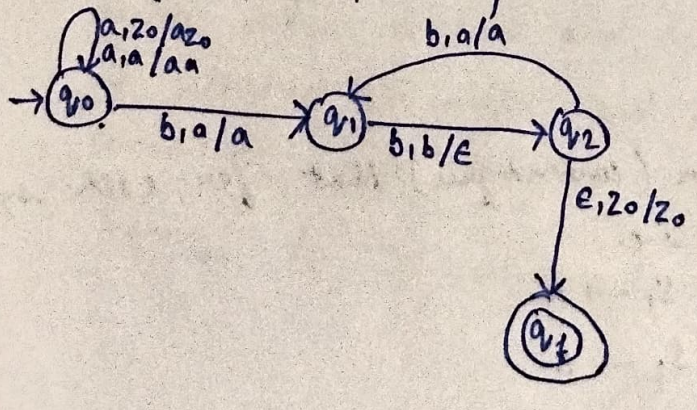
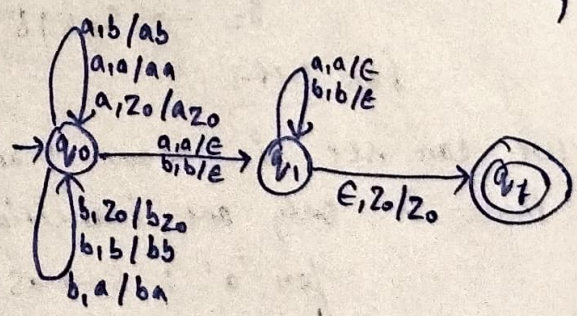
Assignment - 4

Ques1. Differentiate between NPDA and DPDA.

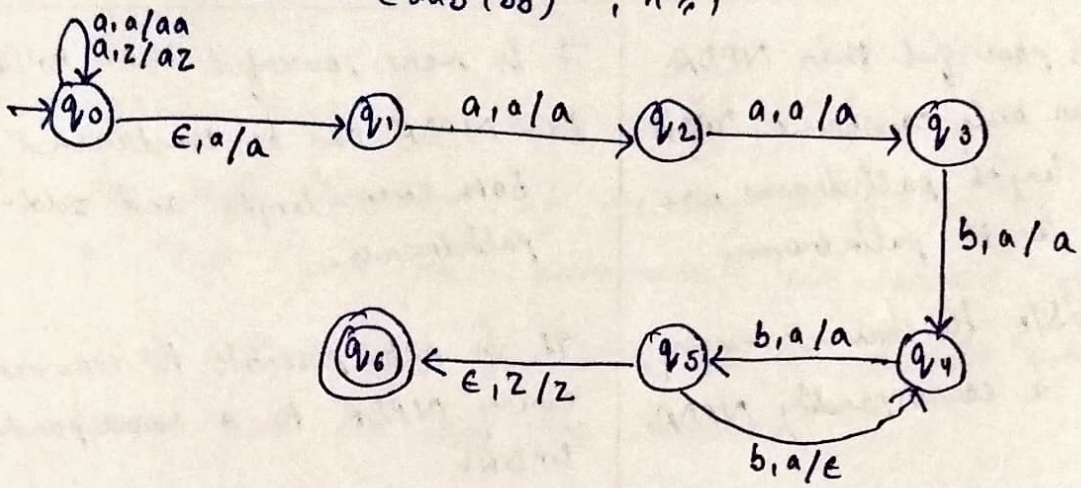
Ans

DPDA	NPDA
i) It is less powerful than NPDA. Ex. We can only construct DPDA for odd-length palindromes not for even length palindromes.	It is more powerful than DPDA. Ex- NPDA can be constructed for both even-length and odd-length palindromes.
ii) It is possible to convert every DPDA to a corresponding NPDA.	It is not possible to convert every NPDA to a corresponding DPDA.
iii) There is only one ~~subset~~ state transition from one state to another state for an input symbol.	There may or maynot be more than one state transition from one state to another state for same input symbol.
iv) It is known as Deterministic Pushdown Automata	It is known as Non-Deterministic Pushdown Automata
Ex- $L = a^n b^{2n}, n > 0$ $L = \{abb, aabb, ---\}$ 	Ex- $L = ww^R$ $w \in (a+b)^*$ $L = \{abba, abbbba, ---\}$ 

Ques 2. Construct an NPDA that accepts the language generated by the grammar:

$$S \rightarrow aSbb \mid aab$$

$$L = a^n \epsilon aab (bb)^n, \; n \geq 1$$



Ques 4. Show that $L = \{ w \in (a,b)^* : n_a(w) \neq n_b(w) \}$ is a deterministic context free language.

