|  |
| --- |
| **Name:Hemant Ranjan**  **Roll No:40** |
| **Exp. No:**2 **Date:** 04/08/2020 |
| **Regular Expressions , Finite Automaton, Conversion of NFA to DFA.** |
| **Aim:** To familiarize with Regular Expressions , Finite Automaton and Conversion of NFA to DFA. |
| **Regular Expression**    Regular expressions are used to represent the language of finite automata. An expression is regular if:   * ɸ is a regular expression for regular language ɸ. * ɛ is a regular expression for regular language {ɛ}. * If a ∈ Σ (Σ represents theinput alphabet), a is a regular expression with language {a}. * If a and b are regular expressions, a + b is also a regular expression with language {a,b}. * If a and b are regular expressions, ab is also a regular expression. * If a is regular expression, a\* (0 or more times a) is also regular.   Eg: Regular expression for strings made of a and b containing at least one a and one b is (a+b)\*a(a+b)\*b + (a+b)\*b(a+b)\*a  **Finite Automata**  Finite Automata(FA) is the simplest machine to recognize patterns.It takes the string of symbols as input and changes its state accordingly. When the desired symbol is found, then the transition occurs.At the time of transition, the automata can either move to the next state or stay in the same state.Finite automata have two states, **Accept state** or **Reject state**. When the input string is processed successfully, and the automata reaches its final state, then it will accept.  A finite automata is a collection of 5 tuples(Q, ∑, δ, q0, F), where:  Q: finite set of states  ∑: finite set of the input symbol  q0: initial state  F: final state  δ: Transition function  There are two types of finite automata:  DFA(Deterministic Finite Automata)  NFA(Non-Deterministic Finite Automata)  **DFA(Deterministic Finite Automata)**  DFA is used in Lexical Analyzer in Compiler. The finite automata are called deterministic finite automata if the machine is read an input string one symbol at a time.In DFA, there is only one path for specific input from the current state to the next state.DFA does not accept the null move.DFA can contain multiple final states.  A DFA is a collection of 5-tuples M=(Q, ∑, δ, q0, F)  Q: finite set of states  ∑: finite set of the input symbol  q0: initial state  F: final state  δ: Transition function -δ: Q x ∑→Q  **NFA(Non-Deterministic Finite Automata)**  It is easier to construct an NFA than a DFA for a given regular language. The finite automata are called NFA when there exist many paths for specific input from the current state to the next state. Every NFA is not a DFA, but each NFA can be translated into a DFA. NFA is defined in the same way as DFA but with the following two exceptions, it contains multiple next states, and it contains ε transitions.  A NFA is a collection of 5-tuples M=(Q, ∑, δ, q0, F)  Q: finite set of states  ∑: finite set of the input symbol  q0: initial state  F: final state  δ: Transition function - δ: Q x ∑ →2Q  **Conversion of NFA to DFA**  In NFA, when a specific input is given to the current state, the machine goes to multiple states. It can have zero, one or more than one move on a given input symbol. On the other hand, in DFA, when a specific input is given to the current state, the machine goes to only one state. DFA has only one move on a given input symbol.  Let, M = (Q, ∑, δ, q0, F) is an NFA which accepts the language L(M). There should be equivalent DFA denoted by M' = (Q', ∑', q0', δ', F') such that L(M) = L(M').  **Steps for converting NFA to DFA**  **Step 1:** Initially Q' = ϕ  **Step 2:** Add q0 of NFA to Q'. Then find the transitions from this start state.  **Step 3:** In Q', find the possible set of states for each input symbol. If this set of states is not in Q', then add it to Q'.  **Step 4:** In DFA, the final state will be all the states which contain F(final states of NFA)  Eg :Conversion of NFA to DFA of string ending with abb. |
| **Result:** Familiarized with Regular Expressions ,Finite Automaton and Conversion of NFA to DFA. |
| **Remarks:**(To be filled by faculty) |