Lecture two

Topics that must be covered in this lecture:

- ***Regular expression (RE)***
- ***Formal definition of RE:***
- ***Examples of Kleene star***
- ***Examples of concatenation***
- ***Examples of union***
- ***More Examples of RE***

Regular Expression

In arithmetic, we can use the operations + and × to build up expressions such as

$$(5 + 3) \times 4.$$

The value of the arithmetic expression is the number 32. The value of a regular. Similarly, we can use the regular operations to build up expressions describing languages, which are called regular expressions. An example is:

$$(0 \cup 1)0^*$$

In this case, the value is the language consisting of all strings starting with a 0 or a 1 followed by any number of 0s. We get this result by dissecting the expression into its parts:

- First, the symbols 0 and 1 are shorthand for the sets {0} and {1}. So (0∪ 1) means ({0} ∪ {1}). The value of this part is the language {0,1}. The part 0* means {0}*, and its value is the language consisting of all strings containing any number of 0s.
- Second, like the × symbol in algebra, the concatenation symbol ° often is implicit in regular expressions. Thus (0∪1)0* actually is shorthand for (0∪1) ° 0*. The concatenation attaches the strings from the two parts to obtain the value of the entire expression.

In arithmetic, we say that × has precedence over + to mean that when there is a choice, we do the × operation first. Thus in 2+3×4, the 3×4 is done before the addition. To have the addition done first, we must add parentheses to obtain (2 + 3) × 4. In regular expressions, the star operation is done first, followed by concatenation, and finally union, unless parentheses change the usual order.

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, Regular Expression is a set of symbols, Thus if alphabet Σ= {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 (λ) which has no symbols in it.

FORMAL DEFINITION OF A REGULAR EXPRESSION:

Let Σ be an alphabet. The regular expressions over Σ are defined recursively as follows:

i) Basis: ϕ, λ, and a, for every a ∈ Σ, are regular expressions over Σ.
ii) Recursive step: Let u and v be regular expressions over Σ. The expressions

(u U v)
(uv)
(u*)

are regular expressions over Σ.
iii) Closure: u is a regular expression over Σ only if it can be obtained from the basis elements by a finite number of applications of the recursive step.

Every regular expression represents a language according to interpretation of the symbol ∪ and ∗ as set union and kleen star.

In i, the regular expressions a and λ represent the languages {a} and { λ }, respectively. In i, the regular expression ϕ represents the empty language. In items ii, the expressions represent the languages obtained by taking the union or concatenation of the languages R1 and R2, or the star of the language R1, respectively.

Don't confuse the regular expressions λ and ϕ. The expression λ represents the language containing a single string—namely, the empty string—whereas ϕ represents the language that doesn't contain any strings. When we want to distinguish

between a regular expression R and the language that it describes, we write L(R) to be the language of R.

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, 000000,110011, 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, +

Example 2:

The symbol * is called the Kleene star.
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} find x+y x + y = {a b c d}

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: let L1 = {a, b}. L2 = {c, d}. find L1L2

L1.L2 = {ac, ad, bc, bd} Note: ab differ from ba.

Example 2:

Let ∑= {x}.

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

L2 = {set of all even words over ∑ with odd length}.

L1= {x, xxx, xxxxx, xxxxxxx……}.

L2= { λ, xx, xxxx, xxxxxx…}.

L1.L2 = {x, xxx, xxxxx, xxxxxxx…}.

Example 3:

Let L1 = {x, xxx}. L2 = {xx}. Find L1L2

L1.L2 = {xxx, xxxxx}.

Some rules on concatenation:

λ.x = x

L1.L2 = {set of elements}

Example 4:

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

W1W2 = 0011001100000

W2W1 = 0000000110011

W1λ = W1 = 00110011

λW2 = W2 = 00000

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

EXAMPLE :

In the following instances, we assume that the alphabet Σ is {0,1}, describe the sets from the following regular expression.

1. 0*10* = {w| w contains a single 1}.
2. Σ *1 Σ *= {w| w has at least one 1}.
3. Σ *001 Σ * = {w| w contains the string 001 as a substring}.
4. 1*(01+)* = {w| every 0 in w is followed by at least one 1}.
5. (ΣΣ)* = {w| w is a string of even length}.
6. (ΣΣΣ)* = {w| the length of w is a multiple of 3}.
7. 01 ∪ 10 = {01, 10}.
8. 0Σ *0 ∪ 10Σ *1 ∪ 0 ∪ 1 = {w| w starts and ends with the same symbol}.
9. (0 ∪ λ)1* = 01* ∪ 1*.

The expression 0 ∪ λ describes the language {0, λ }, so the concatenation operation adds either 0 or λ before every string in 1*.

10. (0 ∪ λ)(1 ∪ λ) = { λ, 0, 1, 01}.
11. 1* ϕ = ϕ.

Concatenating the empty set to any set yields the empty set.

12. ϕ * = { λ }.

The star operation puts together any number of strings from the language to get a string in the result. If the language is empty, the star operation can put together 0 strings, giving only the empty string.

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*	{ λ, 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, λ, b, bb, bbb, ... }
(a|b)*	{λ, a, b, aa, ab, ba, bb, ... }
aa(ba)*bb	{ aabb, aababb, aabababb, ... }
(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)