Ans Let, $L_1 = \{ w \in (a+b)^* \mid n_a(w) > n_b(w) \}$

$L_2 = \{ w \in (a+b)^* \mid n_a(w) < n_b(w) \}$

$\therefore \; L = L_1 \cup L_2$

For L_1: $S_1 \rightarrow aS_1b \mid bS_1a \mid S_1a \mid a$

For L_2: $S_2 \rightarrow aS_2b \mid bS_2a \mid S_2b \mid b$

For L: $S_1 \cup S_2$

$\Rightarrow$ $S \rightarrow S_1 \mid S_2 \mid aSb \mid bSa$

$S_1 \rightarrow S_1a \mid a$

$S_2 \rightarrow S_2b \mid b$

for DCFL

we can see in above grammar (context free) that for each symbol there is only one transition, i.e.

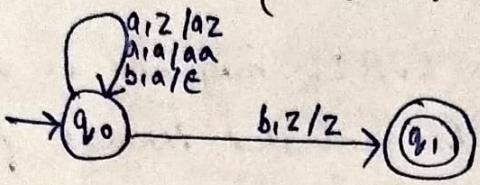
for 'a': $S \rightarrow aSb$ $\quad S_1 \rightarrow a$

for 'b': $S \rightarrow bSa$ $\quad S_2 \rightarrow b$

$\therefore$ This language is context free language.

Ques 3. Find an NPDA with two states for the language

$$L = \{ a^n b^{n+1} : n \geq 0 \}$$

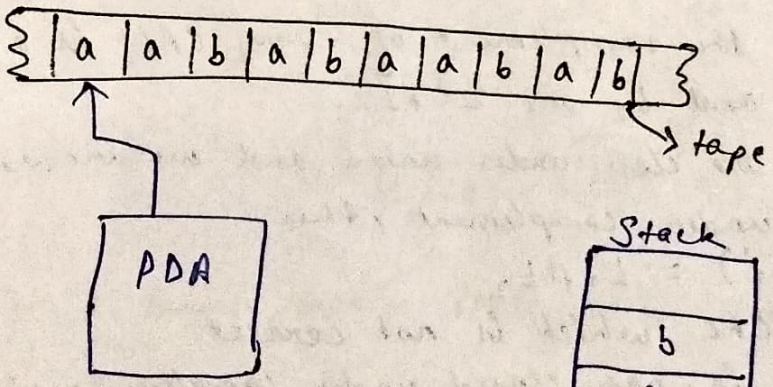


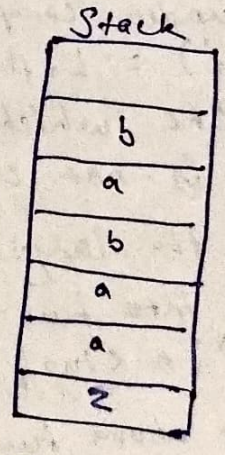
Ques 5. Show that $L = \{ ww : w \in \{a,b\}^* \}$ is not context free.

For this language, if we try to design or construct a PDA (push down automata) we cannot do it.

This is because the PDA cannot decide where a word ended and its copy has started.

Ex- Suppose, $w = aabab$ $\quad ww = aababaabab$





Here, if PDA reads w it cannot decide where w ends and next w started. It will read it as a complete single string.

Thus, it is not a CFL.

Ques 6. Show that the family of CFL is not closed under intersection and complementation.

Ans For Intersection

If L_1 and L_2 are CFLs then $L_1 \cap L_2$ may not be a CFL.

Proof: $L_1 = \{a^n b^n a^m \mid n, m \geq 0\}$ is generated through following CFG

$$S \rightarrow XA$$
$$X \rightarrow aXb \mid \epsilon$$
$$A \rightarrow Aa \mid \epsilon$$

$L_2 = \{a^n b^m a^m \mid n, m \geq 0\}$ is generated by the following CFG

$$S \rightarrow AX$$
$$X \rightarrow bXa \mid \epsilon$$
$$A \rightarrow Aa \mid \epsilon$$

$L_1 \cap L_2 = \{a^n b^n a^n \mid n \geq 0\}$ which is known not to be a CFL.

Therefore, CFL is not closed under intersection

For Complementation

If L_1 is CFL then L_1' may not be a CFL.

Proof: Assume the complement of every CFL is a CFL.

Let L_1 and L_2 are 2 CFLs.

