# **Regular expression**

* Regular expression is a sequence of pattern that defines a string. It is used to denote regular languages.
* It is also used to match character combinations in strings. String searching algorithm used this pattern to find the operations on string.
* In regular expression, x\* means zero or more occurrence of x. It can generate {e, x, xx, xxx, xxxx,.....}
* In regular expression, x+ means one or more occurrence of x. It can generate {x, xx, xxx, xxxx,.....}

## **Operations on Regular Language**

The various operations on regular language are:

**Union:** If L and M are two regular languages then their union L U M is also a union.

1. L U M = {s | s is in L or s is in M}

**Intersection:** If L and M are two regular languages then their intersection is also an intersection.

1. L ⋂ M = {st | s is in L and t is in M}

**Kleene closure:** If L is a regular language then its kleene closure L1\* will also be a regular language.

1. L\* = Zero or more occurrence of language L.

### **Example**

Write the regular expression for the language:

L = {abn w:n ≥ 3, w ∈ (a,b)+}

### **Solution:**

The string of language L starts with "a" followed by atleast three b's. Itcontains atleast one "a" or one "b" that is string are like abbba, abbbbbba, abbbbbbbb, abbbb.....a

So regular expression is:

r= ab3b\* (a+b)+

Here + is a positive closure i.e. (a+b)+ = (a+b)\* - ∈

# **Examples of Regular Expression**

### **Example 1:**

Write the regular expression for the language accepting all the string which are starting with 1 and ending with 0, over ∑ = {0, 1}.

**Solution:**

In a regular expression, the first symbol should be 1, and the last symbol should be 0. The r.e. is as follows:

1. R = 1 (0+1)\* 0

### **Example 2:**

Write the regular expression for the language starting and ending with a and having any having any combination of b's in between.

**Solution:**

The regular expression will be:

1. R = a b\* b

### **Example 3:**

Write the regular expression for the language starting with a but not having consecutive b's.

**Solution:** The regular expression has to be built for the language:

1. L = {a, aba, aab, aba, aaa, abab, .....}

The regular expression for the above language is:

1. R = {a + ab}\*

### **Example 4:**

Write the regular expression for the language accepting all the string in which any number of a's is followed by any number of b's is followed by any number of c's.

**Solution:** As we know, any number of a's means a\* any number of b's means b\*, any number of c's means c\*. Since as given in problem statement, b's appear after a's and c's appear after b's. So the regular expression could be:

1. R = a\* b\* c\*

### **Example 5:**

Write the regular expression for the language over ∑ = {0} having even length of the string.

**Solution:**

The regular expression has to be built for the language:

1. L = {ε, 00, 0000, 000000, ......}

The regular expression for the above language is:

1. R = (00)\*

### **Example 6:**

Write the regular expression for the language having a string which should have atleast one 0 and alteast one 1.

**Solution:**

The regular expression will be:

1. R = [(0 + 1)\* 0 (0 + 1)\* 1 (0 + 1)\*] + [(0 + 1)\* 1 (0 + 1)\* 0 (0 + 1)\*]

### **Example 7:**

Describe the language denoted by following regular expression

1. r.e. = (b\* (aaa)\* b\*)\*

**Solution:**

The language can be predicted from the regular expression by finding the meaning of it. We will first split the regular expression as:

r.e. = (any combination of b's) (aaa)\* (any combination of b's)

L = {The language consists of the string in which a's appear triples, there is no restriction on the number of b's}

### **Example 8:**

Write the regular expression for the language L over ∑ = {0, 1} such that all the string do not contain the substring 01.

**Solution:**

The Language is as follows:

1. L = {ε, 0, 1, 00, 11, 10, 100, .....}

The regular expression for the above language is as follows:

1. R = (1\* 0\*)

### **Example 9:**

Write the regular expression for the language containing the string over {0, 1} in which there are at least two occurrences of 1's between any two occurrences of 1's between any two occurrences of 0's.

**Solution:** At least two 1's between two occurrences of 0's can be denoted by (0111\*0)\*.

Similarly, if there is no occurrence of 0's, then any number of 1's are also allowed. Hence the r.e. for required language is:

1. R = (1 + (0111\*0))\*

### **Example 10:**

Write the regular expression for the language containing the string in which every 0 is immediately followed by 11.

**Solution:**

The regular expectation will be:

1. R = (011 + 1)\*



# **Conversion of RE to FA**

To convert the RE to FA, we are going to use a method called the subset method. This method is used to obtain FA from the given regular expression. This method is given below:

**Step 1:** Design a transition diagram for given regular expression, using NFA with ε moves.

**Step 2:** Convert this NFA with ε to NFA without ε.

**Step 3:** Convert the obtained NFA to equivalent DFA.

### **Example 1:**

Design a FA from given regular expression 10 + (0 + 11)0\* 1.

**Solution:** First we will construct the transition diagram for a given regular expression.

**Step 1:**



**Step 2:**



**Step 3:**



**Step 4:**



**Step 5:**



Now we have got NFA without ε. Now we will convert it into required DFA for that, we will first write a transition table for this NFA.

|  |  |  |
| --- | --- | --- |
| **State** | **0** | **1** |
| →q0 | q3 | {q1, q2} |
| q1 | qf | ϕ |
| q2 | ϕ | q3 |
| q3 | q3 | qf |
| \*qf | ϕ | ϕ |

The equivalent DFA will be:

|  |  |  |
| --- | --- | --- |
| **State** | **0** | **1** |
| →[q0] | [q3] | [q1, q2] |
| [q1] | [qf] | ϕ |
| [q2] | ϕ | [q3] |
| [q3] | [q3] | [qf] |
| [q1, q2] | [qf] | [qf] |
| \*[qf] | ϕ | ϕ |

### **Example 2:**

Design a NFA from given regular expression 1 (1\* 01\* 01\*)\*.

**Solution:** The NFA for the given regular expression is as follows:

**Step 1:**



**Step 2:**



**Step 3:**

We will first construct FA for R = 0\*1 + 10 as follows:

**Step 1:**



**Step 2:**



**Step 3:**



**Step 4:**

