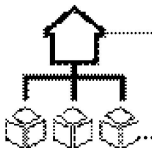


THEORY OF COMPUTATION

Introduction

Alphabet: An alphabet is a finite non-empty set of symbols.

Example: Portion of a calculator: {0, 1, 2, 3,….,9,+,/,(,)}

Note: 1. At least one symbol is necessary.

2. 'Σ' denote Alphabet.

String: A string over an alphabet 'A' is a finite ordered sequence of symbols from 'A'. The length of string is number of symbols in string, with repetitions counted.

Example: If Σ = {0 - 9, /,=,-,+,*,(,)} then Strings valid: 12+34, 12-2 etc.

Strings Invalid: sin (45), log (10) etc. These strings are not valid because sin (), log () are not defined over the alphabet set.

Note: The Empty string denoted by 'Ɛ', is the (unique) string of length zero. Empty string, Ɛ≠ empty set.

Example: Language (L) for string that consist of only 0's or only 1's and have an **odd length** over alphabet {0,1} is—

Ans: {0,1,000,111,11111,00000…}

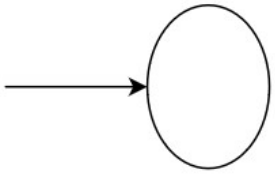
Solution:

Only 0's → should have only 0's. It should not be combination of 0's and 1's.

Only 1's → should have only 1's. It should not be combination of 0's and 1's.

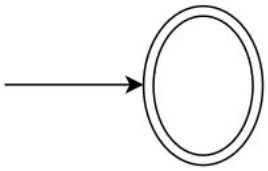
Odd length → only odd number of 0's or odd number of 1's i.e., length of string should be odd.

An Empty Languages An empty language is a language which does not accept any strings includinge. The Finite automata for empty language can be represented as



(i.e., One state, non-accepting and no transitions)

A language which only accepts (Ɛ) The language which only accepts ' Ɛ ' can be represented as



This machine accepts Ɛ - only.

Σ^*: The set of all strings over an alphabet Σ will be denoted by Σ^*

Σ^+: This will denote the set Σ^*- { Ɛ }.

Ex: If = {0, 1) then

Σ^* = { Ɛ,0,1,00,01,10,11,000,001,…..}

Σ^+={0,1,00,01,10,11,000,001,……}

Operations: Operations on strings

1. Concatenation: Combines two strings by putting one after other.

Example: Two strings are defined as x = Cloud, y = IT. The concatenation (x.y) of two strings results in= x.y =Cloud.IT =CloudIT

Note: Concatenation of empty string with any other string gives string itself. i.e x. Ɛ= Ɛ.x=x

2. Kleen Star operation: Let 'w' be a string, w* is set of string obtained by applying any number of concatenations of w with itself, including empty string.

Example: a*={ Ɛ,a,aa,aaa,…… }

Operations: Operations on language

1. Union
2. Intersection
3. Difference
4. Concatenation

Example: Let L_1={00,11}, L_2={01,10}. Then L_1o L_2=? (Concatenation operation)

Ans.: L_1O L_2={0001,0010,1101,1110}

Finite Automata

- ✓ Finite automata are used to recognize patterns.
- ✓ It takes the string of symbol as input and changes its state accordingly. When the desired symbol is found, then the transition occurs.
- ✓ At the time of transition, the automata can either move to the next state or stay in the same state.
- ✓ Finite automata have two states, **Accept state** or **Reject state**. When the input string is processed successfully, and the automata reached its final state, then it will accept.

Formal Definition of FA

A finite automaton is a collection of 5-tuple (Q, ∑, δ, q0, F), where:

1. Q: finite set of states
2. ∑: finite set of the input symbol
3. q_0: initial state
4. F: final state
5. δ: Transition function

