Scheme and Solution

Internal Test – III, Nov. 2018

Sub: Automata Theory and Computability (15CS54)

Prepared By: Sagarika Behera, Dept. of CSE, CMRIT, Bangalore

USN

Internal Assessment Test III –Nov. 2018

Sub:	Automata Theory and Computability					Sub Code:	15CS54	Branch:	CSE
Date:	20/11/18	Duration:	90 min's	Max Marks:	50	Sem/Sec:	5/CSE(A,B,C)		OBE

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1	Design a Turing Machine for **L={$a^nb^nc^n$|n≥1}.** Write the transition function and transition diagram for the same.	[10]	CO5	L3
2	Explain the Multi tape Turing Machine and Non-deterministic Turing Machine with neat block diagram.	[10]	CO4	L1
3	State and prove pumping lemma for Context Free Language(CFL). Show that **L= {$a^nb^nc^n$|n≥1}** is not context free.	[10]	CO4	L1 &L3
4	Write down the closure properties of CFL. Prove that the family of CFL's are closed under union, concatenation and star closure.	[10]	CO4	L2
5	Write Short notes on: (a) Post correspondence problem (b) Halting problem of TM	[2*5]	CO5	L1
6	Define a Turing machine. Explain the working of a basic TM with a neat diagram. Also define the language accepted by TM.	[10]	CO4	L1
7	Design a TM for L={**w**| w is a palindrome and **w Ɛ {a,b}*}.** Write the transition function for the constructed TM and write the sequence of ID's for the input string **w=baab .**	[10]	CO5	L3
8	Design a TM to perform subtraction of two integers. Write the transition function and transition diagram for the same.	[10]	CO5	L3
9	Define Chomsky Normal Form (CNF). Convert the following CFG to CNF. S→ABC | BaB A→aA|BaC|aaa B→bBb|a|D C→CA|AC D→ϵ	[10]	CO4	L3
10	Define Greibach Normal Form(GNF). Convert the following CFG to GNF. S→aAa|bBb|ϵ	[10]	CO4	L3

$A \to C|a$
$B \to C|b$
$C \to CDE|\epsilon$
$D \to A|B|ab$

Q.1. Design a Turing Machine for $L=\{a^n b^n c^n | n \geq 1\}$. Write the transition function and transition diagram for the same. **[5+5]**

Ans:

The transition function is given below.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Note: Draw the transition diagram according to the above transition function.

Q.2. Explain the Multi tape Turing Machine and Non-deterministic Turing Machine with neat block diagram. [5+5]

Ans:

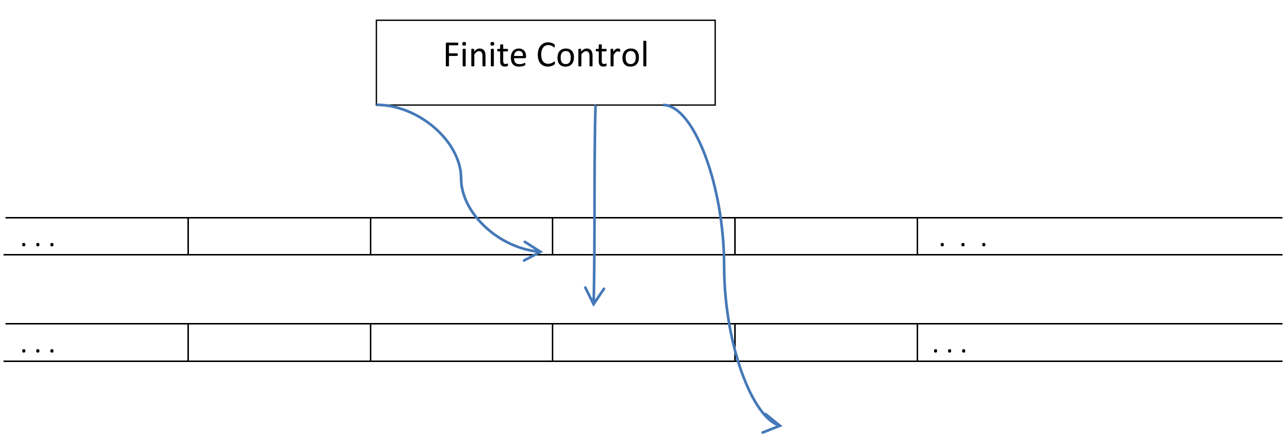
<u>VARIANTS OF TURING MACHINE:</u>

There are two new models of Turing machines:

1. MULTITAPE TURING MACHINE
2. NON-DETERMINISTIC TURING MACHINE

MULTITAPE TURING MACHINE

Multi-tape Turing Machines have multiple tapes where each tape is accessed with a separate head. Each head can move independently of the other heads. Initially the input is on tape 1 and others are blank. At first, the first tape is occupied by the input and the other tapes are kept blank. Next, the machine reads consecutive symbols under its heads and the TM prints a symbol on each tape and moves its heads.



. . .					. . .

A Multi-tape Turing machine can be formally described as a 7-tuple (Q,Σ,Γ, B, δ, q_0, F) where −

- **Q** is a finite set of states
- Σ is a finite set of inputs
- Γ is the tape alphabet
- **B** is the blank symbol
- **δ** is a relation on states and symbols where

  $\delta: Q \times \Gamma^k \rightarrow Q \times (\Gamma \times \{\text{Left, Right, Stationary}\})^k$

  where there is **k** number of tapes
- $\mathbf{q_0}$ is the initial state
- **F** is the set of final states

  In each move the machine M:

  (i) Enters a new state

  (ii) A new symbol is written in the cell under the head on each tape

  (iii) Each tape head moves either to the left or right or remains stationary.

NON-DETERMINISTIC TURING MACHINE

In a Non-Deterministic Turing Machine, **for every state and symbol, there are a group of actions the TM can have. So, here the transitions are not deterministic.** The computation of a non-deterministic Turing Machine is a tree of configurations that can be reached from the start configuration.

An input is accepted if there is at least one node of the tree which is an accept configuration, otherwise it is not accepted. If all branches of the computational tree halt on all inputs, the non-deterministic Turing Machine is called a **Decider** and if for some input, all branches are rejected, the input is also rejected.

A non-deterministic Turing machine can be formally defined as a 7-tuple (Q, $\sum$, Γ, δ, q_0, B, F) where −

- **Q** is a finite set of states
- Γ is the tape alphabet
- $\sum$ is the input alphabet
- **δ** is a transition function;

  $\delta : Q \times \Gamma \rightarrow 2^{(Q \times \Gamma \times \{\text{Left, Right}\})}$

- $\mathbf{q_0}$ is the initial state
- **B** is the blank symbol
- **F** is the set of final states

Q.3. State and prove pumping lemma for Context Free Language (CFL). Show that $L = \{a^n b^n c^n | n \geq 1\}$ is not context free. [5+5]

Ans:

If **L** is a context-free language, there is a pumping length **p** such that any string **w ∈ L** of length **≥ p** can be written as **w = uvxyz**, where **vy ≠ ε**, **|vxy| ≤ p**, and for all **i ≥ 0,** $uv^ixy^iz \in L$.

Let **L** is context free. Then, **L** must satisfy pumping lemma.

At first, choose a number **n** of the pumping lemma. Then, take z as $a^nb^nc^n$.

Break **z** into **uvwxy,** where

|vwx| ≤ n and vx ≠ ε.

Hence **vwx** cannot involve both as and cs, since the last a and the first c are at least (n+1) positions apart. There are two cases −

Case 1 − **vwx** has no cs. Then **vx** has only as and bs. Then **uwy**, which would have to be in **L**, has **n** cs, but fewer than **n** as or bs.

