Regular Expressions and Languages

Regular Language

- A language is said to be regular if there is a finite acceptor for it
- Every Regular Language can be described by using NFA/DFA

Regular Expression

- One way of describing regular languages which consists of strings, symbols and operators
- Why do we use Regular Expressions?
  - We study Regular Expressions because it helps us define how the computers should interpret the regularity in forms such as programming languages and more
  - It has been applied to the lexical analysis of the computer programming language compiler, switch circuit design, syntactic pattern recognition, and so on.
- How to use Regular Expressions? (Notation)
  - Combination of strings, symbols and from some alphabet Σ
  - Parenthesis
  - Operators '+' , '. ' and '*'
    - + represents union
    - . represents concatenation
    - * represents star-closure
- Examples — R = (a + (b . c))* — L = {ε0, a, aa, bc, bcbc, bca, abc, aabc...}
- Closures
  - Kleen — r* is known as kleen closure which indicate occurrences of r for ∞ number of times
    - If Σ = { a } and R = a*
    - Then R = { €, a, aa, aaa, aaaa, ... }
  - Positive
    - If Σ = { a } and R = a+
    - Then R = { a, aa, aaa, aaaa, ... }

Language associated with regular expressions

- Regular expressions can be used to describe some simple languages
- If r is the regular expression then L(r) is the language associated with it
- The language L(r) is defined using the following rules
  - Φ is a regular expression denoting the empty set
  - € is a regular expression denoting { € }
  - For every a ∈ Σ, a is a regular expression denoting { a }

Few Examples on Regular Expressions

- Design the regular expression for the language containing all the strings having any number of a's and b's over the set Σ = { a, b }
  - Σ = { a, b }
  - Possible set of language L = { ε, a, b, aa, ab, ba, bb, aaa, ... }
  - R = (a + b)*
- Construct the regular expression for the language containing all the strings having any number of a's and b's except the null string over the set Σ = { a, b }
  - Σ = { a, b }
  - Possible set of language L = { a, b, aa, ab, ba, bb, aaa, ... }
  - R = (a + b)+
- Write regular expression to denote the language L over Σ* where Σ = { a, b, c } in which every string will be such that at least one a followed by at least one b followed by at least one c
  - Σ = { a, b, c }
  - R. E. = a+b+c+
  - Possible set of language L = { abc, aabc, abbc, aabbc, abbccc, aabbcc, ... }
- In Reverse: Describe in simple English language Given regular expression is r = (a + ab)*
  - Σ = { a, b }
  - R. E. = (a + ab)*
  - Possible set of language L = { ε, a, ab, aba, aab, abab, aaa, ... }
  - The language is beginning with zero or any number of a's but not having consecutive b's
- Write regular expression to denote the language L over Σ* where Σ = { a, b } such that the third character from right end of the string is always a
  - Σ = { a, b }
  - R. E. = (Any number of character a's and b's) . a . (a or b) . (a or b)
  - R = (a + b)* . a . (a + b) . (a + b)
  - Possible set of language L = { aaa , aba, abb, aaba, aabb, bababb, ... }
- Find a regular expression for the language L = { w ε { 0, 1 }* | w has no pair of consecutive zeros }
  - The regular expression involves repetition of the strings of the form 1 ... 101 ... 1
  - We can represent this as r1 = (1*011*)*
  - If the string can be terminated by 0 then we can modify the expression as
  - r1 = (1*011*)* . (0 + ε)
  - The expression does not represent the strings with all 1's and strings with all 1's ending with 0
  - It can be represented as
  - r2 = 1* . (0 + ε)
  - We can give resultant regular expression as
  - R = r1 + r2
  - R = (1*011*)* . (0 + ε) + 1* . (0 + ε)
  - Alternate Solution: r = (1 + 01)* . (0 + ε)
  - r = (1* (01)* 1*)*. (0 + ε)
- Obtain regular expression over Σ = { 0 , 1 } such that the set of all strings begin or end with 00 or 11
  - Hint: Consider 6 different cases
    - R1 = Strings beginning with 00 and ending with 11
    - R2 = Strings beginning with 11 and ending with 00
    - R3 = Strings beginning with 00
    - R4 = Strings beginning with 11
    - R5 = Strings ending with 00
    - R6 = Strings ending with 11