Unit - I | Lecture- 02

Introduction to Regular Expressions

Dr. R. Madana Mohana
Professor, Artificial Intelligence & Data Science | I/c-Head, Artificial Intelligence & Machine Learning
CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY
Hyderabad - 500 075, Telangana, INDIA
www.cbit.ac.in

Regular Expressions

- Recursive definition of Regular Expressions
- Regular Sets /Regular Language
- Identities or Algebraic laws of Regular Expressions
- Applications of Regular Expressions

Regular Expressions

- The language accepted by **Finite Automata (FA)** can be easily described by simple expressions called **Regular Expressions**.
- **Regular Expressions** are useful for representing certain sets of strings in an algebraic fashion. These describes the languages accepted by **Finite Automata (FA)**.

Recursive definition of Regular Expressions

A recursive definition of **Regular Expression** over an alphabet **Σ** is as follows:

1. Any terminal symbol '**a**' (i.e., an element of **Σ**, **a∈Σ**), an empty set **Φ** and empty string **ϵ** or **λ** are **regular expressions**.
2. The UNION of two regular expressions R_1 and R_2 written as R_1+R_2 is also a **regular expression**.

Recursive definition of Regular Expressions

3. The CONCATENATION of two regular expressions R_1 and R_2 written as R_1R_2 is also a **regular expression**.
4. The CLOSURE or ITERATION (*) of a regular expression R written as R^* is also a **regular expression**.
5. If R is a regular expression, then (R) is also a **regular expression**.

Recursive definition of Regular Expressions

6. The **regular expressions** over an alphabet **Σ** are precisely those obtained recursively by the applications of the rules 1 to 5 once or several times.

If the parenthesis are not there, then the hierarchy of operations is as follows:

i. Iteration or Closure

ii. Concatenation

iii. Union

Regular Sets

Any set represented by a **Regular Expression** is a **Regular Set**. Regular sets are the sets which are accepted by **Finite Automata (FA)**. But the actual definition is a special class of sets of strings or words over **Σ** are called **regular sets** and is defined recursively as follows:

1. Every finite sets of strings over **Σ** including the empty set **Φ** is a **regular set**.

Regular Sets

2. If **u** and **v** are two regular sets over **Σ** , then the UNION of two regular sets **u** and **v** written as **uuv** is also a **regular set**.
3. If **u** and **v** are two regular sets over **Σ** , then the CONCATENATION of two regular sets **u** and **v** written as **uv** is also a **regular set**.

Regular Sets

4. If **S** is a regular set over **Σ**, then its closure i.e., **S*** is also a **regular set**.
5. No set is regular unless it is obtained by a finite number of applications of definitions of rules 1 to 4.

Examples of Regular Sets and Regular Expressions

S. No.	Regular Sets	Regular Expressions
1	{101}	101
2	{abba}	abba
3	{01, 10}	01+10
4	{ϵ, ab}	ϵ+ ab
5	{aba, a, b, bba}	aba+a+b+bba
6	{ϵ, 0, 00, 000, …..}	ϵ+0+00+000+….. $= 0^*$
7	{1, 11, 111, …..}	1+11+111+….. = 11^*
8	{a, b}	a+b
9	$\{a, b\}^*$	$(a+b)^*$

Examples of Regular Sets and Regular Expressions

S. No.	Regular Sets	Regular Expressions
10	L_1 = {The set of all strings of 0's and 1's ending in 00}	***Solution:*** $\Sigma = \{0, 1\}$ $\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, ...\}$ $= \{\epsilon, 0, 1, 00, 01, 10, 11, 000, ...\}\ 00$ $= \mathbf{(0+1)^*00}$
11	L_2 = {The set of all strings of 0's and 1's beginning with 0 and ending with 1}	***Solution:*** $\Sigma = \{0, 1\}$ $\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, ...\}$ $= 0\{\epsilon, 0, 1, 00, 01, 10, 11, 000, ...\}\ 1$ $= \mathbf{0(0+1)^*1}$

University Previous Examination Questions on Regular Expressions

Q1. Describe the following sets by regular expressions {101}

Solution:

The set contains the string `101`. That means the language `L` can be defined as `L = {The language contains the substring 101}` .

