Relatório de Compiladores

Rafael Carreira, Bruno Carvalho
26 de janeiro de 2017

Analisador Sintático

Utilizamos o método ascendente para fazer o analisador sintático, com auxílio do software GNU Bison. A gramática livre de contexto é definida pela tupla:

$$G = (V,T,P,S)$$

onde V é o conjunto de símbolos não terminais, T é o alfabeto da linguagem, P é o conjunto de regras de produção e S é o símbolo de partida.

As definições de cada elemento da tupla são apresentadas abaixo:

V = { INICIO, PARAM_DECL, MAIS_PARAM_DECL, CORPO_PROG, DECLARACAO, NOVA_DECL, COMANDO, ATRIBUICAO, EXPRESSAO, EXPRESSAO_COMP, CONDICIONAL, ELSE_DECL, CONDISSAO, MAIS_COND, REPETICAO, CHAMADA, LISTA_PARAM, MAIS_PARAM
}

T = { A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, W, X, Y, Z, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -, *, /, %, {, }, =, <, >, (,), ,, ;, ', ”
}

P = {

```
1  INICIO → TIPO IDENTIFICADOR ABRE_PAR PARAM_DECL FECHA_PAR ABRE_CHA CORPO_PROG FECHA_CHA INICIO
2         | TIPO IDENTIFICADOR ABRE_PAR PARAM_DECL FECHA_PAR ABRE_CHA CORPO_PROG FECHA_CHA

3  PARAM_DECL → TIPO IDENTIFICADOR MAIS_PARAM_DECL
4             | %empty

5  MAIS_PARAM_DECL → COMMA TIPO IDENTIFICADOR MAIS_PARAM_DECL
6                  | %empty

7  CORPO_PROG → DECLARACAO CORPO_PROG
8             | DECLARACAO
9             | COMANDO CORPO_PROG
10            | COMANDO

11 DECLARACAO → TIPO IDENTIFICADOR NOVA_DECL
12            | TIPO IDENTIFICADOR ATRIB EXPRESSAO NOVA_DECL

13 NOVA_DECL → SEMI_COLON
14           | COMMA IDENTIFICADOR NOVA_DECL
15           | COMMA IDENTIFICADOR ATRIB EXPRESSAO NOVA_DECL

16 COMANDO → ATRIBUICAO
17         | CONDICIONAL
18         | REPETICAO
19         | CHAMADA
20         | RETURN ABRE_PAR EXPRESSAO FECHA_PAR SEMI_COLON

21 ATRIBUICAO → IDENTIFICADOR ATRIB EXPRESSAO SEMI_COLON
22            | IDENTIFICADOR ATRIB STRING SEMI_COLON

23 EXPRESSAO → EXPRESSAO OPERADOR_ARIT EXPRESSAO_COMP
24           | EXPRESSAO_COMP

25 EXPRESSAO_COMP → ABRE_PAR EXPRESSAO FECHA_PAR
26                | INTEIRO
27                | REAL
28                | IDENTIFICADOR

29 CONDICIONAL → IF ABRE_PAR CONDISSAO FECHA_PAR ABRE_CHA CORPO_PROG FECHA_CHA ELSE_DECL

30 ELSE_DECL → ELSE ABRE_CHA CORPO_PROG FECHA_CHA
31           | ELSE CONDICIONAL
32           | %empty

33 CONDISSAO → EXPRESSAO OPERADOR_REL EXPRESSAO MAIS_COND
34           | EXPRESSAO

35 MAIS_COND → OPERADOR_LOG CONDISSAO
36           | %empty

37 REPETICAO → WHILE ABRE_PAR CONDISSAO FECHA_PAR ABRE_CHA CORPO_PROG FECHA_CHA
38           | FOR ABRE_PAR ATRIBUICAO SEMI_COLON CONDISSAO SEMI_COLON EXPRESSAO FECHA_PAR ABRE_CHA CORPO_PROG FECHA_CHA

39 CHAMADA → IDENTIFICADOR ABRE_PAR LISTA_PARAM FECHA_PAR SEMI_COLON
40         | IDENTIFICADOR ATRIB IDENTIFICADOR ABRE_PAR LISTA_PARAM FECHA_PAR SEMI_COLON

41 LISTA_PARAM → EXPRESSAO MAIS_PARAM
42             | STRING MAIS_PARAM
43             | %empty

44 MAIS_PARAM → COMMA EXPRESSAO
45            | COMMA STRING MAIS_PARAM
46            | %empty
```

}

Utilização

Para criar o analisador entre na pasta e execute o comando “make”. É necessário que as ferramentas “flex” e “bison” estejam instaladas:

```
cd src
make
```

Para utilizar o analisador é só executar o programa “analisador” passando o arquivo a ser analisado como entrada:

```
./analisador < textoDeEntrada.txt
```

Para gerar a descrição do autômato da análise sintática em formato HTML (tal como o incluso neste pacote) é só executar a diretiva “html” do comando “make”. Para a geração da descrição em HTML é necessário a instalação do programa “xsltproc”.

```
make html
```

Depois é só abrir o arquivo gerado usando algum visualizador de páginas web, como por exemplo o “firefox”:

```
firefox ansin.html
```

Para limpar os arquivos temporários resultante das compilações é só executar a diretiva “clean” do comando “make”:

```
make clean
```