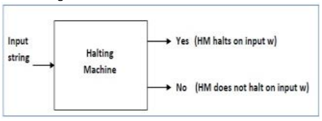
The Turing machine M = (Q. ∑., 6, q0 , B , F) Where, Q is the set of finite states
Z is the set of input alphabets is the set of tape symbols
6 is the transition function Q x to Q x x (L,R) q is the start state
B is the special symbol indicating blank character. F is the set of final state

Different types of Turing Machine?
-Multi tape Turing Machine
-Non-deterministic Turing Machine
-Multi-dimensional Turing Machine
- Multi Read Turing Machine

Halting Problem of Turing machine
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 –



Chomsky Hierarchy
Chomsky Hierarchy represents the class of languages that are accepted by the different machine. The category of language in Chomsky's Hierarchy is as given below:

1. Type 0 known as Unrestricted Grammar.
2. Type 1 known as Context Sensitive Grammar.
3. Type 2 known as Context Free Grammar.
4. Type 3 Regular Grammar.

Type 0 Grammar:
Type 0 grammar is known as Unrestricted grammar. There is no restriction on the grammar rules of these types of languages. These languages can be efficiently modeled by Turing machines.
For example:
bAa → aa
S → s

Type 1 Grammar:
Type 1 grammar is known as Context Sensitive Grammar. The context sensitive grammar is used to represent context sensitive language. The context sensitive grammar follows the following rules:
1.The context sensitive grammar may have more than one symbol on the left hand side of their production rules.
2.The number of symbols on the left-hand side must not exceed the number of symbols on the right-hand side.
3.The rule of the form A → ε is not allowed unless A is a start symbol. It does not occur on the right-hand side of any rule.**4.**The Type 1 grammar should be Type 0. In type 1, Production is in the form of V → T
Where the count of symbol in V is less than or equal to T.

- **Explain the various applications of regular expressions.**
  Design of compilers
- To define languages
- Declarative way to express set of strings
- Validation — i.e., checking the correction of i/p
- Tokenization: i.e.,conversion of string of characters into a sequence of words for later interpretation in pattern matching.
- Test for a pattern within a string.
- Replace text in a document.
- Extract a substring from a string based upon a pattern match.
- Used in languages like JScript and e for string handling.
- Helps in implementing complex match logic in databases.

Dfa vs Nfa
DFA: M=(Q, Σ, δ,q0,F) δ: Q X Σ → Q
1. DFA stands for Deterministic Finite Automata.
2. For each symbolic representation of the alphabet, there is only one state transition in DFA.3. DFA cannot use Empty String transition.4. DFA can be understood as one machine. 5. In DFA, the next possible state is distinctly set.6. DFA is more difficult to construct.
NFA : M=(Q, Σ, δ,q0,F) δ: Q X Σ → 2^Q 1. NFA stands for Nondeterministic Finite Automata. 2. No need to specify how does the NFA react according to some symbol. 3. NFA can use Empty String transition. 4. NFA can be understood as multiple little machines computing at the same time. 5. In NFA, each pair of state and input symbol can have many possible next states. 6. NFA is easier to construct.

What is trap state
A state for which there exists transitions to itself for all the input symbols chosen from ∑
Define regular expression.
The language accepted by finite automata is called regular language. A regular language can be described using regular expressions, consisting of alphabets in Z and the operators *','.','+'. The order of evaluation of regular expression is determined by parenthesis and the operator precedence '*','.' And '+' respectively.
What is Pumping lemma Pumping lemma is a method of pumping (generating) many input string from a given string it is used to show that certain languages are not regular.
State Arden's theorem. If P and Q are two regular expressions over , and if P does not contain , then the following equation in R given by R = Q + RP has an unique solution i.e., R = QP*." That means, whenever we get any equation in the form of R = Q + RP, then we can directly replaced by R = QP*. So, here first we will prove that R = QP* is the solution of this equation and then we will also prove that it is the unique solution of this equation.
Define GNF: Let G= (V,T,P,S) be a CFG. The CFG is a said to be in GNF if all the production are of the form Adam Where aeT«eV' i.e., the first symbol on the right hand side of the production must be a terminal & it can be followed by 0 or more variable.
What are useful and useless symbols in grammer?
In a CFG, G=(V,T,P, S), x is useless, if it does not satisfy either of the following condition
(a)m'=> w, where w is in T .

