LANGUAGES & THIER REPRESENTS IN DIFFERENT WAYS

Formal Language:

There are three types of language that are;

- Single letter language
- Finite letter language
- Infinite letter language

Single Letter language:

Such language which have only one alphabet.

For Example:

- If ∑ = {a}
- If ∑ = {b}
- If ∑ = {0}
- If ∑ = {1}
- If ∑ = {λ}

Finite letter language:

Such language from which generated strings can be countable.

For Example:

- If ∑ = {a , b}
- All strings of length two.
- {aa , bb , ab , ba}

(Except these strings we can't make any other strings, these strings are countable)

Infinite letter language:

Such language from which generated strings can be uncountable.

For Example

- If ∑ = {a,b}
- All strings that containing A's & B's.
- {a, b, aa, bb, aaa, bbb, aabb,……}

There are two properties of infinite language that are;

1. Kleene closure (star operator):

It takes / choose **zero** or more occurrence of an alphabet or string.

It is represented by *****

It includes null (λ).

For Example:

- If ∑ = {a} Write all string including λ.
- { λ , a, aa, aaa, aaaa, ……...}
- **a* (Regular Expression)** ***we will discuss it later***

2. Kleene Plus:

It takes / choose one or more occurrence of an alphabet or string.

It is represented by +.

It doesn't include null (λ).

For Example:

- If ∑ = {a} Write all string without λ.
- {a, aa, aaa, aaaa, ……..}
- **a+ (Regular Expression)** ***we will discuss it later***

How to describe a language?

Following are the ways to describe a language:

- Descriptive definition
- Recursive definition
- Regular expression
- Finite Automata

DESCRIPTIVE DEFINITION:

It describing the conditions imposed on its words.

Example 1:

- If ∑ = {a , b} All String end with "**a**".
- {a, aa, ba, aaa, aba, bba, baa…} **(Correct way of descriptive method)**
- {a, aaa, aa, ba, bba, aba, baa…} **(Wrong way of descriptive method)**

- **Notes:** We can write the description in above both ways but the correct one is the first one, its means that order of strings should be ascending.

Example 2:

- If ∑ = {0 , 1} Write all String which contains "**00**".
- {00, 000, 001, 100, 0100…}
- ***Note:*** ***we can also represent descriptive way in the below method.***

Example 3:

- L={a, aa, aaa, aaaa,…….....}
- Language "L" is a set of all strings containing only A's

Example 4:

- L={aabb, aaabbb, aaaabbbbb,…...…}
- Language "L" is a set of all strings in which all No of A's and B's are equal.

RECURSIVE DEFINITION

Language is defined using recursive definition by the following three steps.

- Step 1: some basic words are specified in the language
- Step 2: rules for constructing more words are defined in the language
- Step 3: No Strings except those constructed in above, are allowed to be in the language.

EXAMPLES

1. Defining language of INTEGER

- Step 1: 1 is in **INTEGER**.
- Step 2: If x is in **INTEGER** then x+1 and x-1 are also in **INTEGER**.
- Step 3: No strings except those constructed in above, are allowed to be in **INTEGER**.

2. Defining language of EVEN

- Step 1: 2 is in **EVEN**.
- Step 2: If x is in **EVEN** then x+2 is also in **EVEN**.
- Step 3: No strings except those constructed in above, are allowed to be in **EVEN**.

3. Defining the language $\{a^nb^n\}$, n=1,2,3,… , of strings defined over Σ={a,b}

- Step 1: ab is in $\{a^nb^n\}$
- Step 2: if x is in $\{a^nb^n\}$, then axb is in $\{a^nb^n\}$
- Step 3: No strings except those constructed in above, are allowed to be in $\{a^nb^n\}$

4. Defining the language L, of strings ending in a , defined over Σ={a,b}

- Step 1: a is in L
- Step 2: if x is in L then s(x) is also in L, where s belongs to Σ*
- Step 3: No strings except those constructed in above, are allowed to be in L

5. Defining the language L, of strings beginning and ending in same letters , defined over Σ={a, b}

- Step 1: a and b are in L
- Step 2: (a)s(a) and (b)s(b) are also in L, where s belongs to Σ*
- Step 3: No strings except those constructed in above, are allowed to be in L

6. Defining the language L, of strings containing aa or bb , defined over Σ={a, b}

- Step 1: aa and bb are in L
- Step 2: s(aa)s and s(bb)s are also in L, where s belongs to Σ*
- Step 3: No strings except those constructed in above, are allowed to be in L

REGULAR EXPRESSION (REGEX OR REGEXP)

It is another way of representing language as descriptive method.

With the help of regular expression we can express strings in short form.

Recursive definition of Regular Expression:

A **Regular Expression** can be recursively defined as follows -

- **ε** is a Regular Expression indicates the language containing an empty string. **(L (ε) = {ε})**
- **x** is a Regular Expression where **L = {x}**
- If **X** is a Regular Expression denoting the language **L(X)** and **Y** is a Regular Expression denoting the language **L(Y)**, then

- **X + Y** is a Regular Expression corresponding to the language **L(X) ∪ L(Y)**
- Where **L(X+Y) = L(X) ∪ L(Y)**.
- **X.Y** is a Regular Expression corresponding to the language **L(X) . L(Y)**
- Where **L (X.Y) = L(X) . L(Y)**
- **R*** is a Regular Expression corresponding to the language **L(R*)**

- where **L(R*) = (L(R))***

Why we use regular expression?

