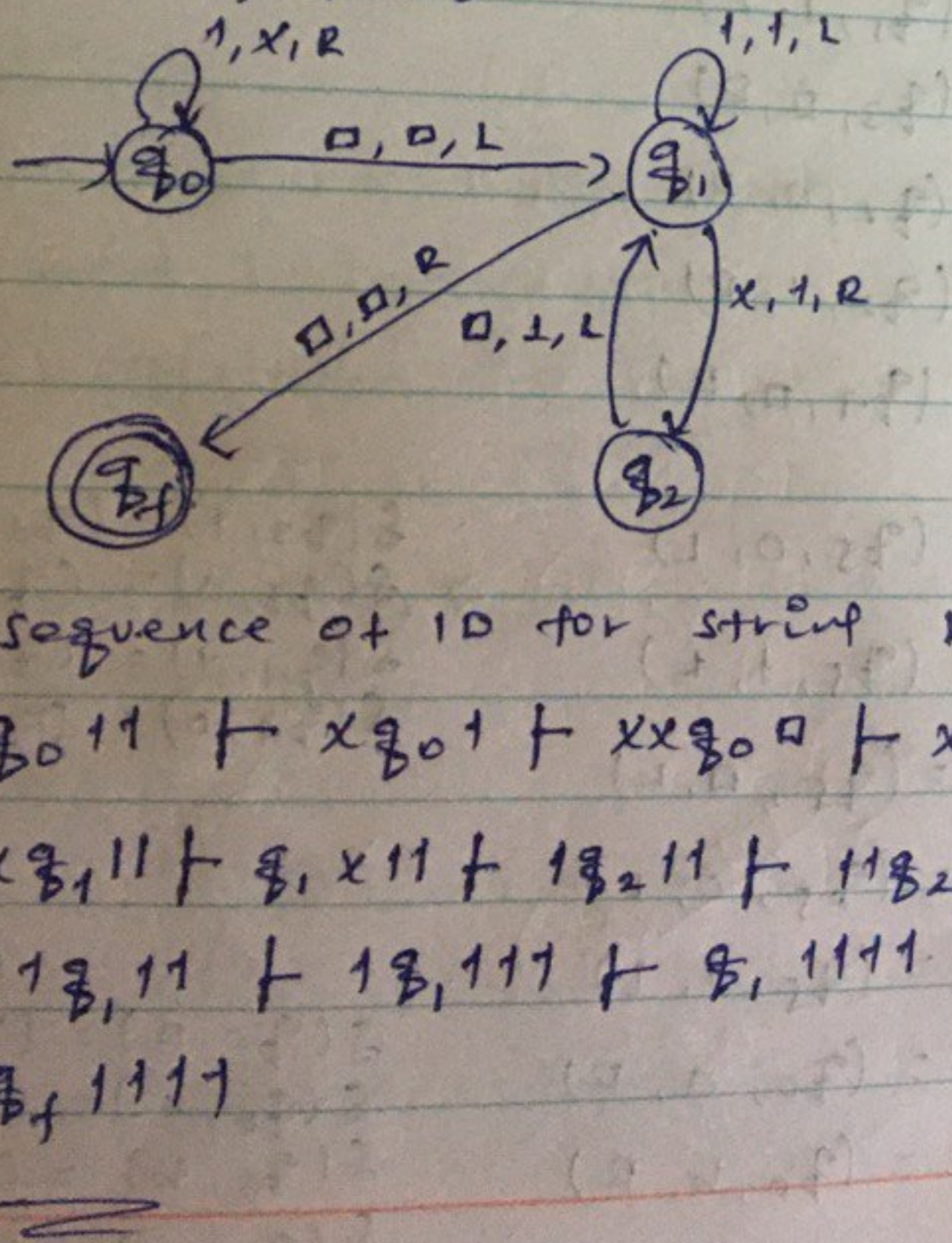
$\delta(q_2, 1) = (q_2, 1, R)$

$\delta(q_2, \square) = (q_1, 1, L)$

$\delta(q_1, 1) = (q_1, 1, L)$

$\delta(q_1, \square) = (q_f, \square, R)$

→ Transition graph :



The sequence of ID for string $W = 11$

$q_0 11 \vdash X q_0 1 \vdash X X q_0 \square \vdash X q_1 X \vdash X 1 q_2 \square \vdash$

$X q_1 11 \vdash q_1 X 11 \vdash 1 q_2 11 \vdash 11 q_2 1 \vdash 111 q_2 \square \vdash$

