5 Turing Machine

5.1 BASICS OF TURING MACHINE

Introduction:

- A Turing machine is a control unit with a finite number of states and unbounded tape(s).
- Although, the Turing machine seems to be very simple from the structural point of view, it is as powerful as simple it looks.
- There are so many problems that are unsolvable by push down automaton. Such problems are solved using a Turing Machine.
- According to the Turing thesis, the only general type of automaton whose power almost matches with a computer is nothing but the Turing machine.

Standard turing machine:

- Turing machine uses the tape as a buffer, in other words, we can say that it is a temporary storage area.
- There is a read-write head which is a part of the Turing machine, and it is associated with the tape to read and write one symbol at a time onto the tape at the time of traversing the tape in the right or left direction.

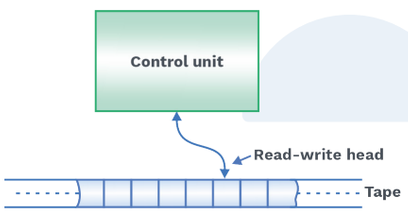


Fig. 5.1 Diagram of a Turing Machine

If M is a Turing machine, then the definition of M should be
$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

Where
Q is the set of finite states,
Σ is the set of input alphabets,
Γ is a finite set of symbols called the tape alphabet,
δ is the transition function, defined as
$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
where L = left movement & R = right movement,
$B \in \Gamma$ is a special symbol called the blank,
$q_0 \in Q$ is the initial state,
$F \subseteq Q$ is the set of final states.

Note: In the definition of a Turing machine, we assume that E $\Sigma \subseteq \Gamma$- {B}, that the input alphabet is a subset of the tape alphabet, not including the blank.

- The interpretation of the transition function δ on Q × Γ states the working principle of the Turing machine.
- The present state and the recent tape symbol, which is being read, are passed as the arguments of the transition function.
- As a result, the control of the Turing machine will move to a new state, the previous tape symbol may be changed with a new one, and the read write head will be shifted to the left or to the right.
- The shift or move symbol will decide about the movement of the read write head after the write of the new symbol in the place of the old one, whether to shift left or right by one cell.

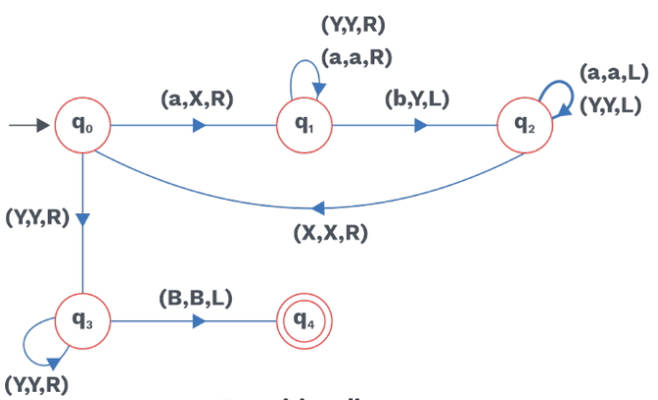
Note: Turing machine can't accept epsilon.

Turing machine as acceptor:

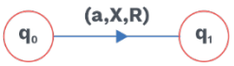
SOLVED EXAMPLES

Q1 **Design a TM for L = $a^n b^n$ / n ≥ 1.**

Sol:



Transition diagram

The transition 

means q_0 by getting 'a'; it is replacing 'a' with 'X' & going to state q_1 and read-write head move one cell right.

We can even give transition table for $L = a^n b^n / n \geq 1$ as shows below:

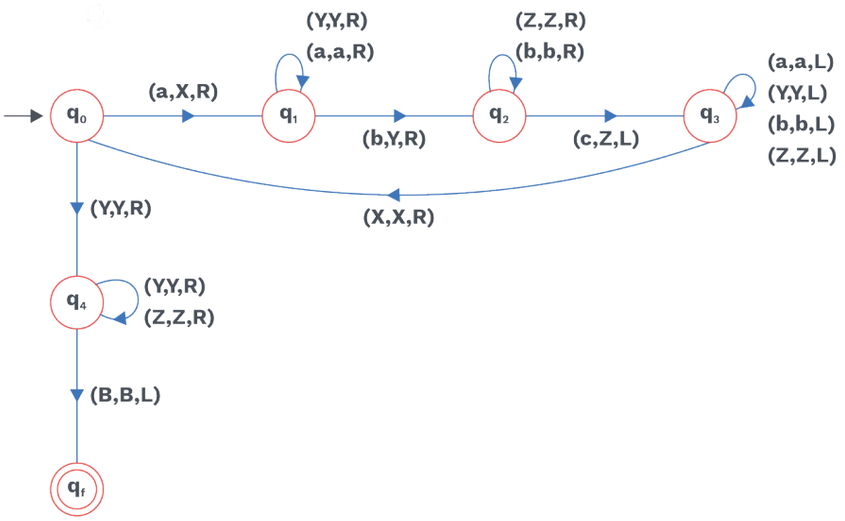
$\delta(q_0, a) \rightarrow (q_1, X, R)$
$\delta(q_0, Y) \rightarrow (q_3, Y, R)$
$\delta(q_1, a) \rightarrow (q_1, a, R)$
$\delta(q_1, Y) \rightarrow (q_1, Y, R)$
$\delta(q_1, b) \rightarrow (q_2, Y, L)$
$\delta(q_2, a) \rightarrow (q_2, a, L)$
$\delta(q_2, Y) \rightarrow (q_2, Y, L)$
$\delta(q_2, X) \rightarrow (q_0, X, R)$
$\delta(q_3, Y) \rightarrow (q_3, Y, R)$
$\delta(q_3, B) \rightarrow (q_4, B, L)$

If the string will be in language then it will get accepted and halt at state q_4 otherwise it will either go dead state or dead configuration.

Note: The transition table and transition diagram both are equivalent in power.

Q2 Design a Turing machine for $L = \{a^n b^n c^n / n \geq 1\}$.

Sol:



Transition diagram

The transition table for $L = \{a^n b^n c^n \mid n \geq 1\}$ is given below:

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

$\delta(q_0, Y) \rightarrow (q_4, Y, R)$
$\delta(q_4, Y) \rightarrow (q_4, Y, R)$
$\delta(q_4, Z) \rightarrow (q_4, Z, R)$
$\delta(q_4, B) \rightarrow (q_f, B, L)$
$\delta(q_1, b) \rightarrow (q_2, Y, R)$
$\delta(q_1, a) \rightarrow (q_1, a, R)$
$\delta(q_1, Y) \rightarrow (q_1, Y, R)$
$\delta(q_2, b) \rightarrow (q_2, b, R)$
$\delta(q_2, Z) \rightarrow (q_2, Z, R)$
$\delta(q_2, c) \rightarrow (q_3, Z, L)$
$\delta(q_3, b) \rightarrow (q_3, b, L)$
$\delta(q_3, Z) \rightarrow (q_3, Z, L)$
$\delta(q_3, Y) \rightarrow (q_3, Y, L)$
$\delta(q_3, a) \rightarrow (q_3, a, L)$
$\delta(q_3, X) \rightarrow (q_0, X, R)$

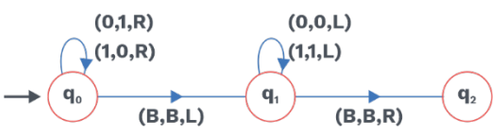
Transducer:
This is a more general automaton,which is able to produce strings of symbols as output and it is called a transducer.

Turing machine as transducer:

SOLVED EXAMPLES

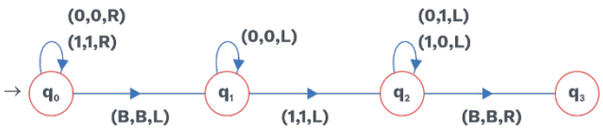
Q1 **Design a TM to find 1's complement of a binary number.**

Sol:



Q2 **Design a TM to find 2's complement of a binary number.**

Sol:

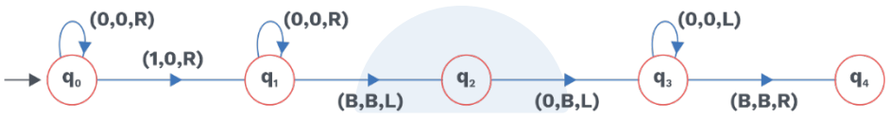


Note: Always, the read-write head of Turing machine will point to starting of the input.