Pushdown Automata(PDA)
- Pushdown automata is a way to implement a CFG in the same way we design DFA for a regular grammar. A DFA can remember a finite amount of information, but a PDA can remember an infinite amount of information.
- Pushdown automata is simply an NFA augmented with an "external stack memory". The addition of stack is used to provide a last-in-first-out memory management capability to Pushdown automata. Pushdown automata can store an unbounded amount of information on the stack. It can access a limited amount of information on the stack. A PDA can push an element onto the top of the stack and pop off an element from the top of the stack. To read an element into the stack, the top elements must be popped off and are lost.
- A PDA is more powerful than FA. Any language which can be acceptable by FA can also be acceptable by PDA. PDA also accepts a class of language which even cannot be accepted by FA. Thus PDA is much more superior to FA.

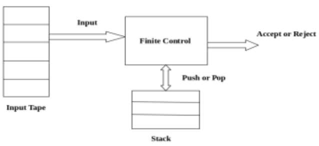


Fig: Pushdown Automata

Pushdown Automata(PDA)
- Pushdown automata is a way to implement a CFG in the same way we design DFA for a regular grammar. A DFA can remember a finite amount of information, but a PDA can remember an infinite amount of information.
- Pushdown automata is simply an NFA augmented with an "external stack memory". The addition of stack is used to provide a last-in-first-out memory management capability to Pushdown automata. Pushdown automata can store an unbounded amount of information on the stack. It can access a limited amount of information on the stack. A PDA can push an element onto the top of the stack and pop off an element from the top of the stack. To read an element into the stack, the top elements must be popped off and are lost.
- A PDA is more powerful than FA. Any language which can be acceptable by FA can also be acceptable by PDA. PDA also accepts a class of language which even cannot be accepted by FA. Thus PDA is much more superior to FA.

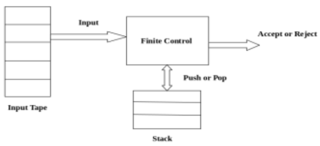
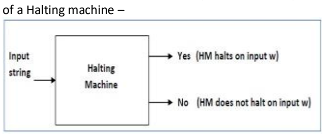


Fig: Pushdown Automata

The Turing machine M = (Q. ∑., 6, q0 , B , F) Where, Q is the set of finite states
Z is the set of input alphabets is the set of tape symbols
6 is the transition function Q x to Q x x (L,R) q is the start state
B is the special symbol indicating blank character. F is the set of final state

Different types of Turing Machine?
-Multi tape Turing Machine
-Non-deterministic Turing Machine
-Multi-dimensional Turing Machine
- Multi Read Turing Machine

Halting Problem of Turing machine
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 –



Chomsky Hierarchy
Chomsky Hierarchy represents the class of languages that are accepted by the different machine. The category of language in Chomsky's Hierarchy is as given below:

5. Type 0 known as Unrestricted Grammar.
6. Type 1 known as Context Sensitive Grammar.
7. Type 2 known as Context Free Grammar.
8. Type 3 Regular Grammar.

Type 0 Grammar:
Type 0 grammar is known as Unrestricted grammar. There is no restriction on the grammar rules of these types of languages. These languages can be efficiently modeled by Turing machines.
For example:
bAa → aa
S → s

Type 1 Grammar:
Type 1 grammar is known as Context Sensitive Grammar. The context sensitive grammar is used to represent context sensitive language. The context sensitive grammar follows the following rules:
1.The context sensitive grammar may have more than one symbol on the left hand side of their production rules.
2.The number of symbols on the left-hand side must not exceed the number of symbols on the right-hand side.
3.The rule of the form A → ε is not allowed unless A is a start symbol. It does not occur on the right-hand side of any rule.**4.**The Type 1 grammar should be Type 0. In type 1, Production is in the form of V → T
Where the count of symbol in V is less than or equal to T.