$11 q_1 11 \vdash 1 q_1 111 \vdash q_1 1111 \vdash q_1 \square 1111 \vdash$

$q_f 1111$

Eg 1 : Design a TM to copy a string of {0,1}

M: (Q, Σ, Γ, δ, q_0, □, F)

Q: {q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_f}

Σ: {1, 0}

Γ: {1, 0, □, a, b}

q_0 : {q_0}

F = {q_f}

□ : {□}

δ:

$\delta(q_0, 0) = (q_1, a, R)$

$\delta(q_0, 1) = (q_2, b, R)$

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

$\delta(q_1, 1) = (q_1, 1, R)$

$\delta(q_1, □) = (q_3, □, R)$

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

$\delta(q_2, 1) = (q_2, 1, R)$

$\delta(q_2, □) = (q_4, □, R)$

$\delta(q_3, □) = (q_5, 0, L)$

$\delta(q_4, □) = (q_5, 1, L)$

$\delta(q_5, □) = (q_5, □, L)$

$\delta(q_5, 0) = (q_5, 0, L)$

$\delta(q_5, 1) = (q_5, 1, L)$

$\delta(q_5, a) = (q_0, a, R)$

$\delta(q_5, b) = (q_0, b, R)$

$\delta(q_3, 1) = (q_3, 1, R)$

$\delta(q_3, 0) = (q_3, 0, R)$

$\delta(q_4, 1) = (q_4, 1, R)$

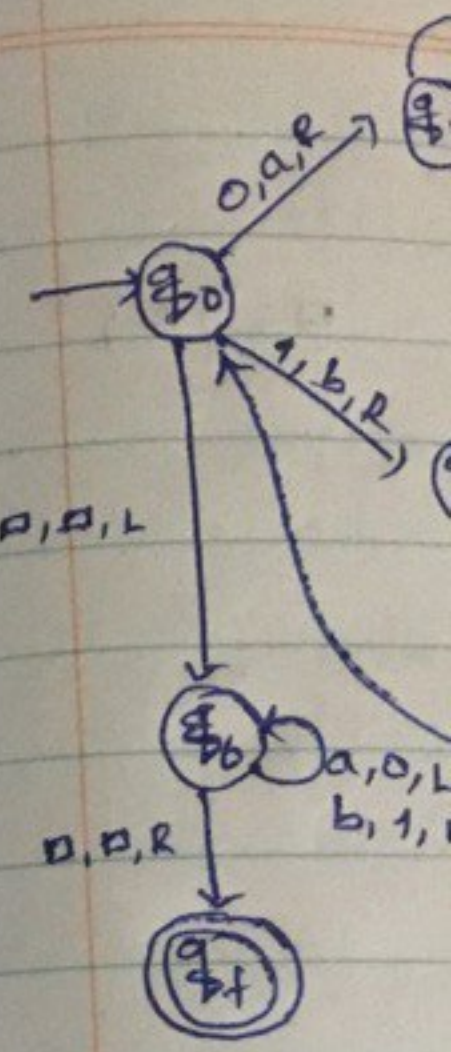
