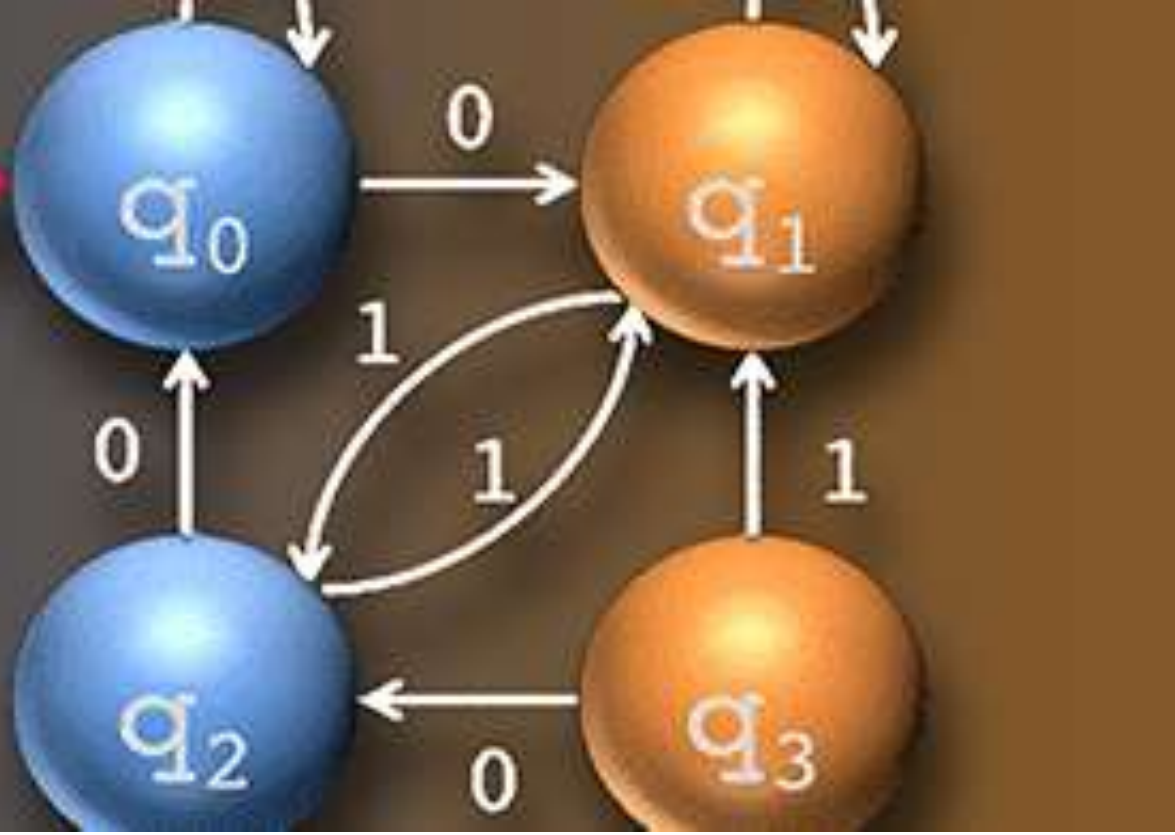
CSE 305

Theory of COMPUTATION



Lecture 20

Regular Expressions (1)

Md. Mijanur Rahman, Prof. Dr.

Dept. of Computer Science and Engineering, Jatiya Kabi Kazi Nazrul Islam University, Bangladesh.
www.mijanrahman.com

Contents

Regular Expressions

- Regular Expressions
- Regular Languages
- Operation on Regular Languages
- Extensions of Regular Expressions
- Regular Sets and Properties of Regular Sets
- Identities Related to Regular Expressions
- Conversion of Finite Automata into Regular Expressions
- Conversion of Regular Expressions into Finite Automata
- Pumping Lemma for Regular Languages
- Closure Properties of Regular Languages
- Relationship with other Computation Models

Regular Expressions

- Regular expressions are an important notation for defining patterns. **Each pattern connects a set of strings. Therefore, regular expressions will give as names for sets of strings.**
- It supports an appropriate and useful notation for describing tokens. **Regular Expressions define the language accepted by finite Automata (Transition Diagram).**

- Regular Expressions are defined over an alphabet $\sum$. If R is a Regular Expression, therefore L(R) represents language denoted by the regular expression.

