2 Computability

Automaton

- An automaton is defined as a system, where energy, materials and information are transformed, transmitted and used for performing some functions without direct participation of man.
- It other words, we can say that

  "A computational model or a recognition model can be described by automata theory".
- In computer science field, the word "AUTOMATA" is used for recognizer (or accepter) as well as a transducer (machines with output capability).

Model of Finite Automaton (FA)

- A finite automaton consists of an input tape, a finite non-empty set of states, an input alphabet, a read-only head a transition function which defines the change of configuration an initial state and a finite non-empty set of final states.

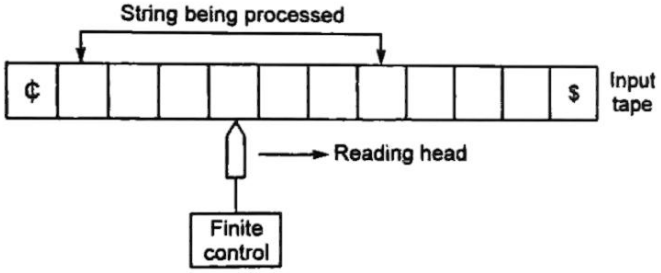


Fig. 1 Block diagram of a finite automaton

- A finite automaton can be represented by a 5 tuples $(Q, \Sigma, \delta, q_0, F)$, where
  - (i) Q is a finite non-empty set of states.
  - (ii) Σ is a finite non-empty set of inputs called input alphabet.
  - (iii) δ is a function which maps $\delta\ (Q \times \Sigma) \to Q$ and is usually called direct transition function.
  - (iv) $q_0 \in Q$, is the initial state.
  - (v) $F \subseteq Q$ is the set of final states. It is assumed have that there may be more than one final state.

Note The transition function which maps $\delta(Q \times \Sigma^*) \to Q$ (*i.e.*, maps a state and a string of input symbols including the empty string into a state) is called indirect transition function.

Syllabus

- Models of computation : finite automata, pushdown automata
- Non-determinism and NFA
- DPDA and PDAs and language accepted by these structures
- Grammars, Languages
- Non-computability and examples of non-computable problems

Transition System

- A transition graph or a transition system is a finite directed labelled graph in which each vertex (or node) represents a state and a directed edges indicate the transition of a state and the edges are labelled with input/output.

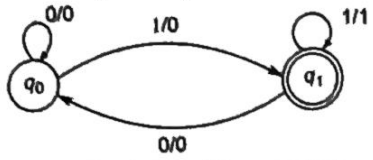


Fig. 2 A transition system

- The initial state is represented by a circle with an arrow pointing towards it, the final state by two concentric circles and the other states are represented by just a circle.
- The edges are labelled by input/output (*e.g.*, by 1/0 or 1/1). For example, if the system is in state q_0 and the input 1 is applied, the system moves to state q_1 as there is a directed edge from q_0 to q_1 with label 1/0. It outputs 0.

Properties of Transition Functions

1. $\delta(q, \wedge) = q$ in a finite automaton, where $\wedge$ means null string.
2. For all string w and input symbol a

$$\delta(q, aw) = \delta(\delta(q, a), w)$$
$$\delta(q, wa) = \delta(\delta(q, w), a)$$

Non-Deterministic Finite Automaton (NDFA)

A non-deterministic finite automaton M can be described by 5-tuples $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a non-empty and finite set of states
2. Σ is an input alphabet
3. δ is transition function which maps $\delta(Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^Q$ *i.e.*, the head reads a symbol in its present state and moves into the set of next state(s). 2^Q is the power set of Q
4. $q_0 \in Q$ known as initial state or starting state
5. $F \subseteq Q$ known as set of final states

It is also known as non-deterministic finite state machine.

The Equivalence of DFA and NDFA

- A DFA can simulate the behaviour of NDFA by increasing the number of states.
- Any NDFA is a more general machine without being more powerful.

Transducer or Finite State Machine (FSM)

A finite state machine is similar to finite automata (FA) except that it has the additional capability of producing output.

FSM =FA + Output capability

A finite state machine is described by 6 tuples.

$(Q, \Sigma, \Delta, \delta, \lambda, S)$, where

1. Q is finite and non-empty set of states
2. Σ is input alphabet
3. Δ is output alphabet
4. δ is a transition function which maps present state and input symbol on to the next state or $(Q \times \Sigma) \rightarrow Q$
5. λ is the output function
6. $S \in Q$ is the initial state or starting state

Types of Finite State Machine (FSM)

There are two types of FSMs.

1. Mealy Machines

- If the output of finite state machine is dependent on present state and present input, then this model of finite state machine is known as Mealy machine.
- A Mealy machine can be described by 6 tuples $(Q, \Sigma, \Delta, \delta\, \lambda, S)$, where
  1. Q is finite and non-empty set of states
  2. Σ is input alphabet
  3. Δ is output alphabet
  4. δ is transition function which maps present state and input symbol on to the next state or $(Q \times \Sigma) \rightarrow Q$
  5. λ is the output function which maps $(Q \times \Sigma) \rightarrow \Delta$, ((present state, present input symbol) $\rightarrow$ output) and
  6. $S \in Q$ is the initial state or starting state.
- If $z(t)$, $q(t)$ and $x(t)$ are output, present state and present input respectively at time t
  Then, $z(t) = \lambda(q(t), x(t))$
- For input $\wedge$ (null string), $z(t) = \wedge$

2. Moore Machines

- If the output of finite state machine is dependent on present state only, then this model of finite state machine is known as Moore machine.
- A Moorre machine can be described by 6 tuples $(Q, \Sigma, \Delta, \delta, \lambda, S)$, where
  1. Q is a finite and non-empty set of states
  2. Σ is the input alphabet

