Theory of Computation

For

Computer Science

&

Information Technology

By

www.thegateacademy.com

✆ 080-40611000

Syllabus for Theory of Computation

Regular Expressions and Finite Automata, Context-Free Grammar's and Push-Down Automata, Regular and Context-Free Languages, Pumping Lemma, Turing Machines and Undecidability.

Previous Year GATE Papers and Analysis

GATE Papers with answer key

thegateacademy.com/gate-papers

Subject wise Weightage Analysis

thegateacademy.com/gate-syllabus

Contents

"Yesterday I dared to struggle. Today I dare to win."
.....Bernadette Devlin

CHAPTER 1

Introduction/Preliminaries

Learning Objectives

After reading this chapter, you will know:

1. Relations

Introduction

String: A string is a finite sequence of symbols put together.
E.g. 'bbac' the length of this string is '4'.
Note: The length of empty string denoted by ∈, is the string consisting of zero symbols.
Thus $|\in| = 0$.

Alphabet: An alphabet is a finite set of symbols.
E.g.: {a, b, 0, 1, B}

Formal language: A formal language is a set of strings made up of symbols taken from some alphabet.
E.g.: The language consisting of all strings formed by the alphabet {0, 1}
Note:

1. The empty set, ϕ, is a formal language. The cardinality (size) of this language is zero.
2. The set consisting of empty string, {ϕ} is a formal language. The cardinality (size) of this language is one.

Set: A set is a collection of objects (members of the set) without repetition.

i. **Finite Set:** A set which contains finite number of elements is said to be finite set.
E.g.: {1, 2, 3, 4}

ii. **Countably Infinite Set:** Sets that can be placed in one-to-one correspondence with the integers are said to be countably infinite or countable or denumerable.
E.g.: The set $\in *$ of the finite-length strings from an alphabet ∈ are countably infinite, (if ∈: {0, 1} Then ∈∗: {0, 01, 1, 10, 011.....} i.e., all possible strings with '0' and'1')

iii. **Uncountable Set:** Sets that can't be placed in one-to-one correspondence with the integers are said to be uncountable sets.
E.g.: The set of real numbers.

Relations

A (binary) relation is a set of ordered tuples. The first component of each tuple is chosen from a set called the domain and the second component of each pair is chosen from a (possibly different) set called the range.

E.g. Let A {1, 2, 3, 4} be a set. A relation R, on 'A' can be defined as R: {(1, 2), (1, 3), (1, 1), (2, 3)}

Properties of Relations

We say a relation R on set S is

1. **Reflexive:** If aRa for all a in S
2. **Irreflexive:** If aRa is false for all a is S
3. **Transitive:** If aRb and bRc implies aRc
4. **Symmetric:** If aRb implies bRa
5. **Asymmetric:** If aRb implies that bRa is false
6. **Anti-symmetric:** If aRb and bRa implies a = b

Equivalence Relation: A relation is said to be equivalence relation if it satisfies the following properties.

1. Reflexive
2. Symmetric
3. Transitive

An important property of equivalence relation 'R' on set 'S' is that R partitions 'S' into disjoint nonempty equivalence classes (may be infinite)

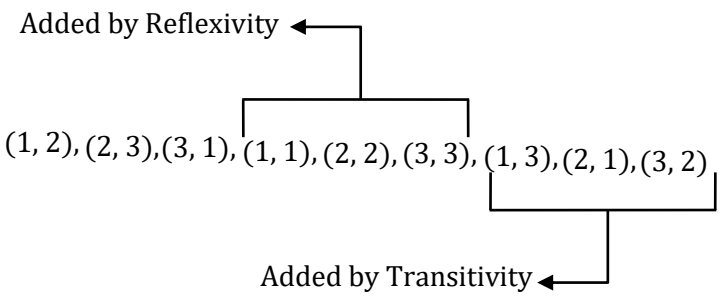
i.e., $S = S_1 \cup S_2$....., where for each i and j, with $i \neq j$

1. $S_i \cap S_j = \emptyset$
2. For each 'a' and 'b' in S_iaRb is true.
3. For each 'a' in Si and 'b' in Sj, aRb is false.
   The S_i's are called equivalence classes.
   E.g.: The relation 'R' on people defined by pRq if and only if 'p' and 'q' were born at the same hour of the same day of some year:
   The number of equivalence classes are: 24 (no. of hours) × 7 (no. of days in a week).

