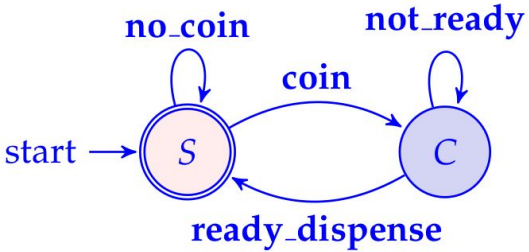
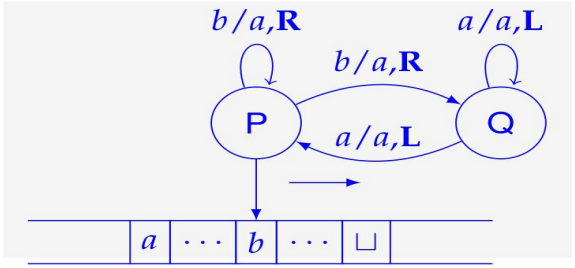
TURING MACHINE

From finite to infinite memory

Finite instruction machine with finite memory (Finite State Automata)



Finite instruction machine with unbounded memory (Turing machine)



From finite automata to Turing machine

1.A Turing machine can both write on the tape and read from it.

2. The read-write head can move both to left and right.

3. The tape is infinite.

4. Once accept/reject states are reached, the computation terminates at once.

Formal definition of Turing machine

A Turing machine M has five elements:

M.alphabet: Denoted by Σ, the tape alphabet is the set of symbols used to write input, output and intermediate strings on the tape.

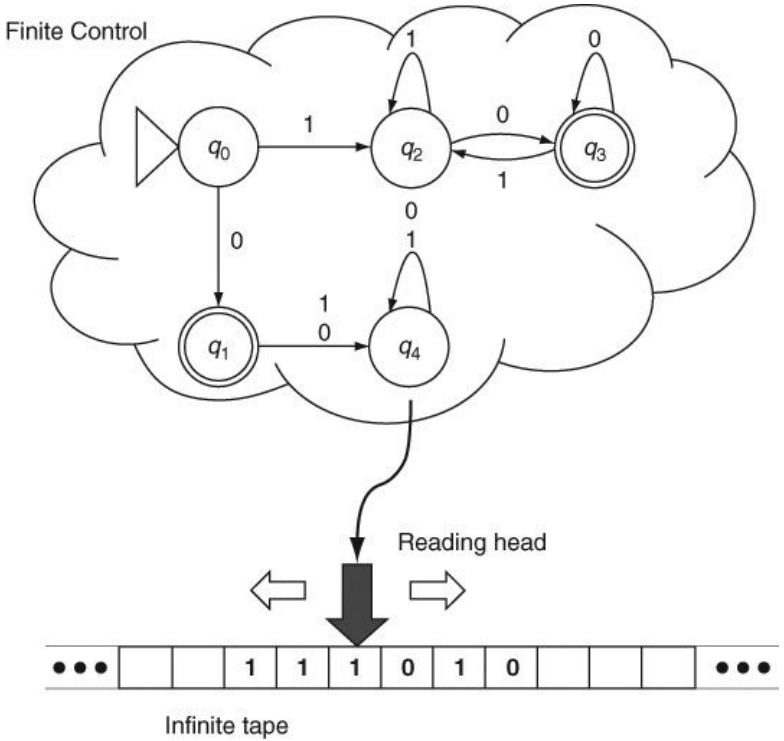
M.states: Also denoted by Q, it is the set of all states in the automaton.

M.startState: Usually denoted by q0, it is the start state of the automaton in the control unit.

M.finalStates: Denoted by QF, it is the subset of M.states that are final states of the automaton.

