AUTOMATA FORMAL LANGUAGES AND LOGIC

Lecture notes
Regular Expression

Prepared by:
Kavitha K N
Assistant Professor

Department of Computer Science & Engineering
PES UNIVERSITY
(Established under Karnataka Act No.16 of 2013)
100-ft Ring Road, BSK III Stage, Bangalore – 560 085

Table of Contents:

Examples Solved:

12	Length of the string is greater than or equal to 2(at least 2)	**9**
13	Length of the string is at most 2	**9**
14	Length of the string is even	**9**
15	Length of the string is odd	**9**
16	Length of the string |w| mod 3=0	**9**
17	Length of the string |w| mod 3=2	**9**
18	Number of a's in the strings is equal to 2	**9**
19	Number of a's in the string should be greater than or equal to 2	**10**
20	Number of a's in the string should be utmost 2	**10**
21	L = { n_a (w) mod 2 = 0}	**10**
22	L = { n_a (w) mod 2 = 1}	**10**
23	L=($a^n b^m$: n+m is even}	**10**
24	L=($a^{2n} b^{2m}$: n,m >=2 }	**10**
25	L=($a^n b^m$: n,m >=1 and n*m >=3}	**10**
26	L=($a^n b^m$: n>=4 and m<=3}	**11**
27	L={$a^n b^m c^k$: n+m is odd and k is even}	**11**
28	L={$a^n b^m c^k$: n<=4, m>=2, k<=2}	**11**
29	L={aba, a^5 ,aba^7, a^{11},aba^{13} , …}	**11**
30	L ={VUV : U,V ∈ {a,b}* and |V| =2}	**11**
31	If the binary string is even then the length of the string is even, and if the binary string is odd then the length of the string is odd.	**11**

32	**L={ Binary string whose decimal equivalent is twice an odd number}**	**12**
33	L={binary strings that do not have two consecutive zero}	**12**
34	L= {binary strings that have exactly one pair of 0's}	**12**
35	L= (Strings over the alphabet a and where it does not have 2 consecutive a's and 2 consecutive b's}	**12**

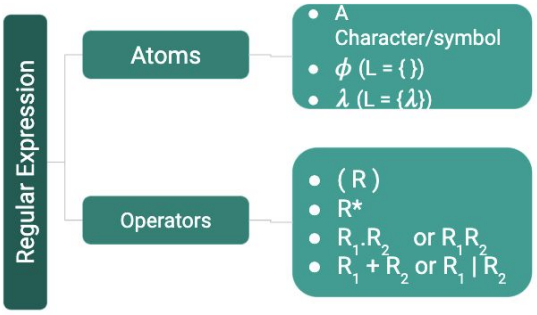
1. Regular expression

Regular expressions provide a convenient and concise notation for describing regular languages. They are Used as the basis for numerous software systems (Perl, flex, grep, etc.)

A regular expression is a pattern consisting of a sequence of characters that are matched against a text, more formally a string. If the string and the pattern match, the string belongs to the language denoted by the Regular expression.

More formally we can say regular expressions are an algebraic way to describe regular languages. It is like a mathematical expression.

Like how a mathematical expression is made of operands (data) and operators.
A regular expression is made of atoms and operators.



The atom specifies what we are looking for and where in the text the match is to be made.
The operator combines atoms into complex expressions.

Atomic Regular Expressions :
The regular expressions begin with three simple building blocks.
A character or a symbol for example (a, b, 0, 1 etc)
● The symbol Ø is a regular expression that represents the empty language Ø.
● The symbol lambda is a regular expression that represents the language which contains empty string{ λ }

We can make complex/Compound Regular Expressions by combining these atomic regular expressions using operators in four ways:

1) If R is a regular expression, (R) is a regular expression with the same meaning as R.
2) If R is a regular expression, R* is a regular expression for the Kleene closure of the language of R.
3) If R1 and R2 are regular expressions, R1 | R2 is a regular expression for the union of the languages of R1 and R2 .
4) If R1 and R2 are regular expressions, R1R2 is a regular expression for the concatenation of the languages of R1 and R2 .

