P.L.

It is a finiteset of strings.

∑ = { x }

L1 = {x,xx,xxx,….} = [x^n|n>=l}

Property? Close under concat?

L2 = {x^i|I e N and I is odd} = {x,xxx,xxxxx,….} – This is not closed because if you concatenate two of the variables its possible for it to not be odd, for example the first two terms would equal xxxx.

Example:

∑ = {a,b}

Palindrome = {epsilon,all string x = reverse(X)}

= [e,a,b,aa,bb,aba,…} this is not closed concatenation

Algebra to describe higher type of languages?

Exercise 1 (DIY):

∑={a,b,c}

L={x|x starts with a and ends with c}

Find which string does not belong to L.L ? abcabc,acaac,abcbcac,abcbacbc

Exercise 2 (DIY)

∑ = same as ex 1

L = same as ex 1

L\*

Which of these strings are not valid for L\* -epsilon,acaca,abcbc,acacacacac

Regular Lenguages?

It is a language over alphabet ∑ that is defined recursively:

Ф {Emtpy language} = RL

{ϵ} {empty string} = RL

For each a ϵ∑ {a} = RL

If L1 and L2 are RLs then

L1 u L2 = RL

L1 \* L2 = RL

L1\* = RL

Notes

RL are these languages that can be represented with finite (ie fixed) memory.

Not every finite languae is a regular language and not all RL are necessarily finite.

Not Regular Languages?

Theorem: A language L over alphabet ∑ is Non regular if and only if there is an infinite subset of ∑\*

L = {a^nb^n|n is a positive integer}

Kleene Theorem

Regular language <-> Regular Grammar <-> Regular Expression <-> Finite State Automata<-> Transition Graph

<-> means equivalence

Regular Expression?

-every character in ∑ is a RE

-If R1 and R2 both RE’s then

(R1) -> RE

R1\*R2 -> RE(concatenation)

(R1+R2) R1|R2 ->RE (alteration)

R1\* ->RE (repetition)

RE RL

a {a}

a|b {a,b}

ϵ|a {ϵ,a}

a|b|a {a,b}

ab {ab}

(a|b)(c|d) {ac,ad,bc,bd]

(ab)\* {ϵ,ab,abab,ababab,…}

L((a|b).a\*) = L(a|b).L(a\*)

= (L(a)UL(b).(L(a))\*

=({a}U{b}).({a})\*

={a,b].{ϵ,a,aa,aaa,…}

RE = (aa)\*(bb)\*b

L(RE) = {a^(2m)b^(2m)b

Theorem:

Two regular expressions are equivalent iff they genereate the same regular language

Example:

R1 = (l+ol)\*(o+µ)

R2 = (l+oll\*)\*(o+µ)+l\*(o+ µ)

L(R1)=L(R2)

Regex API Util Java

Example1: Matching and Validation

Import javautil.ArrayList;

Import javautil.List;

Import javautil.regex;

Public class Validate{

Public static void main(String[]args){

List <String> input = new ArrayList<String>();

Inpute.add(“123-45-6789”);

Input.add(“9876-5-4321”);

Input.add(“192-89-7465”);

//format must be 3#s-2#s-4#s so the first and last one would be accepted but not the middle one

for(string ssn : input){

if ssn.matches(“^(\\d{3}-?\\d{2}-?\\d{4})$”0)

System.out.println(“Format good SSN” + ssn);

}}}}

////Not sure what this is, second java file

Public class Extract {

Public static void main (String[] args)[

String input = “ I have a cat, but I like my dog better”;

Pattern p = Pattern.compile(“(mouse|cat|dog|wolf|bear)”);

Matcher m = p.matcher(input);

List <String> animals = new ArrayList<String>();

While(m.find()){

System.out.pirntln(“Found a” + m.group() +’.”);

Animals.add(m.group());}}}}