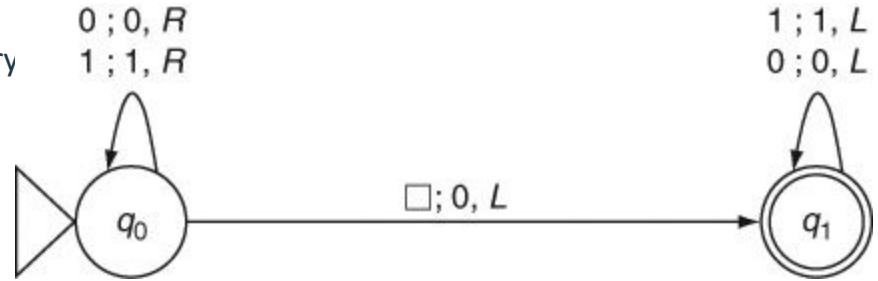
M.transitionFunction: It is a function δ from Q X Σ to Q X Σ X {Left, Right}, that is, a mapping from the current state and the current symbol under the reading head on the tape to a new state, a new symbol in the cell and a movement of the reading head by one cell to the left or to the right.

Turing machine

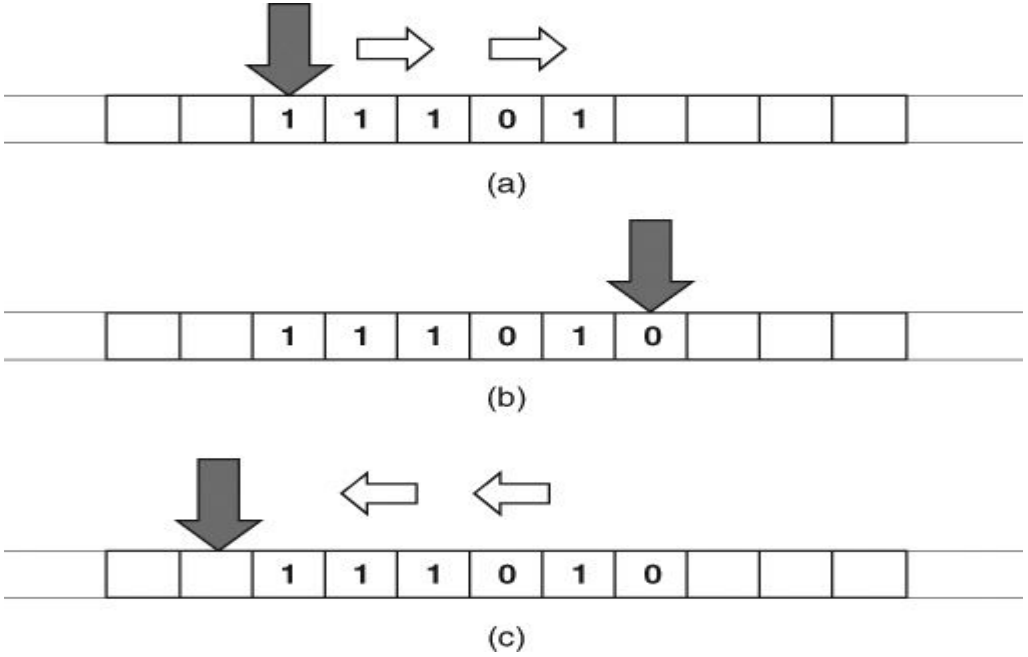


Turing machine example 1

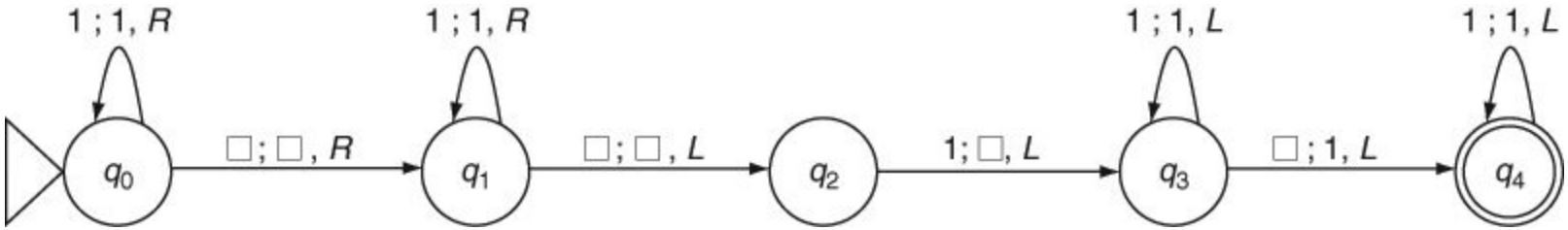
To double a given binary