Q3 **Design a turing machine to add two number and the numbers are represented as unary number and use '1' as a separator between two number (for ex. 3 will be represented as '000' in unary).**

Sol: **Ex.:** 3 + 4 can be represented as input tape as BB00010000BB and the output will be BB0000000BB.

So, it is the same as a concatenation of two numbers.



Q4 **Design a Turing machine to compare two number and two numbers are like a (3) b (5) 111 0 11111 . Here, '0' act as separator.**

Sol: Here, we have three cases:

Case 1:

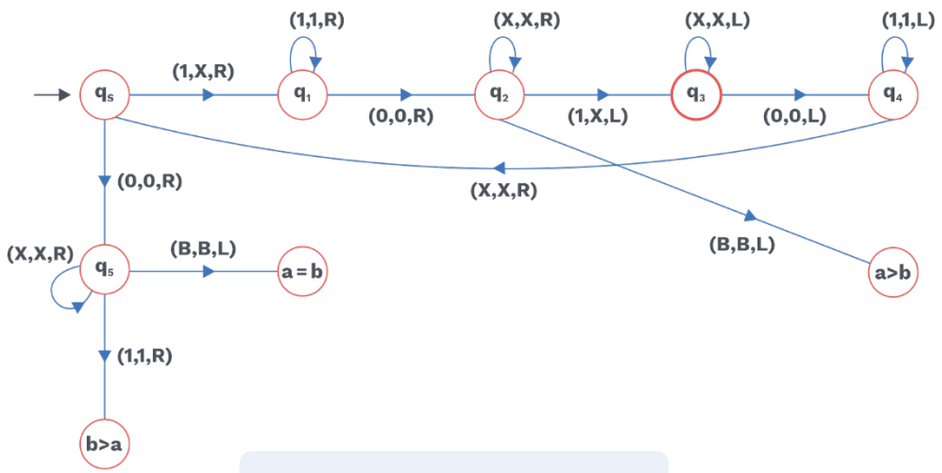
When a = b, it means that it should halt in a state which will indicate both are equal.

Case 2:

When a > b, it means that it should halt in a state which will indicate 'a' is greater than b.

Case 3:

When a < b, it means that it should halt in a state which will indicate 'b' is greater than 'a'.

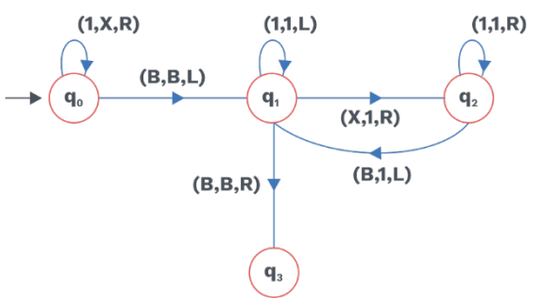


Note:

- Turing machine is capable of doing addition and comparison.
- So, if we could do addition and comparison we could do any other mathematical operation. So, every mathematical function either multiplication, division, logarithm or exponential can be written in terms of comparison and addition.
- So, Turing machine can implement any mathematical function. Therefore, Turing machine is mathematical complete.
- Turing machine can compute mathematical function even using any other number system.

Q5 **Design a turing machine which will do copies of the input for example if the input is B11B then output will be BB1111BB.**

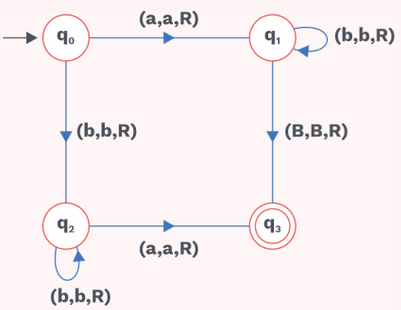
Sol:



Rack Your Brain

1) For $\sum = \{0, 1\}$, design a Turing machine that accepts the language denoted by the regular expression 00^*.

2) What language is accepted by the Turing machine whose transition graph is in the figure below?



Main features of standard turing machine:

Apart from all the different descriptions of a Truing machine, the standard one can be defined as

- As δ defines at most one move always for each configuration, the Turing machine becomes deterministic by default.
- The definition doesn't incorporate any specific input or output file. There is an assumption that- initially, the tape holds some characters. A specific number of these characters can be treated as an input. In a similar manner, a definite series of characters belonging to the tape maybe visualized as the output.

Turing machine as language accepters:

The Turing machine M $(Q, \sum, \Gamma, \delta, q_0, B, F)$ will accept the language called $L(M) = \{w \in \sum^+ : q_0 w \vdash^* x_1 q_f x_2$ for some $q_f \in F, x_1, x_2 \in \Gamma^*\}$

Computable:

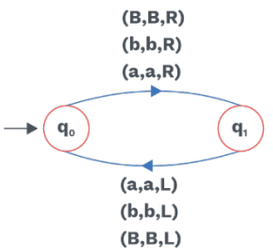
The necessary condition of Turing computable or computable for a function with a given domain is, there should be some Turing machine $M(Q, \sum, \Gamma, \delta, q_0, B, F)$ such that,

$q_0 w \vdash^*_M q_f f(w)$, $q_f \in F$, for all $w \in D$.

Halting problem:

- There are some Turing machines which never halt. It may go into infinite loop.

Example:



- In a Turing machine, the not halting issue i.e whether the TM is ever halted or not, which is commonly known as the halting problem.
- This type of problem does not occur either in Finite Automata and Pushdown Automata because there we have only one move in a forward direction.
- So, when input gets over, either the FA or PDA will definitely halt, but in the case of TM it can go either forward or backwards.

Variations of turing machines:

Equivalence of classes of automata:

If two automatons have acceptance for the same language, then they are called equivalent. For example, suppose there are two automata classes, C_1 and C_2 and two automatons M_1 and M_2. If for every M_1 in C_1 there is an M_2 in C_2 such that $L(M_1) = L(M_2)$ then we can equalize the power of both the classes. And, if the vice versa is also hold then the automata classes, i.e. C_1 and C_2 are equivalent.

Turing machines with a stay-option:

- The standard Turing machine definition says that the movement of the read-write head must be to the left or to the right.
- Another most convenient option provided, is to do nothing or to stay at the current place even after updating the symbol.
- Hence, the Turing machine with a do-nothing or stay option can be defined by modifying δ

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, S\}$$

S represents do-nothing or no movement of the R/W head.

- Anyway, this option will not add extra power to the automaton.

Turing machines with semi-infinite tape:

- The tape is bounded only in one direction.
- Let's visualize a tape with a left boundary.

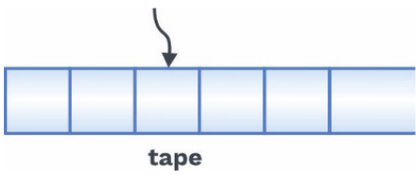


Fig. 5.2 Diagram of Turing machine with Semi-Infinite Tape

The off-line turing machine:

- Here, input is given in a separate file, and we are not allowed to modify the input.
- In case, we have to do some modification on input, then we have to 1st copy entire input into the tape, and then we have to do modifications to the tape.
- Even after computation is over, the input is going to remain intact on the input file.

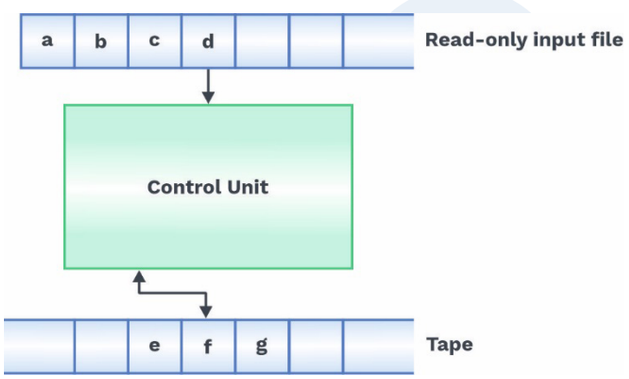


Fig. 5.3 Diagram of The off-Line Turing Machine

- This modified TM is also equivalent in power to standard TM.

