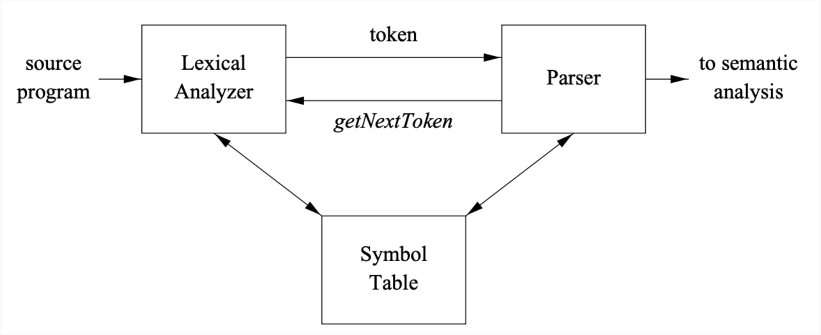
Compiladores: Análise Léxica

Christiano Braga

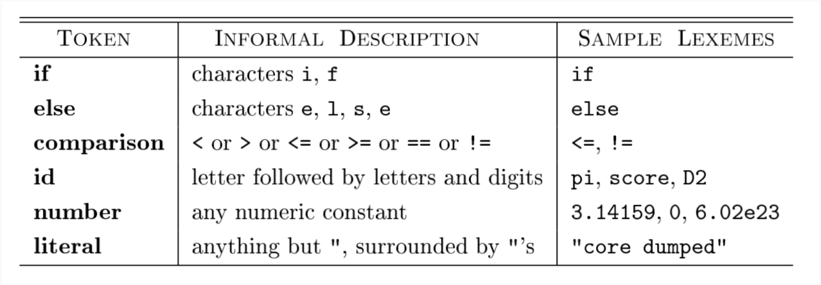
Universidade Federal Fluminense

Fevereiro 2021

Analisador léxico

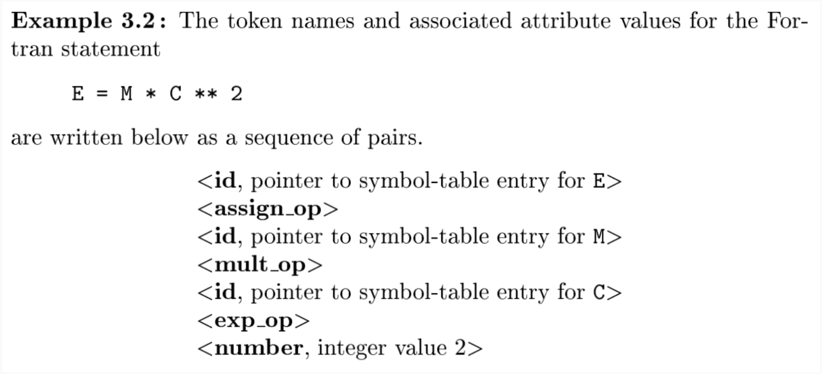


Tokens



TOKEN	INFORMAL DESCRIPTION	SAMPLE LEXEMES
if	characters `i`, `f`	`if`
else	characters `e`, `l`, `s`, `e`	`else`
comparison	`<` or `>` or `<=` or `>=` or `==` or `!=`	`<=`, `!=`
id	letter followed by letters and digits	`pi`, `score`, `D2`
number	any numeric constant	`3.14159`, `0`, `6.02e23`
literal	anything but `"`, surrounded by `"`'s	`"core dumped"`

Exemplo



Example 3.2 : The token names and associated attribute values for the Fortran statement

```
E = M * C ** 2
```

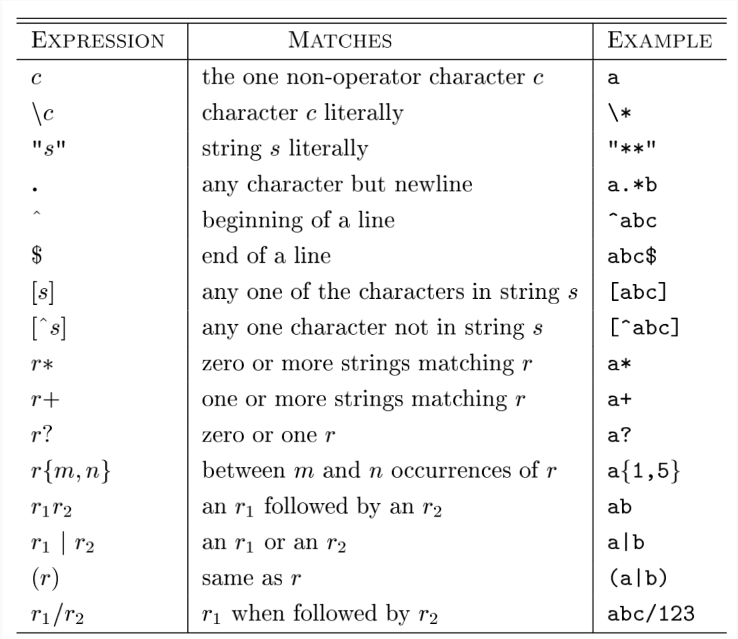
are written below as a sequence of pairs.