Hence the regular expression is

= `(0+1)*101(0+1)*`

University Previous Examination Questions on Regular Expressions

Q2. Find regular expression for the following:

L ={ w/every odd position of w is 1} defined over Σ = {0, 1}

Solution:

The regular expression for given language **L**:

= 1 (odd position) 0 (even position) 1 (odd position) 0 (even position)….

= 10101010….

= (10)*

University Previous Examination Questions on Regular Expressions

Q3. Find a regular expression corresponding to each of the following subsets over {0, 1}*

a) The set of all strings containing no three consecutive 0's.

b) The set of all strings where the 10^{th} symbol from right end is 1.

c) The set of all strings over {0, 1} having even number of 0's and odd number of 1's.

d) The set of all strings over {0, 1} in which the number of occurrences are divisible by 3.

University Previous Examination Questions on Regular Expressions

Q3. Find a regular expression corresponding to each of the following subsets over $\{0, 1\}^*$

a) The set of all strings containing no three consecutive 0's.

Answer:

Regular Expression = $1^*01^*01^*+1^*(0+00+\epsilon)1^*$

b) **The set of all strings where the 10^{th} symbol from right end is 1.**

Answer:

Regular Expression =

$$(0+1)^*1(0+1)(0+1)(0+1)(0+1)(0+1)(0+1)(0+1)(0+1)(0+1)(0+1)$$

University Previous Examination Questions on Regular Expressions

c) The set of all strings over {0, 1} having even number of 0's and odd number of 1's.

Answer:

The even number of 0's can be represented as

Regular Expression = $(00)^*$

The odd number of 1's can be represented as

Regular Expression = $1(11)^*$

Hence Regular Expression = $[(0+11)(0+1)]^*+1$

University Previous Examination Questions on Regular Expressions

d) The set of all strings over {0, 1} in which the number of occurrences are divisible by 3.

Answer:

i) If the number of 0's are divisible by 3.

Regular Expression = `(1*01*01*01*)*`

The possible strings are `{0001, 0100, 010101, …..}`

ii) If the number of 1's are divisible by 3.

Regular Expression = `(0*10*10*10*)*`

The possible strings are `{01011, 1101, 101010, …..}`

Identities or Algebraic Laws of Regular Expressions

Two **Regular Expressions** **P** and **Q** are equivalent (**P ≈ Q**) if **P** and **Q** represent the same set of strings.

The following are the Identities or Algebraic Laws of Regular Expressions:

I_1: $\Phi + R = R$

I_2: $\Phi R = R\Phi = \Phi$

I_3: $\epsilon R = R\epsilon = R$

Identities or Algebraic Laws of Regular Expressions

I_4: $\epsilon^* = \epsilon$ and $\Phi^* = \epsilon$

I_5: $R + R = R$

I_6: $R^*R^* = R^*$

I_7: $RR^* = R^*R = R^*$

I_8: $(R^*)^* = R^*$

I_9: $\epsilon + RR^* = \epsilon + R^*R = R^*$

I_{10}: $(PQ)^*P = P(QP)^*$

Identities or Algebraic Laws of Regular Expressions

I_{11}: $(P + Q)^* = (P^*Q^*)^* = (P^* + Q^*)^*$

I_{12}: $(P + Q)R = PR + QR$ and $R(P + Q) = RP + RQ$

Here P, Q and R are **Regular Expressions**.

Example problems on simplification of Regular Expressions using Algebraic Laws

Example-1. Prove $\epsilon + 1^*(011)^*(1^*(011)^*)^* = (1 + 011)^*$

Solution:

L.H.S = $\epsilon + 1^*(011)^*(1^*(011)^*)^*$

Let R = $1^*(011)^*$

L.H.S = $\epsilon + RR^*$

L.H.S = R* [using I_9: $\epsilon + RR^* = \epsilon + R^*R = R^*$]

L.H.S = $(1^*(011)^*)^*$

Let P = 1, Q = 011

L.H.S = $(P^*Q^*)^*$

L.H.S = $(P + Q)^*$ [using I_{11}: $(P + Q)^* = (P^*Q^*)^* = (P^* + Q^*)^*$]

L.H.S = $(1 + 011)^*$ = R.H.S

Hence proved.