Multitape turing machines:

- More than one tape is there in the case of a multitape Turing machine, and the most important part is every tape has its dedicated R/W head.
- We define an n-tape machine by $M=(Q,\Sigma,\Gamma,\delta,q_0,B,F)$ where Q, Σ, Γ, q_0, F are the same as standard TM, but where

$$\delta : Q \times \Gamma^n \rightarrow Q \times \Gamma^n \times \{L,R\}^n$$

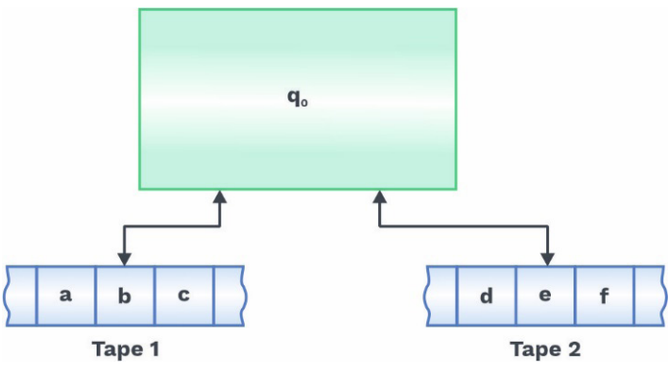


Fig. 5.4 Diagram of Multitape Turing Machines

- Multitape turing machine is also equivalent to standard turing machine.

Multidimensional turing machines:

- In such Turning architecture, infinite extension of the tape is allowed in any direction.

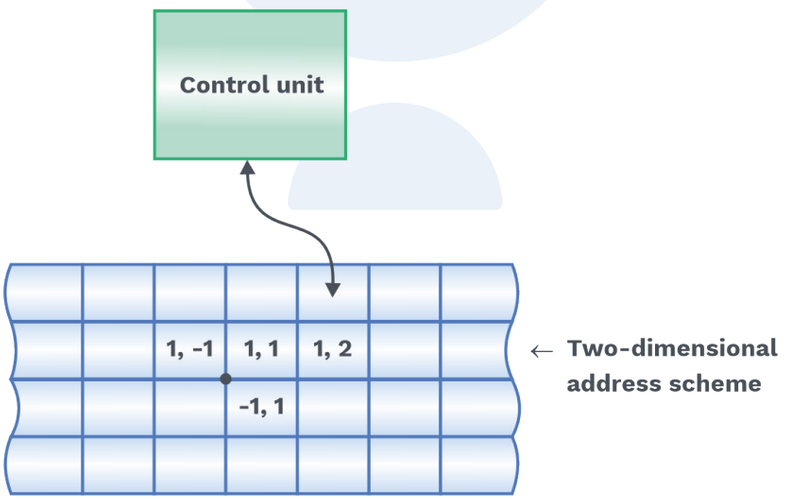


Fig. 5.5 A Diagram Of A Two-Dimensional Turing Machine Is Shown Above.

- The transition function δ of a 2D Turing machine can be defined as

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, U, D\}$$

where, U represents the up movement of the read-write head and D represents the down movement.

- Multidimensional Turing machine is equivalent to the standard Turing machine.

Jumping turing machine:

- In this jumping TM, instead of moving just one step ahead either to the left or to the right, it can move many steps in one go.
- Here, the δ is given below

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\} \times \{n\}$$

$$\uparrow$$

Here, 'n' number of steps.

- This jumping TM is also equivalent in power to standard TM.

Non erasing turing machine:

- In non-erasing TM, on the input symbol, we are not allowed to write Blank(B) but we can write any other symbol.
- This non-erasing TM is also equivalent in power to standard TM.

Always writing TM:

- In always writing Turing machine, it is always needed to modify the input with different symbols whenever the Turing Machine sees any input. Turing Machine cannot leave it as it is. Example: if TM get 'a' as input, then it will write something which is not 'a'.
- This Turing machine is also equivalent to standard TM.

Multihead TM:

- This modified TM have more than 1 read-write head.
- Multiple heads simultaneously look after the scanned symbol and make the needful, i.e. move or write onto the tape independently.
- A Single head Turing machine can simulate a multi head Turing machine.

Automata with a queue:

- If there are finite Automata, and if we are adding a queue, then it is equivalent to standard TM.

Turing machine with only 3 states:

- Any Turing machine can be minimized to a Turing machine which has only 3 states.
- If a Turing machine has only three states, then it is also equivalent to a standard Turing machine.

Multitape TM with stay option and atmost 2 states:

- Any Turing machine can be as multitape TM with stay option and almost 2 states.
- This modified TM also have the same power as standard TM.

Non deterministic turing machine:

- A non deterministic Turing machine is an automaton is defined by

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

Where
Q = finite set of internal states
Σ = finite set of input alphabets
Γ = Finite set of symbols or tape alphabets
δ = the transition function

$$\delta : Q \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{L,R\}}$$

B = blank space (special symbol) from tape symbol.
q_0 = initial state $\in$ Q
F = Set of final states $\subseteq$ Q

- In a non-deterministic machine, the range of δ is a set of all possible paths, so the machine can take any path out of all possible paths.
- The deterministic Turing machines classes are equivalent to the non deterministic Turing machines classes.
- If at least one configuration of a non deterministic Turing machine accepts a string w belongs to L then that language L is said to be accepted by the same Turing Machine.
- "A non deterministic Turing machine M is said to decide a language L if, for all w $\in$ Σ*, There is a path that leads either to the acceptance or rejection."

Note: A NPDA with one extra independent stack. The δ is given below:-
δ: $Q \times (U\{\in\}) \times \Gamma \times \Gamma \;\; 2^{Q \times \Gamma^* \times \Gamma^*}$
This is also equivalent to standard TM.

Multistack machines:

- The tapes of a multi tape Turing machine in such a way, that it starts behaving like a stack.
- A one-stack machine is a DPDA, while a machine with two stacks is a Turing Machine.

Grey Matter Alert!

Simulating a Turing Machine by a real computer

- It is possible, in principle, to simulate a TM by a real computer if we accept that there is a potentially infinite supply of a removable storage device such as a disk, to simulate the non-blank will be better portion of the TM tape.
- Since the physical resources to make disks are not infinite, this argument is questionable.
- However, since the limits on how much storage exists in the universe are unknown and undoubtedly vast, the assumption of an infinite resource, as in the TM tape, is realistic in practice and generally accepted.

Simulating a computer by a turing machine:

- A Turing Machine can simulate the storage and control of a real computer if it uses one tape to store all the locations and their contents registers, main memory, disk and other storage devices.
- Thus, we can surely say that something that cannot be done by a TM also cannot be done by a real computer.

Rack Your Brain

Consider the nondeterministic Turing machine
$M = (\{q_o, q_1, q_2, q_f\}, \{O, 1\}, \{O, 1, B\}, \delta, q_o, B, \{q_f\})$
Informally but clearly describe the language L(M) if 8 consists of the following sets of rules:

$\delta(q_0, O) = \{(q_0, 1, R), (q_1, 1, R)\}$
$\delta(q_1, 1) = \{(q_2, 0, L)\}$
$\delta(q_2, 1) = \{(q_0, 1, R)\}$
$\delta(q_1, B) = \{(q_f, B, R)\}$

Universal turing machine:

- In a universal Turing machine M_u, a standard Turing machine(M) will be given as input along with an input string w. We can simulate the computation of M on w.
- To give a standard Turing machine as an input, first, a standard representation of a Turing machine is required.
- Assume $Q = \{q_1, q_2, \ldots\ldots, q_n\}$
- where, q_1 = initial state, q_2 = single final state, and

$$\Gamma = \{a_1, a_2, \ldots\ldots, a_m\},$$

Where, a_1 represents the blank symbol.

- Suppose, there is an encoding where q_1 can be represented as 1, q_2 can be represented as 11 and so on.
- Similarly, the tape input symbols can be encoded as $a_1 = 1$, $a_2 = 11$, and so on.
- To separate the 1's or the strings of 1's can be distinguished using 0 as a symbol.
- Any Turing machine can be defined by the transition function δ along with the initial, final state and the blank symbol defined.
- "The transition function is encoded according to this scheme, with the arguments and result in some prescribed sequence". For example, $\delta(q_1,a_2) = (q_2,a_3,L)$ might appear as

...10110110111010.....

