****

# Assignment

(Compiler Construction)

TOPIC: - ***Convert NFA into DFA***

# Submitted by : M.FAIZAN ARSHAD

# Submitted to : Mr. Abid

# Course : MCS

# Registration No : 141-NUN-0120

Due Date **: 26 Dec, 2015**

Date of Submission **: 26 Dec, 2015**

## *NORTHERN UNIVERSITY NOWSHERA, HAKIMABAD CAMPUS*

**CONVERSION FROM NFA to DFA**

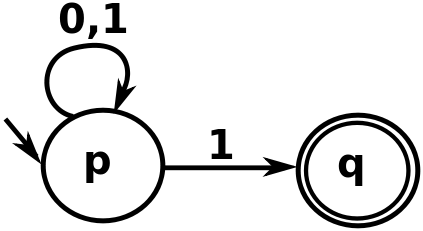
**Definition of NFA:**

In a *nondeterministic* finite automaton (NFA), for each state there can be zero, one, two, or more transitions corresponding to a particular symbol.

If NFA gets to state with more than one possible transition corresponding to the input symbol, we say it *branches*.

If NFA gets to a state where there is no valid transition, then that branch *dies*.

It is the example of the non deterministic automata theory (NFA).



|  |  |  |
| --- | --- | --- |
|  | **0** | **1** |
| *p* | {*p*} | {*p*,*q*} |
| *q* | ∅ | ∅ |

NFA→DFA

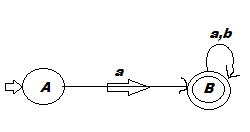
DFA→NFA

NFA ≈DFA

**EXPLAIN:**

**Let ∑= {a,b}**

**L1=** {Start with “a”}

****

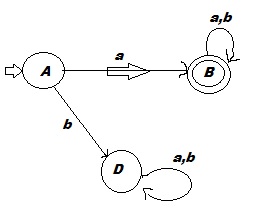
**Transition table for NFA:**

|  |  |  |
| --- | --- | --- |
|  | a | b |
| A | B | **ϕ** |
| B | **B** | **B** |

**Transition graph for DFA:**

|  |  |  |
| --- | --- | --- |
|  | a | b |
| A | B | **D** |
| B | **B** | **B** |
| D | **D** | **D** |

**DIAGRAM of DFA:**

****

**DFA Formal Definition:**

A deterministic finite automaton (DFA) is a 5-tuple

Where,

* Q is a finite set called the states,
* ∑ is a finite set called the alphabet,
* ∂ is the transition function,
* Q1 One is the start state, and
* “F” One or more than one final or set of accept states.

DFA has exactly only transition for each State/symbol pair

**Difference between a DFA and an NFA:**

* DFA has exactly only transition for each State/symbol pair
* NFA has 0, 1 or more transitions for each State/symbol pair

**Application of NFA:**

NFAs and DFAs are equivalent in that if a language is recognized by an NFA, it is also recognized by a DFA and vice versa. The establishment of such equivalence is important and useful. It is useful because constructing an NFA to recognize a given language is sometimes much easier than constructing a DFA for that language. It is important because NFAs can be used to reduce the complexity of the mathematical work required to establish many important properties in the[**theory of computation**](https://en.wikipedia.org/wiki/Theory_of_computation). For example, it is much easier to prove[**closure properties**](https://en.wikipedia.org/wiki/Regular_languages#Closure_properties)of[**regular languages**](https://en.wikipedia.org/wiki/Regular_language)using NFAs than DFAs.