$\delta(q_4, 0) = (q_4, 0, R)$

$\delta(q_0, □) = (q_6, □, L)$

$\delta(q_6, a) = (q_6, 0, L)$

$\delta(q_6, b) = (q_6, 1, L)$

$\delta(q_6, □) = (q_f, □, R)$



Shift

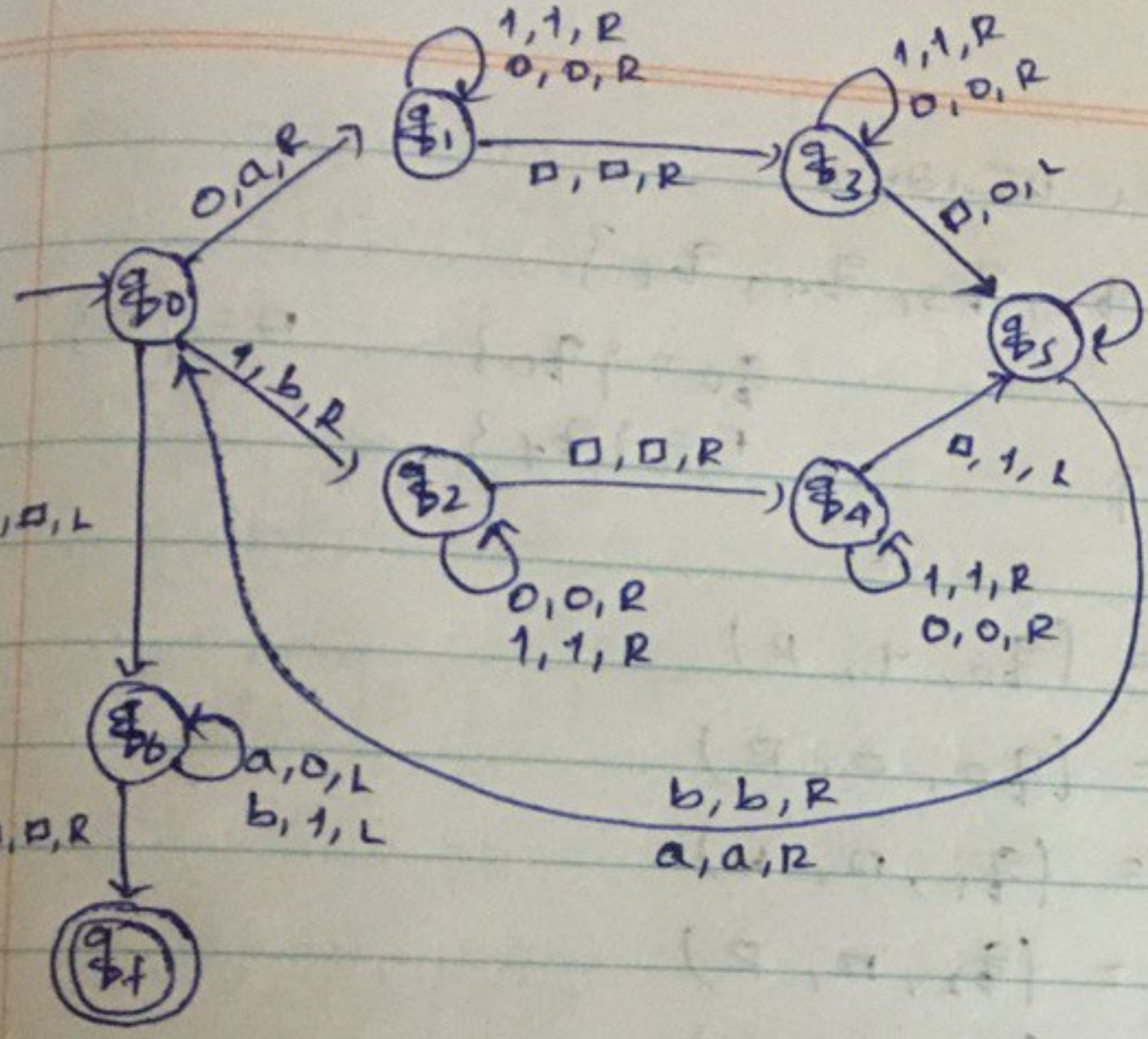
Design a

i) If inpu

Q =

Σ

δ:



Shift

Design a TM to shift a string one cell to the right

i) If input is in unary

□ | 1 | 1 | ---- | □

replaced by □ (first 1) ; replaced by 1 (last □)

$Q = \{q_0, q_1, q_f\}$ $\quad \Gamma = \{1, \square\}$ $\quad F = \{q_f\}$

$\Sigma = \{1\}$ $\quad q_0 = \{q_0\}$ $\quad \square = \{\square\}$

δ:

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

$\delta(q_1, 1) = (q_1, 1, R)$

$\delta(q_1, \square) = (q_f, 1, L)$

(ii) If input is in binary

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

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

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

$q_0 = \{q_0\}$

$F = \{q_f\}$

$\square = \{\square\}$

δ:

$\delta(q_0, 1) = (q_0, 1, R)$

$\delta(q_0, 0) = (q_0, 0, R)$

$\delta(q_0, \square) = (q_1, \square, L)$

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

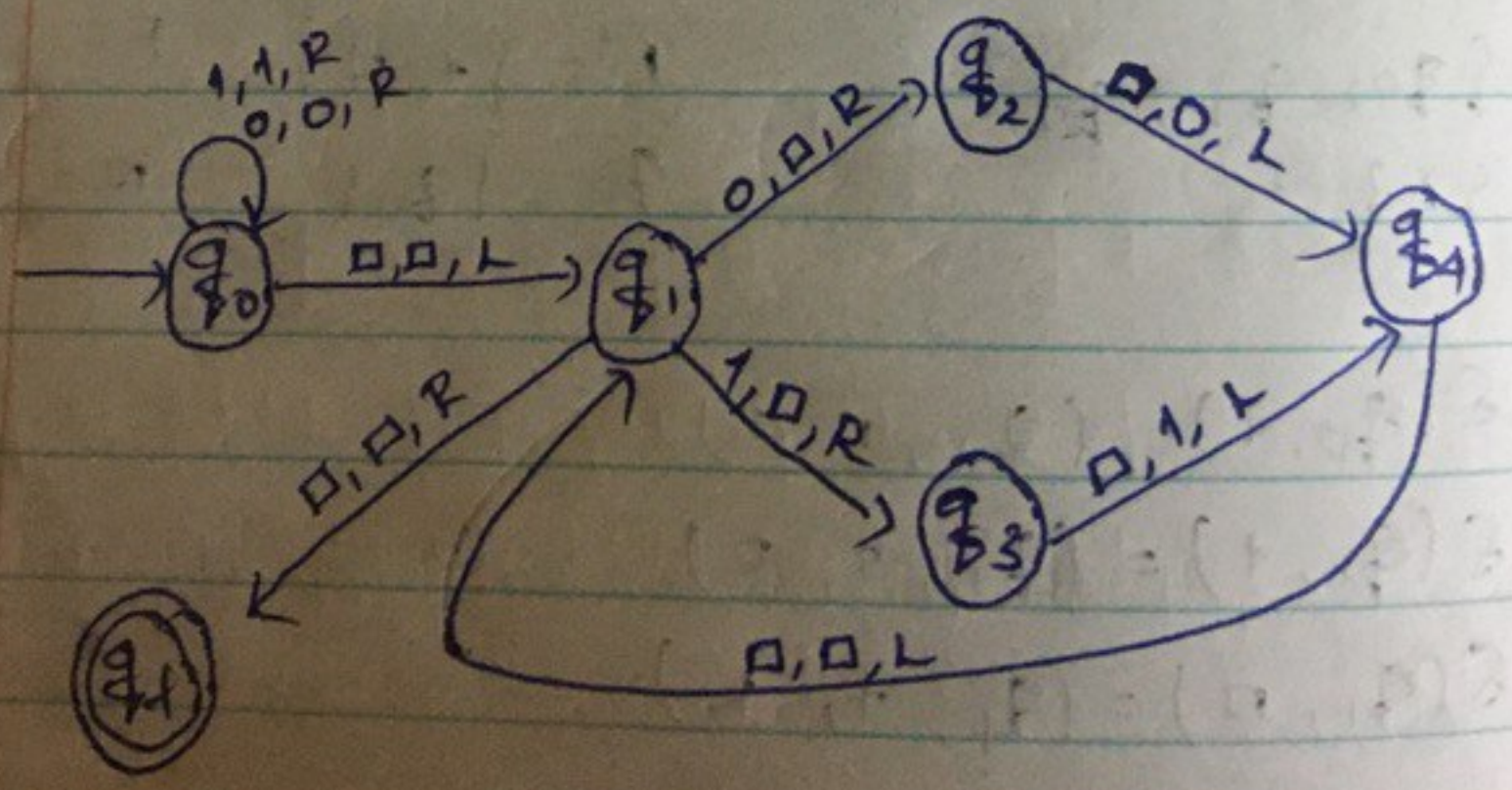
$\delta(q_1, 1) = (q_3, \square, R)$

$\delta(q_2, \square) = (q_4, 0, L)$

$\delta(q_3, \square) = (q_4, 1, L)$

$\delta(q_4, \square) = (q_1, \square, L)$

$\delta(q_1, \square) = (q_f, \square, R)$



ex: w = 110

pos: 0 1 2 3

1	1	0	1

↑ q0

1	1	0	

1	1	0	

1	1	0	

↑ q1

1	1	

1	1	

↑ q2

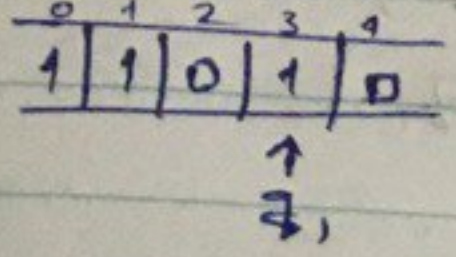
(*) Design

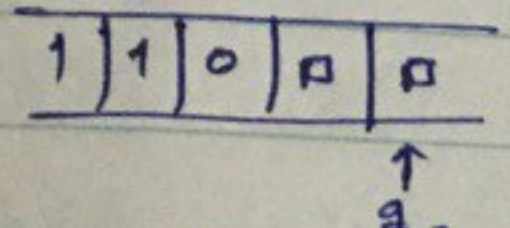
number

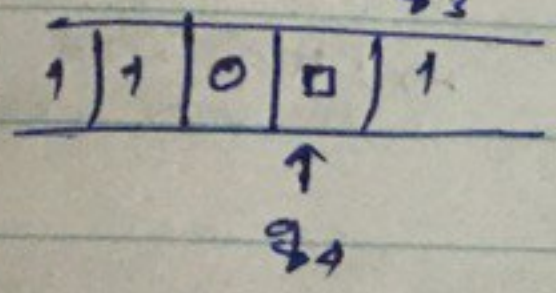
