Problem no: 01

Implementing DFA

Program Title:

Implementing and analyzing DFA.

Objectives:

To learn to implement DFA using python.

Theory:

**DFA** stands for **Deterministic Finite Automaton**. 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.**

**Formal Definition of a DFA**

A DFA can be represented by a 5-tuple (Q, ∑, δ, q0, F) where −

* **Q** is a finite set of states.
* **∑** is a finite set of symbols called the alphabet.
* **δ** is the transition function where δ: Q × ∑ → Q
* **q0** is the initial state from where any input is processed (q0 ∈ Q).
* **F** is a set of final state/states of Q (F ⊆ Q).

**Example DFA**

**b b a**

**a** **a** **b**

**3**

**a**

**b**

**DFA 5-tuple**

**Q = {0, 1, 2, 3}**

**∑ = {a, b}**

**q0 = {0}**

**F = {3}**

**Transition Function δ showed by following table:**

|  |  |  |
| --- | --- | --- |
| **Present State** | **Next State for Input a** | **Next State for Input b** |
| →0 | 1 | 0 |
| 1 | 2 | 1 |
| 2 | 1 | \*3 |
| \*3 | \*3 | 0 |

Code:

1. #pip install pyformlang
3. #importing necesssary modules
4. **from** pyformlang.finite\_automaton **import** DeterministicFiniteAutomaton
5. **from** pyformlang.finite\_automaton **import** State
6. **from** pyformlang.finite\_automaton **import** Symbol
8. #declaration of dfa
10. dfa = DeterministicFiniteAutomaton()
12. #Creation as states
13. state0 = State(0)
14. state1 = State(1)
15. state2 = State(2)
16. state3 = State(3)
18. #Creation of symbols
20. sym\_a = Symbol("a")
21. sym\_b = Symbol("b")
23. #Add start state
24. dfa.add\_start\_state(state0)
26. #Add final state
27. dfa.add\_final\_state(state3)
29. #Creation of transition function...
30. dfa.add\_transition(state0, sym\_a, state1)
31. dfa.add\_transition(state0, sym\_b, state0)
32. dfa.add\_transition(state1, sym\_a, state2)
33. dfa.add\_transition(state1, sym\_b, state1)
34. dfa.add\_transition(state2, sym\_a, state1)
35. dfa.add\_transition(state2, sym\_b, state3)
36. dfa.add\_transition(state3, sym\_a, state3)
37. dfa.add\_transition(state3, sym\_b, state0)
39. #check if a word is accepted
40. dfa\_result1 = dfa.accepts([sym\_a, sym\_a, sym\_b])
41. dfa\_result2 = dfa.accepts(['a', sym\_b, sym\_a, sym\_b])
42. dfa\_result3 = dfa.accepts(["a", "b", "a", "a", "b"])
43. dfa\_result4 = dfa.accepts(['b', "a", "a", "a"])
45. **print**('Results: ', dfa\_result1, ' ', dfa\_result2, ' ', dfa\_result3, ' ', dfa\_result4)

Output:

Results: True True False False

Program Title:

Implementing and analyzing DFA with list.

Theory:

**List** is one of the python data types. **List** is a collection which is ordered and changeable. Allows duplicate members. In Python lists are written with square brackets.

thislist=["apple", "banana", "cherry"]  
print(thislist)

['apple', 'banana', 'cherry']

Code:

1. **import** numpy as np
2. **from** pyformlang.finite\_automaton **import** DeterministicFiniteAutomaton
3. **from** pyformlang.finite\_automaton **import** State
4. **from** pyformlang.finite\_automaton **import** Symbol
6. #declaration of dfa
8. dfa = DeterministicFiniteAutomaton()
10. #Creation as states
11. state0 = State(0)
12. state1 = State(1)
13. state2 = State(2)
14. state3 = State(3)
16. #Creation of symbols
18. sym\_a = Symbol("a")
19. sym\_b = Symbol("b")
21. #Add start state
22. dfa.add\_start\_state(state0)
24. #Add final state
25. dfa.add\_final\_state(state3)
27. #Creation of transition function...
28. dfa.add\_transition(state0, sym\_a, state1)
29. dfa.add\_transition(state0, sym\_b, state0)
30. dfa.add\_transition(state1, sym\_a, state2)
31. dfa.add\_transition(state1, sym\_b, state1)
32. dfa.add\_transition(state2, sym\_a, state1)
33. dfa.add\_transition(state2, sym\_b, state3)
34. dfa.add\_transition(state3, sym\_a, state3)
35. dfa.add\_transition(state3, sym\_b, state0)

38. data = np.array(['aab', 'abab', 'abaab', 'baaa', 'ababab', 'aabbbb', 'aaaab'])
40. **for** i **in** data:
41. list = []
42. **for** j **in** i:
43. list.append(j)
45. **if** dfa.accepts(i):
46. **print**(list, ' is accepted')
47. **else**:
48. **print**(list, 'is not accepted')

Output:

['a', 'a', 'b'] is accepted

['a', 'b', 'a', 'b'] is accepted

['a', 'b', 'a', 'a', 'b'] is not accepted

['b', 'a', 'a', 'a'] is not accepted

['a', 'b', 'a', 'b', 'a', 'b'] is not accepted

['a', 'a', 'b', 'b', 'b', 'b'] is not accepted

['a', 'a', 'a', 'a', 'b'] is accepted