Regular Expressions

- **Language** – It is a collection of strings over some fixed alphabet. The empty string can be indicated by ε.
- **Example** – If L (Language) = set of strings of 0's **& 1's** of length two

  then L = {00, 01, 10, 11}

- **Example** – If L = {1}

  then $L^* = L^0 \cup L^1 \cup L^2 \cup \ldots..$

  Here * can be 0, 1, 2, 3……..

  ∴ $L^* = \{\varepsilon\} \cup \{1\} \cup \{11\} \cup \ldots..$

  ∴ $L^* = \{\varepsilon, 1, 11, 111, \ldots..\}$

Regular Expressions

A Regular Expression (RE) can be recursively defined as follows −

- **ε** is a Regular Expression indicates the language containing an empty string. **(L (ε) = {ε})**
- **φ** is a Regular Expression denoting an empty language. **(L (φ) = { })**
- **x** is a Regular Expression where **L = {x}**
- If **X** is a Regular Expression denoting the language **L(X)** and **Y** is a Regular Expression denoting the language **L(Y)**, then
  - **X + Y** is a Regular Expression corresponding to the language **L(X) ∪ L(Y)** where **L(X+Y) = L(X) ∪ L(Y)**.
  - **X . Y** is a Regular Expression corresponding to the language **L(X) . L(Y)** where **L(X.Y) = L(X) . L(Y)**
  - **R*** is a Regular Expression corresponding to the language **L(R*)**where **L(R*) = (L(R))***
- If we apply any of the rules several times from 1 to 5, they are Regular Expressions.

Some RE Examples

Regular Expressions

and their Regular Sets:

Regular Expressions	Regular Set
(0 + 10*)	L = { 0, 1, 10, 100, 1000, 10000, … }
(0*10*)	L = {1, 01, 10, 010, 0010, …}
(0 + ε)(1 + ε)	L = {ε, 0, 1, 01}
(a+b)*	Set of strings of a's and b's of any length including the null string. So L = { ε, a, b, aa , ab , bb , ba, aaa…….}
(a+b)*abb	Set of strings of a's and b's ending with the string abb. So L = {abb, aabb, babb, aaabb, ababb, ……..}
(11)*	Set consisting of even number of 1's including empty string, So L= {ε, 11, 1111, 111111, …….}
(aa)*(bb)*b	Set of strings consisting of even number of a's followed by odd number of b's , so L = {b, aab, aabbb, aabbbbb, aaaab, aaaabbb, ……..}
(aa + ab + ba + bb)*	String of a's and b's of even length can be obtained by concatenating any combination of the strings aa, ab, ba and bb including null, so L = {aa, ab, ba, bb, aaab, aaba, ……..}

Regular Languages

- **A regular language is a language that can be expressed with a regular expression or a deterministic or non-deterministic finite automata or state machine.**
- A **language** is a set of strings which are made up of characters from a specified alphabet, or set of symbols.
- Regular languages are a subset of the set of all strings. Regular languages are used in parsing and designing programming languages and are one of the first concepts taught in computability courses.
- These are useful for helping computer scientists to recognize patterns in data and group certain computational problems together — once they do that, they can take similar approaches to solve the problems grouped together.
- **Regular languages are a key topic in computability theory.**

Operation on Regular Languages

- **The various operations on the regular language are as follows:**

  If L_1={00,10} & L_2={01,11}

Operation	Description	Example
Union	Union of two languages L_1 and L_2 produce the set of strings which may be either in language L_1 or in language L_2or in both. $L_1 \cup L_2$= {set of string in L_1 and string in L_2}	$L_1 \cup L_2$={00, 10, 01, 11}
Concatenation	Concatenation of two languages L_1 and L_2 create a set of strings which are formed by combining the strings in L_1 with strings in L_2 (strings in L_1 should be followed by strings in L_2). L_1L_2={Set of string in L_1 followed by strings in L_2}.	L_1L_2={0001, 0011, 1001,1011}

Operation on Regular Languages

- **The various operations on the regular language are as follows:**

  If L_1={00,10} & L_2={01,11}

Kleen closure of L_1 L_1^*	Kleene closure defines zero or more appearance of input symbols in a string. It consists of an empty string Ɛ (a set of strings with 0 or more occurrences of input symbols). $L_1^*=L_1^0 \cup L_1^1 \cup L_1^2 \cup \ldots..$ $L_1^* = \bigcup_{i=0}^{\infty} L_1^i$	L_1^*={ε,00,10,1010, 0010,1000,0000,000000, 001000,….}
Positive Closure L_1^+	Positive closure indicates one or more occurrences of input symbols in a string. It eliminates empty string Ɛ(set of strings with 1or more appearance of input symbols). $L_1^+=L_1^1 \cup L_1^2 \cup \ldots..$ $L_1^+ = \bigcup_{i=0}^{\infty} L_1^i$	L_1^+={00,10,1010, 0010,1000,0000,000000, 001000,….}