3. Δ is output alphabet
4. δ is transition function which maps present state and input symbol on to the next state or $(Q \times \Sigma) \to Q$
5. λ is the output function which maps $Q \to \Delta$, (present state → output) and
6. $S \in Q$ is the initial state or starting state

- If $z(t)$ and $q(t)$ are output and present state respectively at time t. Then, $z(t) = \lambda\ (q(t))$
- For input $\wedge$ (null string), $z(t) = \lambda$ (starting state)

For example,

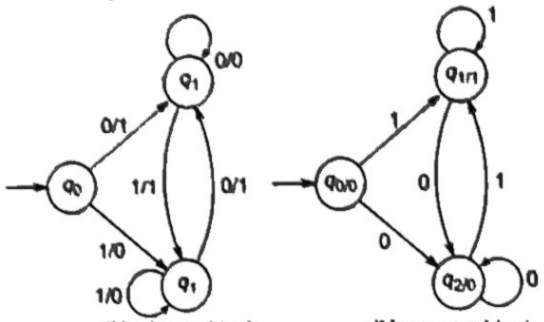


Fig. 3

Remark For a Moore machine, if the input string is of length n, the output string is of length $n + 1$. In case of Mealy machine, if the input string is of length n the output string is also of the same length n.

Grammar

A grammar G can be defined as (V_N, Σ, P, S), where

1. V_N is a finite non-empty set whose elements are called variables.
2. Σ is a finite non-empty set whose elements are called terminals.
3. $V_N \cap \Sigma = \phi$
4. P is a finite set whose elements are $\alpha \to \beta$, where α and β are strings on $V_N \cup \Sigma$. α has atleast one symbol from V_N. Elements of P are called productions or production rules or rewriting rules.

Chomsky Classified of Grammars/Languages

Noam Chomsky has classified the grammars in for categories (type 0, type 1, type 2 and type 3) based on the right hand side forms of the productions.

1. Type 0

These types of grammars are also known as phrase structured grammars and RHS of these are free from any restrictions.

2. Type 1

These types of grammars are also known as Context Sensitive Grammar (CSG). This is a Grammar in which each production rule is of the form $\alpha \to \beta$, where length of $\alpha \leq$ length of β.

e.g., Consider the grammar :

$$A \to aABc$$
$$cB \to Bc$$
$$A \to abc$$
$$bB \to bb$$
$$B \to cba$$

3. Type 2

These types of grammars are also known as context free grammar (CFG). This is a grammar in which every production rule is of the form $\alpha \to \beta$ where, $\alpha \in a$ single V_N and $\beta \in V$

e.g., Consider the grammar :

$$S \to aSaSb$$
$$S \to a$$

where, S is the starting symbol.

4. Type 3

These types of grammars are also known as regular grammar or linear grammar. This is a grammar in which every production rule is either of the form

Right linear grammar

$$A \to aB$$
$$A \to a$$

Left linear Grammar

$$A \to Ba$$
$$A \to a$$

Ambiguous Sentence

- If a particular sentence has more than one leftmost or rightmost derivation, then the sentence is called ambiguous sentence.
- If a grammar generates an ambiguous sentence then it is called ambiguous.

e.g., Consider the grammar

$$E \to E + E \mid E * E \mid (E)\ id$$

Consider the sentence $id + id * id$. It has two parse tree or it has two leftmost derivation.

(i)
$$E \to E + E$$
$$E \to id + E$$
$$E \to id + E * E$$
$$E \to id + id * E$$
$$E \to id + id * id$$

(ii)
$$E \to E * E$$
$$E \to E + E * E$$
$$E \to id + E * E$$
$$E \to id + id * E$$
$$E \to id + id * id$$

Unambiguous Sentence

A sentence in a language is said to be unambiguously sentence, if it has a unique leftmost or rightmost derivation.

Language and Their Related Automaton

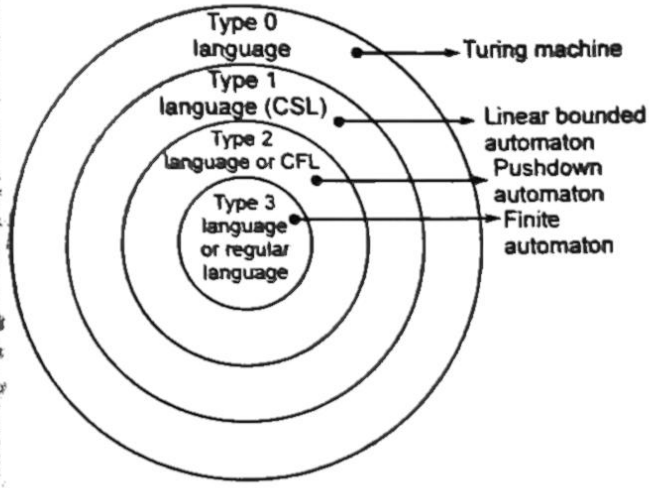


Fig. 4

Type 3 ⊆ Types 2 ⊆ Type 1 ⊆ Type 0.

Here, type 0 is the super set and type 1 is contained in type 0, type 2 is contained in type 1 and type 3 is contained in type 2.