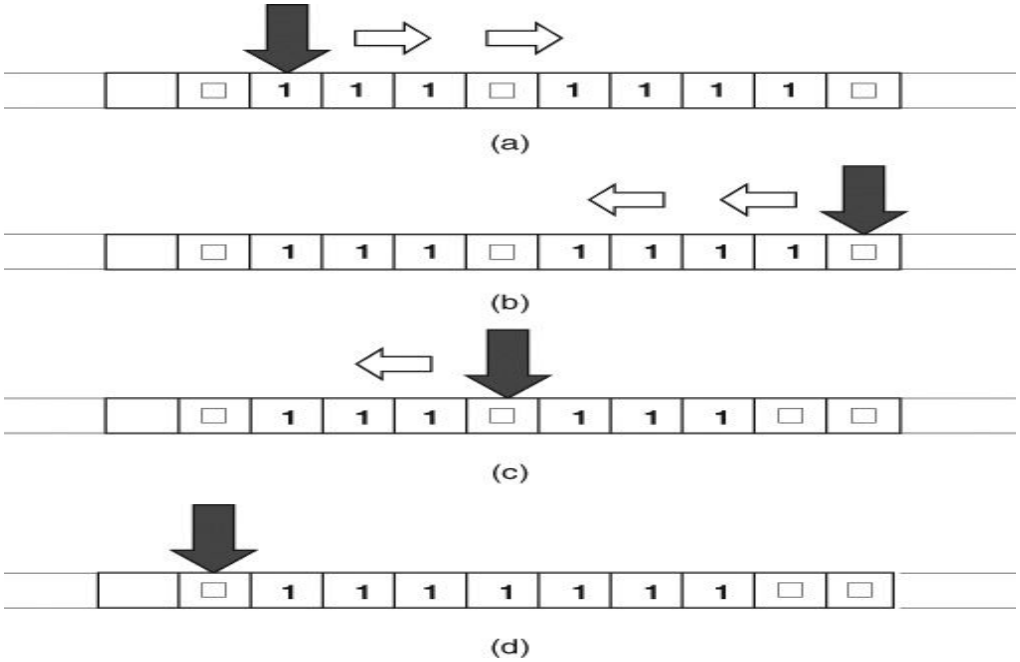
Tape contents



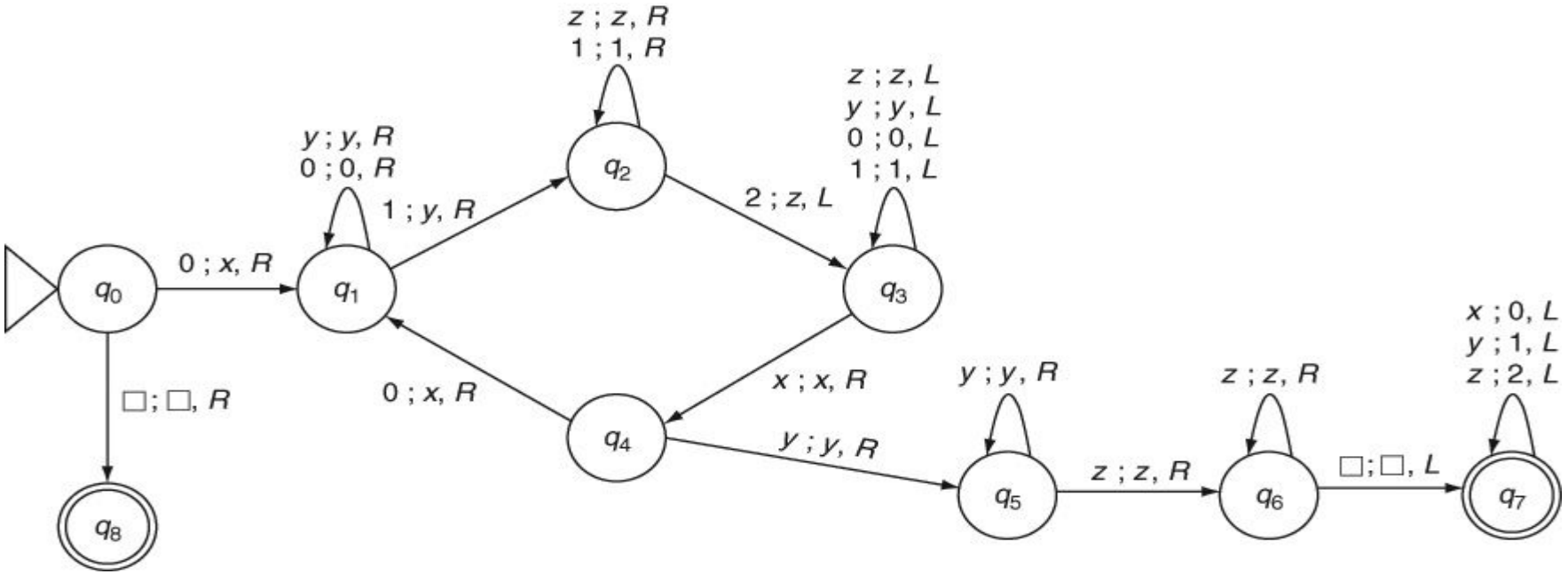
Adding Unary Numbers



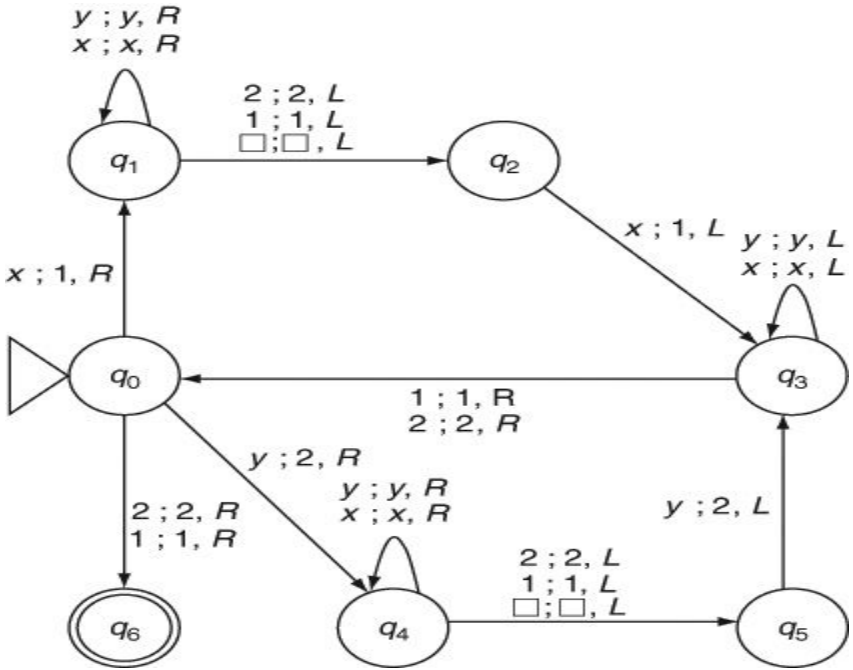
Tape contents



Turing machine for $0^n1^n2^n$



Turing machine for ww^R where w $\sum$={x,y}



Importance of Turing machine

- Turing Machine is an abstract computing machine with the power of real computer and mathematical definitions that can be computed
- TM consists of Finite state control and an infinite tape divided into cells.