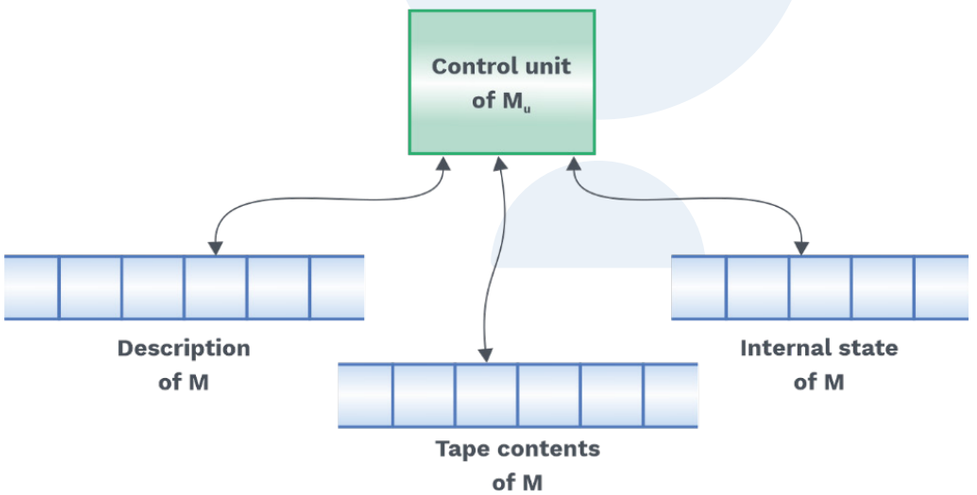


Fig. 5.6

Note:

- It follows from this that any turing machine has a finite encoding as a string on $\{0,1\}^+$ and that, given any encoding of M, we can decode it uniquely.
- But any combination of strings will not represent any Turing machine (e.g. the string 00011).

Working procedure of universal turing machine:

- The encoded definition of M will be kept in the tape 1, given an input M and w.
- The tape symbols of M will be placed in the tape 2 and tape 3 will contain the states of M.
- The universal Turing machine identifies the configuration of M by looking at the tape 2 and tape 3.
- The tape 1 will help for the transition by seeing tape 2 and tape 3.
- Finally, the changes will be reflected on the tape 2 and tape 3, the results will be reflected in tape 2.
- As it can be implemented using any programming language, we can implement the same using any standard Turing machine also.

Previous Years' Question

A single tape Turing Machine M has two states q_0 and q_1, of which q_0 is the starting state. The tape alphabet of M is {0, 1, B} and its input alphabet is {0, 1}.
The symbol B is the blank symbol used to indicate end of an input string. The transition function of M is described in the following table. *(GATE-2003)*

	0	1	B
q0	q1, 1, R	q1, 1, R	Halt
q1	q1, 1, R	q0, 1, R	q_0, B, L

The table is interpreted as illustrated below.
The entry (q1, 1, R) in row q0 and column 1 signifies that if M is in state q0 and reads 1 on the current page square, then it writes 1 on the same tape square, moves its tape head one position to the right and transitions to state q1.
Which of the following statements is true about M?
1) M does not halt on any string in $(0 + 1)^+$
2) M does not halt on any string in $(00 + 1)^*$
3) M halts on all strings ending in a 0.
4) M halts on all strings ending in a 1.
Sol: Option 1)

Note:

- FA + tape = FA + 2stack = FA + queue = TM
- TM = FA+ 2 stack
  = FA +3 stack
  = FA + 4 stack
  .
  .
  .
  = FA+ n stack (n ≥ 2)

Recursively enumerable:

- To ensure a language L is recursively enumerable, we have to find a Turing machine for it.
- Recursive Enumerable can be defined as a Turing machine M, where, for every w ∈ L,

$$q_0 w \vdash^*_M x_1 q_f x_2,$$

with q_f a final state.

- If string w is not in the language, then the definition doesn't say whether it will go in a non-final state and halt or it may go into an infinite loop.

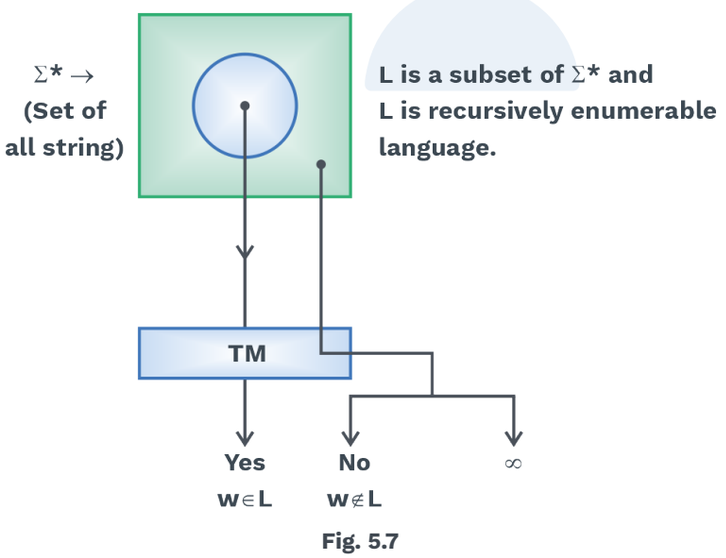


Fig. 5.7

Recursive language:

1) A formal language for which a standard Turing machine exists, i.e., any strings from that language will be accepted by the Turing machine and the machine halts as well is called recursive language.

2) "The Turing machine that always halt is called Halting TM/decider/Total Turing machine."

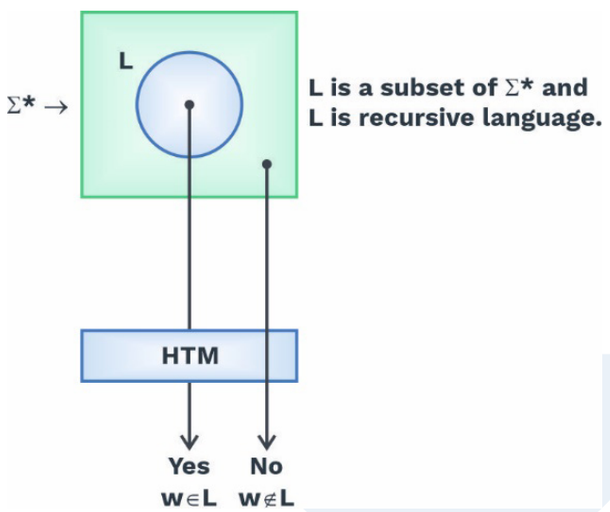


Fig. 5.8

Conclusion

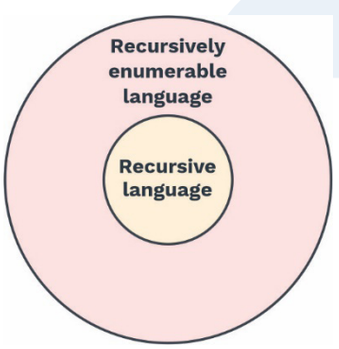


Fig. 5.9

"Recursive Language is a subset of recursively enumerable language".

Chomsky hierarchy:

1) All formal languages are divided into 4 classes by Chomsky and those class hierarchy known as “Chomsky Hierarchy”.

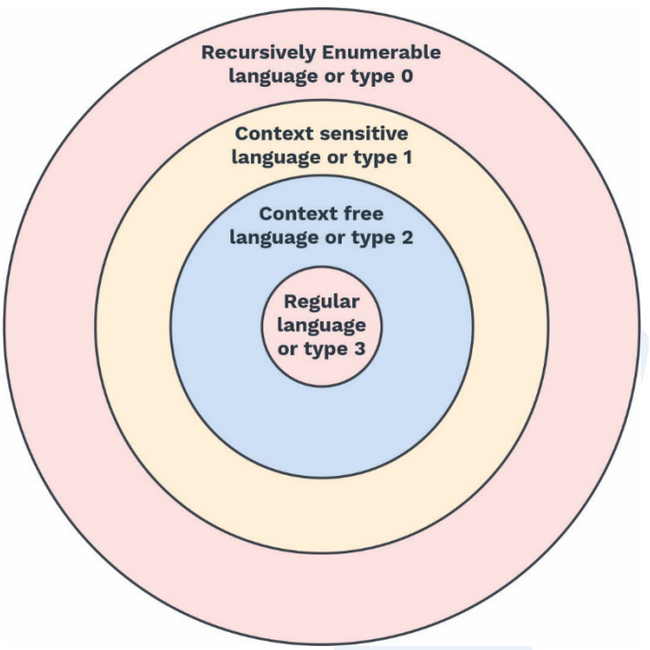


Fig. 5.10

2) Actually Type 0 represents recursively enumerable language, type 1 represents context-sensitive language, type 2 represents context free language and type 3 represents regular language.