Extensions of Regular Expressions

- **Kleene suggests regular expression in the 1950s with the primary operation for a union, concatenation, and Kleene closure.**

- **There is some notational extension specified that are directly in use −**
  - **One or more instance:** Unary postfix operator + displays positive closure of a regular expression and its language. It defined that if a is the regular expression, then (a) + indicates the language (L(a) +. There are two algebraic laws r∗r∗ = r+|e and r+ =rr∗r∗ = r∗r∗r relate the positive closure and Kleene closure.
  - **Zero or one instance:** Unary postfix operator? define zero or one appearance. It define that r? is similar to r|e or L(r?) = L(r) U {e}. This operator has the equal precedence and associativity as * and +./

Regular Sets and Properties of Regular Sets

- A set containing all the strings generated by a Regular Expression is known as a **Regular Set.**
- **Properties of Regular Sets:**

Property 1. *The union of two regular set is regular.*

Proof –

Let us take two regular expressions

$RE_1 = a(aa)^*$ and $RE_2 = (aa)^*$

So, L_1 = {a, aaa, aaaaa,.....} (Strings of odd length excluding Null)

and L_2 ={ ε, aa, aaaa, aaaaaa,.....} (Strings of even length including Null)

$L_1 \cup L_2$ = { ε, a, aa, aaa, aaaa, aaaaa, aaaaaa,.....}

(Strings of all possible lengths including Null)

RE ($L_1 \cup L_2$) = a^* (which is a regular expression itself)

Hence, proved.

Regular Sets and Properties of Regular Sets

- **Properties of Regular Sets:**

Property 2. *The intersection of two regular set is regular.*

Proof –

Let us take two regular expressions

$RE_1 = a(a^*)$ and $RE_2 = (aa)^*$

So, L_1 = { a,aa, aaa, aaaa, ...} (Strings of all possible lengths excluding Null)

L_2 = { ε, aa, aaaa, aaaaaa,.....} (Strings of even length including Null)

$L_1 \cap L_2$ = { aa, aaaa, aaaaaa,.....} (Strings of even length excluding Null)

RE ($L_1 \cap L_2$) = aa(aa)* which is a regular expression itself.

Hence, proved.

Regular Sets and Properties of Regular Sets

- **Properties of Regular Sets:**

Property 3. *The complement of a regular set is regular.*

Proof –

Let us take a regular expression –

RE = (aa)*

So, L = {ε, aa, aaaa, aaaaaa, ...} (Strings of even length including Null)

Complement of **L** is all the strings that is not in **L**.

So, L' = {a, aaa, aaaaa, ...} (Strings of odd length excluding Null)

RE (L') = a(aa)* which is a regular expression itself.

Hence, proved.

Regular Sets and Properties of Regular Sets

- **Properties of Regular Sets:**

Property 4. *The difference of two regular set is regular.*

Proof –

Let us take two regular expressions –

$RE_1 = a\,(a^*)$ and $RE_2 = (aa)^*$

So, L_1 = {a, aa, aaa, aaaa, ...} (Strings of all possible lengths excluding Null)

L_2 = { ε, aa, aaaa, aaaaaa,.....} (Strings of even length including Null)

$L_1 - L_2$ = {a, aaa, aaaaa, aaaaaaa, ...}

(Strings of all odd lengths excluding Null)

RE ($L_1 - L_2$) = a (aa)* which is a regular expression.

Hence, proved.

Regular Sets and Properties of Regular Sets

- **Properties of Regular Sets:**

Property 5. *The reversal of a regular set is regular.*

Proof –

We have to prove $\mathbf{L^R}$ is also regular if **L** is a regular set.

Let, L = {01, 10, 11, 10}

RE (L) = 01 + 10 + 11 + 10

L^R = {10, 01, 11, 01}

RE (L^R) = 01 + 10 + 11 + 10 which is regular

Hence, proved.

Regular Sets and Properties of Regular Sets

- **Properties of Regular Sets:**

Property 6. *The closure of a regular set is regular.*

Proof –

If L = {a, aaa, aaaaa, ...} (Strings of odd length excluding Null)

i.e., RE (L) = a (aa)*

L* = {a, aa, aaa, aaaa , aaaaa,.....} (Strings of all lengths excluding Null)

RE (L*) = a (a)*

Hence, proved.

