5.1 Introduction to Turing Machine

- In 1936, Mr. Alan Turing introduced advanced mathematical model for modern digital computer which was known as Turing machine. Turing machine is advanced machine of FA and PDA.
- This simple mathematical model has no difference between input and output set. This mathematical model can be constructed to accept a given language or to carry out some algorithm.
- This model sometimes uses it's own output as input for further computation.
- This machine or model can select current location and also decides location of memory by moving left or right. This mathematical model known as Turing machine has become an effective procedure.

- In short Turing machine is equal to,

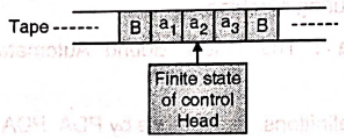
  TM = FA + Tape

  FA = Finite Automata

  Tape

  1. Infinite memory unit
  2. Infinite cell (Each cell contains only one alphabet at one time)
  3. Head of records move left or right.

- Turing machine's mathematical model consists of **Head** (moves right or left or stays in position), **infinite tape** and **finite set of states.**



- **Working of Turing machine** can be explained as follows, where, one move of TM shows

  1. Changing state location
  2. Replacing old symbol by new
  3. Left or right move of head

Definition of Turing Machine:

Let M define TM with seven Tuple,

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

Where,

Q = Finite set of states

Γ = Finite set of tape symbols

$B \in \Gamma$ = Blank space symbol

Σ = Set of input symbols where $\Sigma \subseteq \Gamma - \{B\}$ (excluding blank symbols)

δ = Function for next move $[Q \infty \Gamma$ to $Q \infty \Gamma \infty \{S\ R\}]$

q_0 = Start state

$F \subseteq Q$ = Final state set

Turing Machine (Formal Definition)

Representations of Turing Machines:

1. Instantaneous Description:

2. Transition Table:

- We can define δ in the form of table is denoted as transition table.

	a
q_0	x, q_1, R

- Transition function δ gives a alphabet from q_0 stated replaced by x, next state is q_1 and move to the right.

3. Transition Graph

We can use transition diagram to represent Turing machine. Directed edges are used to represent transition of states.

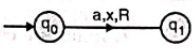


Fig. 5.1.12

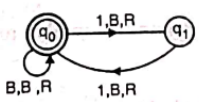
Where, q_0 and q_1 are transition states

a is current alphabet in insert tape

x is replacing alphabet

R means move to the right

E.g. Construct TM for string having even number of 1's over $\Sigma = \{1\}$.



Designing Turing machine

1. Despite their simplicity R/W Turing machines are very powerful computing devices.
2. Head of Turing machine scans the symbol and machine remembers it, for future.
3. Minimization of number of states is activity by changing the states only when there is a change in the written symbol or when there is a change in the movement of the R/W head.
4. Turing machine works as mathematical model for solving complexity of language recognition problem.
5. TM is used for solving arithmetic operations, problems.
6. Diagram of language recognizing by Turing machine.

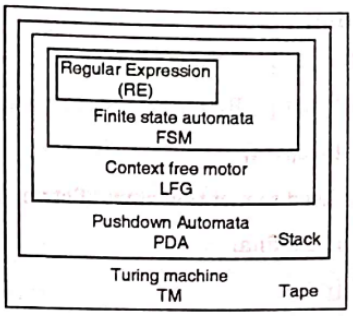


Fig. 5.2.1

All Regular Expressions and Context Free language are recognized by a particular type of machine, regular and not context free are also accepted by Turing machine.

Types of Turing Machines:

1. Multitrack Turing machine
2. Multitape Turing machine
3. Non-deterministic Turing machine
4. Multistack Turing machine
5. Counter machine.

Multitrack Turing machine:
A multi-track Turing machine is a type of Turing machine with multiple tracks, each having its own head for reading and writing symbols. The transition function for a multi-track Turing machine specifies how it behaves when transitioning between states based on the current symbols on all of its tapes. In brief, the transition function can be described as follows:

For a multi-track Turing machine:
1. The transition function takes as input the current state, the symbols on each of the tapes under their respective heads, and the machine's current internal state.

