**REGULAR EXPRESSION**

Just as finite automata are used to *recognize* patterns of strings, regular expressions are used to *generate* patterns of strings. A regular expression is an algebraic formula whose value is a pattern consisting of a set of strings, called the language of the expression.

Operands in a regular expression can be:

* *characters* from the alphabet over which the regular expression is defined.
* *variables* whose values are any pattern defined by a regular expression.
* *epsilon* which denotes the empty string containing no characters.
* *null* which denotes the empty set of strings.

Operators used in regular expressions include:

* Union: If R1 and R2 are regular expressions, then R1 | R2 (also written as R1 U R2 or R1 + R2) is also a regular expression.

L(R1|R2) = L(R1) U L(R2).

* Concatenation: If R1 and R2 are regular expressions, then R1R2 (also written as R1.R2) is also a regular expression.

L(R1R2) = L(R1) concatenated with L(R2).

* Kleene closure: If R1 is a regular expression, then R1\* (the Kleene closure of R1) is also a regular expression.

L(R1\*) = epsilon U L(R1) U L(R1R1) U L(R1R1R1) U ...

Closure has the highest precedence, followed by concatenation, followed by union.

**Examples**

The set of strings over {0,1} that end in 3 consecutive 1's.

(0 | 1)\* 111

The set of strings over {0,1} that have at least one 1.

0\* 1 (0 | 1)\*

The set of strings over {0,1} that have at most one 1.

0\* | 0\* 1 0\*

The set of strings over {A..Z,a..z} that contain the word "main".

Let <letter> = A | B | ... | Z | a | b | ... | z

<letter>\* main <letter>\*

The set of strings over {A..Z,a..z} that contain 3 x's.

<letter>\* x <letter>\* x <letter>\* x <letter>\*

The set of identifiers in Pascal.

Let <letter> = A | B | ... | Z | a | b | ... | z

Let <digit> = 0 | 1 | 2 | 3 ... | 9

<letter> (<letter> | <digit>)\*

The set of real numbers in Pascal.

Let <digit> = 0 | 1 | 2 | 3 ... | 9

Let <exp> = 'E' <sign> <digit> <digit>\* | epsilon

Let <sign> = '+' | '-' | epsilon

Let <decimal> = '.' <digit> <digit>\* | epsilon

<digit> <digit>\* <decimal> <exp>

### Example 1:

Write the regular expression for the language accepting all combinations of a's, over the set ∑ = {a}

**Solution:**

All combinations of a's means a may be zero, single, double and so on. If a is appearing zero times, that means a null string. That is we expect the set of {ε, a, aa, aaa, ....}. So we give a regular expression for this as:

1. R = a\*

That is Kleen closure of a.

### Example 2:

Write the regular expression for the language accepting all combinations of a's except the null string, over the set ∑ = {a}

**Solution:**

The regular expression has to be built for the language

1. L = {a, aa, aaa, ....}

This set indicates that there is no null string. So we can denote regular expression as:

R = a+

### Example 3:

Write the regular expression for the language accepting all the string containing any number of a's and b's.

**Solution:**

The regular expression will be:

1. r.e. = (a + b)\*

This will give the set as L = {ε, a, aa, b, bb, ab, ba, aba, bab, .....}, any combination of a and b.

The (a + b)\* shows any combination with a and b even a null string.

# **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)\*