The precedence of the operators is as follows:

- Brackets have the highest precedence(R)
- Then kleene closure R*
- Followed by concatenation R1R2
- And lastly union has the least precedence, that means it will be evaluated at last.R1 | R2

1.2 Algebraic laws of Regular Expressions :

a) **Identity of Regular Expression:**
An Identity of an operator is a value that when the operator is applied to the identity and some other value, the result is the other value
-> Union:
Identity of union operator is Ø
Ø + A = A + Ø = A
-> Concatenation:
Identity of Concatenation is λ
λ .A = A. λ = A

-> R . ϕ = ϕ . R = ϕ :

Let L_1, L_2 be languages, then the concatenation $L_1 \circ L_2 = \{w \mid w = xy, x \in L_1, y \in L_2\}$. If $L_2 = \emptyset$, then there is no string $y \in L_2$ and so there is no possible w such that $w = xy$. Thus for any L_1, we'll have $L_1 \circ \emptyset = \emptyset$.

b) **Associativity Laws for Regular Expressions**
-> Union:
Union operator is associative
A + (B + C) = (A + B) + C
-> Concatenation:
Concatenate operator is associative
A.(B.C) = (A.B).C

c) **Commutativity for Regular Expressions**
-> Union:
Union operator is Commutative
A + B = B + A
-> Concatenation:
Concatenate operator is not Commutative
AB ≠ BA

1.3 Basic regular expressions

1) **a**

If the regular expression is 'a' then the language has only one single String and can be represented as L={ a}

2) **a***

This regular expression means 0 or more occurrences of a

L={ λ , a,aa,aaa,aaaa…}

3) **a+**

1 or more more occurrences of a

L={a,aa,aaa,aaaa…}

4) **(a+b) can also be represented as (a|b)**

This regular expression means either a or b

L={a,b}

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

Set of strings greater than equal to 0 over the alphabet a or b

L={ λ ,a,b,aa,bb,ab,ba.aab…}

$(a+b)^0$

String of length 0

L={ ε }

= Σ^0

$(a+b)^1$

Set of all strings with length 1

L={a,b}

$(a+b)^2$

Set of all strings of length 2

(a+b)(a+b) or L={aa,ab,ba,bb}

1.2 Construction of regular expressions

-> Obtain regular expression for the following over the alphabet Σ {a,b}:

1) **Strings ending with ab**

The string should end with ab, it can start with any number of a's and b's in any order of any length

RE= (a+b)*ab

2) **Strings starts with ab**

The string should start with ab, it can end with any number of a's and b's in any order of any length

RE=ab(a+b)*

3) **String contains ab**

The string should mandatorily contain the substring ab, before the substring ab, there can be an any number of a's and b's in any order of any length and after the

substring ab, there can be an any number of a's and b's in any order of any length

RE= (a+b)* ab (a+b)*

4) **String should end with ab or ba**

The string should either end with ab or ba, it can start with any number of a's and b's in any order of any length

RE=(a+b)*(ab+ba)

5) **Strings that start and end with same symbol**

The string can either start and end with a or it can start and end with b and can have any number of a's and b's in any order of any length in between.

RE=a (a+b)* a + b(a+b)*b

6) **Strings that start and end with different symbol**

The string can either start with a and end with b or it can with start b and end with a and can have any number of a's and b's in any order of any length in between.

RE=a(a+b)*b + b(a+b)*a

7) **String where the 3rd symbol from the left (from starting) is 'a'(LHS)**

The first and second symbol can be either a or b 3 rd symbol should be a after that the string can have any number of a's and b's in any order of any length

RE=(a+b)(a+b)a(a+b)*

8) **String where the 2nd symbol from the right (from ending) is a(RHS)**

The last symbol can be either a or b, last but one symbol should be 'a' , the string can start with number of a's and b's in any order of any length

RE=(a+b)*a (a+b)

9) **String where at least one 'a' in the first three symbols and at least one 'b' in the last two Symbols.**

'a' can be 1st , 2nd or 3rd symbol in the first three symbols and be can be last or last but one symbol in the last two symbols , between first three and last 2 symbols there can be any number of a's and b's in any order of any length

RE= (a(a+b)(a+b)) + ((a+b) a(a+b)) + (a+b)(a+)a (a+b)* (b(a+b)) + (a+b)b

10) Strings contain at least one a and at least one b.