Since CFLs are close under union and we are assuming they are closed under complement, then

$(L_1' \cup L_2') = L_1 \cap L_2$

is a CFL which is not correct

Therefore, CFL is not closed under complementation

Ques 7. Show that the language $L = \{a^n b^n : n \geq 0, n \neq 100\}$ is context free

Ans Let divide L into two languages: say L_1 & L_2

$L_1 = \{a^n b^n \mid n < 100\}$ $\quad L_2 = \{a^n b^n \mid n > 100\}$

We know that above two languages L_1 and L_2 are CFLs.

Also, we know CFLs are closed under union, that means

$L_1 \cup L_2$ is a CFL.

and $L = L_1 \cup L_2$, therefore L is CFL.

Assignment - 5

Ques 1. What is ID (Instantaneous Description)?

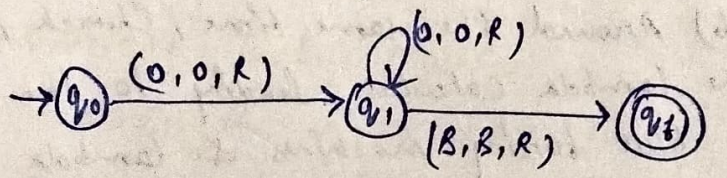
Ans The Instantaneous Description is called as an informal notation & explains how a pushdown automata computes the given input string and makes a decision that the given string is accepted or rejected.

A move from one instantaneous description to another is denoted by the symbol '$\vdash$'.

Generally, the leftmost symbol indicates the top of the stack & and the bottom at the right end. This type of triple notation is called an ID (Instantaneous Description) of PDA.

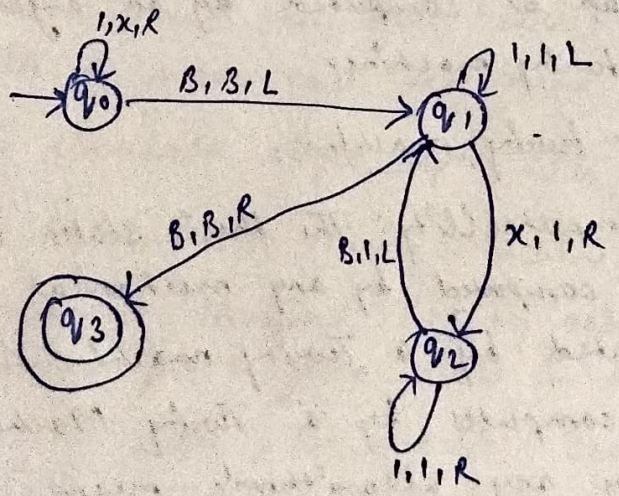
Ques 2. Construct a turing machine that accepts the language denoted by the regular expression 00^*?

Ans



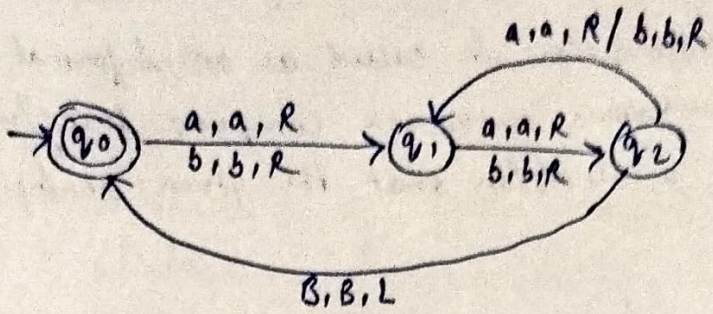
Ques 3. Design a turing machine that copies strings of 1's.

Ans



Ques4. Design a turing machine that copies will accept the language $L = \{w : |w| \text{ is even}\}$ on $\{a, b\}$?

Ans



Ques5. What is Turing's thesis / Church's thesis?

Ans Church / Turing Thesis

- Historical Context

  i) Alan Turing (1936) In his seminar paper "On computable numbers, with an Application" Turing introduced the concept of Turing Machines & formulated his thesis.

  ii) Alonzo Church (1936) Around the same time, Church proposed his thesis using the lambda calculus, leading to the recognition of the equivalence b/w turing machines & lambda calculus.

- Formal Definition

  Any function that can be computed by an algorithm can be computed by a turing machine.

- Key points of Church - turing machine

  i) Equivalence of computability: The thesis states that if a function can be computed by any mechanical procedure, it can be computed by a Turing machine & any function that cannot be computes by a Turing Machine cannot be computed by any algorithmic mean.

  ii) Formulations: -

  a) Turing formulation Introduced by Alan Turing, this involves the concept of the turing machine.

b) Church's formulation Independently proposed by Alonzo Church this involves the lambda calculus, a formal system in mathematical logic & computer science for expression.

iii) Implication:

* Universality of Turing Machine
* Limitations of Computability

Ques6. Explain the various types of Turing Machine?