- Blank is one of the tape symbol but not the input symbol
- TM accepts its input if it ever enters an accepting state.
- Language accepted by TM is known as Recursively Enumerable Language.
- Instantaneous description of TM describes current configuration of a TM by finite length string

Turing Machine can be represented in the following notations:

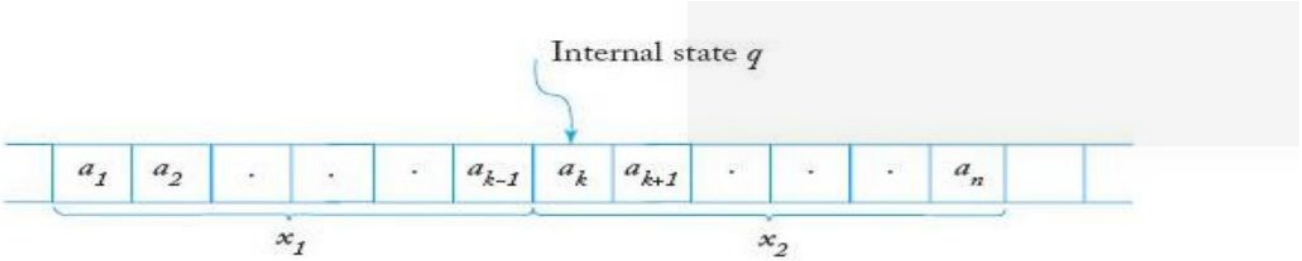
- Transitions in Turing Machine
- Instantaneous Description of Turing Machine
- Transition Diagram of Turing Machine