Regular Expression

- The language accepted by finite automaton are regular languages and these languages are easily expressed in regular expressions.
- Regular expressions are means to represent certain sets of strings in some algebraic fashion or regular expressions describe the language accepted by FA.
- Any set represented by a regular expression is called a regular set. For example, if $a, b \in \Sigma$, then
  - (i) a denotes the set $\{a\}$
  - (ii) $a + b$ denotes the set $\{a, b\}$
  - (iii) $a \cdot b$ denotes the set $\{ab\}$
  - (iv) a^* denotes the set $\{\wedge, a, aa, aaa, \ldots\}$
  - (v) $(a + b)^*$ denotes the set $\{a, b\}^*$

Identifies for Regular Expressions

- Two regular expressions P and Q are equivalent if P and Q represent the same set of strings.
- We now give the identities for regular expression;

$(I_1)\ \phi + R = R$ Here, ϕ means empty set
$(I_2)\ \phi R = R\phi = \phi$ $\wedge$ means null string
$(I_3)\ \wedge R = R \wedge = R$
$(I_4)\ \wedge^* = \wedge$ and $\phi^* = \wedge$
$(I_5)\ R + R = R$
$(I_6)\ R^* R^* = R^*$
$(I_7)\ RR^* = R^* R$
$(I_8)\ (R^*)^* = R^*$
$(I_9)\ \wedge RR^* = R^* = \wedge + R^* R$
$(I_{10})\ (PQ)^* P = P(QP)^*$
$(I_{11})\ (P + Q)^* = (P^* Q^*)^* = (P^* + Q^*)^*$
$(I_{12})\ (P + Q)R = PR + QR$
and $R(P + Q) = RP + RQ$

Transition Systems and Regular Expressions

The following explanation describes the relation between transition systems and regular expressions.

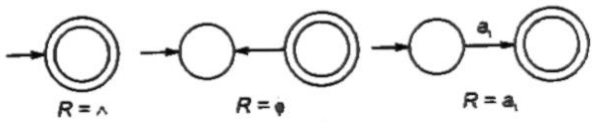


Fig. 5

Some Theorems For Regular Grammars

Theorem 1

If L is a regular language recognized by finite automation M, then there exists a regular grammar G which generates L.

Theorem 2

If L is a regular language generated by some regular grammar G, then there exists a FA which recognizes L.

Regular and Non-Regular Languages

- The pumping lemma is a useful tool to prove that a certain language is not regular language.
- The pumping lemma is always negative in its result. It is used exclusively to show that certain languages are not regular because they can't meet its requirements.
- But Myhill-Nerode theorem tells, that the given language might be regular or not regular and it has constructive aspect. This theorem is used in proving certain language is regular or not.

Context Free Language

A context free grammar can be defined as follows : If every production is of the form $A \to \alpha$, where $A \in V_N$ and $\alpha \in (V_N \cup \Sigma)^*$.

Derivation Trees

The derivation in a CFG can be represented using trees. Such trees representing derivations are called derivation trees.

Yield

The yield of a derivation tree is the concatenation of the lables of the leaves without repetition in the left to right ordering.

Example of CFG

$$G = (\{S\}, \{0, 1\}, P, S)$$

P : consist of S

1. $S \to 0S1$ or just simply $S \to 0S1 \mid S$
2. $S \to \wedge$

Example of derivations

$$S \Rightarrow 0S1$$
$$S \Rightarrow 0S1$$
$$\Rightarrow 01$$
$$S \Rightarrow 0S1$$
$$\Rightarrow 00S11$$
$$\Rightarrow 000S111$$
$$\Rightarrow 000111$$

Normal Forms for Context Free Grammars

- In a context free grammar the RHS of a production can be any string of variables and terminals.
- When the production in G satisfy certain restrictions, then G is said to be in a 'normal form'.

1. Chomsky Normal Form

- In the Chomsky normal form, we have restrictions on the length of RHS and the nature of symbols in the RHS of productions.
- A context free grammar G is in Chomsky normal form if every production is of the form $A \to a$ or $A \to BC$ and $S \to \wedge$ is in G if $\wedge \in L(G)$.
- For a grammar in CNF, the derivation tree has the following property :

  Every node has at most two descendants either two internal vertices or a single leaf.

2. Greibach Normal Form (GNF)

- Greibach Normal Form (GNF) is another norma form.
- A context free grammar is in GNF if ever production is of the form $A \to a\alpha$, where $\alpha \in V_N$ and $a \in \Sigma(\alpha$ may be $\wedge)$ and $S \to \wedge$ is in G $\wedge \in L(G)$.

  For example, G given by $S \to aAB \mid \wedge$, $A \to bC$, $B \to b$, $C \to c$ is in GNF.

Pushdown Automaton (PDA)

- Pushdown Automaton (PDA) is considered as a device that recognizes context tree grammars.
- It has a read-only input tape, input alphabet, finite state control, a set of final states and an initial state as in the case of an FA.
- In addition to these, it has a stack called the pushdown store (PDS). It is a read write pushdown store as we add elements to PDS or remove elements from PDS.

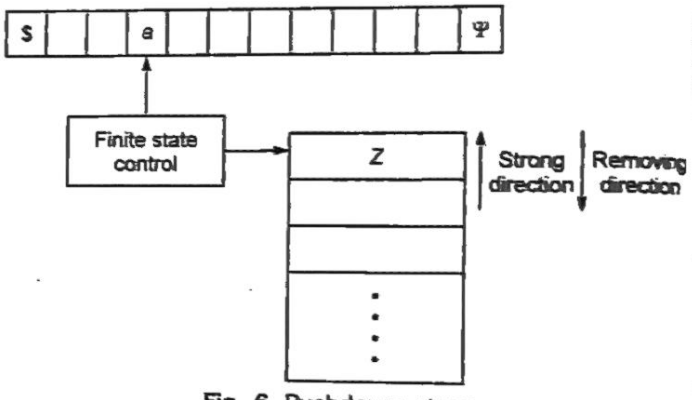


Fig. 6 Pushdown store

- A pushdown automaton consists of
  - (i) a finite non-empty set of states denoted by Q.
  - (ii) a finite non-empty set of input symbols denoted by Σ.
  - (iii) a finite non-empty set of pushdown symbols denoted by Γ.
  - (iv) a special state called the initial state denoted by q_0.
  - (v) a special pushdown symbol called the initial symbol on the pushdown store denoted by z_0.
  - (vi) the set of final states, a subset of Q denoted by F and
  - (vii) the transition function δ from $Q \times (\Sigma \cup (\wedge)) \times \Gamma$ to the set of finite subsets of $Q \times \Gamma^*$.