Note:

Recursive language is not type (0) language. Only recursively enumerable language is type (0) language.

Extended Version of Chomsky Hierarchy:

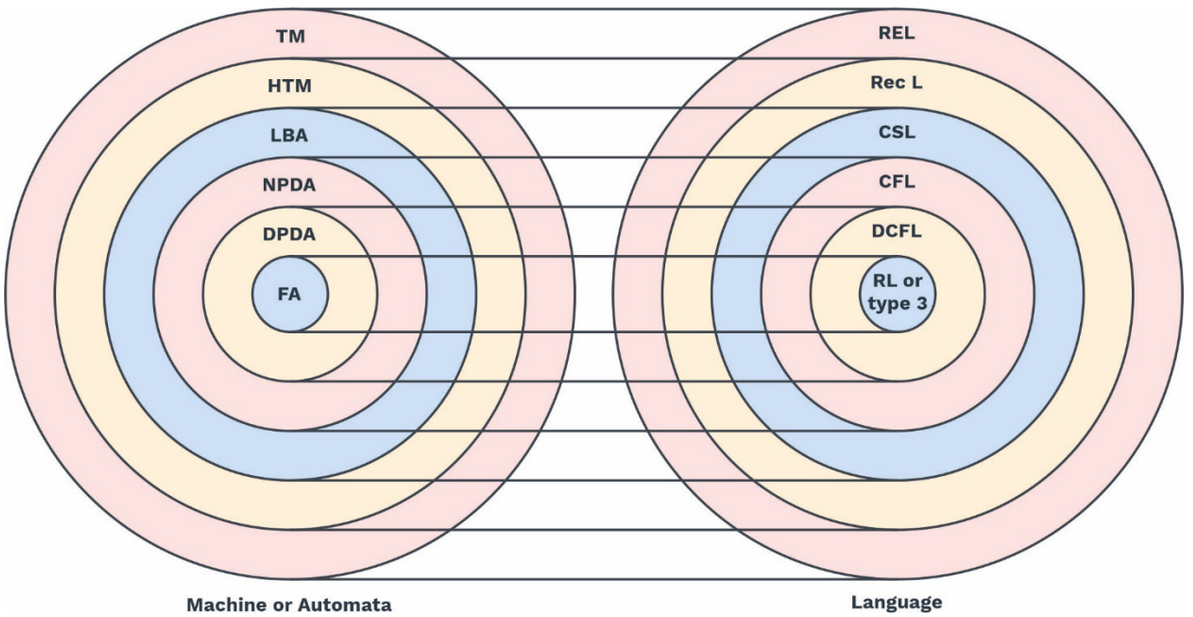


Fig. 5.11

Language and their corresponding grammar:

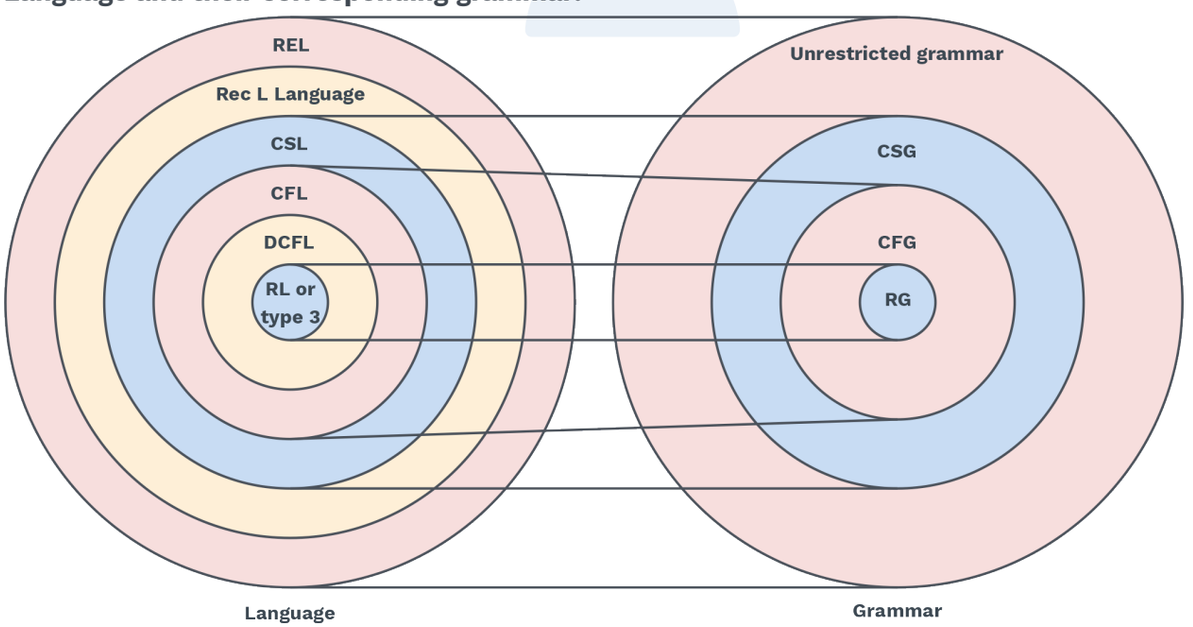


Fig. 5.12

Theorem 1:
"If a language L and its complement $\bar{L}$ are both recursively enumerable then both languages are recursive."

Theorem 2:
"If L is recursive, then $\bar{L}$ is also recursive and consequently both are recursively enumerable."

Unrestricted grammar/type 0 grammar:
When all the productions in a grammar are of the form $X \rightarrow Y$
where $X \in (V + T)^+$ and $Y \in (V + T)^*$
[Here, V indicate set of variables and 'T' indicate set of terminals]

- Unrestricted grammar can be defined '$\in$' but TM is not configured in order to accept a '$\in$' string. So, we assume that we don't consider language deriving '$\in$'.

Ex.: $S \rightarrow S_1B$

$S_1 \rightarrow aS_1b$

$bB \rightarrow bbB$

$aS_1B \rightarrow aa$

$B \rightarrow \in$

Closure properties of Recursive and Recursive Enumerable Language

Property	Recursive	RE language
Union	Yes	Yes
Intersection	Yes	Yes
Set Difference	Yes	No
Complementation	Yes	No
Intersection with regular language	Yes	Yes
Union with regular language	Yes	Yes
Concatenation	Yes	Yes
Kleen Closure	Yes	Yes
Kleen Plus	Yes	Yes
Reversal	Yes	Yes
Homomorphism	No	Yes
$\in$-Free Homomorphism	Yes	Yes
Inverse Homomorphism	Yes	Yes
Substitution	No	Yes

Table 5.1 Cosure Properties

Theorem 1:

"For all recursive language 'L' there is an enumeration technique for it."

Proof:

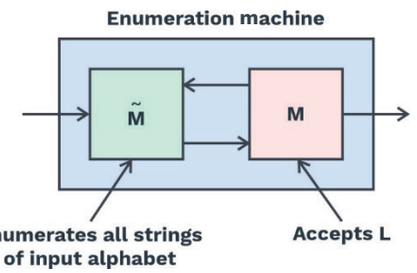


Fig. 5.13

If the alphabet is {a, b} then $\tilde{M}$ enumerate strings as follows:

'a'
'b'
'aa'
'ab'
'ba'
'bb'
'aaa'
'aab'
...

Enumeration Procedure
Repeat:
$\tilde{M}$ generates a string w
M checks if w ∈ L
Yes: Print w to output
No: Ignore w
End of proof.

Example: L = {b, ab, bb, aaa, ...}

$\tilde{M}$	L(M)	Enumeration Output
a		
b	b	b
aa		
ab	ab	ab
ba		
bb	bb	bb
aaa	aaa	aaa
aab		
...		

Note:

Theorem 2:

"A language is recursively enumerable if and only if there is an enumeration procedure for it."

Previous Years' Question

Define languages L_0 and L_1 as follows: **(GATE-2003)**

L_0 = {<M, w, 0> | M halts on w}

L_1 = {<M, w, 1> | M does not halts on w}

Here <M, w, i> is a triplet, whose first component M is an encoding of a Turing Machine, second component w is a string, and third component i is a bit.

Let L = L_0 U L_1. Which of the following is true?

