Date: 07/08/2024

Experiment 1.4

AIM

To convert a given NFA to DFA

ALGORITHM

1. Start
2. Create utility functions for NFA data structure to read and write transitions.
3. Read NFA input as follows:
    1. The first line contains the number of states (n) , number of final states (f) , number of input alphabets(m), and number of transitions(t).
    2. The next line contains f space separated integers denoting the final states.
    3. The next line contains the m input alphabets as a single string.
    4. The next t lines contain transitions as “qi qj c” representing a transition from qi to qj on input alphabet c. Here, the alphabet ‘e’ denotes epsilon.
4. Remove epsilon transitions from the NFA.
5. Initialize state mapping as linked list to map NFA state set to DFA states.
    1. NFA states are represented as bit mask.
    2. Transitions are stored as a vector of size m containing the transition from this state to the next
6. Define a recursive function to create state mapping from NFA starting with a given state set:
    1. Check if the mapping for the given state set exists in the linked list.
    2. If found, terminate function call.
    3. Otherwise, add node n mapping the state set to new DFA state to the linked list.
    4. For each input symbol i:
        1. Compute set of reachable states for given state set using input symbol i.
        2. Recursively process the new state set.
        3. Add transition from current DFA state to the state corresponding to the given state set.
7. Recursively convert NFA starting with state set {q0} represented as bit mask of 1.

8. Create DFA data structure with number of states equalling size of the state mapping linked list.
    1. Copy input alphabets from the NFA.
    2. For each state in state mapping:
        1. Mark DFA state as final if any NFA state in the set is final.
        2. Populate DFA transition using transitions in the state mapping.
9. Priint the DFA.
10. Stop

RESULT

Successfully converted the given NFA to DFA.