$\Sigma =$

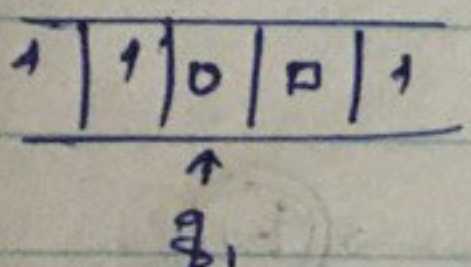
ex w = 1101

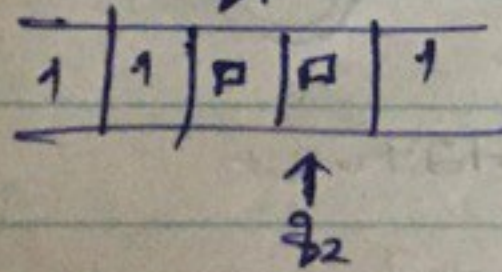
pos: 0 1 2 3 4

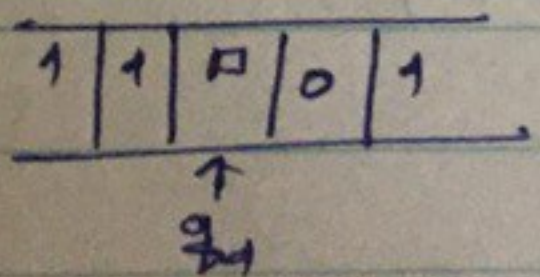


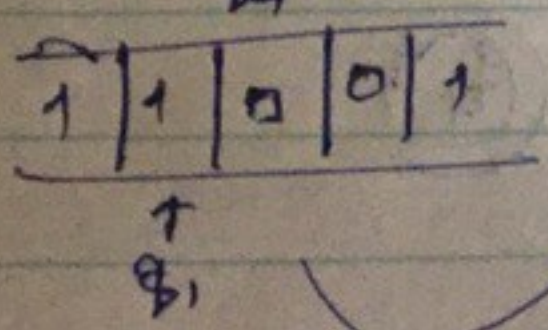


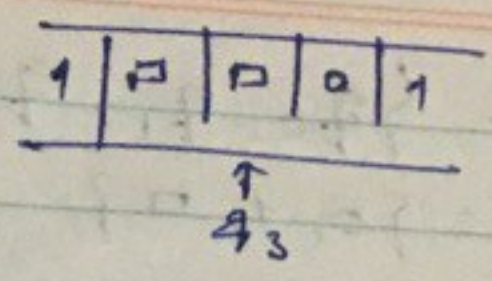


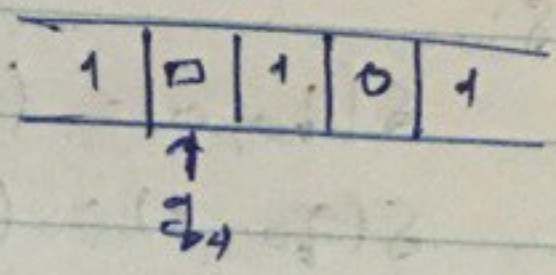


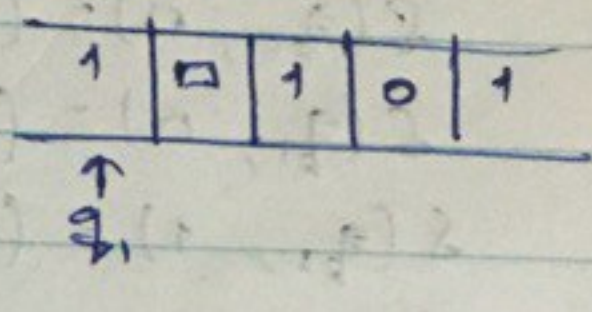


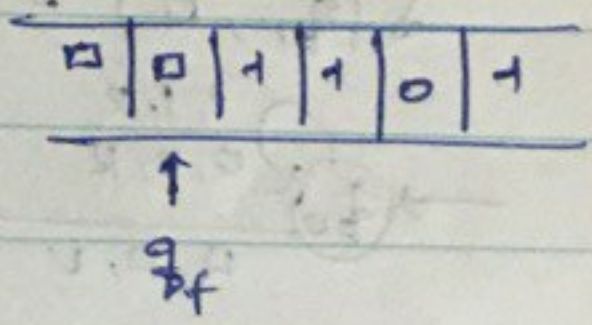












(*) Design a TM that adds 1 to a given binary number.

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

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

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

