RGPV 2010, 02

Q. Write short note on equivalent of DFA and NDFA ?

Solution:

Every DFA is an NDFA.

If from a regular set an NDFA is created than there may be chances of existence of DFA.

DFA is 5 tuple machine:

M = (Q, Σ, δ, q0, F)

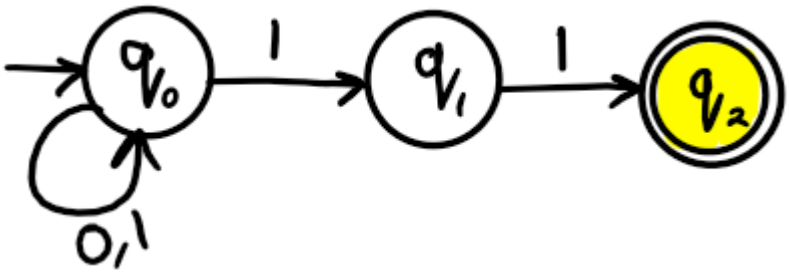
1. Q is a finite non empty set of states.
2. Σ is a finite non empty set of input symbols.
3. δ is a transition function, QXΣ int to Q
4. q0 is an initial state belong to Q.
5. F is the set of final states belong to Q.

NDFA is 5 tuple machine:

1. M = (Q, Σ, δ, q0, F)
2. Q is a finite non empty set of states.
3. Σ is a finite non empty set of input symbols.
4. δ is a transition function, QXΣ int to 2Q
5. q0 is an initial state belong to Q.
6. F is the set of final states belong to Q.

Component	DFA	NFA
Definition	$(Q, \Sigma, \delta, q_0, F)$	$(Q, \Sigma, \delta, q_0, F)$
States (Q)	Finite set of states	Finite set of states
Alphabet (Σ)	Finite alphabet (input symbols)	Finite alphabet (input symbols)
Transition Function (δ)	$Q \times \Sigma \rightarrow Q$	$Q \times \Sigma \rightarrow P(Q)$
Initial State (q_0)	Initial state in Q	Initial state in Q
Accepting States (F)	Set of accepting (final) states	Set of accepting (final) states
Determinism	Transitions are deterministic	Transitions can be non-deterministic
Transition Ambiguity	None	Multiple transitions possible
Uniqueness of Paths	Unique path for each input symbol	Multiple paths for each input symbol
Language Recognition	Determines acceptance or rejection	Determines acceptance or rejection
Equivalence	Equivalent to NFA with one state per DFA state	Equivalent to DFA
Memory Usage	Generally more memory efficient	May use more memory due to multiple transitions
Size (States)	May have more states due to determinism	May have fewer states due to non-determinism

Problem: Convert the following Non-Deterministic Finite Automata (NDFA) to Deterministic Finite Automata (DFA).



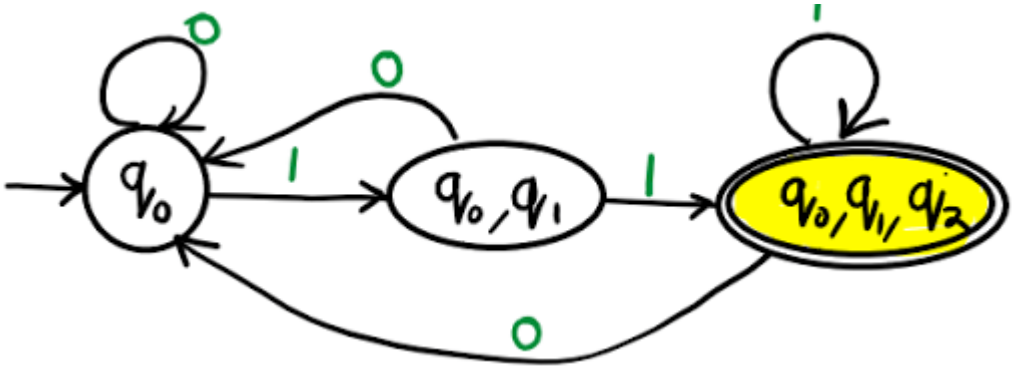
Transition table for NDFA from above NDFA transition diagram

State	Input 0	Input 1
->q0	q0	q0, q1
q1	–	*q2
q2	–	–

Transition table for DFA from above NDFA transition table

State	Input a	Input b
->q0	q0	{q0, q1}
{q0, q1}	q0	*{q0, q1, q2}
*{q0, q1, q2}	q0	*{q0, q1, q2}

Transition diagram from above DFA transition table



Q1: What is the difference between a DFA and an NFA?

Ans: A DFA (deterministic finite automaton) is a type of finite automaton that has a single unique next state for every input symbol in each state, while an NFA (nondeterministic finite automaton) can have multiple possible next states for a given input symbol in each state.

Q2: Can every NFA be converted to an equivalent DFA?

Ans: Yes, every NFA can be converted to an equivalent DFA. This is done using the subset construction algorithm.

Q3: What is the subset construction algorithm?

Ans: The subset construction algorithm is a method for converting an NFA to an equivalent DFA. It works by simulating the behavior of the NFA on all possible input strings, keeping track of the set of states that the NFA could be in after reading each input symbol. The

resulting set of states corresponds to a single state in the DFA, and transitions in the DFA are defined based on the transitions between sets of states in the NFA.

Q4: Are all languages recognizable by NFAs also recognizable by DFAs?

Ans: Yes, all languages that can be recognized by an NFA can also be recognized by a DFA, since every NFA can be converted to an equivalent DFA.

Q5: Are all languages recognizable by DFAs also recognizable by NFAs?

Ans: Yes, all languages that can be recognized by a DFA can also be recognized by an NFA.

Q6: Can an NFA recognize languages that a DFA cannot recognize?

Ans: No, an NFA cannot recognize languages that a DFA cannot recognize. Although an NFA can have more expressive power than a DFA in terms of the complexity of languages it can recognize, any language recognized by an NFA can also be recognized by a DFA.

Q7: Can a DFA recognize languages that an NFA cannot recognize?

Ans: Yes, there are languages that can be recognized by a DFA but not by an NFA. One example of such a language is the language {0^n1^n | n >= 0} which is not recognized by any NFA, but can be recognized by a DFA.

Q8: Can an NFA have fewer states than a DFA that recognizes the same language?

Ans: Yes, it is possible for an NFA to have fewer states than a DFA that recognizes the same language. This is because NFAs can use nondeterminism to explore multiple possible paths through the input, potentially reducing the number of states needed to recognize the language.

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. 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.