Regular Sets and Properties of Regular Sets

- **Properties of Regular Sets:**

Property 7. *The concatenation of two regular sets is regular.*

Proof –

Let $RE_1 = (0+1)^*0$ and $RE_2 = 01(0+1)^*$

Here, $L_1 = \{0, 00, 10, 000, 010, ...\}$ (Set of strings ending in 0)

and $L_2 = \{01, 010,011,.....\}$ (Set of strings beginning with 01)

Then, $L_1 L_2 = \{001,0010,0011,0001,00010,00011,1001,10010,.....\}$

Set of strings containing 001 as a substring which can be represented by an RE − (0 + 1)*001(0 + 1)*

Hence, proved.

Identities Related to Regular Expressions

- **Given R, P, L, Q as regular expressions, the following identities hold −**

  - $\emptyset^* = \varepsilon$
  - $\varepsilon^* = \varepsilon$
  - $RR^* = R^*R$
  - $R^*R^* = R^*$
  - $(R^*)^* = R^*$
  - $RR^* = R^*R$
  - $(PQ)^*P = P(QP)^*$
  - $(a+b)^* = (a^*b^*)^* = (a^*+b^*)^* = (a+b^*)^* = a^*(ba^*)^*$
  - $R + \emptyset = \emptyset + R = R$ (The identity for union)
  - $R\,\varepsilon = \varepsilon\,R = R$ (The identity for concatenation)
  - $\emptyset\,L = L\,\emptyset = \emptyset$ (The annihilator for concatenation)
  - $R + R = R$ (Idempotent law)
  - $L\,(M + N) = LM + LN$ (Left distributive law)
  - $(M + N)\,L = ML + NL$ (Right distributive law)
  - $\varepsilon + RR^* = \varepsilon + R^*R = R^*$

Examples: Regular Expressions

- **Example 1: Write the regular expression for the language accepting all the string which are starting with 1 and ending with 0, over ∑ = {0, 1}.**

Solution:

In a regular expression, the first symbol should be 1, and the last symbol should be 0. The r.e. is as follows:

R = 1(0+1)*0

- **Example 2: Write the regular expression for the language starting and ending with a and having any combination of b's in between.**

Solution:

The regular expression will be:

R = ab*a

Examples: Regular Expressions

- **Example 3: Write the regular expression for the language starting with a but not having consecutive b's.**

Solution: The regular expression has to be built for the language:

L = {a, aba, aab, aba, aaa, abab, ...}

The regular expression for the above language is:

R = {a + ab}*

- **Example 4: Write the regular expression for the language accepting all the string in which any number of a's is followed by any number of b's is followed by any number of c's.**

Solution: As we know, any number of a's means a* any number of b's means b*, any number of c's means c*. Since as given in problem statement, b's appear after a's and c's appear after b's. So the regular expression could be:

R = a* b* c*

Examples: Regular Expressions

- **Example 5: Write the regular expression for the language over ∑ = {0} having even length of the string.**

Solution:

The regular expression has to be built for the language:

L = {ε, 00, 0000, 000000, ...}

The regular expression for the above language is:

R = (00)*

- **Example 6: Write the regular expression for the language having a string which should have at least one 0 and at least one 1.**

Solution:

The regular expression will be:

R = [(0 + 1)* 0 (0 + 1)* 1 (0 + 1)*] + [(0 + 1)* 1 (0 + 1)* 0 (0 + 1)*]

Examples: Regular Expressions

- **Example 7: Describe the language denoted by following regular expression**

  R = (b* (aaa)* b*)*

Solution:

The language can be predicted from the regular expression by finding the meaning of it. We will first split the regular expression as:

R = (any combination of b's) (aaa)* (any combination of b's)

L = {The language consists of the string in which a's appear triples, there is no restriction on the number of b's}

- **Example 8: Write the regular expression for the language L over ∑ = {0, 1} such that all the string do not contain the substring 01.**

Solution:

The Language is as follows:

L = {ε, 0, 1, 00, 11, 10, 100, ...}

The regular expression for the above language is as follows:

R = (1* 0*)

Examples: Regular Expressions

- **Example 9: Write the regular expression for the language containing the string over {0, 1} in which there are at least two occurrences of 1's between any two occurrences of 0's.**

Solution:

At least two 1's between two occurrences of 0's can be denoted by the following regular expression:

R = (0111*0)*.

- **Example 10: Write the regular expression for the language containing the string in which every 0 is immediately followed by 11.**

Solution:

The regular expectation will be:

R = (011 + 1)*

?

THE END