Instantaneous Description of Turing Machine

- Instantaneous description of TM describes the current state of the control unit, the contents of the tape and the position of the read-write head.
- The notation $x_1\ q_1 x_2$

  Or $a_1\ a_2 \ldots\ldots\ldots a_{k-1}\ q\ a_k a_{k+1} \ldots\ldots\ldots\ldots a_n$

  Is an instantaneous description of a machine in state q with the tape depicted as follows:

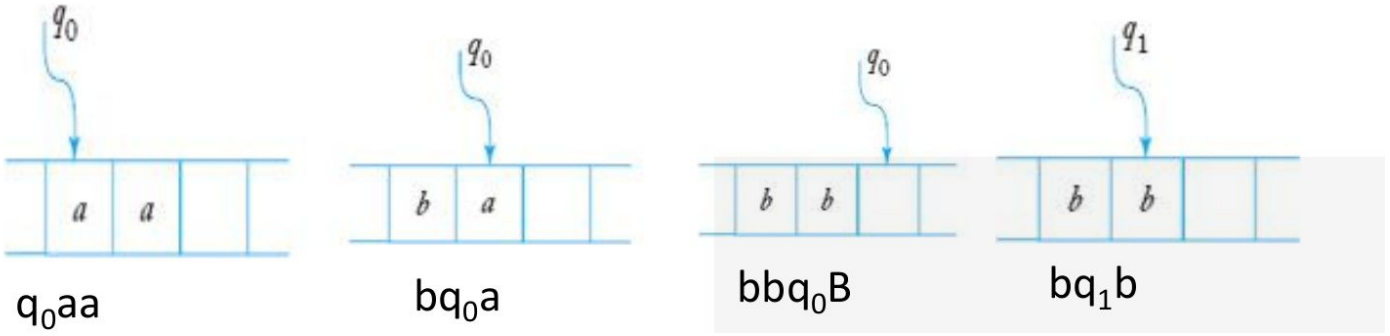


The symbols a_1 ,a_2..... ,a_n shows the tape contents, while q defines the state of control unit.

The convention is chosen so that the position of the read- write head is over the cell containing the symbol immediately following q.

For example The instantaneous description qBw indicates that the read- write head, is on the cell, to the immediate left of the first symbol, of w and that cell contains a blank.

Instantaneous Description corresponds to the pictures



$q_0\,aa \vdash bq_0\,a \vdash bbq_0\,B \vdash bq_1b$

A move from one configuration to another will be denoted by $\vdash$.

Thus if $\delta(q_1 , c) = (q_2 , e, R)$

Then the move $abq_1 cd \vdash abeq_2 d$ is made.

Similarly the symbol * over $\vdash$ represents the arbitrary number of moves in the description.

Example $q_0 aa \vdash bq_0 a \vdash bbq_0 B \vdash bq_1 b$ can be written as $q_0 aa \vdash^* bq_1 b$

The language Accepted by a Turing machine

- Turing machines can be viewed as language acceptor
- The language accepted by a Turing machine is the set of all strings which cause the machine to halt in a final state, when started in its standard initial configuration (q_0 , leftmost input symbol)
- A string is rejected if

  • The machine halts in a non final state, or

  • The machine never halts

Church Turing Thesis

Alonso Church (1903–1995)

Alan Turing (1912–1954)

- How powerful are Turing machines?
- Turing's Thesis contends that any computation carried out by mechanical means can be performed by some Turing machine
- An acceptance of Turing's Thesis leads to a definition of an algorithm:

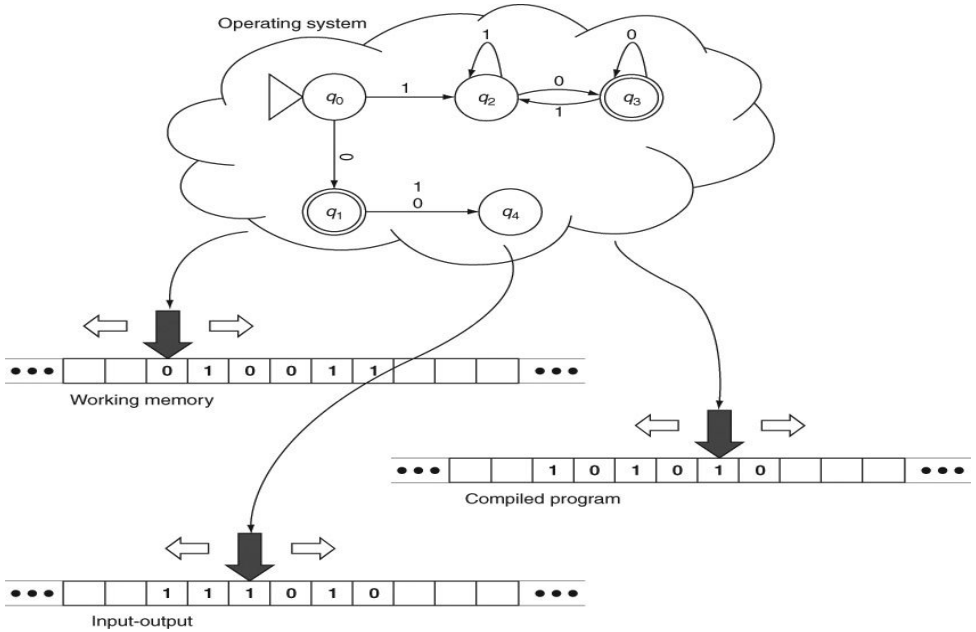
  An algorithm for a function $f : D \longrightarrow R$ is a Turing machine M, which given any d D on its tape, eventually halts with the correct answer $f(d) \in R$ on its tape

- Turing and Church showed that anything that can be computed can be computed by Turing Machine.
- No one so far has been able to find a computing problem that Turing machines can not compute but some other machine, mechanism can.
- This claim is known as **The Church-Turing Thesis.**

Universal Turing machine

- A Universal Turing Machine is a “programmable” machine that can compute anything that is computable
- A universal machine MU should be able to simulate the computation carried out by any hardcoded Turing machine M.
- The universal Turing machine MU has three tapes:
  - The input-output tape is used to simulate the tape of the particular machine M (or computable function) which it is simulating. The input as well as any output is written on this tape.
  - The compiled program tape stores “the program,” that is, the table of instructions or an encoding of the transition functions of the Turing machine M which it is simulating. This tape is a read-only tape.
  - The working memory tape, on which the current state of the machine M being simulated is maintained, along with any working memory (i.e., scratchpad) it needs.

Universal Turing machine



Binary Encoding of a Turing Machine

Standardize all elements of the Turing machine M except its transition function – let us assume that the alphabet is always {a1, a2, a3, …}; that its states are always numbered q1, q2, q3, etc. where q1 is always the start state and q2 is always the single final state.

Each transition from the state qi with current symbol aj going to state qk and new symbol on the tape al with the reading head moving left (L) or right (R), that is, (qi, aj) (qk, al, R/L) is encoded as a sequence of numbers (i, j, k, l, 1/11) where 1 stands for R and 11 for L. This sequence can be encoded as unary numbers separated by a single 0. For example, the transition (q3, a1) (q4, a2, R) would be encoded as 111010111101101.

Each such encoding of a transition in M can be padded by two 0 s on either side, for example, as 0011110101111010110100 and the encodings of all the transitions can be concatenated to create a single binary string that represents the entire Turing machine M. The resulting string is a compiled program or executable binary program of the machine M.

Recursive and Recursively Enumerable Languages

- Recursive Languages

A Language L is said to be recursive if there exists a Turing machine which will accept all the strings in 'L' and reject all the strings not in 'L'. The Turing Machine will halt every time and give an answer (accepted or rejected) for each string input.