Example problems on simplification of Regular Expressions using Algebraic Laws

Example-2. Prove $(1 + 00^*1) + (1 + 00^*1)(0 + 10^*1)^*(0 + 10^*1) = 0^*1(0 + 10^*1)^*$

Solution:

L.H.S = $(1 + 00^*1) + (1 + 00^*1)(0 + 10^*1)^*(0 + 10^*1)$

L.H.S = $(1 + 00^*1)(\epsilon + (0 + 10^*1)^*(0 + 10^*1))$ // Taking $(1 + 00^*1)$ common at left side

Let $R = 0 + 10^*1$

L.H.S = $(1 + 00^*1)(\epsilon + R^*R)$

L.H.S = $(1 + 00^*1)(R^*)$ [using I_9: $\epsilon + RR^* = \epsilon + R^*R = R^*$]

L.H.S = $(1 + 00^*1)(0 + 10^*1)^*$

L.H.S = $(\epsilon + 00^*)1(0 + 10^*1)^*$ // Taking 1 common from $(1 + 00^*1)$ at right side

Let $R = 0$

L.H.S = $(\epsilon + RR^*)1(0 + 10^*1)^*$

L.H.S = $R^*1(0 + 10^*1)^*$ [using I_9: $\epsilon + RR^* = \epsilon + R^*R = R^*$]

L.H.S = $0^*1(0 + 10^*1)^*$ = R.H.S

Hence proved.

Applications of Regular Expressions

- A **`Regular Expression`** that gives a picture of the pattern we want to recognize is the medium of choice for applications that search for patterns in text.
- The **`Regular Expressions`** are then compiled, behind the scenes, into **`deterministic`** or **`nondeterministic automata`**, which are then simulated to produce a program that recognize patterns in text.

Applications of Regular Expressions

The following are the applications of **`Regular Expressions:`**

- Regular Expressions in UNIX
- Lexical Analysis
- Finding Patterns in Text

Applications of Regular Expressions

1. Regular Expressions in UNIX:

There are various UNIX notations used for regular expressions.

- The symbol `.` **(dot)** stands for "any character".
- The sequence `[a1a2a3….ak]` stands for the regular expression

  $a_1+a_2+a_3+\ldots.+a_k$.
- The range of characters can be specified using square brackets.

  For example: `[a-z]` denotes set of lower case letters.
- For matching with some special character a backslash `(\)` is used.

  For example: `\-` means match with character `-`.

Applications of Regular Expressions

1. Regular Expressions in UNIX: cont.

- There are Special notations for several of the most common classes of characters.

  For example: `[:digit:]` is the set of ten digits, the same as `[0-9]`.

  The `[:alpha:]` stands for any alphabetic character, as same as `[A-Za-Z]`. The `[:alnum:]` stands for the digits and letters (alphabetic and numeric characters), as same as `[A-Za-Z0-9]`

Applications of Regular Expressions

1. Regular Expressions in UNIX: cont.

In addition, there are several operators that are used in UNIX regular expressions.

- The operator `|` is used in place of `+` to denote union.
- The operator `?` Means "zero or one of." Thus, `R?` in UNIX is the same as $\epsilon + R$.
- The operator `+` means "one or more of". Thus, `R+` in UNIX is shorthand for RR^* in our notation.
- The operator `{n}` means "`n` copies of". Thus `R{5}` in UNIX is shorthand for `RRRRR`.

Applications of Regular Expressions

2. Lexical Analysis:

- Compiler uses **`lexical analyzer`** in the process of compilation.
- The task of **`lexical analyzer`** is to scan the input program and groups them into tokens.
- The tokens are defined by a regular expression for each pattern.

  For example: an **`identifier`** is a token which has a pattern "An alphabet followed by any number of alphanumeric character".