We use regular expression because with descriptive method all words (strings) of the language can't be written (because they are infinite), so avoid this problem we use regular expression.

For Example:

- If ∑ = {a , b}
- All strings end with "a".
- Descriptive method = {a, aa, ba, aba,……..}
- **Regular Expression = (a+b)*a**

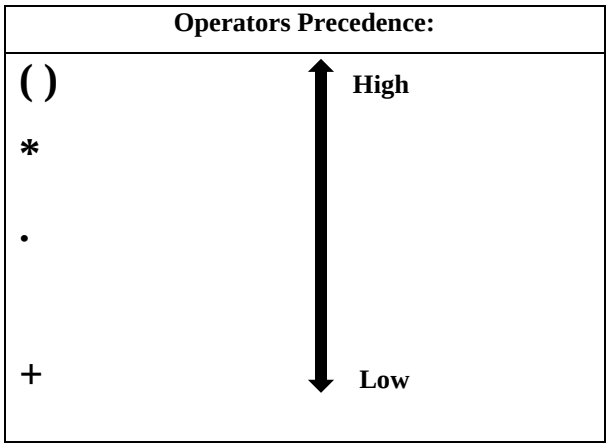
In the above example, in descriptive language we can't express all words (strings), but in regular expression we covered / expressed all the possible strings in short form.

Following are operators used in Regular expression.

+ , | , * , ( ) , .

For example:

- (a + b)*
- ab.ba
- (a | b)



+ Symbol in Regular Expression:

For Example: a + b

It means we can take one side at a time, either left side or right side.

Possible strings of a+b

Let's clear it with examples

- (a+b)***** = {λ, ab, aababb……..}
- (ab)***** = {λ, ab, abababb……..}

Let's clear it more, in some cases it may be like that

(a + b) = (a) **OR** (b)

It means from it, only two strings can be make, either "a" OR "b"

(a + b)* **=** **(**λ, ab, aababb**)**

It means from it, all possible strings can be make.

Examples of Regular Expression:

1. If ∑ = {a , b} Write all strings

D.M = {λ, a, b, ab, ba, aab, abb …..}

R.E = (a+b)*

2. If ∑ = {a , b} All possible string that starts & end with "**a**".

D.M = {aa, aba, aaa, abba, abaa …..}

R.E = a (a+b)* a

3. If ∑ = {a , b} All possible string that contain exactly two "a"

D.M = {aa, baa, aba, aab, abab, …..}

R.E = b* a b* a b* (*Here b* = λ, b, bb, bbb,…… (in it, b can be any value of it)*

4. If ∑ = {a , b} All possible string that contain double "bb"

D.M = {bb, abb, bba, bbb,…..}

R.E = (a+b)* bb (a+b)*

Here (a+b) = λ, a, b, ab, ba,..... (in it* ***a+b*** *can be any value of it)*

5. If ∑ = {a , b} All possible string that start & end with same letter.

We have two cases

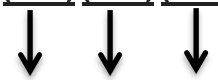
a (a+b)* **a** **b** (a+b)* **b**

So, R.E = a (a+b)* a + b (a+b)* b

- ***Note:*** ***about + symbol we have already discuss, that what does it mean.***

6. If ∑ = {0 , 1} All possible strings whose length is 3

R.E = **(0+1) (0+1) (0+1)**



0 1 0

D.M = {000, 010, 011, 111,110,100,… }

7. If ∑ = {0 , 1} All possible strings whose 3rd letter is 1.

R.E = (0+1) (0+1) **1** (0+1)*

How to make correct regular expression?

Make such regular expression which is suitable for all required strings.

Let's clear it with example.

All strings which contain exactly **two a** where ∑ = {a , b}

Strings: aa, aab, baba, bbaa, babab

Now we will make such Regular expression which express all above strings.

R.E = b* a b* a b*

Regular Languages

The language generated by any regular expression is called a regular language.

- If r1, r2 are regular expressions, corresponding to the languages L1 and L2 then the languages generated by r1+ r2, r1r2(or r2r1) and r1*(or r2*) are also regular languages.
- If L1 and L2 are expressed by r1and r2, respectively then the language expressed by:
  - r1+ r2, is the language L1 + L2 or L1 Union L2
  - r1r2, , is the language L1L2, of strings obtained by prefixing every string of L1 with every string of L2
  - r1*, is the language L1*, of strings obtained by concatenating the strings of L, including the null string.

Examples

- If r1 = (aa+bb) and r2 = (a+b) then the language of strings generated by r1+r2, is also a regular language, expressed by (aa+bb) + (a+b)

- If r1 = (aa+bb) and r2 = (a+b) then the language of strings generated by r1r2, is also a regular language, expressed by (aa+bb)(a+b)
- If r = (aa+bb) then the language of strings generated by r*, is also a regular language, expressed by (aa+bb)*