Two types of Finit automata: DFA and NFA

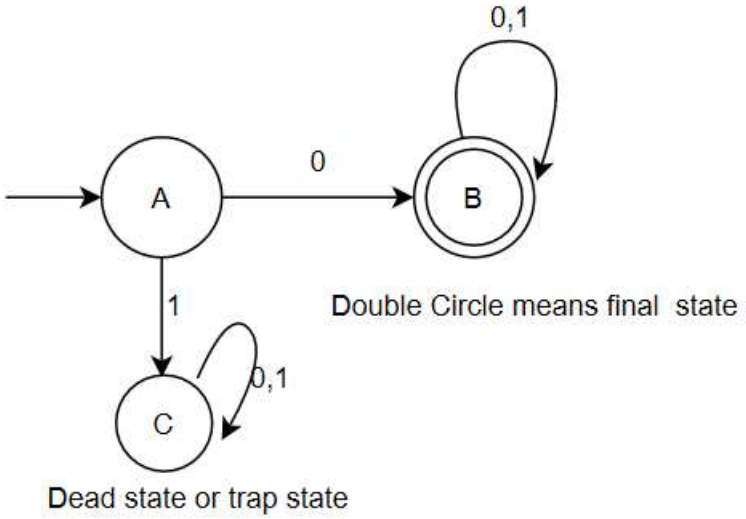
Deterministic Finite Automaton (DFA)

In DFA, for each input symbol, one can determine the state to which the machine will move. Hence, it is called **Deterministic Automaton**. As it has a finite number of states, the machine is called **Deterministic Finite Machine** or **Deterministic Finite Automaton. DFA machine can exists in only one state at any given time. DFA is defined bt 5-tuple: {Q,** ∑, q_0,F, δ **}**

Example 1: **Cosider a DFA that accepts sets of all string over {0,1} which start with 0 .**

Let L_1 = {0,1,00,01,000,010,011,0000,……}

Let string start with= A



Here all input start with 0 goes to the final state B. But input start with 1 goes to the C which is dead of trap state.

Example 2: DFA with ∑ = {0, 1} accepts all strings starting with 1.

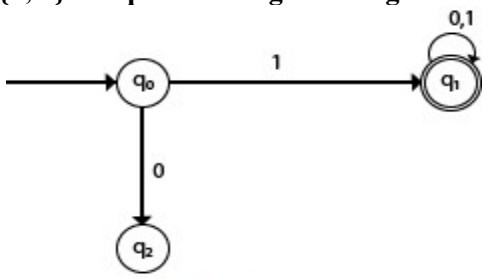


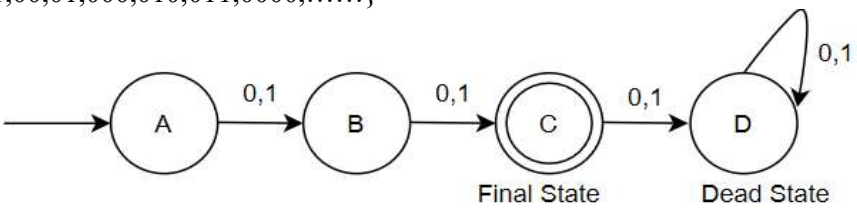
Fig: Transition diagram

The finite automata can be represented using a transition graph. In the above diagram, the machine initially is in start state q0 then on receiving input 1 the machine changes its state to q1. From q0 on receiving 0, the machine changes its state to q2, which is the dead state. From q1 on receiving input 0, 1 the machine changes its state to q1, which is the final state. The possible input strings that can be generated are 10, 11, 110, 101, 111....., that means all string starts with 1.

Example 3: **Cosider a DFA that accepts sets of all string over {0,1} of lenth 2.**

Sol^{n}: lenth 2 menas 00,01,11,10….

Let L_1 = {0,1,00,01,000,010,011,0000,……}



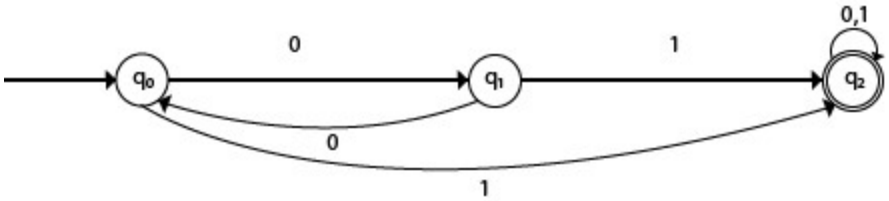
Here if we take 0,1 the for 0 it goes to B then for 1 goes to C. This is final state. If we take input 001 then goes B then C then for 1 goes to D which is dead state.

Example 4: Make Transition Table

The transition table is basically a tabular representation of the transition function. It takes two arguments (a state and a symbol) and returns a state (the "next state").

A transition table is represented by the following things:

- ✓ Columns correspond to input symbols.
- ✓ Rows correspond to states.
- ✓ Entries correspond to the next state.
- ✓ The start state is denoted by an arrow with no source.
- ✓ The accept state is denoted by a star.