  Symbolically, a PDA is a 7 tuple *i.e.*, $(Q, \Sigma, \Gamma, \delta, Q_0, Z_0, F)$.

Moves of PDA

- In the first move, an input symbol is read from tape, it means the head is advanced and depending upon the topmost symbol on the stack and present state, PDA has number of choices to proceed further.
- In the second type of move, the input symbol is not read from the tape, it means head is not advances and the topmost symbol of stack is used. The topmost of stack is modified without reading the input symbol. It is also known as $\wedge$-move.

Model of Pushdown Automaton (PDA)

1. Non-deterministic PDA (NPDA)

- Like NFA (non-deterministic finite automata), NPDA has number of choices for its inputs. As we have discussed transition function δ which maps from $Q \times (\Sigma \cup \{\wedge\}) \times \Gamma$ to (finite subset of) $Q \times \Gamma^*$.
- A NPDA accepts an input if a sequence of choices leads to some final state or causes PDA to empty it stack.

2. Deterministic PDA (DPDA)

- Deterministic PDA (DPDA) has the same definition as a PDA but with 2 additional constraints.
- A PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$ is deterministic if it satisfies both the conditions given below.
  (a) $\delta(q, a, b)$ contains at most one element.
  (b) If $\delta(q, \wedge, b)$ is not empty, then $\delta(q, c, b)$ must be empty for every $c \in \Sigma$

LL Parser

- In computer science, an LL parser is a top-down parser for a subset of the context free grammars. It parses the input from left to right and constructs a leftmost derivation of the sentence. This class of grammars which are parsable in this way is known as the LL grammars.
- In other words we can say LL parser as recursive descent parser.
- An LL parser is called an LL (K) parser if it uses K tokens of look ahead when parsing a sentence. If such a parser exists for a certain grammar and it can parse sentences of this grammar without backtracking, then it is called an LL(K) grammar. A language that has an LL(K) grammar is known as an LL(K) language
- To explain its works, we will consider the following small grammar :
  1. $S \rightarrow F$
  2. $S \rightarrow (S + F)$
  3. $F \rightarrow A$

  and parse the following input :
  $(a + a)$

Accession No 88016

The parsing table for this grammar looks as follows :

	(	)	a	+	$
S	2	-	1	-	-
F	-	-	3	-	-

- Note that there is also a column for the special terminal represented here as $, that is used to indicate the end of the input stream.
- In each step, the parser reads the next available symbol from the input stream and the top-most symbol from the stack. If the input and the stack top symbol match the parser discards them both, leaving only the unmatched symbols in the input stream and on the stack.
- Thus, in its first step, the parser reads the input symbol '(' and the stack top symbol 'S'. The parsing table instruction comes from the column headed by the input symbol '(' and the row headed by the stack-top symbol 'S', this cell contains '2' which instructs the parser to apply rule (2). Parser has to rewrite 'S' to '$(S + F)$' on the stack and write the rule number 2 to the output. The stack then becomes [(, S, +, F,), $].
- This is indeed a list of rules for a leftmost derivation of the input string which is
  $S \rightarrow (S + F) \rightarrow (F + F) \rightarrow (a + F) \rightarrow (a + a)$

LR Parser

In computer science, an LR parser is a parser that reads input from left to right and produces a rightmost derivation.

The term LR(K) parser is also used, where the K refers to the number of unconsumed 'look ahead' input symbols that are used in making parsing decisions. Usually K is 1 and the term LR parser is often intended to refer to this case.

- An LR parser is said to perform bottom up parsing because it attempts to deduce the top level grammar production building up from the leaves.

Turing Machine

- A Turing Machine (TM) is a generalization of a PDA, which uses a tape instead of a tape and stack. The potential length of the tape is assumed to be infinite and divided into cells.
- The head of Turing machine is capable to read as well as write on the tape and can move left or right or remain static.

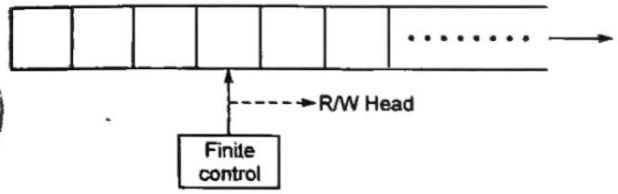


Fig. 7 Model of Turing machine

- A Turing machine can be described by 7 tuples $(Q, \Sigma, \Gamma, \delta, q_0, \square, F)$, where
  1. Q is the finite set of states, not including the halt state (h).
  2. A finite set Σ of symbls (the alphabet).
  3. A finite set Γ of tape symbols, $\Sigma \subseteq \Gamma$.
  4. A transition function
     $$\delta\,(Q \times \Gamma) \to Q \times \Gamma \times \{L, R\}$$
  5. $\square \in \Gamma$ is a special symbol called blank.
  6. $q_0 \in Q$ is the initial state.
  7. $F \in Q$ is the set of final states.
- A Turing machine M decides a language if it accepts it and it never loops.
- Depending upon the number of moves in transition a TM may be deterministic or non-deterministic. If TM has at most one move in a transition, then it is called. Deterministic Turing Machine (DTM), if one or more than one move then, it is called Non-Deterministic Turing Machine (NTM or NDTM).
- A language accepted by Turing machine is known as Recursively Enumerable (RE). Being enumerable means, a TM precisely lists all strings of that language.