Closure of Relations: Suppose P is a set of properties of relations. The P-closure of a relation R is the smallest relation R that includes all the pairs of R and possesses the properties in P.

E.g.: Let R = {(1, 2), (2, 3), (3, 1)} be a relation on set {1, 2, 3}

i. The reflexive-transitive closure of R is denoted by R* is



ii. The symmetric closure of R is
{(1, 2), (2, 3), (3, 1), (2, 1), (3, 2), (1, 3)}

Term	Definition
prefix of s	A string obtained by removing zero or more trailing symbols of string s; e.g., ban is a prefix of banana.
suffix of s	A string formed by deleting zero or more of the leading symbols of s; e.g., nana is a suffix of banana.
substring of s	A string obtained by deleting a prefix and a suffix from s; e.g., nan is a substring of banana. Every prefix and every suffix of s is a substring of s, but not every substring of s is a prefix or a suffix of s. For every string s, both s and ∈ are prefixes, suffixes, and substrings of s.
proper prefix, suffix, or substring of s	Any nonempty string x that is, respectively, a prefix, suffix, or substring of s such that s ≠ x.
subsequence of s	Any string formed by deleting zero or more not necessarily contiguous symbols from s; e.g., 'baaa' is a subsequence of banana.

"Shoot for the moon. Even if you miss, you will land among the stars."

...Les Brown

CHAPTER 2

Finite Automata

Learning Objectives

After reading this chapter, you will know:

1. Finite Automata
2. Construction of DFA from NFA (Sub Set Construction)
3. Epsilon(∈)-Closures
4. Eliminating ∈-Transitions (Construction of DFA from ∈-NFA)

Finite Automata

A finite automaton involves states and transitions among states in response to inputs. They are useful for building several different kinds of software, including the lexical analysis component of a complier and systems for verifying the correctness of circuits or protocols. They also serve as the control unit in many physical systems including: vending machines, elevators, automatic traffic signals, computer microprocessors, and network protocol stacks.

Deterministic Finite Automata

A DFA captures the basic elements of an abstract machine: it reads in a string, and depending on the input and the way the machine was designed, it outputs either true or false. A DFA is always in one of N states, which we usually name 0 through N-1. Each state is labeled true or false. The DFA begins in a designated state called the start state. As the input characters are read in one at a time, the DFA changes from one state to another in a pre-specified way. The new state is completely determined by the current state and the character just read in. When the input is exhausted, the DFA outputs true or false according to the label of the state it is currently in.

A Deterministic finite automaton is represented by a Quintuple (5-tuple): $(Q, \sum, \delta, q_0, F)$
Where,
Q: Finite set of states
$\sum$: Finite set of input symbols called the alphabet.
δ: $Q \times \sum \Rightarrow Q$ (δ is a transition function from $Q \times \sum$ to Q)
q_0: A start state, one of the states in Q
F: A set of final states, such that $F \subseteq Q$

Induction

Suppose w is a string of the form xa; that is a is the last symbol of w, and x is the string consisting of all but the last symbol. For example $w = 1101$ is broken into $x = 110$ and $a = 1$. Then

$\hat{\delta}(q, w) = \delta(\hat{\delta}(q, x), a)$

Now, to compute $\hat{\delta}(q, w)$ first compute $\hat{\delta}(q, x)$the state that the automaton is in after processing all but the last symbol of w. Suppose this state is p; that is, $\hat{\delta}(q, x) = p$. Then $\hat{\delta}(q, w)$is what we get by making a transition from state p on input a. That is, $\hat{\delta}(q, w) = \hat{\delta}(p, a)$

E.g.: Let us design a DFA to accept the language

$L = \{w \mid w$ has both an even number of 0's and an even number of 1's$\}$

It should not be surprising that the job of the states of this DFA is to count both the number of 0's and the number of 1's, but count them modulo 2. That is, the state is used to remember whether the number of 0's seen so far is even or odd, and also to remember whether the number of 1's seen so far is even or odd. There are thus four states, which can be given the following interpretations:

q_0: Both the number of 0's seen so far and the number of 1's seen so far are even.

q_1: The number of 0's seen so far is even, but the number of 1's seen so far is odd.

q_2: The number of 1's seen so far is even, but the number of 0's seen so far is odd.