1) L is recursively enumerable, but L′ is not.

2) L′ is recursively enumerable, but L is not.

3) Both L and L′ are recursive.

4) Neither L and L′ is recursively enumerable.

Sol: Option 4)

Previous Years' Question

Consider the following languages. **(GATE-2016 (set 2))**

- L_1 = {<M> | M takes at least 2016 steps on some input},
- L_2 = {<M> | M takes at least 2016 steps on all inputs} and
- L_3 = {<M> | M accepts $\in$},

where for each Turing machine M, <M> denotes a specific encoding of M.

Which one of the following is TRUE?

1) L_1 is recursive and L_2, L_3 are not recursive

2) L_2 is recursive and L_1, L_3 are not recursive

3) L_1, L_2 are recursive and L_3 is not recursive

4) L_1, L_2, L_3 are recursive

Sol: Option 3)

Linear bound automata:

- We can limit the power of a Turing machine by restricting the way in which we are going to use tape.
- If we limit the Non-deterministic TM in such a way that it can use the tape like a stack then it converges to a PDA.

- If we limit the Turing machine to use only finite amount of cell and only unidirectional tape movement is allowed then it can converges into finite automata.
- If we limit tape of Turing machine to use only that part of the tape where the input is present and not beyond it, then it converge to LBA (Linear Bounded Automata).
- A linear bounded automata is designed with an unbounded tape. It extents up to which the tape needs to be made use of purposely, with an instant of the input. Practically, the unbounded tape is restricted by the input string.
- To enforce this, we can envision the input as bracketed by the two special symbols, the left-end marker and the right-end marker.
- LBA is less powerful than TM and it is more powerful than PDA. Ex.: $a^nb^nc^n \mid n \geq 1$ is accepted by LBA but not by PDA.
- Here, we don't know whether the deterministic LBA are equivalent in power to non-deterministic LBA. It is undecidable.
- Complement of the CFL is CSL.
- Language accepted by LBA is called context sensitive language.
- LBA is halting, Turing machine as context sensitive language is also a recursive language.

Countability

- **Countable set definition:** "A set 's' is said to be countable, if the elements of the set can be put in one to one correspondence with the set of natural numbers."
- By this we mean that the elements of the set can be written in some order, say, x_1, x_2, x_3, ..., so that every element of the set has some finite index.

for ex.: E = set of even number = {0, 2, 4, 6, 8...}

For any element in E of form 2i (i starts with 0) we give a corresponding element (i + 1) in N (Natural Number). So, for every element in even number set we can associate a natural number as a correspondence. Therefore, we can say that set of even number is countable.

eg.: O = set of odd number

= {1, 3, 5, 7, 9, 11, ...}

Here, we can represent set of odd number in terms of (2i + 1) ($i \geq 0$). We can map every odd number of the form (2i + 1) with (i + 1) (Natural number).

$$O = \{1, 3, 5, 7, 9, 11, \ldots\}$$
$$\downarrow \downarrow \downarrow \downarrow \downarrow \downarrow$$
$$N = \{1, 2, 3, 4, 5, 6, \ldots\}$$

Hence, the set of odd number is countable.

Uncountable set definition: "A set is uncountable if it is infinite and not countable."

Rack Your Brain

Given R = set of real number. Is R countable set?

Alternative definition of countability:

- A set is said to be countable if there exists an enumeration method using which all the elements of the set can be generated and for any particular element, it takes only a finite number of steps to generate it.
- The finite number of steps taken to generate an element can be used as its index and hence a mapping into a natural number set.

for ex.:

i) Set of even number

Enumeration procedure: for (q = 0 to ∞)

printf (2 × q);

Output :	0	2	4	6	8	...
Steps :	1	2	3	4	5	...

So, the index for (0, 2, 4, 6, 8, ...) is (1, 2, 3, 4, 5, ...).

Hence, set of even number is countable.

Similarly;

ii) Set of odd number

for (q = 0 to ∞)

printf (2 × q + 1)

Output :	1	3	5	7	9	11	...
Steps :	1	2	3	4	5	6	

SOLVED EXAMPLES

Q1 **Let S = set of all quotients in the form of p/q, where p and q are positive integers and q ≠ 0. Check whether S is countable or not?**

Sol: The set S is countable since we are having an enumeration procedure for set S.
Here, we don't go either in increasing order of numerator or denominator, we just add numerator and denominator and we just go in the increasing order of sum.
Here, the smallest number is 1/1.
So, the 1st sum is 2.
Within the sum we can enumerate in increasing order.
So, the enumeration order is $\frac{1}{1}, \frac{1}{2}, \frac{2}{1}, \frac{1}{3}, \frac{2}{2}, \frac{3}{1}, \ldots$
Hence, it is countable.

Lexicographic ordering:

- Lexicographic ordering is nothing but the alphabetical order, i.e., the sequence of letters present in the dictionary.
- Suppose we have a set Σ = {a, b} and Σ* in lexicographical order will be {∈, a, b, aa, ab, ba, bb, aaa, aab, ...}

Rack Your Brain

Given a set S = set of all string over {a, b} is
1) Countable
2) Uncountable
3) Finite
4) Neither countable nor uncountable

Note:

- Set of all string possible over any alphabet is countable.
- So, Σ* = set of all string over Σ is countable.
- Every subset of countable set is either finite or countable.
- Since every language is a subset of Σ*. Therefore, every language is countable.
- It means given any language it is definitely either countable language or finite language.

Q2 **Check whether T = set of all Turing machine is countable or not?**

Sol: Since every Turing machine can be encoded as a string of 0's and 1's.
Let $\Sigma = \{0, 1\}$ then Σ^* = set of all strings over {0, 1}.
Since, Σ^* is countable and T is subset of Σ^*.
Hence, T = set of all Turing machines.

Implication of the fact that the set of all turing machines are countable:

- Since we know that the set of Turing machines are countable, then the set of recursively enumerable language are also countable.
- Since the set of recursive language is subset of the set of recursively enumerable language. Hence, a set of recursive languages is countable.
- Similarly, a set of context free language, a set of context sensitive language, and the set of regular language are also countable.
- We can also modify a turing machine in such a way that it can act as PDA. Therefore, if set of TM is countable it implies that the set of PDA is also countable.
- Since, all machines are subset of set of Turing machine. So, the set of LBA, and set of FA are also countable.

Theorem: Set of all languages possible are uncountable.

Implication from theorem:

- Set of TM is countable, and the set of all languages possible is uncountable.
- Therefore, there are some languages for which there are no machines yet. It means there are some languages which are not even recursively enumerable.

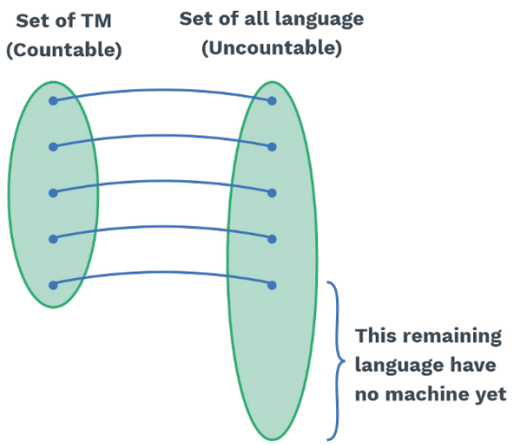


Fig. 5.14

Theorem:
"set of all languages are uncountable".

Proof:
Suppose there is a set $\Sigma = \{a, b\}$.
We need to prove that, 2^{Σ^*} is uncountable.
Let us assume, 2^{Σ^*} is countable.
As, $\Sigma = \{a, b\}$
$\Sigma^* = \{\in, a, b, aa, ab, ba, bb, aaa, aab,....\}$

Let us represent each string using 0 or 1, if the string is present in the language $\in 2^{\Sigma^*}$ then mark that string using '1' otherwise '0'.

Σ^*	$\in$	a	b	aa	ab	ba	bb	aaa	aab	.....
Language 1	(0)	1	1	0	1	0	1	0	0	.....
Language 2	1	(0)	1	1	1	1	1	0	0	.....
Language 3	0	1	(1)	0	0	1	1	0	0	.....
Language 4	1	1	1	(1)	1	0	1	0	1	.....
Language 5	1	1	1	1	(1)	1	1	1	1	.....
.	.	.	.	.	.	.	.	.	.	.....
.	.	.	.	.	.	.	.	.	.	.....
.	.	.	.	.	.	.	.	.	.	.....