Church's Thesis

The principle that TM's and formal version of algorighms and no computational procedure will be considered on algorithms unless it can be implemented as a TM is known as church's thesis.

Universal Turing Machine (UTM)

- A universal Turing machine is a specified Turing machine that can simulate the behaviour of any TM. Alan Turing described such a machine. Its existence means that there is a Turing machine known as UTM, which is capable of running any algorithm.

Computability and Recognizability of Turing Machines

- If a function is effectively calculable, then it is Turing computable conversely. If a function is not Turing computable, then it is not effectively calculable.
- An effectively computable function is one in which there is a finite number of steps of find an answer if there is no errors in the computation, it always has right output in a finite number of steps.

Recursively Enumerable (RE) and Recursive Languages

- When a TM executes an input, there are four possible outcomes of execution. The TM–
  1. Halts and accepts the input
  2. Halts and rejects the input
  3. Never halts (fall into loop)
- We also know that the language accepted by TM is known as Recursively Enumerable (RE) and the set of recursive languages is subset of recursively enumerable set.
- A language is recursive, if there is a Turing machine the accepts every string of the language and rejects every string that is not in the language.

Decidability and Undecidability

- A problem is said to be decidable if
  1. Its language is recursive, or
  2. It has solution of answer (algorithm)
- If for a problem both the answer are possible sometimes 'YES' and sometimes 'NO', then problem is undecidable.

NP-Completeness

- A language L is said to be in class P, if there exists a (deterministic) Turing machine M such that M is of time complexity $p(n)$ for some polynomial p and M accepts L.
- A language L is in class NP if there is a non-deterministic TM M such that M is of time complexity $p(n)$ for some polynomial p and M accepts L.
- Hence, there are so many problems that are listing below, are NP-complete problems.

1. Travelling Salesman Problems

Given, n cities, the distance between them and a number D, does there exist a tour programe for a salesman to visit all the cities so that the total distance travelled is at most D.

2. Satisfiability Problem

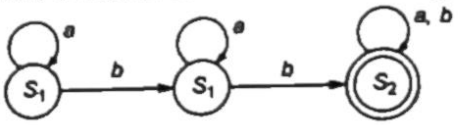
Given, a formula involving propositional variables and logical connectives does there exist a choice of truth values for the variables for which the given formula assumes the truth value T.

3. Vertex Cover Problem

Given, a graph G and a natural number k, does there exist a vertex cover for G with k vertices.

Exercise

1. The language accepted by finite automata is
   (a) context-free (b) regular
   (c) non-regular (d) None of these
2. Which of the following is true?
   (a) All NFA are DFA
   (b) All DFA are NFA
   (c) Both (a) and (b)
   (d) FNA and DFA have different power
3. Two-way finite automata has
   (a) two tapes
   (b) two heads
   (c) bi-directional head movement
   (d) All of the above
4. The major difference between a Moore and a Mealy machine is that
   (a) the output of Moore machine depends upon present state and input
   (b) the output of Moore machine depends upon present state only
   (c) Both (a) and (b)
   (d) None of the above
5. If an input string w has n symbols and can be recognized by a Mealy machine M and equivalent Moore machine M_2, then number of output symbols by M_1 and M_2 are respectively
   (a) n, n (b) $n+1, n$
   (c) $n, n+1$ (d) $n+1, n+1$
6. Regular languages are recognized by
   (a) finite automaton (b) pushdown automaton
   (c) Turing machines (d) All of these
7. Which of the following regular expressions denotes a language comprising all possible strings over the alphabet $\{a, b\}$?
   (a) a^*b^* (b) $(a|b)^*$
   (c) $(ab)^*$ (d) $(a|b^*)$
8. Which of the following is not true?
   [UGC NET, June 2005]
   (a) Power of deterministic automata is equivalent to power of non-deterministic automata
   (b) Power of deterministic pushdown automata is equivalent to power of non-deterministic pushdown automata
   (c) Power of deterministic turing machine is equivalent to power of non-deterministic turing machine
   (d) All of the above
9. Identify the language which is not context-free.
   [UGC NET, June 2005]
   (a) $L = \{WW^R \mid W \in \{0, 1\}^*\}$
   (b) $L = \{a^n b^n \mid n \geq 0\}$
   (c) $L = \{WW \mid W \in \{0, 1\}^*\}$
   (d) $L = \{a^n b^m c^m d^n \mid n, m \geq 0\}$
10. Top down parsers use
    (a) leftmost derivation
    (b) rightmost derivation
    (c) rightmost derivation in reverse order
    (d) None of the above
11. Bottom up parsers use
    (a) leftmost derivation
    (b) rightmost derivation
    (c) right most derivation in reverse order
    (d) None of the above
12. A context-free grammar is
    (a) type 0 (b) type 1
    (c) type 2 (d) type 3
13. Consider a Moore machine M whose digraph is



