lec 5:Regular Expression

1.Regular Expression

Regular Expression have an important role in computer science applications. in application involving text, users may want to search for strings that satisfy certain pattern. regular expression provided a powerful method for describing such pattern such as AWK and GREP in UNIX,

Regular Expression is a set of symbols, Thus if alphabet= {a, b}, then aab, a, baba, bbbbb, and baaaaa would all be strings of symbols of alphabet.

Regular expressions can be used to define languages. A regular expression is like a "pattern"; strings that match the pattern are in the language, strings that do not match the pattern are not in the language. The construction of regular expressions is defined recursively, starting with primitive regular expressions, which can be composed using typical operators to form more complex regular expressions.

In addition we include an empty string denoted by (_) which has no symbols in it.

2.Operation on Language

Regular operations on languages (sets of strings). Suppose L and K are languages.

- ***Union***: $L \cup K = \left\{ x \,\middle|\, x \in L \quad \text{or} \quad x \in K \right\}$.
- ***Concatenation***: $L \circ K = LK = \left\{ xy \,\middle|\, x \in L \quad \text{and} \quad y \in K \right\}$.
- ***Star*** (***Kleene star***):

$$L^* = \left\{ w_1 w_2 \dots w_n \,\middle|\, w_1, \dots, w_n \in L \text{ and } n \geq 0 \right\}.$$

We (hopefully) all understand what union does. The other two have some subtleties. Let

$$L = \{\texttt{under}, \texttt{over}\}, \quad \text{and} \quad K = \{\texttt{ground}, \texttt{water}, \texttt{work}\}.$$

Then

$$LK = \{\texttt{underground}, \texttt{underwater}, \texttt{underwork}, \texttt{overground}, \texttt{overwater}, \texttt{overwork}\}.$$

Similarly,

$$K^* = \left\{ \begin{array}{l} \epsilon, \texttt{ground}, \texttt{water}, \texttt{work}, \texttt{groundground}, \\ \texttt{groundwater}, \texttt{groundwork}, \texttt{workground}, \\ \texttt{waterworkwork}, \dots \end{array} \right\}.$$

3. Definition of Regular Expressions :

Let us fix an alphabet $\sum$. Here are the basic regular expression.

regex	conditions	set represented
a	$a \in \Sigma$	{a}
ϵ		$\{\epsilon\}$
$\emptyset$		{}

Thus, ; represents the empty language. But _ represents that language which has the empty word as its only word in the language..

In particular, for a regular expression hexpi, we will use the notation L(hexpi) to denote the language associated with this regular expression. Thus,

$$L(\varepsilon) = \{ \varepsilon\} \quad \text{and} \quad L(\Phi) = \{ \} ;$$

which are two different languages.

Suppose that L(R) is the language represented by the regular expression R. Here are recursive rules that make complex regular expressions out of simpler ones.

regex	conditions	set represented
$R \cup S$ or $R + S$	R, S regexes	$L(R) \cup L(S)$
$R \circ S$ or RS	R, S regexes	$L(R)L(S)$
R^*	R a regex	$L(R)^*$

And some handy shorthand notation:

regex	conditions	set represented
R^+	R a regex	$L(R)L(R)^*$
Σ		Σ

Some specific boundary case examples:

1. $R\epsilon = R = \epsilon R$.
2. $R\emptyset = \emptyset = \emptyset R$.

   This is a bit confusing, so let us see why this is true, recall that

$$R\emptyset = \left\{ xy \;\middle|\; x \in R \text{ and } y \in \emptyset \right\}.$$

   But the empty set ($\emptyset$) does not contain any element, and as such, no concatenated string can be created. Namely, its the empty language.

3. $R \cup \emptyset = R$ (just like with any set).
4. $R \cup \epsilon = \epsilon \cup R$.

   This expression can not always be simplified, since ϵ might not be in the language $L(R)$.

5. $\emptyset^* = \{\epsilon\}$, since the empty word is always contain in the language generated by the star operator.
6. $\epsilon^* = \{\epsilon\}$.