Present State	Next State for input 0	Next state for input 1
→q0	q1	q2
q1	q0	q2
*q2	q2	q2

The first column indicates all the current states. From q0 input 0 we can go q1 and for input 1 next state is q2 and so on.

NFA (Non-Deterministic finite automata)

- ✓ NFA stands for non-deterministic finite automata. It is easy to construct an NFA than DFA for a given regular language.
- ✓ The finite automata are called NFA when there exist many paths for specific input from the current state to the next state.
- ✓ Every NFA is not DFA, but each NFA can be translated into DFA.
- ✓ NFA is defined in the same way as DFA but with the following two exceptions, it contains multiple next states, and it contains ε transition.

In the following image, we can see that from state q0 for input a, there are two next states q1 and q2, similarly, from q0 for input b, the next states are q0 and q1. Thus it is not fixed or determined that with a particular input where to go next. Hence this FA is called non-deterministic finite automata.

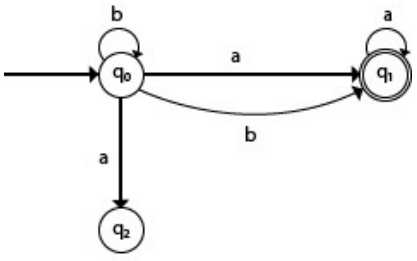


Fig:- NDFA

Example 3: NFA with ∑ = {0, 1} and accept all string of length atleast 2.

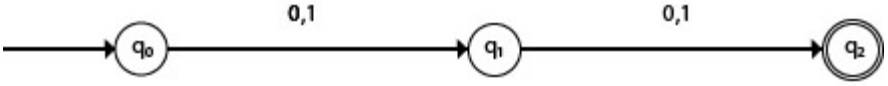


Fig: NFA

Present State	Next state for input 0	Next state for input 1
→q0	q1	q1
q1	q2	q2
*q2	Ɛ	Ɛ

Difference between DFA and NFA: *[Sonali/Janata Bank (Officer (ICT)-2019]*

DFA	NFA
All transitions are deterministic i.e each transition leads to exactly one state.	Transitions could be non deterministic i.e a transition could lead to a subset of state.
For each state, the transition on all possible symbols sould be defined	For each state not all symbols necessarily have to be defined.
Accept input if last state is in 'F'	Acept input if one of the last states is in 'F'
Practical implementation is feasible	Practical implementation has to be deterministic.
Requires more space.	Requires less space.
Backtracking is allowed in DFA	In NDFA, backtracking is not always possible.

Some practices Problems:

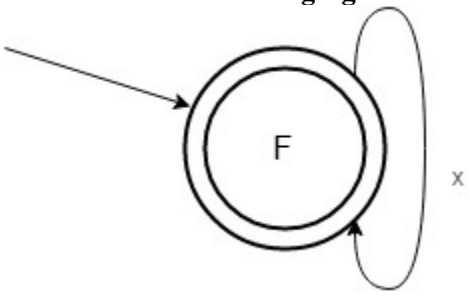
1. A Language for which no DFA exist is a_____

a) Regular Language b) Non-Regular Language
c) May be Regular d) Cannot be said Ans.: **b**

2. A DFA cannot be represented in the following format

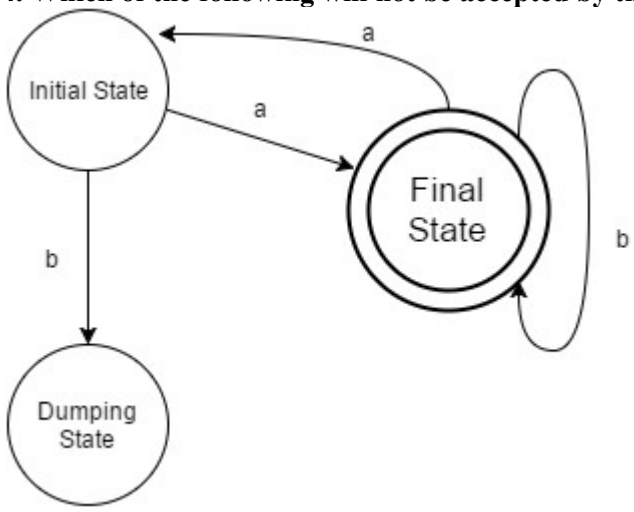
a) Transition graph b) Transition Table
c) C code d) None of the mentioned Ans:**d**

