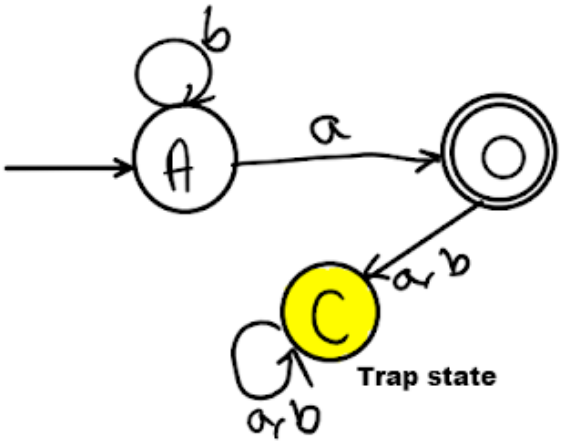
RGPV PYQ 2010

A trap state, which may also be called non-halting or absorbing state, is a state from which a finite automaton (FA) or pushdown automaton (PDA) cannot transit. In the event that an FA or PDA goes to a trap state, it stays there forever and hence making the machine halt. Error conditions or simplifying the automaton design are some of the reasons why trap states are often used.

> *If a transition leads to a state from which it can never escape. such a state is called a trap state. Trap state is also known as Dead state.*

Trap-state example:

- In DFA below, state C is a trap state.
- From C no input is going to another state.



Some properties of trap states include:

1. Trap states are non-halting: Once an FA or PDA enters a trap state, it remains in that state indefinitely.

2. Trap states are absorbing: Once an FA or PDA enters a trap state, it cannot transition to any other state.

3. Trap states can represent error conditions: Trap states can be used to represent error conditions, such as encountering an invalid input symbol or reaching a contradictory state.

4. Trap states can simplify the design of automata: Trap states can be used to simplify the design of automata by eliminating transitions that would lead to error conditions or complex behavior.

References:

- Introduction to the Theory of Computation" by Michael Sipser.

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. Equivalent of DFA and NFA
15. Properties of transition functions
16. Mealy to Moore Machine
17. Moore to Mealy machine
18. Diiference between Mealy and Moore machine
19. Pushdown Automata
20. Remove ∈ transitions from NFA
21. TOC 1
22. Diiference between Mealy and Moore machine
23. RGPV TOC What do you understand by DFA how to represent it
24. What is Regular Expression
25. What is Regular Set in TOC
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
54. Design a NFA that accepts the language over the alphabet, Σ= {0, 1,2} where the decimal equivalent of the language is divisible by 3.