2. It provides instructions for the machine to transition to a new state, write symbols on each tape, move each tape's head (left or right), and possibly change the internal state based on this input.

3. The transition function is typically defined as a set of quintuples of the form (current state, symbols under the heads, new state, symbols to write, head movements) for each possible combination of current state and symbols on the tapes.

4. The transition function allows the machine to navigate and manipulate its multiple tapes, making it a powerful computational model for solving various computational problems.

In summary, the transition function of a multi-track Turing machine governs how the machine operates by specifying state transitions and tape manipulations based on the current symbols and the internal state of the machine.

Multitape Turing machine:
A multi-tape Turing machine has multiple tapes, each with its own head for reading and writing symbols. The transition function for a multi-tape Turing machine is relatively simple in concept:

The transition function specifies how the machine behaves when transitioning between states based on the current symbols under all of its tape heads. In brief:

1. It takes as input the current state and the symbols currently under each tape head.

2. It provides instructions for the machine to transition to a new state, write symbols on each tape, move each tape's head (left, right, or stay), and possibly change the internal state based on this input.

3. The transition function is typically defined as a set of quintuples of the form (current state, symbols under the tape heads, new state, symbols to write, and head movements) for each possible combination of current state and symbols on the tapes.

4. The transition function allows the machine to navigate and manipulate its multiple tapes, making it a powerful computational model for solving various computational problems.

In summary, the transition function of a multi-tape Turing machine governs how the machine operates by specifying state transitions and tape manipulations based on the current symbols and the internal state of the machine.

Non deterministic Turing machine:
A non-deterministic Turing machine (NDTM) is a theoretical computational model that extends the concept of a deterministic Turing machine (DTM). While DTMs have a single, well-defined transition for each combination of current state and symbol under the tape head, NDTMs allow for multiple possible transitions from a given state and symbol, introducing non-determinism.

Key characteristics of non-deterministic Turing machines include:

1. **Non-Deterministic Choices:** In an NDTM, at any given configuration, there can be multiple possible transitions. The machine can, at any point, have multiple choices on which state to move to and what symbol to write on the tape.

2. **Acceptance:** An NDTM accepts an input if there exists at least one sequence of non-deterministic choices that leads to an accepting state.

3. **Rejection:** An NDTM rejects an input if there are no sequences of choices that lead to an accepting state.

Non-deterministic Turing machines are particularly useful in theoretical computer science for analyzing problems and determining whether solutions exist. They are often used in discussions of complexity theory, where non-deterministic computation can help identify upper bounds on the computational complexity of problems.

While NDTMs are powerful for analysis and theoretical discussions, they don't represent physical machines that can be built and operated. Deterministic Turing machines, as well as other models like probabilistic Turing machines and quantum Turing machines, provide insight into the practical limits of computation.

multistack turing machine
A multi-stack Turing machine (MSTM) is an abstract computational model that extends the classical Turing machine by providing multiple stacks for storage and manipulation of symbols. It's a theoretical concept used in computer science and automata theory to study the computational capabilities of machines with additional storage structures beyond the traditional tape used by a standard Turing machine. Here are the key characteristics of a multi-stack Turing machine:

1. **Multiple Stacks:** Instead of a single tape, an MSTM has two or more stacks. Each stack operates independently and can push or pop symbols. These stacks provide extra memory storage for the machine's computation.

2. **State Transitions:** Similar to a regular Turing machine, an MSTM has a transition function that defines how the machine behaves based on the current state and the symbols on the top of each stack. The transition function specifies state changes, symbol pushes or pops, and head movements for each stack.

3. **Computational Power:** Multi-stack Turing machines are more powerful than standard Turing machines in the sense that they can solve a broader class of problems. They are equivalent in computational power to other models like the pushdown automaton, which can recognize context-free languages, and they can be used to analyze more complex grammars and languages.

