11. Regular Expressions

Learning Outcomes: At the conclusion of the chapter, the student will be able to:

- Identify the language associated with a regular expression
- Find a regular expression to describe a given language

Regular expressions are a method of formally describing a regular language through algebraic notations. They serve as an input language for many systems that process strings like UNIX grep command for finding strings. Search systems convert the regular expression into DFA or NFA and simulate that automaton on the file being searched.

1. Definition of Regular Expressions

Basis:

1) Φ is a regular expressions and language of Φ or L(Φ) = Φ
2) ε is a regular expressions and language of ε or L(ε) = {ε}
3) For each symbol a ∈ Σ, **a** is a regular expressions and language of **a** or L(**a**) = {a}

Induction:

4) If R_1 and R_2 are regular expressions, R_1+R_2 is a regular expression and $L(R_1+R_2) = L(R_1) \cup L(R_2)$
5) If R_1 and R_2 are regular expressions, R_1R_2 is a regular expression and $L(R_1R_2) = L(R_1)L(R_2)$
6) If R is a regular expressions, then R* is also a regular expressions and $L(R^*) = (L(R^*))^*$
7) If R is a regular expressions, then (R) is also a regular expressions and L(®) = (L(R))

Precedence: Parentheses have the highest precedence, followed by *, concatenation, and then union.

Examples of some Regular Expressions

- $L(ab^*) = \{ a, ab, abb, \ldots ab^n \}$, $L((ab)^*) = \{\varepsilon, ab, abab, ababab, \ldots \}$, $L(a^* b^*) = \{\varepsilon, a, aaa\ldots, b, bb, bbb\ldots, ab, abb, abb.., aaa\ldots bbb\ldots\}$
- L(001) = {001}
- L(0+10*) = { 0, 1, 10, 100, 1000, 10000, … }
- L(0*10*) = {1, 01, 10, 010, 0010, …} i.e. {w | w has exactly a single 1}
- L(ΣΣ)* = {w | w is a string of even length}
- L((0(0+1))*) = { ε, 00, 01, 0000, 0001, 0100, 0101, …}
- L((0+ε)(1+ ε)) = {ε, 0, 1, 01}

Example 11.1 Design REs for languages described by the following strings (Σ = {a, b})

a) Beginning with aa: aa(a+b)*

b) Beginning with a and ending with bb: $a\,(a+b)^*\,bb$

c) Ending with aba or aaba: $(a+b)^{*}(aba + aaba)$

d) Strings of even length: $((a+b)(a+b))^{*}$

e) At least one a: $(a+b)^{*} a (a+b)^{*}$ OR $b^{*} a (a+b)^{*}$

f) At most one a: $b^{*} + b^{*}a b^{*}$

g) Containing no more than 3 a's: $b^{*} (\varepsilon + a) b^{*} (\varepsilon + a) b^{*} (\varepsilon + a)$

h) Containing at least 2 a's: $b^{*} a b^{*} a (a+b)^{*}$

i) Beginning and ending with the same symbol : $a (a+b)^{*} a + b (a+b)^{*} b$

j) All strings of 0's and 1's without two consecutive 1's $(0+10)^{*} (\varepsilon + 1)$

2. Equivalent Regular expressions

Two REs are equivalent if they describe the same language as in example 9.1 (e). There exist some algebraic laws that may be used to simplify a given RE.

3. Algebraic Laws for Regular Expressions

Let L, M and N be REs

Associativity and Commutativity		
1	$L + M = M + L$	Commutative law for union (Note that $LM \neq ML$)
2	$(L+M) + N = L + (M + N)$	Associative law for union
3	$(LM)N = L(MN)$	Associative law for concatenation
Distributive Laws		
1	$L(M+N) = LM + LN$	Left distributive law for concatenation over union
2	$(L+M)N = LN + MN$	Right distributive law for concatenation over union
Identities and Annhilators		
1	$\Phi + L = L + \Phi = L$	Φ is identity for union
2	$\varepsilon L = L\varepsilon = L$	ε is identity for concatenation (Note that $R+\varepsilon$ may or may not equal R (we are adding ε to the language))
3	$\Phi L = L\Phi = \Phi$	Φ is annihilator for concatenation (Note that $L\emptyset$ will only equal L if L itself is the empty set)

Idempotent Law		
1	$L + L = L$	Idempotent law for union
Laws and identities involving Closures		
1	$(L^*)^* = L^*$	Closing an expression that is already closed does not change the language
2	$\Phi^* = \varepsilon$	The closure of Φ contains only string ε
3	$\varepsilon^* = \varepsilon$	Only string that can be formed by concatenating any number of copies of empty string is the empty string itself
4	$LL^* = L^*L = L^+$	Definition of positive closure
5	$L^* = L^+ + \varepsilon$ or $L^* = LL^* + \varepsilon = \varepsilon + LL^*$	Definition of positive closure
6	$L^* L^* = L^*$	Concatenation of closure of a language will be the closure itself
7	$(LM)^*L = L(ML)^*$	
8	$(L+M)^* = (L^* M^*)^* = (L^*+M^*)^*$	All three represent closure of union