3. What does the following figure most correctly represents?



a) Final state with loop x b) Transitional state with loop x
c) Initial state as well as final state with loop x d) Insufficient Data Ans:**c**

4. Which of the following will not be accepted by the following DFA?



a) ababaabaa b) abbbaa
c) abbbaabb d) abbaabbaa Ans: **a**

5. Which of the following is correct?

Statement 1: ε represents a single string in the set.
Statement 2: Φ represents the language that consist of no string.
a) Statement 1 and 2 both are correct
b) Statement 1 is false but 2 is correct
c) Statement 1 and 2 both are false
d) There is no difference between both the statements, ε and Φ are different notation for same reason Ans: **a**

6. Which of the following represents a language which has no pair of consecutive 1's if ∑= {0,1}?

a) (0+10)*(1+ε) b) (0+10)*(1+ε)*
c) (0+101)*(0+ε) d) (1+010)*(1+ε) Ans: **a**

7. Subset Construction method refers to:

Which two of the following four regular expressions are equivalent?
a) (00)*(e + 0) (ii) (00)* (iii) 0* (iv) 0(00)* (a) (i) and (ii) (b) (ii) and (iii) (c) (i) and (iii) (d) (iii) and (iv) Solution: (i) (00)*(e + 0) represents any number of 0's (ii) (00)* represents even no. of 0's (iii) 0* represents any number of 0's (iv) 0(00)* represents odd number of 0's So, (i) and (iii) are the same expressions.**8. In NFA, this very state is like dead-end non final state:**

a) ACCEPT b) REJECT
c) DISTINCT d) START Ans:**b**

9. Which of the following recognizes the same formal language as of DFA and NFA?

a) Power set Construction b) Subset Construction
c) Robin-Scott Construction d) All of the mentioned Ans: **d**

10. Lexemes can be referred to as:

a) elements of lexicography
b) sequence of alphanumeric characters in a token

c) lexical errors
d) none of the mentioned Ans: **b**

Regular Expression

🕮 Just as finite automata are used to recognize patterns of strings, regular expressions are used to generate patterns of strings. A regular expression is an algebraic formula whose value is a pattern consisting of a set of strings, called the language of the expression.

✍**Operands in a regular expression can be:**

- ✓ ***characters*** from the alphabet over which the regular expression is defined.
- ✓ ***variables*** whose values are any pattern defined by a regular expression.
- ✓ ***epsilon*** which denotes the empty string containing no characters.
- ✓ ***null*** which denotes the empty set of strings.

✍**Operators used in regular expressions include:**

- ✓ Union: If R1 and R2 are regular expressions, then R1 | R2 (also written as R1 U R2 or R1 + R2) is also a regular expression.
  L(R1|R2) = L(R1) U L(R2).
- ✓ Concatenation: If R1 and R2 are regular expressions, then R1R2 (also written as R1.R2) is also a regular expression.
  L(R1R2) = L(R1) concatenated with L(R2).
- ✓ Kleene closure: If R1 is a regular expression, then R1* (the Kleene closure of R1) is also a regular expression.
  L(R1*) = epsilon U L(R1) U L(R1R1) U L(R1R1R1) U ...

N.B: Closure has the highest precedence, followed by concatenation, followed by union.

🕮 **Some Questions:**

1. **Write a regular expression for each of the following sets of binary strings. Use only the basic operations.**
   a) 0 or 11 or 101
   b) only 0s

   Answer: a) 0 | 11 | 101 b) 0*

3. **Write a regular expression for each of the following sets of binary strings. Use only the basic operations.**
   a) all binary strings
   b) all binary strings except empty string
   c) begins with 1, ends with 1
   d) ends with 00
   e) contains at least three 1s

Answers: a) all binary string, so it can contains empty string of combination of binary digit 0and 1. So, Kleene closure is accepted for this case. [L(R1*) = epsilon U L(R1) U L(R1R1) U L(R1R1R1) U ...]. So, answer is: (0|1)*

b) all binary strings except empty string. So, Kleene closure contains empty string(epsilon) but here condition is except empty string. So, answer is =(0|1)(0|1)*.

c) 1 | 1(0|1)*1
d) (0|1)*00
e) (0|1)*1(0|1)*1(0|1)*1(0|1)* or 0*10*10*1(0|1)*.