- **Explain the various applications of regular expressions.**
  Design of compilers
- To define languages
- Declarative way to express set of strings
- Validation — i.e., checking the correction of i/p
- Tokenization: i.e.,conversion of string of characters into a sequence of words for later interpretation in pattern matching.
- Test for a pattern within a string.
- Replace text in a document.
- Extract a substring from a string based upon a pattern match.
- Used in languages like JScript and e for string handling.
- Helps in implementing complex match logic in databases.

Dfa vs Nfa
DFA: M=(Q, Σ, δ,q0,F) δ: Q X Σ → Q
1. DFA stands for Deterministic Finite Automata.
2. For each symbolic representation of the alphabet, there is only one state transition in DFA.3. DFA cannot use Empty String transition.4. DFA can be understood as one machine. 5. In DFA, the next possible state is distinctly set.6. DFA is more difficult to construct.
NFA : M=(Q, Σ, δ,q0,F) δ: Q X Σ → 2^Q 1. NFA stands for Nondeterministic Finite Automata. 2. No need to specify how does the NFA react according to some symbol. 3. NFA can use Empty String transition. 4. NFA can be understood as multiple little machines computing at the same time. 5. In NFA, each pair of state and input symbol can have many possible next states. 6. NFA is easier to construct.

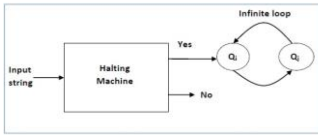
What is trap state
A state for which there exists transitions to itself for all the input symbols chosen from ∑
Define regular expression.
The language accepted by finite automata is called regular language. A regular language can be described using regular expressions, consisting of alphabets in Z and the operators *','.','+'. The order of evaluation of regular expression is determined by parenthesis and the operator precedence '*','.' And '+' respectively.
What is Pumping lemma Pumping lemma is a method of pumping (generating) many input string from a given string it is used to show that certain languages are not regular.
State Arden's theorem. If P and Q are two regular expressions over , and if P does not contain , then the following equation in R given by R = Q + RP has an unique solution i.e., R = QP*." That means, whenever we get any equation in the form of R = Q + RP, then we can directly replaced by R = QP*. So, here first we will prove that R = QP* is the solution of this equation and then we will also prove that it is the unique solution of this equation.
Define GNF: Let G= (V,T,P,S) be a CFG. The CFG is a said to be in GNF if all the production are of the form Adam Where aeT«eV' i.e., the first symbol on the right hand side of the production must be a terminal & it can be followed by 0 or more variable.
What are useful and useless symbols in grammer?
In a CFG, G=(V,T,P, S), x is useless, if it does not satisfy either of the following condition
(a)m'=> w, where w is in T .

terminal and non terminal symbols in grammer?
Non-terminals are syntactic variables that denote sets of strings. The non-terminals define sets of strings that help define the language generated by the grammar. A set of tokens, known as Terminal symbols (Z). Terminals are the basic symbols from which strings are formed.
What is left most derivation in CFG?
A derivation A* => w is called left most derivation if we apply a production only to the left most variable at every step.
What are the different types of grammar?
There are 4 types of grammer:-
•Type 0 Grammer (Phrase structured/ unrestricted grammer)
•Type I Grammer(Context sensitive grammer)
•Type 2 Grammer(Context free grammer)
•Type 3 Grammer(Regular grammer)
Define grammar. Give one example
A grammer is a quad tuple G (V, T, P ,S) where, V is a finite set of variables or non terminals. T is a finite set of terminals.P is a finite set of production rules. Each production is of the following from Aha where, A is a string of symbol from (V IT)*
α Is a string of symbol from (V UT)*
S is the start symbol & S€V Example: SGaAb/€
applications of context free grammar.
1.Parsers
2.Markup language
3.Finite automata
4.Digital design