Multi-stack Turing machines are mainly a theoretical concept and are not meant to represent real-world computers. They are valuable for studying the theoretical limits of computation and for understanding the complexity of problems in the context of formal language theory and automata theory.

Counter Turing Machine:
A counter machine, also known as a counter Turing machine, is a theoretical model of computation that is a simplified variant of a Turing machine. It is specifically designed to perform counting and arithmetic operations. Unlike a traditional Turing machine, which uses an infinite tape for input and storage, a counter machine uses a finite number of counters for its operations.

Here are the key characteristics of a counter machine:

1. **Counters:** A counter machine has a finite number of counters (usually denoted as C1, C2, C3, and so on). Each counter can hold a non-negative integer value.

2. **Operations:** The primary operations of a counter machine are incrementing or decrementing the values in the counters. The machine can also perform conditional branching based on the values in the counters.

3. **Language Recognition:** Counter machines are not general-purpose computers like Turing machines. They are particularly well-suited for recognizing and generating specific languages, especially those that involve counting, such as recognizing palindromes or balanced parentheses.

4. **Simplicity:** Counter machines are simpler in their computational model compared to Turing machines, making them a useful tool for understanding the complexity of problems related to counting and arithmetic.

5. **Limitation:** Counter machines are less powerful than Turing machines. They cannot perform arbitrary computations like a Turing machine can, and they have a finite number of counters, which limits their expressiveness.

Counter machines are used in theoretical computer science and automata theory to study the computational complexity of problems related to counting and arithmetic. They are a valuable tool for understanding the computational bounds of specific language classes and for analyzing problems that can be solved using counter-like operations.

Church-Turing Thesis

- Many mathematicians were finding if there is a procedure, algorithm for solving a problem that can be implemented on Turing machine Alonzo church also worked on same problem and he solved and some problem Chomsky unrestricted grammar recursive recues function theory by λ calculus.
- If a computational problem P is solvable by human being then it is solved by computer and if its solvable by computer then it is solved by Turing machine.
- Mr. Church formalise simple hypothesis over it. If a computational problem P is solvable by human being then it is directly solve by Turing machine.

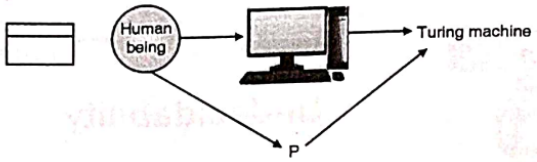


Fig. 6.1.1

Above formalism leads to,

1. **Turing theory (Weak form) :** A turing machine can compute anything that can be composed by digital computer.
2. **Turing theory (strong form) :** A turing machine can compute anything that can be computed.

Define string,

D = {p : P is a polynomial with an integral root }

Consider turing machine,

M = {The input is a polynomial over variable $x_1, x_2, \ldots x_n$ }

- Evaluate P on n tuple of integers.
- If p ever evaluates to 0, accept
- M recognized D but does not decide D.

The Church Turing thesis explain similarities between recursive function and computable one.

1. Recursive function is a mathematical concept.
2. Computable function is more of constructivism.

Now it is universally accepted by computer scientists that turing machine is a mathematical model of an algorithm.

UTM:

1. The limitations of Turing machine is that it only executes one program at a time hence it must be constructed for every new computation to be performed for every input output relation.
2. This is why UTM was introduced as a solution that could be used to worked like a simulator.
3. It is capable of simulating any action of Turing machine on any input.
4. Attributes:
   Reprogrammable machine
   Simulates any other Turing machine

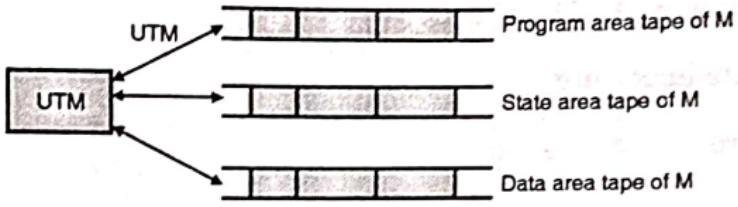


Fig. 6.2.1