4. Write a regular expression to describe inputs over the alphabet {a, b, c} that are in sorted order.

Answer: a*b*c*.

5. Write a regular expression for each of the following sets of binary strings. Use only the basic operations.

a) contains at least three consecutive 1s
b) contains the substring 110
c) contains the substring 1101100
d) doesn't contain the substring 110

Answers: (0|1)*111(0|1)*, (0|1)*110(0|1)*, (0|1)*1101100(0|1)*, (0|10)*1*. The last one is by far the trickiest.

🕮 **Some Example:**

➲**Ex1:** The set of strings over {0,1} that end in 3 consecutive 1's.

Answer: Set of strings over {0,1} means the combination of 0 and 1 (rule of Kleene closure)= (0 | 1)*. The remaining part says it will end with 3 consecutive 1's= 111. So the final answer is= (0 | 1)* 111.

➲**Ex2:** The set of strings over {0,1} that have at least one 1.

Answer: At least 1 mean any number which will generate have at least 1. Without containing 1 this will fail. So, 0* 1 (0 | 1)* is the answer. Here 1 is fixed. If 0*=Є and (0 | 1)*= Є , So result is =Є *1*Є =1

➲**Ex3:** The set of strings over {0,1} that have at most one 1.

Answer: 0* | 0* 1 0*

➲**Ex4:** The set of strings over {A..Z,a..z} that contain the word "main".

Let <letter> = A | B | ... | Z | a | b | ... | z

Answer: <letter>* main <letter>*

➲**Ex5:** The set of strings over {A..Z,a..z} that contain 3 x's.

Answer: <letter>* x <letter>* x <letter>* x <letter>*

➲**Ex6:** What is the meaning of this expression: L((0+1)*101(0+1)*)

Answer: all strings of 0's and 1's having 101 as a substring.

➲**Ex7:** What is the meaning of this expression: L((0+1)*1(0+1)*0(0+1)*1(0+1)*)

Answer: all strings of 0's and 1's having 101 as a subsequence.

➲**Ex8:** What is the meaning of this expression: L(1*(1*01*01*01*)*1*)

Answer: all strings of 0's and 1's having a number of 0's that is a multiple of 3.

Practices Problem

1. Which of the following is a regular expression?

a) L = {0m1n | m, n ≥ 0}
b) L = {0n 1n | n ≥ 0}
c) L = {xx | x ∈ {0, 1}*}
d) L = {1n2 0n | n ≥ 0}

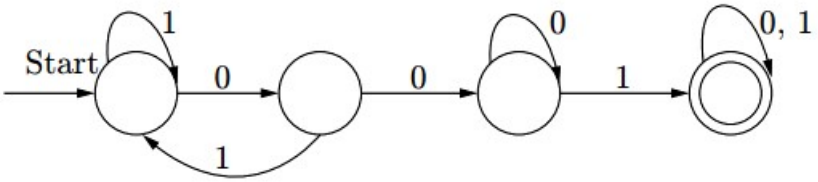
Solution: In option (a), the given expression will generate any number of 0's followed by any number of 1's, which can be accepted by FA. So, it is a regular expression.

In option (b), the given expression will generate number of 0's followed by number of 1's (both should be same in number), which will be accepted by PDA. So, it is not a regular expression.

In option (c), the given expression will generate any string "x" twice, which will be accepted by PDA. So it is not a regular expression.

In option (d), the given expression will generate twice the n number of 1's followed by n number of 0's, which cannot be accepted by FA. So, it is not a regular expression.

2. Consider the following DFS automaton M.



Let S denote the seven bits binary strings in which the first, fourth and last bits are 1. The number of strings in S that are accepted by M is

a) 6 b) 5
c) 4 d) 7

Solution: Strings which will be accepted by machine M can be found by fixing 1 at places first, fourth and last. We will have following four strings: 1001001 1001011 1001111 1111001 **Ans. c**

3. Which two of the following four regular expressions are equivalent?

(i) (00)*(Є + 0) (ii) (00)* (iii) 0* (iv) 0(00)*
a) (i) and (ii) b) (ii) and (iii)
c) (i) and (iii) d) (iii) and (iv)

Solution: (i) (00)*(Є + 0) represents any number of 0's
(ii) (00)* represents even no. of 0's
(iii) 0* represents any number of 0's
(iv) 0(00)* represents odd number of 0's So, (i) and (iii) are the same expressions. **Ans. c**