Types of Turing Machine
1. **Multiple track Turing Machine:**
A k-tack Turing machine(for some k>0) has k-tracks and one R/W head that reads and writes all of them one by one.A k-track Turing Machine can be simulated by a single track Turing machine2. **Two-way infinite Tape Turing Machine:**Infinite tape of two-way infinite tape Turing machine is unbounded in both directions left and right. Two-way infinite tape Turing machine can be simulated by one-way infinite Turing machine(standard Turing machine).3. **Multi-tape Turing Machine:**
It has multiple tapes and controlled by a single head.
The Multi-tape Turing machine is different from k-track Turing machine but expressive power is same.
Multi-tape Turing machine can be simulated by single-tape Turing machine.**4. Multi-tape Multi-head Turing Machine:**
The multi-tape Turing machine has multiple tapes and multiple headsEach tape controlled by separate head Multi-Tape Multi-head Turing machine can be simulated by standard Turing machine.**5. Multi-dimensional Tape Turing Machine:**It has multi-dimensional tape where head can move any direction that is left, right, up or down.
Multi dimensional tape Turing machine can be simulated by one-dimensional Turing machine.**6. Multi-head Turing Machine:**A multi-head Turing machine contain two or more heads to read the symbols on the same tape.
In one step all the heads sense the scanned symbols and move or write independently.Multi-head Turing machine can be simulated by single head Turing machine.
7. Non-deterministic Turing Machine:
A non-deterministic Turing machine has a single, one way infinite tape.A non-deterministic Turing machine is equivalent to deterministic Turing machine.

For example:
S → AT
T → xy
A → a
Type 2 Grammar:
Type 2 Grammar is known as Context Free Grammar. Context free languages are the languages which can be represented by the context free grammar (CFG). Type 2 should be type 1. The production rule is of the form
A → α
Where A is any single non-terminal and is any combination of terminals and non-terminals.
For example:
A → aBb
A → b
B → a
Type 3 Grammar:
Type 3 Grammar is known as Regular Grammar. Regular languages are those languages which can be described using regular expressions. These languages can be modeled by NFA or DFA.
Type 3 is most restricted form of grammar. The Type 3 grammar should be Type 2 and Type 1. Type 3 should be in the form of
V → T*V / T*
For example:
A → xy

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**.
PCP (Post Correspondence Problem).
The Post Correspondence Problem (PCP), introduced by Emi1 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 = (xi, X2, X3, ……..Xn)
$N = (y_1, y_2, y_3, \ldots, y_n)$

We can say that there is a Post Correspondence Solution, if for some i1,i2,…….ik, where 1<=ij<-n the condition xi1……..xik=yil…….yik satisfies

PDA Components:
Input tape: The input tape is divided in many cells or symbols. The input head is read-only and may only move from left to right, one symbol at a time.
Finite control: The finite control has some pointer which points the current symbol which is to be read.
Stack: The stack is a structure in which we can push and remove the items from one end only. It has an infinite size. In PDA, the stack is used to store the items temporarily.
Formal definition of PDA:
The PDA can be defined as a collection of 7 components:
1. **Q:** the finite set of states
2. **∑:** the input set
3. **Γ:** a stack symbol which can be pushed and popped from the stack
4. **q0:** the initial state
5. **Z:** a start symbol which is in Γ.
6. **F:** a set of final states
7. **δ:** mapping function which is used for moving from current state to next state.

Instantaneous Description (ID)
ID is an informal notation of how a PDA computes an input string and make a decision that string is accepted or rejected.
An instantaneous description is a triple (q, w, α) where:
q describes the current state.
w describes the remaining input.
α describes the stack contents, top at the left.
Turnstile Notation:
⊢ sign describes the turnstile notation and represents one move.
⊢* sign describes a sequence of moves.
For example,
(p, b, T) ⊢ (q, w, α)

PDA Components:
Input tape: The input tape is divided in many cells or symbols. The input head is read-only and may only move from left to right, one symbol at a time.
Finite control: The finite control has some pointer which points the current symbol which is to be read.
Stack: The stack is a structure in which we can push and remove the items from one end only. It has an infinite size. In PDA, the stack is used to store the items temporarily.
Formal definition of PDA:
The PDA can be defined as a collection of 7 components:
8. **Q:** the finite set of states
9. **∑:** the input set
10. **Γ:** a stack symbol which can be pushed and popped from the stack
11. **q0:** the initial state
12. **Z:** a start symbol which is in Γ.
13. **F:** a set of final states
14. **δ:** mapping function which is used for moving from current state to next state.

