|  |
| --- |
| CONVERSION FROM NFA TO DFA |
|  |



**A BRIEF INTRODUCTION ON WHAT IS AN NFA AND DFA**

* *A FINITE AUTOMATON IS A MACHINE THAT HAS MEMORY IN THE FORM OF STATES AND NO OTHER POWER THAT A NORMAL COMPUTER HAS.*

*A FINITE AUTOMATON IS OF TWO TYPES*

* *NFA(NON DETERMINISTIC FINITE AUTOMATON).*
* *DFA(DETERMINISTIC FINITE AUTOMATON).*
* *IN DFA FOR A PARTICULAR INPUT SYMBOL, THE MACHINE CAN MOVE TO ANY COMBINATION OF STATES THAT THE MACHINE HAS. A DFA CAN ONLY HAVE ONE TRANSITION FOR A PARTICULAR ALPHABET.*
* *IN NFA FOR A PARTICULAR INPUT SYMBOL, THE MACHINE CAN MOVE TO ANY COMBINATION OF STATES PRESENT IN THE AUTOMATON. AN NFA CAN HAVE MANY TRANSITIONS FOR AN ALPHABET, INCLUDING AN EMPTY ALPHABET (LAMBDA).*

**AN ALGORITHM TO CONVERT AN NFA TO DFA**

1. CONSIDER AN NFA REPRESENTING A LANGUAGE ‘L’ AND HAVING SOME STATES

Q1,Q2,Q3,………,QN.ASSUME THAT THE LANGUAGE CONSISTS OF ONLY 1’S AND 0’S

LET Q1 BE THE START STATE OF THE NFA WE MAKE A NOTE OF THE STATES IT

GOES TO FROM THE CURRENT STATE ON SEEING THE SAME INPUT SYMBOL .

1. NOW FORM A SUBSET OF ALL THE STATES THAT THE NFA GOES TO FROM THE CURRENT STATE ON ONE SEEING THE SAME INPUT SYMBOL AND NAME IT AS

SOME STATE RANGING FROM C1..CN AND MAKE A TRANSITION FROM THE

CURRENT STATE TO THE NEW STATE.

1. NOW GO TO THE NEWLY CONSTRUCTED SUBSET C1 , AND CHECK WHETHER

THERE ARE ANY STATES IN IT FROM WHICH THERE ARE MORE THAN ONE

TRANSITION FOR THE SAME INPUT SYMBOL. IF ANY SUCH STATES ARE FOUND

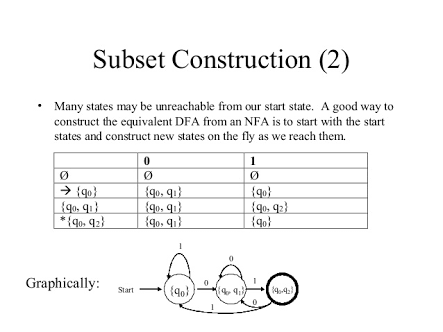
GO TO STEP 2,IF THERE IS A STATE THAT HAS ONLY ONE TRANSITION FROM IT

CREATE A TRANSITION FROM THE CURRENT STATE TO IT.

1. AFTER THIS REPEAT THE WHOLE PROCEDURE UNTILL ALL THE STATES BELONG TO

SUBSET OR NONE OF THE STATES HAVE MULTIPLE TRANSITIONS FROM THEM

FOR THE SAME INPUT SYMBOL.

**AN EXAMPLE OF THE ALGORITHM**

**CODE SUMMARY**

* We used an implementation of an automaton node which has members like transitions to another nodes of the alphabet, a state number, lambda transitions to other nodes, number of lambda transitions and a member to check whether the given state is final or not.
* We also used a set implementation to keep track of the nodes while converting from NFA to DFA.
* We designed the code to be user-friendly and gives a menu with variety of options to choose such as input an NFA, conversion, testing with string, display either in the form of transitions or in the form of a transition table.
* To input an NFA, a file is required and it must follow a pattern as follows:
  + The first line must have the set of alphabets.
  + From the second line, the transitions are included for that particular alphabet in the column.
  + If a transition is not to be mentioned, a hifen(-) is included.
  + In case of multiple transitions, the transitions should be separated by comma(,).
  + After all transitions are included, a blank line is added followed by the list of states to be made as final states.
* For converting an NFA to DFA,
  + We declare a set on which we store a set of possible transitions for a given alphabet.
  + We also declare a large array (power set) on which we store all possible subsets, as well as the transition table for each subset state.
  + First, we include all lambda transitions for the first state into a set.
  + Then, we construct a set of possible transitions for a given alphabet and include lambda transitions of the states stored in the newly created set.
  + We check whether the set is present in the power set, if not found, we include it on the power set.
  + We also correspondingly, map the transitions on the transition table for the power set.
  + The above procedure is repeated for all sets stored in the power set.
  + Once, it is done, a DFA is constructed based on the transition table for power set and mark final states on which a set contains a final state of the original NFA.
* For display, we are given choices whether a transition table should be displayed or a set of transitions should be displayed.
* For testing, we do for both NFA and DFA to verify whether the same string is accepted by both(to check for equivalence).

**DONE BY**

**HARSHITH:01FB14ECS268**

**MAYANK SK:01FB14ECS194**

**SREEVARDAN S:01FB14ECS249**