$q_0 = \{q_0\}$ $\square = \{\square\}$

$F = \{q_f\}$

δ:

$\delta(q_0, 0) = (q_0, 0, R)$

$\delta(q_0, 1) = (q_0, 1, R)$

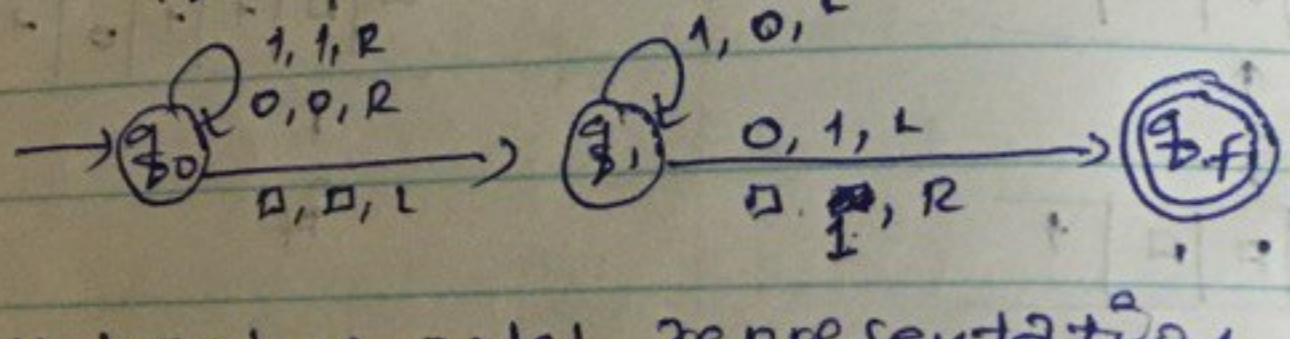
$\delta(q_0, \square) = (q_1, \square, L)$

$\delta(q_1, 0) = (q_f, 1, L)$

$\delta(q_1, 1) = (q_1, 0, L)$

$\delta(q_1, \square) = (q_f, \square, R)$

→ replaces last alphabet 0 skips, just a 1 again

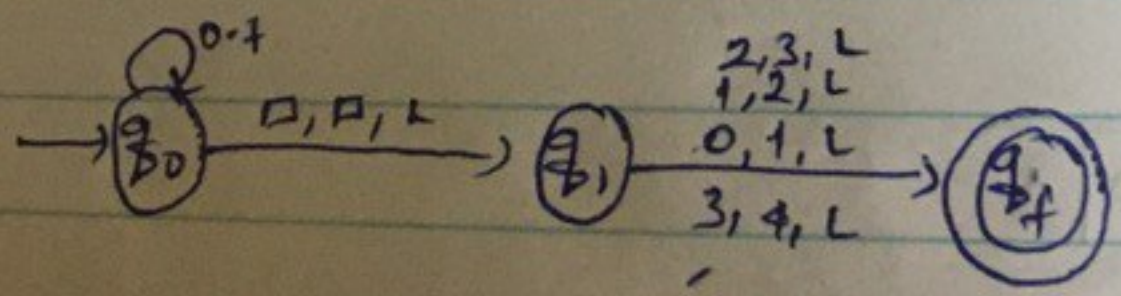


If input => octal representation

$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7\}$

$\Gamma = \{0, 1, - - -, 7, \square\}$

??



Example: Design a TM that converts unary no. to binary no.