    Then, $L(M)$ the language accepted by the machine M is the set of all string having **[UGC NET, Dec 2007]**
    (a) two or more b's (b) three or more b's
    (c) two or more a's (d) three or more a's
14. DPDA accepts any language which is
    (a) DCFL (b) CFL
    (c) regular language (d) Both (a) and (c)
15. Which one of the following is the most powerful parsing method?
    (a) LL(1) (b) Canonical LR
    (c) SLR (d) LALR
16. The grammar G having productions $S \rightarrow aSb \mid ab$ is
    (a) LL (1) grammar (b) LL (2) grammar
    (c) LL (0) (d) None of these
17. Which of the following regular expressions corresponds to the language of all strings over the alphabet $\{a, b\}$ that contains exactly two a's?
    (i) aa (ii) ab^*a (iii) b^*aba^*
    (a) (i) and (ii) (b) (ii) and (iii)
    (c) (i) and (iii) (d) None of these
18. The result of pumping lemma is
    (a) negative always (b) sometime negative
    (c) positive sometime (d) positive always
19. Match List I with List II and select the correct answer using the codes given below the lists.

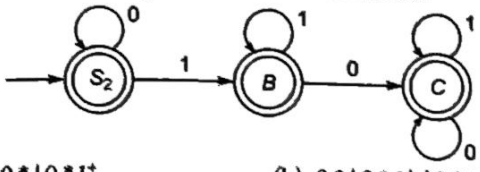
List I	List II
P. Finite automaton	1. Type 0 language
Q. Turing machine	2. Type 1 language
R. Pushdown automaton	3. Type 3 language
S. Linear bounded automaton	4. Type 2 language

Codes

	P	Q	R	S
(a)	3	1	4	2
(b)	4	1	3	2
(c)	3	4	1	2
(d)	3	1	2	4

20. Choose the correct statement.
(a) All regular grammars are CFG
(b) All CFGs are CSG
(c) Regular grammars are most restricted grammars
(d) All of the above

21. The regular expression for the language recognized by the following finite state automata is



(a) $0^*|0^*1^+$ (b) $0^*|0^*1^+|0^*1^+(0+1)^*$
(c) $0^*|11^*$ (d) None of these

22. Which of the following regular expressions corresponds to this grammar?

$S \to AB/AS,\ A \to a/aA,\ B \to b$

[UGC NET, Dec 2006]

(a) aa^*b^+ (b) aa^*b
(c) $(ab)^*$ (d) $a(ab)^*$

23. The language $L=\{a^nb^na^n,\ n \to 1\}$ is recognized by
(a) Turing machine (b) 2PDA
(c) post machine (d) All of these

24. A universal Turing machine is a
(a) reprogrammable Turing machine
(b) two tape TM
(c) single tape TM
(d) None of the above

25. Match List I with List II and select the correct answer using the codes given below the lists.

List I	List II
P. Lexical analyser	1. Pushdown automata
Q. Parsing	2. Turing machine
R. Computing	3. Finite state automata
S. Non-deterministic but finite machine	4. Non-deterministic FA

Codes

	P	Q	R	S
(a)	3	4	2	1
(b)	3	1	2	4
(c)	3	1	4	2
(d)	3	2	1	4

26. A recursive enumerable language is
(a) accepted by TM
(b) not accepted by TM
(c) sometimes accepted and sometimes not accepted
(d) None of the above

27. Which sentence can be generated by

[UGC NET, Dec 2005]

$S \to d|bA$
$A \to d|cA$

(a) *bccddd* (b) *aabccd*
(c) *ababccd* (d) None of these

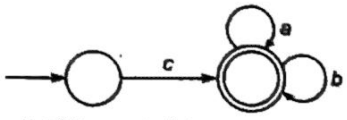
28. Regular expression $a+b$ denotes the set

[UGC NET, Dec 2005]

(a) $\{a\}$ (b) $\{\wedge a, b\}$
(c) $\{a, b\}$ (d) None of these

29. Suppose L is a recursive enumerable language and TM, M accepts L, then for $W \notin L$, the machine M
(a) halts always (b) does not halt
(c) may or may not halt (d) rejects

30. Which of the following strings is accepted by given NDFA?



(a) $c \cdot (a \cup b)^*$ (b) $c \cdot a^* \cdot b^*$
(c) $c \cdot (ab)^*$ (d) None of these

31. The regular expression $(a|b)(a|b)$ denotes the set
(a) $\{a, b, ab, aa\}$ (b) $\{a, b, ba, bb\}$
(c) $\{a, b\}$ (d) $\{aa, ab, ba, bb\}$

32. Which of the following regular expressions denotes a language comprising all possible strings of even length over the alphabet $\{0, 1\}$?
(a) $(0|1)^*$ (b) $(0|1)^*(0|1)^*$
(c) $(00|01|11|10)^*$ (d) $(0|1)(0|1)(0|1)^*$

33. The context-free languages are closed for–

[UGC NET, Dec 2006]

(i) Intersection (ii) Union
(iii) Complementation (iv) Kleene star
(a) (i) and (iv) (b) (i) and (iii)
(c) (ii) and (iv) (d) (ii) and (iii)

34. Which of the following strings is in the language defined by grammar? [UGC NET, June 2006]

$S \to 0A,\ A \to 1A|0A|1$

(a) 01100 (b) 00101
(c) 10011 (d) 11111

35. The logic of pumping lemma is a good example of

[UGC NET, June 2006]

(a) pigeon hole principle
(b) recursion
(c) divide and conquer technique
(d) iteration

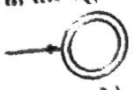
36. Consider the language :