Case 2 − **vwx** has no as.

Here contradiction occurs.

Hence, **L** is not a context-free language.

Q.4. Write down the closure properties of CFL. Prove that the family of CFL's are closed under union, concatenation and star closure. [10]

Ans:

Context-free languages are **closed** under −

- Union
- Concatenation
- Kleene Star operation

Union

Let L_1 and L_2 be two context free languages. Then $L_1 \cup L_2$ is also context free.

Example

Let $L_1 = \{ a^n b^n , n > 0\}$. Corresponding grammar G_1 will have P: $S1 \rightarrow aAb|ab$

Let $L_2 = \{ c^m d^m , m \geq 0\}$. Corresponding grammar G_2 will have P: $S2 \rightarrow cBb|\ \varepsilon$

Union of L_1 and L_2, $L = L_1 \cup L_2 = \{ a^n b^n \} \cup \{ c^m d^m \}$

The corresponding grammar G will have the additional production $S \rightarrow S1 \mid S2$

Concatenation

If L_1 and L_2 are context free languages, then L_1L_2 is also context free.

Example

Union of the languages L_1 and L_2, $L = L_1L_2 = \{ a^n b^n c^m d^m \}$

The corresponding grammar G will have the additional production $S \rightarrow S1\ S2$

Kleene Star

If L is a context free language, then L* is also context free.

Example

Let $L = \{ a^n b^n , n \geq 0\}$. Corresponding grammar G will have P: $S \rightarrow aAb|\ \varepsilon$

Kleene Star $L_1 = \{ a^n b^n \}^*$

The corresponding grammar G_1 will have additional productions $S1 \rightarrow SS_1 \mid \varepsilon$

Context-free languages are **not closed** under −

- **Intersection** − If L1 and L2 are context free languages, then L1 ∩ L2 is not necessarily context free.
- **Intersection with Regular Language** − If L1 is a regular language and L2 is a context free language, then L1 ∩ L2 is a context free language.
- **Complement** − If L1 is a context free language, then L1' may not be context free.

Q.5. (a) **Post Correspondence Problem:**

The Post Correspondence Problem (PCP), introduced by Emil Post in 1946, is an undecidable decision problem. The PCP problem over an alphabet ∑ is stated as follows −

Given the following two lists, **M** and **N** of non-empty strings over ∑ −

$M = (x_1, x_2, x_3, \ldots\ldots\ldots, x_n)$

$N = (y_1, y_2, y_3, \ldots\ldots\ldots, y_n)$

We can say that there is a Post Correspondence Solution, if for some $i_1, i_2, \ldots\ldots\ldots\ldots\ i_k$, where $1 \leq i_j \leq n$, the condition $x_{i1} \ldots\ldots x_{ik} = y_{i1} \ldots\ldots y_{ik}$ satisfies.

Example 1

Find whether the lists

M = (abb, aa, aaa) and N = (bba, aaa, aa)

have a Post Correspondence Solution?

Solution

	x_1	x_2	x_3
M	Abb	aa	aaa
N	Bba	aaa	aa

Here,

$x_2x_1x_3$ = 'aaabbaaa'

and $\mathbf{y_2 y_1 y_3}$ **= 'aaabbaaa'**

We can see that

$\mathbf{x_2 x_1 x_3 = y_2 y_1 y_3}$

Hence, the solution is **i = 2, j = 1, and k = 3.**

Example 2

Find whether the lists **M = (ab, bab, bbaaa)** and **N = (a, ba, bab)** have a Post Correspondence Solution?

