LEX2: Regular Expressions

Lexical Analysis

CMPT 379: Compilers

Instructor: Anoop Sarkar

anoopsarkar.github.io/compilers-class

Regular Languages

- The set of regular languages: each element is a regular language
  - $R = \{R_1, R_2, \ldots, R_n, \ldots\}$
- Each regular language is an example of a (formal) language, i.e. a set of strings

  $R_1 = \{a\}$, $R_2 = \{a, aa, aaa, \ldots\}$, $R_3 = \{b\}$,
  $R_4 = \{ba, ab\}$, $R_5 = \{\varepsilon, b, bb, bbb, \ldots\}$, …

Regular Expressions: Definition

- Meaning function L maps syntax to semantics
  - L(r) = Meaning of regexp r (a regular language)
  - a^* = {ε, a, aa, aaa, …}
  - ϵ = ‘’
  - c = c
  - $A|B = A \cup B$
  - AB = ab | a ∈ A and b ∈ B
  - A^2 = xy | x ∈ A and y ∈ A
  - $A^* = A^0 \cup A^1 \cup A^2 \cup A^3 \ldots$

Regular Expressions: Definition

- Meaning function L maps syntax to semantics
  - L(r) = Meaning of regexp r (a regular language)
  - L(a*) = {ε, a, aa, aaa, …}
  - $L(\epsilon)$ = {''}
  - L(c) = {c}
  - L(A|B) = L(A) ∪ L(B)
  - L(AB) = { ab | a ∈ L(A) and b ∈ L(B) }
  - $L(A^2)$ = { xy | x ∈ L(A) and y ∈ L(A) }
  - L(A*) = $L(A^0) \cup L(A^1) \cup L(A^2) \cup L(A^3)$ ...

Regular Expressions

- Why use meaning function?
  - Make clear what is syntax and what is semantics
  - Allow us to consider notation as a separate issue

Regular Expressions

- Why use meaning function?
  - Make clear what is syntax and what is semantics
  - Allow us to consider notation as a separate issue
    - Identifier:
      - Sequence of letters r_1
      - Sequence of letters and digits r_2

Integer: a non-empty sequence of digits

Integer: a non-empty sequence of digits

(0|1|2|3|4|5|6|7|8|9)

Integer: a non-empty sequence of digits

digit = (0|1|2|3|4|5|6|7|8|9)

Integer: a non-empty sequence of digits

digit = (0|1|2|3|4|5|6|7|8|9)

{digit}*

Integer: a non-empty sequence of digits

digit = (0|1|2|3|4|5|6|7|8|9)

{digit}{digit}*

Integer: a non-empty sequence of digits

digit = (0|1|2|3|4|5|6|7|8|9)

{digit}{digit}*

{digit}+

Identifier: sequence of letters or digits, starting with a letter

Identifier: sequence of letters or digits, starting with a letter

digit = (0|1|2|3|4|5|6|7|8|9)

Identifier: sequence of letters or digits, starting with a letter

digit = (0|1|2|3|4|5|6|7|8|9)

letter = (a|b|c|…|z|A|B|…|Z)

Identifier: sequence of letters or digits, starting with a letter

digit = (0|1|2|3|4|5|6|7|8|9)

letter = (a|b|c|…|z|A|B|…|Z)

[a-z]

Identifier: sequence of letters or digits, starting with a letter

digit = (0|1|2|3|4|5|6|7|8|9)

letter = (a|b|c|...|z|A|B|...|Z)

[a-zA-Z]

Identifier: sequence of letters or digits, starting with a letter

digit = [0-9]

letter = [a-zA-Z]

Identifier: sequence of letters or digits, starting with a letter

digit = [0-9]

letter = [a-zA-Z]

{letter}

Identifier: sequence of letters or digits, starting with a letter

digit = [0-9]

letter = [a-zA-Z]

{letter}({letter}|{digit})*

Whitespace: a non-empty sequence of blanks, newlines and tabs

Whitespace: a non-empty sequence of blanks, newlines and tabs

" "

Whitespace: a non-empty sequence of blanks, newlines and tabs

" "|"\t"

Whitespace: a non-empty sequence of blanks, newlines and tabs

" "|"\t"|"\n"

Whitespace: a non-empty sequence of blanks, newlines and tabs

(" "|"\t"|"\n")+

Regular Expressions: Definition

- Regular expressions describe many language
  - Email username@cs.sfu.ca
  - Filenames
  - Phone numbers
- Regular languages are a language specification
  - We still need an implementation
- Given a string s and a regexp R:

$$s \in L(R)\ ?$$

Definition of Numbers

```
    digit   =   [0-9]
    digits  =  [0-9]+
opt_frac  =  ("."{digits})?
opt_exp  =  (E(\+|\-)?{digits})?
      num  =  {digits}{opt_frac}{opt_exp}
```

Definition of Numbers

```
   digit   =  [0-9]
   digits  =  [0-9]+
opt_frac   =  ("."{digits})?
opt_exp    =  (E(\+|\-)?{digits})?
     num   =  {digits}{opt_frac}{opt_exp}
```

345 , 345.04 , 2.14+e

Regular Expressions for Lexical Analysis

- Write a regexp for the lexemes of each token class
  - Integer = digit+
  - Identifier = letter(letter|digit)+
  - OpenPar = '('
  - . . .
- Construct R, matching all lexemes for all tokens.
  - R = Integer | identifier | . . .

    $= R_1 \mid R_2 \mid R_3 \mid \ldots$

Lexical Analyzer

1. Let input be $x_1\ x_2\ \ldots x_n$

   For $1 \leq i \leq n$ check

   $x_1 \ldots x_i \in L(R)$

   $R = R_1 \mid R_2 \mid \ldots \mid R_n$

2. If success, then we know that

   $x_1 \ldots x_i \in L(R_j)$ for some j

3. Remove $x_1 \ldots x_i$ form input and go to (1)

Lexical Analyzer

- How much input is used ?

$x_1 \ldots x_i \in L(R)$

$x_1 \ldots x_j \in L(R)$

$i \neq j$

Maximal Munch

$\overbrace{x_1\, x_2\, x_3}\, x_4\, x_5\, x_6\, x_7 \ldots$

$\underbrace{x_1\, x_2\, x_3\, x_4}$

==

T_ASSIGN **T_EQ**

Lexical Analyzer

- Which token is used?

$x_1 \ldots x_i \in L(R)$

$x_1 \ldots x_i \in L(R_j)$

$x_1 \ldots x_i \in L(R_k)$

Choose the one listed first!

$R = R_1 \mid R_2 \mid . . . \mid R_n$

IF = if

identifier = letter(letter|digit)*

if $\in$ L(IF)

L(Identifier)

Lexical Analyzer

- What if no rule matches?

$$x_1 \ldots x_i \notin L(R)$$

Error = all strings not in specification

Put it last in priority

Regexps in Lexical Analysis

- Regular expressions are a concise notation for string patterns
- Use in lexical analysis requires small extensions
  - Resolve ambiguities
  - Handle errors
- Good Algorithm known
  - Require only single pass over the input
  - Few operations per character (lookup table)