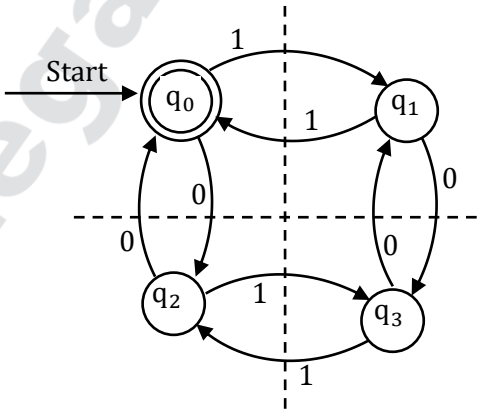
q_3: Both the number of 0's seen so far and the number of 1's seen so far are odd.

State q_0 is both the start state and the alone the accepting state. It is the start state, because before reading any inputs, the numbers of 0's and 1's seen so far are both zero, and zero is even. It is the only accepting state, because it describes exactly the condition for a sequence of 0's and 1's to be in language L.

We now know almost how to specify the DFA for language L. It is,

$A = (\{q_0, q_1, q_2, q_3\}, \{0,1\}, \delta, q_0, \{q_0\})$

Deterministic Finite Automata



Transition Diagram for the DFA

Where the transition function 'S' is described by the transition diagram notice that, how each input '0' causes the state to cross the horizontal, dashed line. Thus, after seeing an even number of 0's we are always above the line, in state q_0 or q_1 while after seeing an odd number of 0's we are always below the line, in state q_2 or q_3. Likewise, every 1 causes the state to cross the vertical, dashed line.

Thus, after seeing an even number of 1's. We are always to the left, in state q_0 or q_2. While after seeing an odd number of 1's we are to the right, in state q_1 or q_3. These observations are an informal proof that the four states have the interpretations attributed to them. However, one could prove the correctness of our claims about the states formally, by a mutual induction with respect to example.

We can also represent this DFA by a transition table shown below. However, we are not just concerned with the design of this DFA; we want to use it to illustrate the construction of $\hat{\delta}$ from its transition function δ. Suppose the input is 110101. Since this string has even numbers of 0's and 1's both, we expect it is in the language. Thus, we expect that $\hat{\delta}\ (q_0, 110101) = q_0$ since q_0 is the only accepting state. Let us now verify that claim.

		0	1
$*\ \rightarrow$	q_0	q_2	q_1
	q_1	q_3	q_0
	q_2	q_0	q_3
	q_3	q_1	q_2

Transition Table for the DFA

The check involves computing $\hat{\delta}\ (q_0, w)$ for each prefix w of 110101, starting at ∈ and going in increasing size. The summary of this calculation is:

$\hat{\delta}(q_0, \in) = q_0$.

$\hat{\delta}(q_0, 1) = \delta(\hat{\delta}(q_0, \in), 1) = \delta(q_{0,}1) = q_1$

$\hat{\delta}(q_0, 11) = \delta(\hat{\delta}(q_0, 1), 1) = \delta(q_{1,}1) = q_0$

$\hat{\delta}(q_0, 110) = \delta(\hat{\delta}(q_0, 1), 0) = \delta(q_{0,}0) = q_2$

$\hat{\delta}(q_0, 1101) = \delta(\hat{\delta}(q_0, 110), 1) = \delta(q_{2,}1) = q_3$

$\hat{\delta}(q_0, 11010) = \delta(\hat{\delta}(q_0, 1101), 0) = \delta(q_{3,}0) = q_1$

$\hat{\delta}(q_0, 110101) = \delta(\hat{\delta}(q_0, 11010), 1) = \delta(q_{1,}1) = q_0$

Acceptance by an Automata

- A string "x" is said to be accepted by a finite automaton $M = (Q, \sum, \delta, q_0, F)$ if $\delta\ (q_0, x) = \rho$ for some ρ in F. The language accepted by M, designated L (M), is the set $\{x \mid \delta\ (q_0, x)$ is in F$\}$.
- A language is a regular set (or just regular) if it is the set accepted by some automaton.
- There are two preferred notations for describing Automata
  1. Transition diagram
  2. Transition table