Solution

	x_1	x_2	x_3
M	ab	bab	bbaaa
N	a	ba	bab

In this case, there is no solution because −

$\mathbf{| x_2 x_1 x_3 | \neq | y_2 y_1 y_3 |}$ (Lengths are not same)

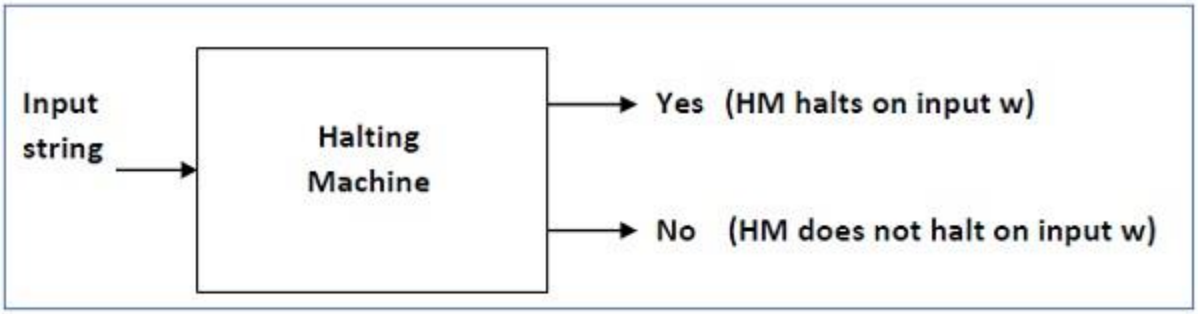
Hence, it can be said that this Post Correspondence Problem is **undecidable**.

(b) Halting Problem of TM:

Input − A Turing machine and an input string **w**.

Problem − Does the Turing machine finish computing of the string **w** in a finite number of steps? The answer must be either yes or no.

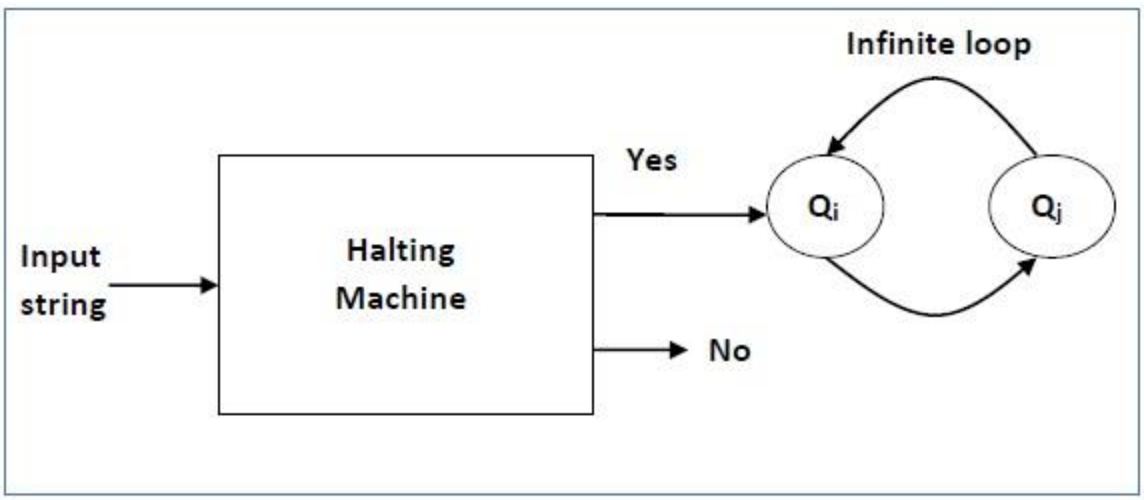
Proof − At first, we will assume that such a Turing machine exists to solve this problem and then we will show it is contradicting itself. We will call this Turing machine as a **Halting machine** that produces a 'yes' or 'no' in a finite amount of time. If the halting machine finishes in a finite amount of time, the output comes as 'yes', otherwise as 'no'. The following is the block diagram of a Halting machine −



Now we will design an **inverted halting machine (HM)'** as −

- If **H** returns YES, then loop forever.
- If **H** returns NO, then halt.

