UCS1602:
COMPILER DESIGN

Specification of tokens

Session Objectives

- To learn concepts specification of tokens
- To study about the regular expressions

Session Outcomes

- At the end of this session, participants will be able to
  - Understand the concepts of regular expression

Outline

- Specification of tokens
- Regular expressions

Specification of Tokens

Specification of tokens

- Alphabet or Character Class
  - Σ is a finite set of symbols (characters)
  - {0,1} is a binary alphabet
- String or Sentence or word
  - A *string s* is a finite sequence of symbols from Σ
    - $|s|$ denotes the length of string *s*
    - ε denotes the empty string, thus $|\varepsilon| = 0$
    - banana → |banana|=6
- Language
  - A *language* is a specific set of strings over some fixed alphabet Σ
  - Σ={0,1}
  - L={0,1,00,11,01,10,000,001,010,011,…}

Specification of tokens Cont...

- **Prefix of s**
  - A string obtained by removing 0 or more trailing symbols of s
  - b, ba, ban, bana, banan, banana
- **Suffix of s**
  - A string formed by deleting 0 or more leading symbols of s
  - a, na, ana, nana ...
- **Substring of s**
  - A string obtained by removing the suffix and prefix from s
  - ana, nan etc
- **Proper prefix and Proper Suffix**
  - Any prefix or suffix other than the string itself
  - b, ba, a, nana ...
- **Subsequence of s**
  - Any string formed by deleting zero or more not necessarily contiguous symbols from s.
  - baaa, ann...

Language Operations

- *Union*

  $$L \cup M = \{s \mid s \in L \text{ or } s \in M\}$$

- *Concatenation*

  $$LM = \{xy \mid x \in L \text{ and } y \in M\}$$

- *Kleene closure*

  $$L^* = \cup_{i=0,\ldots,\infty} L^i$$

- *Positive closure*

  $$L^+ = \cup_{i=1,\ldots,\infty} L^i$$

Regular Expressions

Rules for Regular Expression

- ε is a regular expression, $L(\varepsilon) = \{\varepsilon\}$
- If a is a symbol in Σ then a is a regular expression, $L(a) = \{a\}$
- (r) | (s) is a regular expression denoting the language $L(r) \cup L(s)$
- (r)(s) is a regular expression denoting the language L(r)L(s)
- (r)* is a regular expression denoting (L(r))*
- (r) is a regular expression denoting L(r)

Ex : Identifier → letter (letter |digit) *

Precedence

* (Closure) has the higher precedence

. (Concatenation) has the next higher precedence

| (Union) has the least precedence

Remove unnecessary parentheses

(a)|((b)*c) →a | b * c

Σ={a,b}

RE a|b   {a,b}

(a/b)(a/b) {aa,ab,ba,bb}

(a/b)*   ?   a/a*b   ?

If 2 r.e ***r*** and ***s*** denote the same language then ***r*** and ***s*** are said to be **equivalent** ie. ***r=s*** ex. ***a/b = b/a***

Regular definitions

- Regular definitions introduce a naming convention:

  $d_1 \rightarrow r_1$

  $d_2 \rightarrow r_2$

  …

  $d_n \rightarrow r_n$

  where each r_i is a regular expression over

  $\Sigma \cup \{d_1, d_2, \ldots, d_{i-1}\}$

- Any d_j in r_i can be textually substituted in r_i to obtain an equivalent set of definitions

Regular definitions Cont…

- Example:

$$\textbf{letter} \rightarrow \texttt{A} \mid \texttt{B} \mid \ldots \mid \texttt{Z} \mid \texttt{a} \mid \texttt{b} \mid \ldots \mid \texttt{z}$$
$$\textbf{digit} \rightarrow \texttt{0} \mid \texttt{1} \mid \ldots \mid \texttt{9}$$
$$\textbf{id} \rightarrow \textbf{letter} \; (\, \textbf{letter} \mid \textbf{digit} \,)^{*}$$

- Regular definitions are not recursive:

$$\textbf{digits} \rightarrow \textbf{digit digits} \mid \textbf{digit} \qquad \textit{wrong!}$$

Notational Shorthand

- One or more instances: (r)+
- Zero of one instances: r?
- Character classes: [abc]

$$r^{+} = rr^{*}$$

$$r? = r \mid \varepsilon$$

$$[\mathbf{a\text{-}z}] = \mathbf{a} \mid \mathbf{b} \mid \mathbf{c} \mid \ldots \mid \mathbf{z}$$

Notational Shorthand

- letter_ -> [A-Za-z_]
- digit   -> [0-9]
- id   -> letter_(letter|digit)*

- Examples:
  digit → **[0-9]**
  num → **digit**$^{+}$ (**.** **digit**$^{+}$)? (**E** (**+** | **-**)? **digit**$^{+}$)?

Summary

- Alphabet
- String
- Language
- Language operations
- Regular expression

Check your understanding?

1. Write the language generated by the following regular expression.

   (i) (a/b)*

   (ii) (a*/b*)*

2. Write the regular expression to generate date in the following format

   DD-MM-YYYY