$\Sigma = \{1\}$

$q_0 = \{q_0\}$ $F = \{q_f\}$ $\square = \{\square\}$

$\Gamma = \{1, \square, x,$

$Q = \{q_0, q_1,$

δ:

$\delta(q_0, 1) =$

$\delta(q_1, \square)$

$\delta(q_1, 0)$

$\delta(q_1, 1$

$\delta(q_2, 1$

$\delta(q_2,$

$\delta(q_2,$

$\delta(q_3,$

$\delta(q_3,$

$\delta(q_1,$

$\delta(q_3,$

$\Gamma = \{1, \square, X, 0\}$

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

δ:

$\delta(q_0, 1) = (q_1, X, L)$

$\delta(q_1, \square) = (q_2, 1, R)$

$\delta(q_1, 0) = (q_2, 1, R)$

$\delta(q_1, 1) = (q_1, 0, L)$

$\delta(q_2, 1) = (q_2, 1, R)$

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

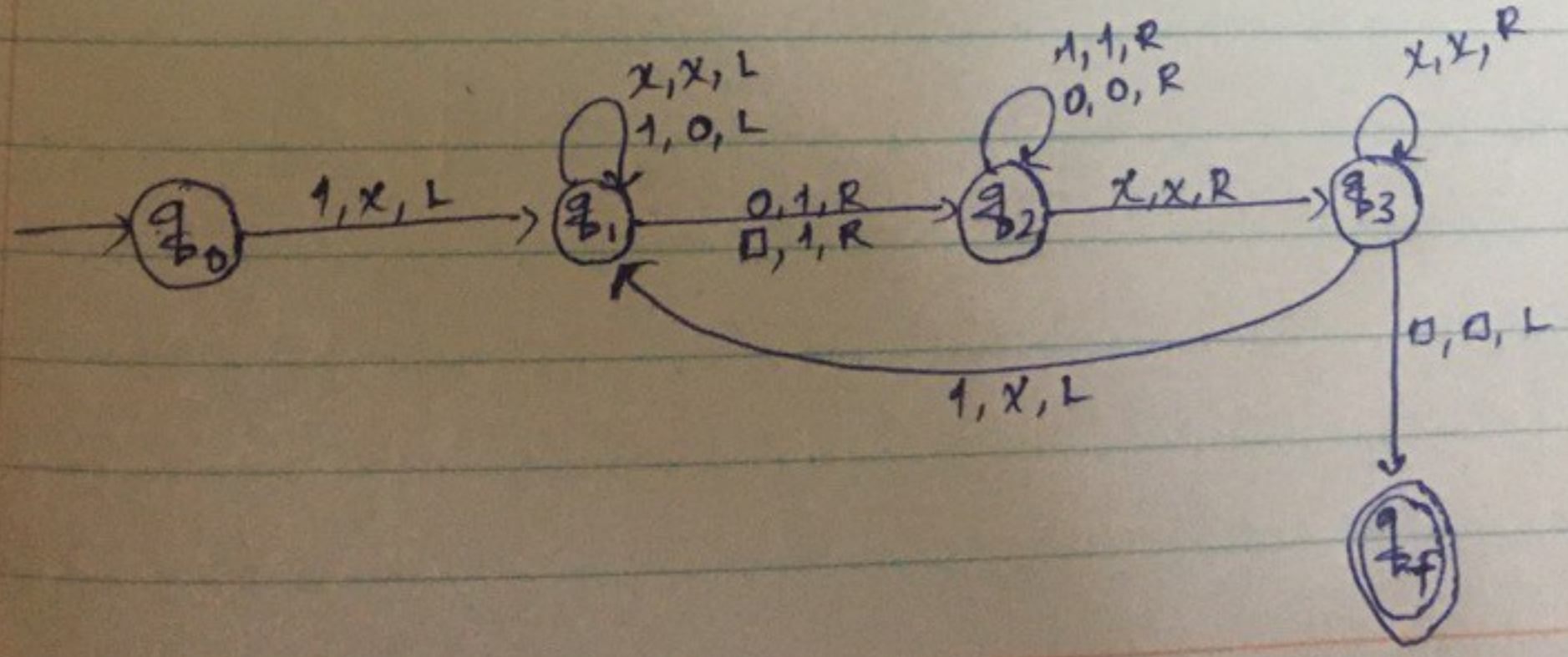
$\delta(q_2, X) = (q_3, X, R)$

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

$\delta(q_3, 1) = (q_1, X, L)$

$\delta(q_1, X) = (q_1, X, L)$

$\delta(q_3, \square) = (q_f, \square, L)$



(*) Design a TM for $L = \{a^n b^n c^n, n \geq 1\}$

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

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

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

$\delta(q_1, b) = (q_2, Y, R)$

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

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

$\delta(q_2, c) = (q_3, z, L)$

$\delta(q_3, z) = (q_3, z, L)$

$\delta(q_3, b) = (q_3, b, L)$

$\delta(q_3, Y) = (q_3, Y, L)$

$\delta(q_3, a) = (q_3, a, L)$

$\delta(q_3, x) = (q_0, x, R)$

$\delta(q_0, Y) = (q_4, Y, R)$

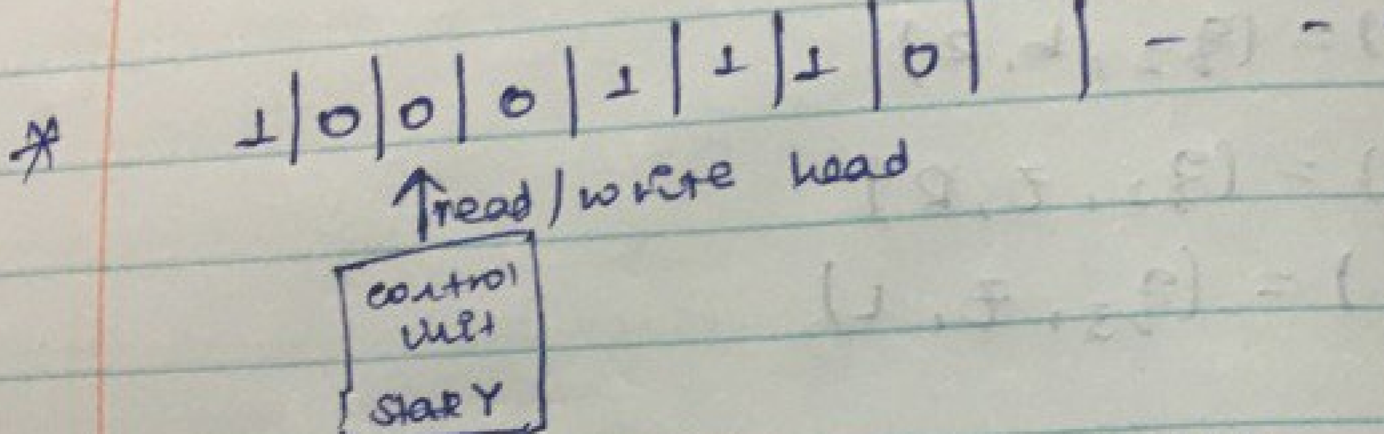
$\delta(q_4, Y) = (q_4, Y, R)$

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

$\delta(q_4, \square) = (q_f, \square, R)$

Turing-recognizable lang

A lang A that is recognized by a TM; i.e., there is a TM M such that M halts and accepts on any input w ∈ A, and M rejects or <u>loops</u> on any input w ∉ A.



- a TM is purely theoretical device
- control unit can be in one of a <u>finite</u> number of states
- Each tape cell holds one of a finite n° of symbols