Now, if we consider the diagonal, then we will find a string S = 0 0 1 1 1 ..... Complement of the diagonal 00111... will be S′ = 1 1 0 0 0..... which is not in the language.

This particular language $\overline{S}$ is not included in that list as it is a different form all the languages by at least one string.
So, our assumption was incorrect.
Hence, 2^{Σ^*} is uncountable. (Proved)

Note:

This kind of argument, because it involves a manipulation of the diagonal elements of a table is called diagonalization.

Note:

Theorem:

"There exists a recursively enumerable language whose complement is not recursively enumerable."

Theorem:

"There exists a recursively enumerable language that is not recursive i.e., the family of recursive languages is a proper subset of the family of recursively enumerable languages."

- Diagonalization method is used to prove that a language is not recursively enumerable because if for a language, we proved that it is not countable then it means there is no enumeration procedure for that language.

  And theorem says that a language is Recursive Enumerable Language if and only if there is enumeration procedure for it. Hence, that language is not recursively enumerable.

Theorem:

"If S_1 and S_2 are countable sets, then $S_1 \cup S_2$ and $S_1 \times S_2$ are countable and union and cross-product can be extended to any number of finite sets."

Theorem:

"The set of all languages that are not recursively enumerable is uncountable."

Proof:

The set of all languages is 2^{Σ^*} (set of all subsets of Σ^*), and out of that some languages are already recursive enumerable, because if there is TM for a language then that language is recursively enumerable and we already know that the number of TM is countable. Therefore, the number of languages in the total language which is recursively enumerable are countable and that will be equal to a number of Turing machines.

And in the remaining set, we don't know anything, so we are just assuming that it is countable. So, the resulting set is the union of a countable set; therefore overall, that should be countable. And if that is true, then our initial proof that 2^{Σ^*} is uncountable is a failure as we know that set of all languages 2^{Σ^*} is uncountable. Therefore, our assumption that, the remaining language i.e., the set of all languages that are not- recursively enumerable is countable, is false. So, set of all languages which are not recursively enumerable, must be uncountable.

Previous Years' Question

Consider the following sets:
S1: Set of all recursively enumerable languages over the alphabet {O, 1}
S2: Set of all syntactically valid C programs
S3: Set of all language over the alphabet {O. 1}
S4: Set of all non-regular languages over the alphabet {O, 1)}
Which of the above sets are uncountable? **(GATE-2019)**
1) S1 and S2
2) S3 and S4
3) S2 and S3
4) S1 and S4

Previous Years' Question

Let N be the set of natural numbers. Consider the following sets,
P: Set of Rational numbers (positive and negative) **(GATE-2018)**
Q: Set of functions from {O, 1} to N
R: Set of functions from N to {O, 1}
S: Set of finite subset of N
Which of the above set are countable?
1) Q and S only
2) P and S only
3) P and R only
4) P, Q and S only

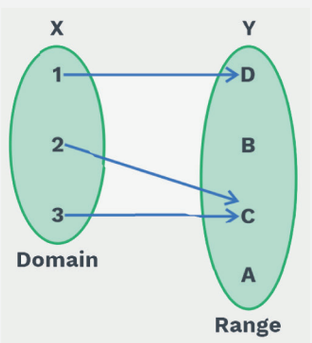
Grey Matter Alert!

Function

- A function is a rule that assigns to elements of one set a unique element of another set.

  OR

  In mathematics, a function is a relation between sets, that associates every element of a first set with exactly one element of the second set.
  Ex.: X = {1, 2, 3} & Y = {A, B, C, D}
  t = {(1, D), (2, C), (3, C)}



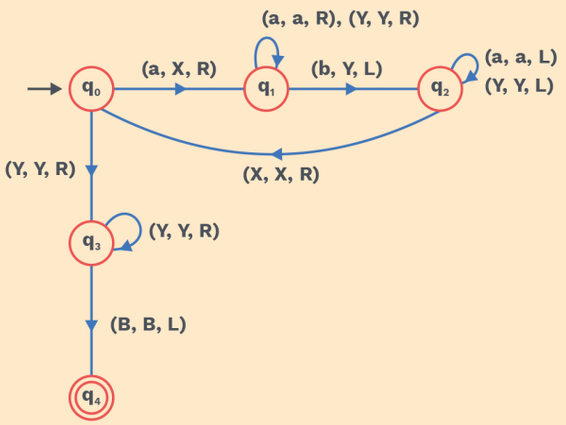
- If f denotes a function, then the first set is called the domain of f and the second set is its range. We write

  $$f : S_1 \rightarrow S_2$$

  to indicate that the domain of f is a subset of S_1 and that the range of f is a subset of S_2.
- If the domain of f is all of S1, we say that f is a total function on S_1, otherwise f is said to be a partial function.

SOLVED EXAMPLES

Q1 **The transition diagram for Turing machine is given below**



Which one of the following strings is accepted by the above TM?

1) ab **2) aab**

3) abb **4) None of the above**

Sol: **1) Explanation:** It is accepting L = $\{a^n b^n \mid n \geq 1\}$. So, (1) is the correct option.

Q2 **Let L = $\{X^{n+m} Y^{n+m} X^m \mid n, m \geq 0\}$. The above language L is**

1) CSL but not CFL **2) Rec L but not CSL**

3) REL but not Rec **4) CFL but not regular**

Sol: **1) Explanation:** It is a CSL because we can give a linear bound automata. We cannot compare number of X's and number of Y's using one stack here.

Q3 **Consider the given statements:**
S1: Set of all irrational number are countable.
S2:If L be a finite language then L^* is recursively enumerable language.
Which of the following statements are correct?

1) S1 **2) S2**
3) S1 & S2 **4) None of the above**

Sol: **2) Explanation:** Statement S1 is incorrect because set of all irrational number are uncountable.

Statement S2: Since L is finite language. So, it is regular. All regular language are recursively enumerable language and recursively enumerable language are closed under Kleene closure. Hence, L^* is recursively enumerable language.

Q4 **Consider the given statements:**
S1: Let L_1 be recursive and L_2 be recursively enumerable, then $L_2 - L_1$ is necessarily recursively enumerable.
S2:$L = \{www^k \mid w \in (a, b)^+\}$ is CSL.
Which of the above statement(s) is/are correct(s)?

1) S1 **2) S2**
3) S1 & S2 **4) None of the above**

Sol: **3) Explanation:**

S1 : $L_2 - L_1 = L_2 \cap \overline{L}_1$

$= REL \cap \overline{Rec\ L}$

$= REL \cap Rec\ L$

$= RE\ L.$

So, statement S1 is correct.

S2 : $L = \{www^k \mid w \in (a, b)^+\}$ is CSL because for this we can give a LBA.

It needs more than one stack to accept this language L, so it is CSL.

Q5 **Consider the given statements:**
S1: Let L_1 be recursive language then its complement $\overline{L}_1$ is also recursive language.
S2: Let L_2 be recursively enumerable language then its complement $\overline{L}_2$ is also recursively enumerable language.
Which of the above statement(s) is/are correct(s)?

1) S1 **2) S2**
3) S1 & S2 **4) None of the above**

Sol: **1) Explanation:** Recursive language is closed under complementation. Hence, $\overline{L}_1$ is also recursive language. Recursively enumerable language is not closed under complementation. Hence, $\overline{L}_2$ may or may not recursively enumerable language.

So, we can't say $\overline{L}_2$ is recursively enumerable language.

Q6 **Consider the following language,**
L_1 = Recursive enumerable language,
L_2 = Recursive language
L_3 = Context free language
Now,
Let $L = (\overline{L}_2 \cap L_1) - L_3$
L is

1) L is CFL **2) L is recursive**
3) L is REL **4) L is not REL**

Sol: **3) Explanation:**

$$L = (\overline{L}_2 \cap L_1) - L_3$$

$$= (\overline{\text{Rec } L} \cap \text{REL}) - \text{CFL}$$

$$= (\text{Rec } L \cap \text{REL}) - \text{CFL}$$

$$= \text{REL} \cap (\overline{\text{CFL}})$$

$$= \text{REL} \cap \text{CSL} = \text{REL}.$$

[Here, Rec L means recursive language,
REL means recursively enumerable language,
CFL means context free language,
CSL means context sensitive language]

Q7 **What is the highest type number that can be assigned to the following grammar?**

$S \rightarrow S_1B$

$S_1 \rightarrow aS_1b$

$bB \rightarrow bbbB$

$aS_1b \rightarrow aa$

$B \rightarrow \in$

1) Type 0 **2) Type 1**

3) Type 2 **4) Type 3**

Sol: **1) Explanation:** The given grammar is in the form of $u \rightarrow v$ where u belongs to $(V+T)^+$ v belongs to $(V+T)^*$ here, V is set of non-terminals and T is set of terminals Hence, it is unrestricted grammar or type-0 grammar.