The minimum length of the string is 2,

RE=(a+b)* a (a+b)* b (a+b)* + (a+b)* b (a+b)* a (a+b)*

11) Length of the string is exactly equal to 2

|w|=2

(a+b)(a+b) or (aa+ab+ba+bb)

12) Length of the string is greater than or equal to 2(at least 2)

|w|>=2

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

13) Length of the string is at most 2

|w|<=2

Here you can have strings of length {0,1,2}

(λ +a+b+aa+ab+ba+bb)

=(λ +(a+b))(λ +(a+b))

Mod counter questions:

14) Length of the string is even

|w| mod 2=0

Generate strings in multiples of 2,

RE = ((a+b)(a+b))*

15) Length of the string is odd

|w| mod 2=1

RE = ((a+b)(a+b))*(a+b)

16) Length of the string |w| mod 3=0

RE = ((a+b)(a+b)(a+b))*

17) Length of the string |w| mod 3=2

RE = ((a+b)(a+b)(a+b))*(a+b)(a+b)

Regular expressions based on restriction to the individual symbols Σ = {a , b}

18) Number of a's in the strings is equal to 2

L = { n_a (w) =2, w ∈ {a,b}*}

Number of a's in the string should be exactly 2, and no restriction on the number of b's in the starting or in the middle or in the end.

RE = b* a b* a b*

19) Number of a's in the string should be greater than or equal to 2

L = { n_a (w) >=2}

We must have at least 2 a's and later we can have any number of a's, there is no restriction on number of b's

RE = b* a b* a b* (a+b)*

20) Number of a's in the string should be at most 2

L = { n_a (w) <=2}

RE = b*+b*ab*+b*ab*ab*

0r

RE = b* (a+ λ)b* (a+ λ)b*

21) L = { n_a (w) mod 2 = 0}

You can have a's which is multiple of 2 any number of times, or you can have only b's

RE = (b* a b* a b*)* + b*

22) L = { n_a (w) mod 2 = 1}

RE = (b*ab*ab*)* b* a b*

23) L=($a^n b^m$: n+m is even}

Length of the string is even, We can get even length if number of a's is even and number of b's is even or number of a's is odd and number of b's is odd so we have two cases,

RE = (aa)*(bb)* + (aa)*a(bb)*b

24) L=($a^{2n} b^{2m}$: n,m >=1 }

The minimum string is aabb

RE = aa(aa)* bb(bb)*

25) L=($a^n b^m$: n,m >=1 and n*m >=3}

Here we have to consider different cases to get the product n * m >=3. If the number of a's is one and number of b;s is greater than or equal to 3 then the product would be greater than or equal to 3 or, number of a's can be greater than or equal to 3 and number of b's is 1 them the product is

greater than or equal to 3, or number a's and b's is greater than or equal to 2 then the product is greater than 3, So,

RE = abbbb* + aaaa*b + aaa*bbb*

26) L=($a^n b^m$: n>=4 and m<=3}

Number of a's is at least 4 and the number of b's is at most 3

RE = aaaaa* (λ + b+bb+bbb)

Or

RE = aaaaa* ((λ + b)(λ + b)(λ + b))

27) L={$a^n b^m c^k$: n+m is odd and k is even}

Note: odd+even=odd and even+odd=odd

RE = ((aa)*a(bb)* + (aa)* (bb)*b) (cc)*

28) L={$a^n b^m c^k$: n<=4, m>=2, k<=2}

Number of a's is at most 4, number of b's is at least 2 and number of c's is at most 2

RE = (λ +a)(λ +a)(λ +a)(λ +a) bbb* (λ +c)(λ +c)

29) L={aba, a^5 , aba^7, a^{11},aba^{13} , …}

RE = (aba + a^5)(a^6)

Or

(aba + aaaaa) (aaaaaa)*

30)L ={VUV : U,V ∈ {a,b}* and |V| =2}

RE = (a+b)(a+b)(a+b)*(a+b)(a+b)

Regular expressions on binary strings Σ ={0,1}

31) If the binary string is even then the length of the string is even, and if the binary string is odd then the length of the string is odd.

RE = ((0+1)(0+1))*(0+1)0 + ((0+1)(0+1))*1

Recognize pattern

32) L={ Binary string whose decimal equivalent is twice an odd number}

2* Odd no	Decimal value	Binary string
2*1	2	10
2*3	6	110
2*5	10	1010
2*7	14	1110
2*9	18	10010

The pattern what we can observe here is all the binary strings ends in 10

RE = (0+1)*(10)

33) L={binary strings that do not have two consecutive zero}

-> 01 as the building block:

RE = (1+01)*(0+ λ)

-> 10 as the building block:

RE = (0+ λ)(1+10)*

34) L= {binary strings that have exactly one pair of 0's}

RE = (1+01)* 00 (1+10)*

35) L= (Strings over the alphabet a,b and where it does not have 2 consecutive a's and 2 consecutive b's}

RE = (b+ λ)ab (a+ λ)

Or

RE =(a+ λ) ba (b+ λ)