Examples of Kleene star:

(1*) is the set of strings {ε, 1, 11, 111, 1111, 11111, etc. }

(1100)* is the set of strings {ε, 1100, 11001100, 110011001100, etc. }

(00+11)* is the set of strings {ε , 00, 11, 0000, 0011, 1100, 1111, 000000, 000011, 001100, etc. }

(0+1)* is all possible strings of zeros and ones, often written as $\sum$ * where $\sum$ = {0, 1}

(0+1)* (00+11) is all strings of zeros and ones that end with either 00 or 11.

(w)+ is a shorthand for (w)(w)* w is any string or expression and the superscript plus, +

Concatenation:

Notation to the concatenation: . (The dot.):

if L1 = {x, xxx} and L2 = {xx} So (L1.L2) means L1 concatenated L2 and it is L1oL2 = {xxx, xxxxx}

Example1:

L1 = {a, b}.

L2 = {c, d}.

L1.L2 = {ac, ad, bc, bd}

Note: ab differ from ba.

Example 2:

∑= {x}.

L1 = {set of all odd words over ∑ with odd length}.

L1 = {set of all even words over ∑ with odd length}.

L1= {x, xxx, xxxxx, xxxxxxx……}.

L2= {_, xx, xxxx, xxxxxx…}.

L1.L2 = {x, xxx, xxxxx, xxxxxxx…}.

Example 3:

L1 = {x, xxx}.

L2 = {xx}.

L1.L2 = {xxx, xxxxx}.

Some rules on concatenation:

λ.x = x

L1.L2 = {set of elements}

Example 4:

A = {a,b} // the alphabet is composed of a and b

A* = {l, a,b,aa,ab,ba,bb,aaa,aab,…}

The symbol * is called the Kleene star.

Ø(empty set)

ε (empty string)

Given regular expressions x and y, x + y is a regular expression representing the set of all strings in either x or y (set union)

x = {a b} y = {c d} x + y = {a b c d}

Example 5:

Let A={0,1}, W1 = 00110011, W2 = 00000

W1W2 = 0011001100000

W2W1 = 0000000110011

W1λ = W1 = 00110011

λW2 = W2 = 00000

x = {a,b} y = {c,d} xy = {ac, ad, bc, bd}

Note: (a + b)* = (a*b*)*

More Examples of regular expressions

Describe the language = what is the output (words, strings) of the following RE

Regular expression	output(set of strings)
λ	{λ }
λ*	{λ }
a	{ a }
aa	{ aa }
a*	{ε, a, aa, aaa, ….}
aa*	{ a, aa, aaa, ... }
a+	{ a, aa, aaa, ...}
ba+	{ ba, baa, baaa, ...}
(ba)+	{ ba, baba, bababa, ...}
(a|b)	{ a, b }

a|b*	{ a, l, b, bb, bbb, ... }
(a|b)*	{λ, a, b, aa, ab, ba, bb, ... }
aa(ba)*bb	{ aabb, aababb, aababababb, ... }
(a + a)	{a}
(a + b)	{a, b}
(a + b)2	(a + b)(a + b) =={aa, ab, ba, bb}
(a + b + c)	{a, b, c}
(a + b)*	{ ε, a, b, aa, bb, ab, ba, aaa, bbb, aab, bba, ….}
(abc)	{abc}
(λ+ a) bc	{bc, abc}
ab*	{a, ab, abb, abbb, …}
(ab)*	{λ, ab, abab, ababab, …}
a + b*	{a, λ, b, bb, bbb, …}
a (a + b)*	{a, aa, ab, aaa, abb, aba, abaa, …} (a + b)* a (a + b)*
	{a, aaa, aab, baa, bab, …}
(a + λ)*	(a)* = {λ, a, aa, aaa, ….}
x* (a + b) + (a + b)	x* (a + b)
x* y + y	x* y
(x + l)x*	x* (x + l) = x*
(x + l)(x + l)* (x + l)	x*

note: symbol λ same mean epsilon symbol (ε)