Ans i) Standard (Single tape) Turing Machine

- A finite set of states.
- An infinite tape divided into cells, each holding a symbol from a finite alphabet.
- A tape head that can read & write symbols on the tape & move left or right.

ii) Multitape Turing Machine

- Extends the single tape Turing Machine to have multiple tapes, each with its own head.
- The transition function depends on the state & the symbols under all tape heads. The machine can read, write & move each tape head independently.
- Can simulate single tape TMs efficiently.

iii) Non-Deterministic Turing Machine

- Extends the standard TM by allowing multiple possible transitions for a given state & tape symbol.
- At each steps, the machine can choose b/w different transitions, effectively exploring multiple computation paths simultaneously.
- Useful for theoretical purposes, particularly in complexity theory.

iv) Universal Turing Machine

- A TM capable of simulating any other TM (Turing Machine)
- Takes a description of a TM & its input as input and simulates the behaviour of the given machine on that input.

v) Multi Dimensional Turing Machine

- Extends the tape to multiple dimensions
- The tape head moves in multiple directions in the multidimensional space.
- Can moves more complex data structures & computations more naturally than a single dimensional type.

Ques 7. Show that the class of deterministic Turing Machines and the class of non-deterministic Turing Machines are equivalent.

Ans The equivalence between the class of deterministic Turing Machine (DTMS) and the class of (NTMS) is established by demonstrating that any computation performed by an NTM can be simulated by a DTM. This result indicates that both types of machines recognize the same class of languages, known as the recursively enumerable languages.

Step by Step outline to show equivalence.

1. Definition

DTM: A DTM has a single transition for each state & input symbol pair. The computation path is unique & follows a single sequence of steps.

NTM: An NTM can have multiple possible transitions for a given state & input symbol pair. It can branch into multiple computation paths simultaneously, exploring all possibilities.

2. Simulation of an NTM by a DTM

To show that any NTM can be simulated by a DTM, we need to construct a DTM that can simulate the behaviour of the NTM. The DTM will systematically explore all possible computation paths of the NTM.

3. Formal Construction

let $N = (Q_N, \Sigma, \Gamma, q_{0N}, q_{accept N}, q_{reject N})$ be an NTM

$D = (Q_D, \Sigma, \Gamma, \delta_D, q_{0D}, q_{accept_D}, q_{reject_D})$ for DTM.

Ques 8. what is linear bounded automata (LBA)? Explain with an example.

Ans A linear bounded automata is similar to turing machine with some properties stated below:

- TM with non-deterministic logic.
- TM with multi-track, and
- TM with a bounded finite length of the tape.

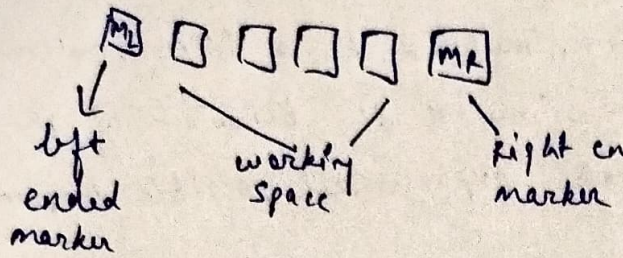
Tuples used in LBA

$M = (Q, T, E, q_0, M_L, M_R, \delta, F)$

where,

$Q \rightarrow$ A finite set of transition states
$T \rightarrow$ Tape alphabet
$E \rightarrow$ Input alphabet
$q_0 \rightarrow$ Initial state
$M_L \rightarrow$ Left bound of tape
$M_R \rightarrow$ Right bound of tape
$\delta \rightarrow$ transition function
$F \rightarrow$ A finite set of final states

Diagramatic Representation



Example

- $L = \{a^{n!} \mid n \geq 0\}$
- $L = \{wn \mid w \text{ from } \{a,b\}^+, n \geq 1\}$
- $L = \{wwwR \mid w \text{ from } \{a,b\}^+\}$

Ques 9. Write short note on Halting Problem.

Ans Given a program / algorithm will ever halt or not?

Halting means that the program on certain input will accept it & halt or reject it & halt & it would never go into an infinite loop. Basically halting means terminating. So can we have an algorithm that will tell that the given program will halt or not. In terms of turing machine, will it terminate when run on some machine with some particular given input strings.

The answer is no we cannot design a generalised algorithm which can appropriately say that given a program will ever halt or not?