- Recursively Enumerable Languages: -

A language L is said to be recursively enumerable language if there exists a Turing machine which will accept (and therefore halt) for all the input strings which are in L ➢But may or may not halt for all input strings which are not in L.

The Grammar for these languages is known as unrestricted grammar.

Decidability and Undecidability

Decidable Languages

➢A Language L is decidable if it is a recursive language. All decidable languages are recursive languages and vice-versa.

Partially Decidable Languages

➢A Language L is partially decidable if ‘L’ is a recursive enumerable language.

• Undecidable Languages

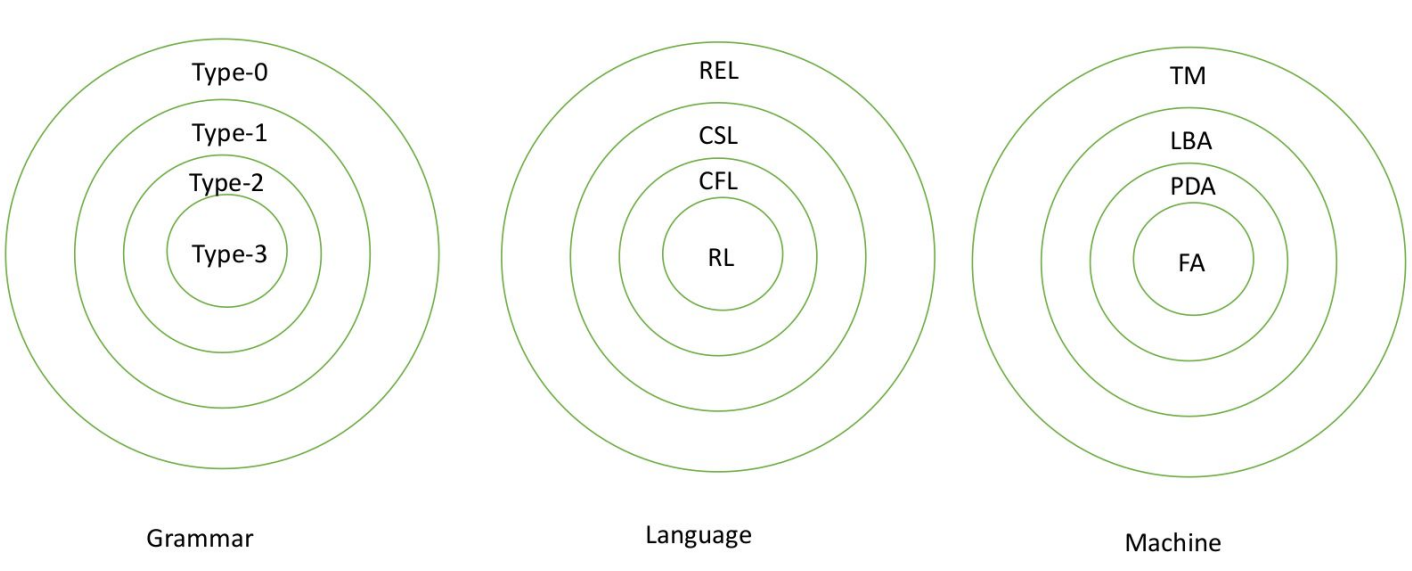
➢A Language is undecidable if it is not decidable.

➢An undecidable language may sometimes be partially decidable but not decidable.