$L_1 = \{a^nb^mc^nd^m \mid n \geq 1, m \geq 1\}$

and $L_2 = \{a^nb^mc^md^n \mid n \geq 1, m \geq 1\}$

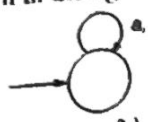
(a) both L_1 and L_2 are context-free
(b) L_1 is not context-free but L_2 is context-free
(c) both are not context-free
(d) L_1 is context-free but L_2 is not context-free

37. The FSM shown in the figure accepts



(a) all strings (b) no strings
(c) ϵ-alone (d) None of these

38. The FSM shown in the figure accepts



(a) all strings (b) no strings
(c) ϵ-alone (d) None of these

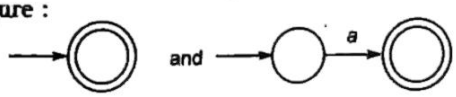
39. The Travelling Salesman Problem (TSP) is
(a) NP but not NP-complete
(b) NP-complete
(c) neither NP nor NP-complete
(d) None of the above

40. The string 1101 does not belong to the set represented by
(a) $110^*(0+1)$ (b) $1(0+1)^*101$
(c) $(00+(11)^*0)^*1$ (d) $(10)^*(01)^*(00+11)^*$

41. Chomsky's hierarchy states that
(a) type 0 $\subset$ type 1 $\subset$ type 2 $\subset$ type 3
(b) type 0 $\subseteq$ type 1 $\subseteq$ type 2 $\subseteq$ type 3
(c) type 0 $\supset$ type 1 $\supset$ type 2 $\supset$ type 3
(d) type 0 $\supseteq$ type 1 $\supseteq$ type 2 $\supseteq$ type 3

42. An FSM with
(a) 1 stack is more powerful than an FSM with no stack
(b) 2 stack is more powerful than a FSM with 1 stack
(c) Both (a) and (b)
(d) None of the above

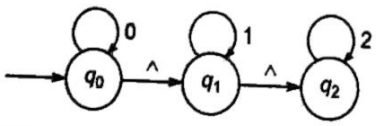
43. Consider the following two FSMs shown in the figure :



Which of the following statements is correct?
(a) Both are equivalent
(b) Second FSM accepts $\wedge$ only
(c) First FSM accepts nothing
(d) None of the above

44. A PDM behaves like a TM when the number of auxiliary memory it has, is
(a) zero (b) 1 or more
(c) 2 or more (d) None of these

45. Give the regular expression described by the following NFA.



(a) $(012)^*$ (b) $(0+1+2)^*$
(c) $0^*1^*2^*$ (d) None of these

46. Which of the following statements is wrong?
(a) Every recursive language is recursively enumerable
(b) A language is accepted by a finite automaton if and only if it is context-free
(c) Recursive languages are closed under intersection
(d) A language is accepted by a finite automaton if and only if it is right-linear

47. Which of the following statements is true?
(a) All languages can be generated by context-free grammar
(b) The number of symbols necessary to stimulate a Turing Machine (TM) with m symbols and n states is mn
(c) Any regular language has an equivalent context-free grammar
(d) The class of context-free grammar is not closed

48. Which sentence can be generated by $S \to aS/bA$ $A \to d/aA$?
(a) *aabccd* (b) *adabcca*
(c) *abcca* (d) *abababd*

49. Which of the following statements is wrong?
(a) A Turing machine cannot solve halting problems
(b) Set of recursively enumerable languages is closed under union
(c) A finite state machine with 2 stacks is more powerful than a finite state machine with 3 stacks
(d) Context sensitive grammar can be recognized by a linearly bounded memory machine

50. Which of the following statements is wrong?
(a) Recursive languages are closed under union
(b) Recursive languages are closed under complementation
(c) If a language and its complement are both regular then the language must be recursive
(d) A language is accepted by a finite automaton if and only if it is recursive

51. Recursively enumerable languages are not closed under
(a) complementation (b) union
(c) intersection (d) None of these

52. Which of the following statements is false?
(a) Regular language is subset of context-free language
(b) Context sensitive language is subset of context-free language
(c) Context-free language is subset of context sensitive language
(d) Context sensitive language is subset of types-0 language

53. Which of the following statements is false?
(a) Pushdown automaton also performs the operation of finite automation
(b) Turing machine can also perform the operation of finite automaton
(c) Linear bounded automaton can also perform the operation of finite automaton
(d) None of the above

54. Which of the following regular expressions denotes a language comprising all possible strings of even length over the alphabet (0, 1)?
(a) 1/0 (1/0)* (b) (0/1) (1/0)*
(c) (1/0) (d) (00/01/11/10)*

55. The regular expression which denotes zero or more instances of an x or y, is
(a) (x/y) (b) $(x/y)^*$
(c) x^*/y (d) $(xy)^*$

56. (ab^*) is regular expression defining a language over $\{a, b\}$. Another regular expression that defines the same language over $\{a, b\}$ is
(a) a^*ba (b) $a^*(ba)^*$
(c) $a(ba)^*$ (d) None of these

57. Which of the following statements is false?
(a) A regular language is also a context-free language
(b) A context-free language is also a regular language
(c) All context-free grammars are ambiguous
(d) Both (b) and (c)

58. For a derivation tree, which of the following statements is false?
(a) The label of each leaf node is x, where x is a terminal
(b) The label of all nodes except leaf nodes (or leaves) is a non-terminal
(c) If the root of a subtree is A, then it is called an A-tree
(d) None of the above

59. A context-free grammar G is said to be ambiguous if
(a) it has two or more leftmost derivations for terminal string $s \in L(G)$
(b) it has two or more rightmost derivations for terminal string $s \in L(G)$
(c) Neither (a) nor (b) is true
(d) Both (a) and (b) are true