<**id**, pointer to symbol-table entry for `E`>
<**assign_op**>
<**id**, pointer to symbol-table entry for `M`>
<**mult_op**>
<**id**, pointer to symbol-table entry for `C`>
<**exp_op**>
<**number**, integer value 2>

Expressões regulares I

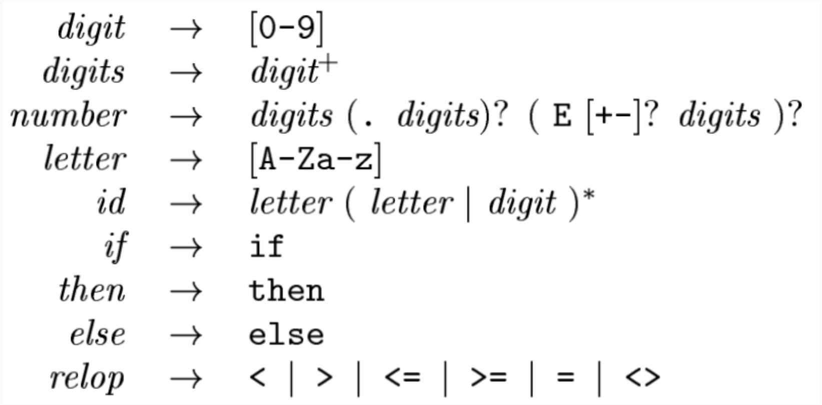
- Reconhecedores: AFD $\equiv$ AFN $\equiv$ AFN$_\epsilon$ $\equiv$ Expressões regulares
- Geradores: Gramáticas lineares

Expressões regulares II



EXPRESSION	MATCHES	EXAMPLE
c	the one non-operator character c	`a`
$\backslash c$	character c literally	`\*`
"s"	string s literally	`"**"`
.	any character but newline	`a.*b`
^	beginning of a line	`^abc`
$	end of a line	`abc$`
$[s]$	any one of the characters in string s	`[abc]`
$[\hat{}s]$	any one character not in string s	`[^abc]`
$r*$	zero or more strings matching r	`a*`
$r+$	one or more strings matching r	`a+`
$r?$	zero or one r	`a?`
$r\{m,n\}$	between m and n occurrences of r	`a{1,5}`
r_1r_2	an r_1 followed by an r_2	`ab`
$r_1 \mid r_2$	an r_1 or an r_2	`a|b`
(r)	same as r	`(a|b)`
r_1/r_2	r_1 when followed by r_2	`abc/123`

Expressões regulares em ação



digit	$\rightarrow$	`[0-9]`
digits	$\rightarrow$	$digit^{+}$
number	$\rightarrow$	*digits* (. *digits*)? (`E` `[+-]`? *digits*)?
letter	$\rightarrow$	`[A-Za-z]`
id	$\rightarrow$	*letter* (*letter* | *digit*)*
if	$\rightarrow$	`if`
then	$\rightarrow$	`then`
else	$\rightarrow$	`else`
relop	$\rightarrow$	`<` | `>` | `<=` | `>=` | `=` | `<>`

Um analisador léxico em Python 3 com PLY I

```
# --------------------------------------------------------
# Dragon book - Exercise 3.5.1
# --------------------------------------------------------
import ply.lex as lex

reserved = {
    'if' : 'IF',
    'then' : 'THEN',
    'else' : 'ELSE'
}
```

Um analisador léxico em Python 3 com PLY II

```python
# List of token names.   This is always required
tokens = [
    'LT',
    'LE',
    'EQ',
    'NE',
    'GE',
    'GT',
    'ID',
    'NUMBER',
    'RELOP',
] + list(reserved.values())
```

Um analisador léxico em Python 3 com PLY III

```
# A string containing ignored characters
# (spaces, tabs and newline)
t_ignore  = ' \t\n'

def t_LE(t):
    r'<='
    t.type = 'RELOP'
    t.value = 'LE'
    return t
```

Um analisador léxico em Python 3 com PLY IV

```python
def t_ID(t):
    r'[a-zA-Z][a-zA-Z0-9]*'
    # Check for reserved words
    t.type = reserved.get(t.value,'ID')
    return t
```

Um analisador léxico em Python 3 com PLY V

```
# A regular expression rule with some action code
def t_NUMBER(t):
    r'\d+'
    t.value = int(t.value)
    return t
```

Um analisador léxico em Python 3 com PLY VI

```
# Error handling rule
def t_error(t):
    print("Illegal character '%s'" % t.value[0])
    t.lexer.skip(1)
```

Um analisador léxico em Python 3 com PLY VII

```
class Ex351Lexer:
    def __init__(self):
        self.lexer = lex.lex()

    def setData(self, data):
        self.data = data
        self.lexer.input(data)
```

Um analisador léxico em Python 3 com PLY VIII

```
def tokenize(self):
    tokens = []
    while True:
        tok = self.lexer.token()
        if not tok:
            break      # No more input
        tokens.append(tok)
    return tokens
```

Um analisador léxico em Python 3 com PLY IX

```
if __name__ == '__main__':
    lex = Ex351Lexer()
    # lex.setData("if")
    lex.setData("if x then 3 <= 4 else 20 >= 1")
    print(lex.tokenize())
```

Um analisador léxico em Python 3 com PLY X

```
$ python3 examplelexer.py
[LexToken(IF,'if',1,0), LexToken(ID,'x',1,3),
LexToken(THEN,'then',1,5), LexToken(NUMBER,3,1,10),
LexToken(RELOP,'LE',1,12), LexToken(NUMBER,4,1,15),
LexToken(ELSE,'else',1,17), LexToken(NUMBER,20,1,22),
LexToken(RELOP,'GE',1,25), LexToken(NUMBER,1,1,28)]
```