  The corresponding regular expression is **`letter(letter + digit)*`**

  i.e., **`[A-Za-z] [A-Za-z + 0-9]*`**

Applications of Regular Expressions

2. Lexical Analysis: cont.

Example-1: `Finite Automata (FA)` for the `regular expression` of token identifier:

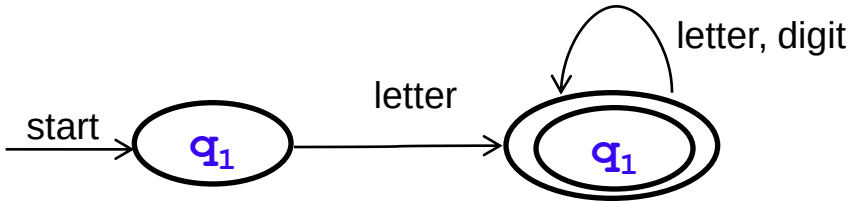

Fig. FA for the token **identifier**

Applications of Regular Expressions

2. Lexical Analysis: cont.

Example-2: `Finite Automata (FA)` for the `regular expression` of token keywords:

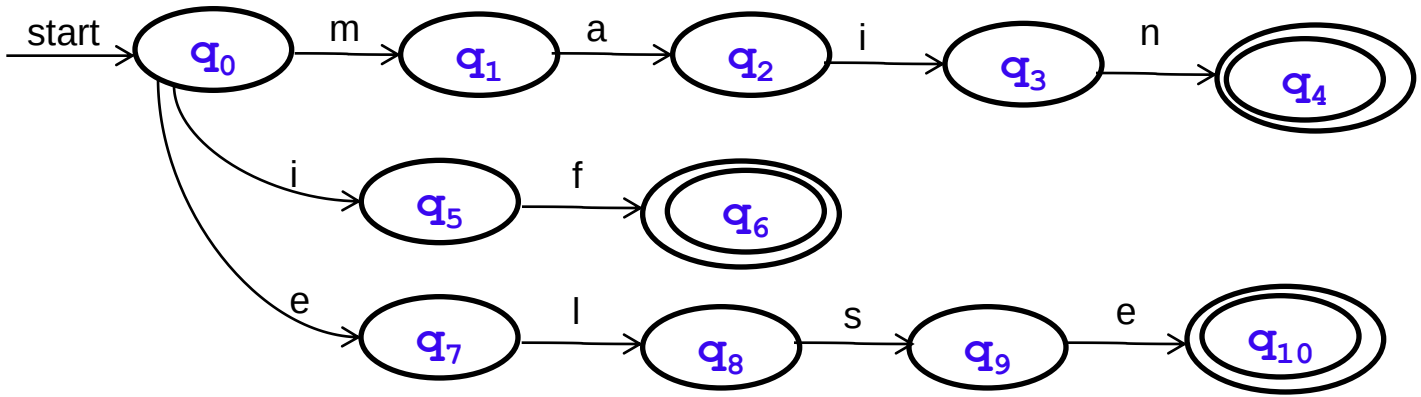

Fig. FA for the token **keywords "main", "if", "else"**

Applications of Regular Expressions

3. Finding Patterns in Text:

- **`Regular expressions`** are useful notations used for finding the patterns from given text .
- A large class of pattern can be identified by regular expression.

For example:

The sting ending with digits is a pattern in the given text which can be recognized by the following regular expression:

```
[a-zA-Z]+[0-9]+
```

Applications of Regular Expressions

3. Finding Patterns in Text: cont.

- **`Regular expressions`** are useful in finding strings corresponding to some pattern to be searched which are defined in unstructured format.
- There are various applications. They are:
  - To scan a very large number of web pages and to detect addresses.
  - To create a mailing list
  - To classify the business by their location and to answer the queries.
  - To search files in the system based on pattern.

Lecture- 02
Introduction to Regular Expressions
Summary

- Recursive definition of Regular Expressions
- Regular Sets /Regular Language
- Identities or Algebraic laws of Regular Expressions
- Applications of Regular Expressions