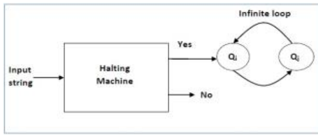
Instantaneous Description (ID)
ID is an informal notation of how a PDA computes an input string and make a decision that string is accepted or rejected.
An instantaneous description is a triple (q, w, α) where:
q describes the current state.
w describes the remaining input.
α describes the stack contents, top at the left.
Turnstile Notation:
⊢ sign describes the turnstile notation and represents one move.
⊢* sign describes a sequence of moves.
For example,
(p, b, T) ⊢ (q, w, α)

terminal and non terminal symbols in grammer?
Non-terminals are syntactic variables that denote sets of strings. The non-terminals define sets of strings that help define the language generated by the grammar. A set of tokens, known as Terminal symbols (Z). Terminals are the basic symbols from which strings are formed.
What is left most derivation in CFG?
A derivation A* => w is called left most derivation if we apply a production only to the left most variable at every step.
What are the different types of grammar?
There are 4 types of grammer:-
•Type 0 Grammer (Phrase structured/ unrestricted grammer)
•Type I Grammer(Context sensitive grammer)
•Type 2 Grammer(Context free grammer)
•Type 3 Grammer(Regular grammer)
Define grammar. Give one example
A grammer is a quad tuple G (V, T, P ,S) where, V is a finite set of variables or non terminals. T is a finite set of terminals.P is a finite set of production rules. Each production is of the following from Aha where, A is a string of symbol from (V IT)*
α Is a string of symbol from (V UT)*
S is the start symbol & S€V Example: SGaAb/€
applications of context free grammar.
1.Parsers
2.Markup language
3.Finite automata
4.Digital design

Types of Turing Machine
1. **Multiple track Turing Machine:**
A k-tack Turing machine(for some k>0) has k-tracks and one R/W head that reads and writes all of them one by one.A k-track Turing Machine can be simulated by a single track Turing machine2. **Two-way infinite Tape Turing Machine:**Infinite tape of two-way infinite tape Turing machine is unbounded in both directions left and right. Two-way infinite tape Turing machine can be simulated by one-way infinite Turing machine(standard Turing machine).3. **Multi-tape Turing Machine:**
It has multiple tapes and controlled by a single head.
The Multi-tape Turing machine is different from k-track Turing machine but expressive power is same.
Multi-tape Turing machine can be simulated by single-tape Turing machine.**4. Multi-tape Multi-head Turing Machine:**
The multi-tape Turing machine has multiple tapes and multiple headsEach tape controlled by separate head Multi-Tape Multi-head Turing machine can be simulated by standard Turing machine.**5. Multi-dimensional Tape Turing Machine:**It has multi-dimensional tape where head can move any direction that is left, right, up or down.
Multi dimensional tape Turing machine can be simulated by one-dimensional Turing machine.**6. Multi-head Turing Machine:**A multi-head Turing machine contain two or more heads to read the symbols on the same tape.
In one step all the heads sense the scanned symbols and move or write independently.Multi-head Turing machine can be simulated by single head Turing machine.
7. Non-deterministic Turing Machine:
A non-deterministic Turing machine has a single, one way infinite tape.A non-deterministic Turing machine is equivalent to deterministic Turing machine.

For example:
S → AT
T → xy
A → a
Type 2 Grammar:
Type 2 Grammar is known as Context Free Grammar. Context free languages are the languages which can be represented by the context free grammar (CFG). Type 2 should be type 1. The production rule is of the form
A → α
Where A is any single non-terminal and is any combination of terminals and non-terminals.
For example:
A → aBb
A → b
B → a
Type 3 Grammar:
Type 3 Grammar is known as Regular Grammar. Regular languages are those languages which can be described using regular expressions. These languages can be modeled by NFA or DFA.
Type 3 is most restricted form of grammar. The Type 3 grammar should be Type 2 and Type 1. Type 3 should be in the form of
V → T*V / T*
For example:
A → xy

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**.
PCP (Post Correspondence Problem).
The Post Correspondence Problem (PCP), introduced by Emi1 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 = (xi, X2, X3, ……..Xn)
$N = (y_1, y_2, y_3, \ldots, y_n)$

We can say that there is a Post Correspondence Solution, if for some i1,i2,…….ik, where 1<=ij<-n the condition xi1……..xik=yil…….yik satisfies