- read/write head overwrites current symbol on each step
- read/write head moves one square to the left or right on each step.

AI and prog. languages use CFGs.

In CS Automata theory (automaton) is an abstract model of a machine that performs computations on an input by moving thru a series of states.

be described as decidable.

limitations of finite Automata

- FA can only count finite input
- There is no FA that can find and recognize set of binary strings of equal 0s & 1s

FA

→ is a machine that recognizes patterns (or abstract)

ex: We can design a DFA to accept the regular lang. a^*

→ But we want to design more powerful machines similar to DFAs that'll accept CFL (ex: $a^n b^n$)

* to handle a lang like $a^n b^n$ (a & b equal in n°. for $n \geq 0$), the machine needs to remember the n° of a's. To do this we use stack.

So PDA = NFA + stack

* Turing-decidable language

A lang A that is decided by a Turing machine; i.e, there is a TM M such that M halts and accepts on any input $w \in A$, and M halts and rejects on input $w \notin A$; i.e., looping cannot happen

Chap 1: Turing machine

→ attempt to understand essential nature of computation by studying properties of simple machine models.

goal: simplest machine that is as powerful as conventional computers.

* Abstract machines are foundation of all modern computers. B/C
  - simple computational models are easier to understand.
  - Leads to deeper understanding of computation

Abstract machines
- FSA
  ↳ can't recognize equal nº of 0's & 1's
- PDA: add read/write memory in z form of stack
  ↳ can't recognize palindromes
- TM: add memory in the form of an arbitrarily large array
  : general purpose computers

What is the use of Automata in CS
- The automata mtd enables computer scientists to understand how machines compute functions and solve problems, as well as what it means for a fu: to be defined as computable or for a question to