The following is the block diagram of an 'Inverted halting machine' −



Further, a machine **$(HM)_2$** which input itself is constructed as follows −

- If $(HM)_2$ halts on input, loop forever.
- Else, halt.

Here, we have got a contradiction. Hence, the halting problem is **undecidable**.

Q.6. Define a Turing machine. Explain the working of a basic TM with a neat diagram. Also define the language accepted by TM. [10]

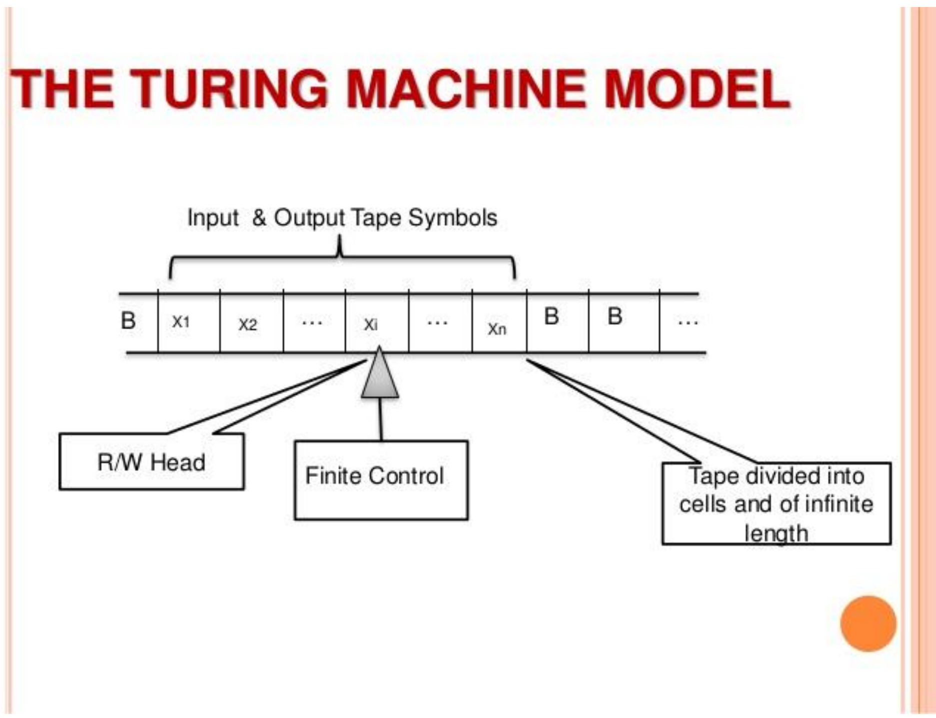
Ans: A Turing Machine is an accepting device which accepts the languages (recursively enumerable set) generated by type 0 grammars. It was invented in 1936 by Alan Turing.

Definition

A Turing Machine (TM) is a mathematical model which consists of an infinite length tape divided into cells on which input is given. It consists of a head which reads the input tape. A state register stores the state of the Turing machine. After reading an input symbol, it is replaced with another symbol, its internal state is changed, and it moves from one cell to the right or left. If the TM reaches the final state, the input string is accepted, otherwise rejected.

A TM can be formally described as a 7-tuple $(Q, X, \sum, \delta, q_0, B, F)$ where −

- **Q** is a finite set of states
- **X** is the tape alphabet
- $\sum$ is the input alphabet
- **δ** is a transition function; $\delta : Q \times X \rightarrow Q \times X \times \{Left\_shift, Right\_shift\}$.
- $\mathbf{q_0}$ is the initial state
- **B** is the blank symbol
- **F** is the set of final states



A TM accepts a language if it enters into a final state for any input string w. A language is **recursively enumerable** (generated by Type-0 grammar) if it is accepted by a Turing machine.

A TM decides a language if it accepts it and enters into a rejecting state for any input not in the language. A language is **recursive** if it is decided by a Turing machine.

There may be some cases where a TM does not stop. Such TM accepts the language, but it does not decide it.