Q8 **If L_1 and L_2 are two recursively enumerable languages, then they are not closed under**

1) Complementation **2) Union**

3) Intersection **4) Concatenation**

Sol: **1) Explanation:** Recursively enumerable language is not closed under "complementation" and it is closed under "union", "intersection", "concatenation", etc.

Q9 **A Turing machine has _____ number of states.**

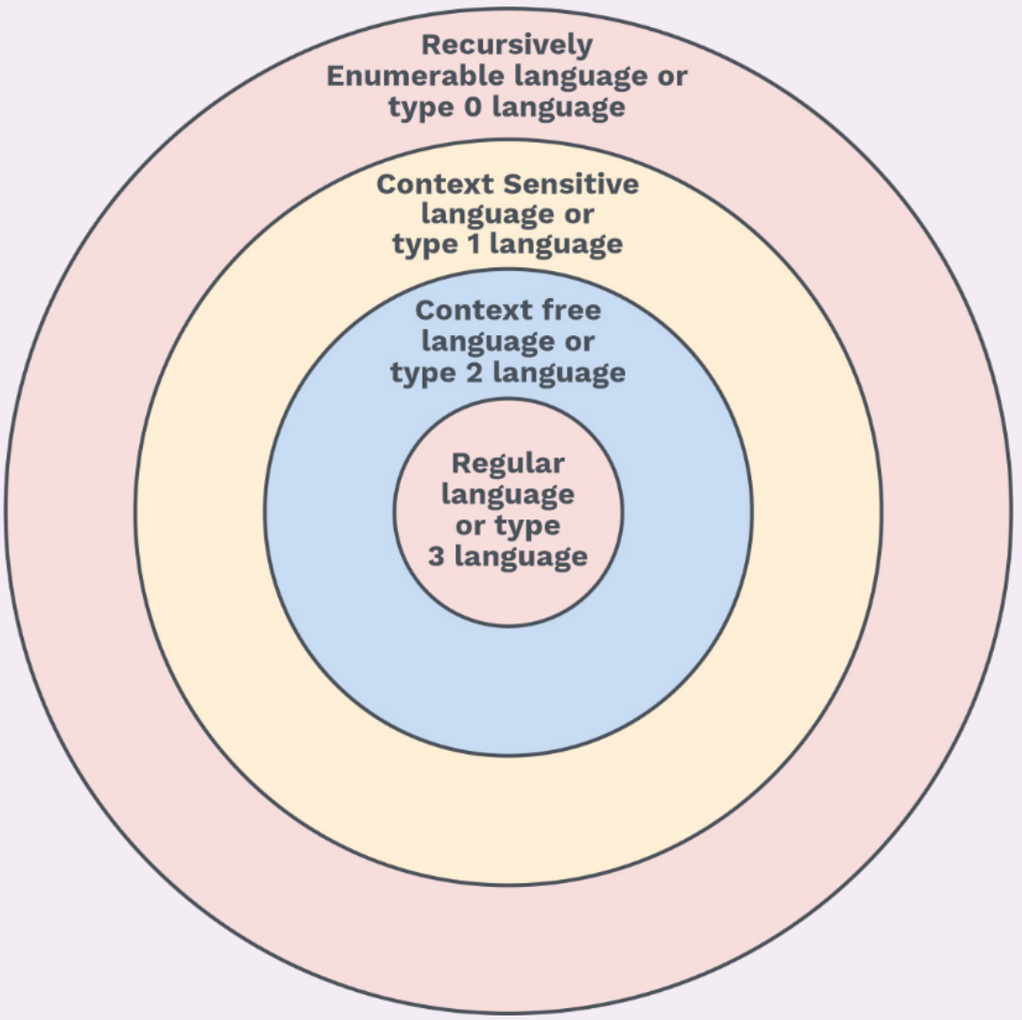
1) Finite **2) Infinite**

3) May be finite **4) None of the mentioned**

Sol: **1) Explanation:** In the definition of Turing machine, 'Q' represents the "Set of finite number of states".

Chapter Summary

- A Turing machine M is defined by M = (Q, Σ, Γ, δ, q_0, B, F) where Q is the set of internal states, Σ is the input alphabets, Γ is a finite set of symbols called the tape alphabet, δ is the transition function, defined as
  δ: {Q × Γ → Q × Γ × {L, R}} where L = left and R = right, B ∈ Γ is a special symbol called the blank, q_0 ∈ Q is the initial state, F ⊆ Q is the set of final states.
- Turing machine can't accept epsilon.
- Turing machine can act both as a transducer as well as acceptor.
- Turing machine is mathematically complete. It can do any mathematical function.
- **Halting Problem:** Some times Turing machine goes into an infinite loop. This problem of not halting of Turing machine is called the halting problem in the Turing machine.
- **There are different variations of Turing machines as shown below:**
  i) Turing machine with a stay option.
  ii) Turing machine with semi-infinite tape.
  iii) The off-line Turing machines.
  iv) Multitape Turing machines.
  v) Multidimensional Turing machines
  vi) Jumping Turing machine
  vii) Non Erasing Turing machine
  viii) Always writing Turing Machine
  ix) Multi head Turing Machine
  x) Turing machine with only 3 states.
  xi) Multitape Turing Machine with stay option and at most 2 states.
  xii) Non-deterministic Turing machine
  All these variations of the Turing machines are equivalent in power to the standard Turing machines.
- We can represent any Turing machine in terms of 0's and 1's but any combination of 0's and 1's is not a Turing machine.
- Finite Automata + n stack (n ≥ 2) = TM
- Finite automata + queue = TM.
- **Recursively Enumerable Language:** For a language L, if there is a Turing machine that can accept L, then the language L is known as recursively enumerable.
- **Halting TM:** Turing machine that always halts is called Halting TM/decider/Total Turing machine.
- **Chomsky Hierarchy:** All formal languages are divided into 4 classes by Chomsky and those class hierarchy known as "Chomsky Hierarchy".



- **Unrestricted Grammar/Type 0 Grammar:** A grammar is called unrestricted if all the productions are of the form $X \rightarrow Y$
- where X is in $(V + T)^+$ and Y is in $(V + T)^*$
  [Here, V indicates set of variables and T indicates set of terminals]
- **Closure Property of Recursive and Recursively Enumerable Language**

Property	Recursive	RE Language
Union	Yes	Yes
Intersection	Yes	Yes
Set Difference	Yes	No
Complementation	Yes	No
Intersection with regular language	Yes	Yes
Union with regular language	Yes	Yes
Concatenation	Yes	Yes
Kleen Closure	Yes	Yes
Kleen Plus	Yes	Yes
Reversal	Yes	Yes
Homomorphism	No	Yes
$\in$-Free Homomorphism	Yes	Yes
Inverse Homomorphism	Yes	Yes
Substitution	No	Yes

- **Linear Bound Automata:** If we limit the tape of the Turing machine to use only where the input is present and not beyond it, then it converges to LBA.
- In LBA, whether deterministic LBA is equivalent to non-deterministic LBA is not known.
- Language accepted by LBA is called context-sensitive language.
- **Countable Set:** If there persists a one-to-one mapping from the elements of a set to the set of positive integers, such a set is classified as countable.
- **Uncountable Set:** A set S is uncountable if it is infinite and not countable.
- Set of all strings possible over any alphabet is countable.

- Set of all languages possible is uncountable.
- There exists a recursively enumerable language whose complement is not recursively enumerable.
- There exists a recursively enumerable language that is not recursive i.e., the family of recursive language is a proper subset of the family of recursively enumerable language.
- If S_1 and S_2 are countable sets, then $S_1 \cup S_2$ and $S_1 \times S_2$ are countable, and union and cross-product can be extended to any number of finite sets.
- The set of all languages that are not recursively enumerable is uncountable.