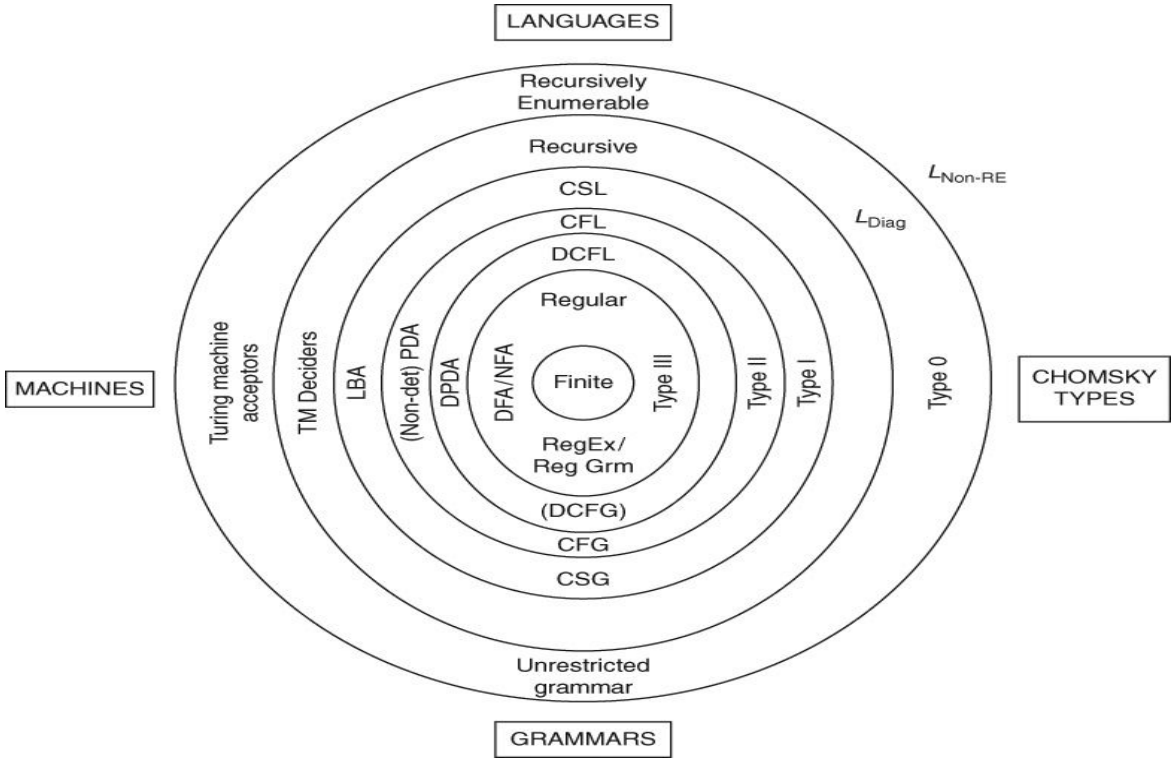
➢If a language is not even partially decidable, then there exists no Turing machine for

that language.

Chomsky Hierarchy



Chomsky Hierarchy



Halting problem of Turing Machine

Halting Problem Says:

- Given a program “Will it halt??”
- Given a Turing Machine, will it halt when running on some given input string?
- Given some program written in some language, will it ever get into an infinite loop or will it always terminate?

Ans:- The best we can do is run the program and see whether it halts.

Post Correspondence Problem

A string matching problem Given two sets (sequences, in fact) of strings W and V (over the same alphabet) each having n strings:

W = {w1, w2, w3, …, wn} and V = {v1, v2, v3, …, vn}

The problem is to find some concatenation of one or more strings from the set W, in any order, where the resulting concatenated string is identical to the corresponding concatenation, in the same order, of strings from the set V.

The simpler version of the PCP is to find some permutation of the indexes of elements of the sets 1, 2, 3, 4, …, say, i1, i2, i3, i4, … so that when concatenated in that order, we get identical concatenations, that is,

wi1.wi2.wi3… = vi1.vi2.vi3…

The V strings must be concatenated in the same order as the W strings. Can we find such a permutation given any two sets of strings W and V each containing n strings?

The general case of the PCP is more complex.

What if we allow repetitions of strings? That is, each wi or vi can be used any number of times in the concatenations.

As such, there is no limit on the length of the concatenations, although we only consider finite numbers of concatenations as solutions to the PCP.

There are instances of the PCP that have no solution if repetitions are not allowed but do present solutions when repetitions are included.

What is a good algorithm to solve the Post Correspondence Problem?

There is no algorithm!

PCP is not computable! It is an undecidable problem!

Consider

W = {bb, baa, bbb}

V = {bbb, aab, bb}

Let us try: $i_1 = 1$, $i_2 = 2$ and $i_3 = 3$, and

$wi_1.wi_2.wi_3$ = bb.baa.bbb =

$vi_1.vi_2.vi_3$ = bbb.aab.bb = bbbaabbb