60. If L is a deterministic CFL and R is regular language, then
(a) $L \cap R$ is a deterministic CFL
(b) $L \cap R$ is a regular language
(c) (a) is true but (b) is not
(d) (b) is true but (a) is not

61. If L_1 and L_2 are two context-free languages, then which of the following is false?
(a) L_1L_2 is a context-free language
(b) $L_1 \cap L_2$ may be or may not be context-free language
(c) $L_1^* L_2^*$ is not a context-free language
(d) None of the above

62. Which of the following statements is true?
(a) Any regular language has an equivalent CFG
(b) Some non-regular languages cannot be generated by any CFG
(c) Some regular languages cannot be generated by any CFG
(d) Both (a) and (b)

63. Which of the following is most powerful?
(a) DFA (b) NDFA
(c) 2PDA (d) DPDA

64. A pushdown automaton is different than a finite automaton by
(a) a read head
(b) a memory in the form of stack
(c) a set of states
(d) All of the above

65. Which of the following statements is false?
(a) For a finite automaton, the working can be desribed in terms of change of states
(b) For a pushdown automaton, the working can be described in terms of change of instantaneous descriptions
(c) Both (a) and (b)
(d) Neither (a) nor (b)

66. Which of the following statements is false?
(a) $LR(k)$ grammars are used in design of compilers
(b) $LR(k)$ grammars are subclasses of CFGs
(c) If a grammar is an $LR(k)$ grammar, then it is also an $LR(k')$ grammar for all $k' < k$
(d) $LR(k)$ grammars play an important role in the study of programming languages

67. The $LR(0)$ grammars
(a) define exactly the deterministic CFLs having the prefix property
(b) may be those grammars which do not allow start symbol in right side of any production
(c) do not allow leftmost derivation
(d) All of the above

68. If a grammar is in $LL(1)$, then it can be compiled by
(a) a recursive decent parser
(b) any parser
(c) cannot be compiled
(d) None of the above

69. Which of the following is a false statement?
(a) An $LL(k)$ grammar has to be CFG
(b) There are some $LL(k)$ grammars which are not CFG
(c) An $LL(k)$ grammar has to be unambiguous
(d) None of the above

70. A Turing machine
(a) is a simple mathematical mode of general purpose computer
(b) models the computing power of any computer
(c) is capable of performing any calculation which can be performed by any computing machine
(d) All of the above

71. A Turing machine is similar to a finite automaton with only the difference of
(a) read/write head (b) input tape
(c) finite state control (d) All of these

72. Universal Turing machine influences the concept of
(a) computability
(b) stored-program computer
(c) Both (a) and (b)
(d) None of the above

73. A finite state machine having finite tape length without rewinding capability and unidirectional tape movement is called
(a) Turing machine (b) PDA
(c) DFA (d) All of these

74. Which of the following statements is false?
(a) A turing machine is more powerful than finite state machine because it has no finite state
(b) A finite state machine can be assumed to be a turing machine of finite tape length without rewinding capability and undirectional tape movement
(c) Both (a) and (b)
(d) None of the above

75. Which of the following statements is false?
(a) The recursiveness problem of type-0 grammar is unsolvable
(b) There is no algorithm that can determine whether or not a given Turing machine halts with complete blank tape when it starts with a given tape configuration
(c) The problem of determining whether or not a given context-sensitive language is context-free is solvable
(d) None of the above

76. Which of the followig statements is false?
(a) If a langauge is not recursively enumerable, then its complement cannot be recursive
(b) The family of recursive languages is closed und... union
(c) The family of recursive languages is closed und... intersection
(d) None of the above

77. Which of the following statements is false?
(a) Every context-sensitive language is recursive
(b) Every recursive language is context-sensitive
(c) Both (a) and (b)
(d) None of the above

78. A problem is NP-complete if
(a) answer can be verified quickly
(b) a quick algorithm to solve this problem can be us... to solve all other NP problems quickly
(c) it can be solved in polynomial time
(d) All of the above

79. Which of the following problems is n... NP-complete?
(a) Satifiability problem
(b) Vertex cover problem
(c) Integer linear programming problem
(d) None of the above

80. Which of the following is incorrect?
(a) The class of decision problem that can be solved ... a deterministic sequential machine in polynom... time is known as P
(b) The class of decision problem that can be verifi... in polynomial time is known as NP
(c) The class of problems that can be understood as ... class NP-hard consequently is NP-complete ... harder
(d) None of the above

Answers

1. (b)	2. (b)	3. (c)	4. (b)	5. (c)	6. (d)	7. (b)	8. (b)	9. (c)	10. (a)
11. (b)	12. (c)	13. (a)	14. (d)	15. (b)	16. (b)	17. (d)	18. (a)	19. (a)	20. (d)
21. (a)	22. (b)	23. (d)	24. (a)	25. (b)	26. (a)	27. (d)	28. (c)	29. (c)	30. (a)
31. (d)	32. (c)	33. (b)	34. (b)	35. (a)	36. (b)	37. (c)	38. (b)	39. (b)	40. (d)
41. (c)	42. (c)	43. (d)	44. (c)	45. (c)	46. (b)	47. (c)	48. (a)	49. (c)	50. (d)
51. (a)	52. (b)	53. (d)	54. (c)	55. (b)	56. (c)	57. (d)	58. (d)	59. (d)	60. (c)
61. (c)	62. (d)	63. (c)	64. (b)	65. (c)	66. (c)	67. (d)	68. (a)	69. (b)	70. (d)
71. (a)	72. (c)	73. (a)	74. (a)	75. (d)	76. (d)	77. (b)	78. (d)	79. (d)	80. (d)