# Lecture # 5.

## NFA :-

- → More powerful than DFA.
- → Used to recognize strings of a lang.
- → Can have 1 start state and 1 or more final states.
- → May have more than one transition for 1 input symbol.
- → May have null transition
  - → no transitions over an input symbol
- * → Possibly have a zero next state on an input symbol or without an input.
  - ↳ A transition without input is called ε-transition.

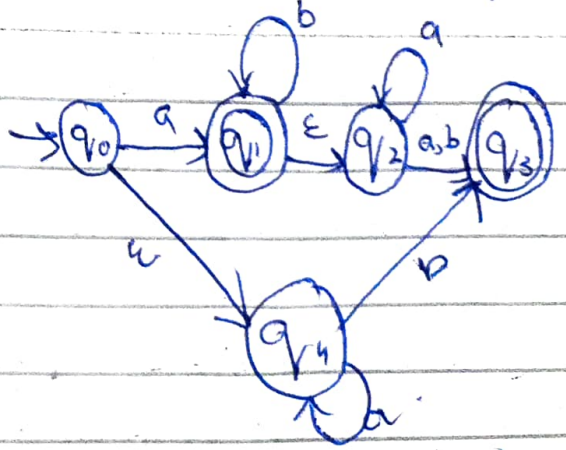
NFA is quintuple. $(Q, \Sigma, \delta, q_0, F)$

$\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \delta(Q)$.

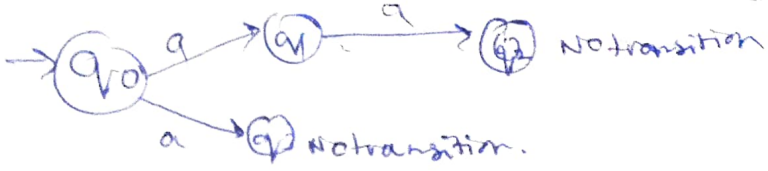
$\delta : Q \times \Sigma$ to $2^Q$.

$Q = \{q_0, q_1, q_2, q_3, q_4\}$ $\Sigma = \{a, b\}$ $F = \{q_1, q_3\}$

| δ | a | b | ε |
|---|---|---|---|
| $q_0$ | $q_1$ | ∅ | $q_4$ |
| $q_1$ | ∅ | $q_1$ | $q_2$ |
| $q_2$ | $q_2, q_3$ | $q_3$ | ∅ |
| $q_3$ | ∅ | ∅ | ∅ |
| $q_4$ | $q_4$ | $q_3$ | ∅ |



NFA




DFA:-

* more restricted than NFA.
* Used to recognize the strings of a lang
* Can have 1 start state and 1 or more final state.
* Must have one transition over each input symbol.
* Have no null ~~tra~~ or empty transition.
* All DFA are NFA's but not all NFA's are DFA's.

Empty string $\epsilon$ and $\Lambda$ are same

$\epsilon$ Empty string = string with 0 occurence of symbols

$\emptyset$ Empty language = a lang with 0 strings or words.

## Algebric Laws for Regular Expressions

* Widely used operator in R.E
  - ↳ Kleene Closure (*), $a^* = \Lambda, a, aa, aaa \ldots$
  - ↳ Concatenation (.), $a.b = ab$
  - ↳ Union (+), $aUb$, $a+b = a, b$.

* Rules for R.E

1. Every letter of $\Sigma$ can be made into a R.E including null string. $L = \{\Lambda\}$.
2. $\emptyset$, empty language itself is a R.E.
3. If R1 and R2 are R.E.
   - ↳ R1 is a R.E
   - ↳ R2 is a R.E
   - ↳ R1+R2 is a R.E
   - ↳ R1.R2 is a R.E
   - ↳ R1* is a R.E.
   - ↳ R1+ is a R.E.

Closure laws:-

- $(r^*)^* = r^*$
- $\emptyset^* = \epsilon$
- $r^+ = r.r^*$
- $r^* = r^* + \epsilon$

$\varepsilon/\Lambda$ is a word while $\emptyset$ is a set of words (a lang)
$\{\varepsilon\}$ is the language containing only the empty word which is different from $\emptyset$ which doesnot contain any word.

## Associative law:

Example $R1 = a$, $R2 = b$, $R3 = c$

If $R1$, $R2$ and $R3$ are R.E.

- $R1 + (R2 + R3) = (R1 + R2) + R3$
- $R1 \cdot (R2 \cdot R3) = (R1 \cdot R2) \cdot R3$
- Associative property doesnot hold for * because its a Unary application.

## Commutative Law / Property:

Example $R1 = a$, $R2 = b$.

If $R1$ and $R2$ are RE then -

- $R1 + R2 = R2 + R1$.
- $R1 \cdot R2 \neq R2 \cdot R1$.

## Distributive law / Property:

If $R1$, $R2$ and $R3$ are RE then.

- $(R1 + R2) \cdot R3 = R1 \cdot R3 + R2 \cdot R3$
- $R1 \cdot (R2 + R3) = R1 \cdot R2 + R1 \cdot R3$
- $(R1 \cdot R2) + R3 \neq (R1 + R3) \cdot (R2 + R3)$

## Idempotent law:

are the set of operations, which can be repeated again & again without changing results.

- $R1 + R1 = R1 \Rightarrow R1 \cup R1 = R1$.

Therefore union operator satisfies idempotent law

- $R1 \cdot R1 \neq R1$

concatenation doesnot satisfy idempotent property.

Ref = Geek for Geeks.

## Identities for R.E.

1. $\emptyset + r = r$
2. $\emptyset \cdot r = r \cdot \emptyset = \emptyset$
3. $\epsilon \cdot r = r \cdot \epsilon = r$
4. $\epsilon^* = \epsilon$ and $\emptyset^* = \epsilon$.
5. $r + r = r$
6. $r^* \cdot r^* = r^*$.
7. $r \cdot r^* = r^* \cdot r = r^+$.
8. $(r^*)^* = r^*$.
9. $\epsilon + r \cdot r^* = r^* = \epsilon + r \cdot r^*$
10. $(p \cdot q)^* \cdot p = p \cdot (q \cdot p)^*$.
11. $(p+q)^* = (p^* \cdot q^*)^* = (p^* + q^*)^*$.
12. $(p+q) \cdot r = p \cdot r + q \cdot r$

## Equivalent R.E

$(0^* 1^*)^* = (0+1)^*$.

$(0^* 1^*)^* = \{\wedge, 0, 00, 1, 11, 01, 10, \ldots\}$

$(0+1)^* = \{\wedge, 0, 00, 1, 11, 01, 10 \ldots\}$.

L.H.S = R.H.S.

## Application of R.E.

1) **Egrep tool**
   - → unix tool
   - → Searches a text file for lines that contain a substring matching a specified pattern.
   - → Echoes all such lines to standard output.

2) **Application in Search Engine**
   - → Archie, 1st S.E used R.E to search through database of files on server.
   - → Used earlies in S.E because of their power and easy implementation.

3) Find and Replace tools

4) Lexical Analysis
- → Scanner and tokenizer.
- → Tokenization is to categorize lexemes to sort them by meaning e.g variable, constant or keyword etc.
- → Set of R.E are used to match the valid set of lexemes that belong to this token type.