In the field of theoretical computer science, particularly in the area of formal languages and automata theory, a regular set refers to a set of strings that can be recognized or described by a regular expression, a finite automaton, or an equivalent formalism.

Formally, a regular set over an alphabet Σ is defined as follows:

1. The empty set ∅ and the set containing the empty string {ε} are regular sets.
2. For each symbol a ∈ Σ, the singleton set {a} is a regular set.
3. If A and B are regular sets, then their union A ∪ B, concatenation A · B, and Kleene star A* are also regular sets.

In other words, regular sets can be built from basic building blocks (symbols, the empty set, and the empty string) using operations such as union, concatenation, and the Kleene star. These operations allow the construction of regular expressions, which are symbolic representations of regular sets.

Regular sets have many useful properties, including closure under complementation, intersection, and difference. They form the foundation of regular languages and are fundamental in the study of formal languages and automata theory. Regular sets are used to describe and analyze various aspects of computation, such as pattern matching, lexical analysis, and regular expressions in programming languages.

What is Regular Set in TOC ?

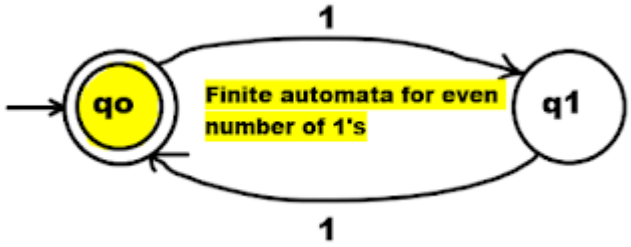
Ans. Regular sets are set which are accepted by FA (finite automata).

For example:

L = {ε, 11, 1111, 111111, 11111111…}

Here L is language set of even number of 1's.

Finite automata for above language L is,



The give set L is regular set because we can represent it using finite automta.

Related Posts:

1. Definition of Deterministic Finite Automata
2. Notations for DFA
3. How do a DFA Process Strings?
4. DFA solved examples
5. Definition Non Deterministic Finite Automata
6. Moore machine
7. Mealy Machine
8. Regular Expression Examples
9. Regular expression
10. Arden's Law
11. NFA with ∈-Moves
12. NFA with ∈ to DFA Indirect Method
13. Define Mealy and Moore Machine

14. What is Trap state ?
15. Equivalent of DFA and NFA
16. Properties of transition functions
17. Mealy to Moore Machine
18. Moore to Mealy machine
19. Diiference between Mealy and Moore machine
20. Pushdown Automata
21. Remove ∈ transitions from NFA
22. TOC 1
23. Diiference between Mealy and Moore machine
24. RGPV TOC What do you understand by DFA how to represent it
25. What is Regular Expression
26. RGPV short note on automata
27. RGPV TOC properties of transition functions
28. RGPV TOC What is Trap state
29. DFA which accept 00 and 11 at the end of a string
30. CFL are not closed under intersection
31. NFA to DFA | RGPV TOC
32. Moore to Mealy | RGPV TOC PYQ
33. DFA accept even 0 and even 1 |RGPV TOC PYQ
34. Short note on automata | RGPV TOC PYQ
35. DFA ending with 00 start with 0 no epsilon | RGPV TOC PYQ
36. DFA ending with 101 | RGPV TOC PYQ
37. Construct DFA for a power n, n>=0 || RGPV TOC
38. Construct FA divisible by 3 | RGPV TOC PYQ
39. Construct DFA equivalent to NFA | RGPV TOC PYQ
40. RGPV Define Mealy and Moore Machine

41. RGPV TOC Short note on equivalent of DFA and NFA
42. RGPV notes Write short note on NDFA
43. Minimization of DFA
44. Construct NFA without ∈
45. CNF from S–>aAD;A->aB/bAB;B->b,D->d.
46. NDFA accepting two consecutive a’s or two consecutive b’s.
47. Regular expresion to CFG
48. Regular expression to Regular grammar
49. Grammar is ambiguous. S → aSbS|bSaS|∈
50. leftmost and rightmost derivations
51. Construct Moore machine for Mealy machine
52. RGPV TOC PYQs
53. Introduction to Automata Theory