Universidade Federal de Juiz de Fora - UFJF - Departamento de Ciência da Computação

Teoria dos Compiladores - 2021/1

Alunos: Arthur de Freitas Dornelas 201735004
Vinicius da Cruz Soranço 201735003

1. Introdução

Nesta etapa do trabalho, foi desenvolvido um Analisador sintático (parser) para a linguagem **lang**, dessa forma, foi utilizado a linguagem Java e o auxílio da ferramenta ANTLR 4. Como já vimos, essa ferramenta gera automaticamente os arquivos que são necessários para a implementação, facilitando as etapas que são mais repetitivas e simplesmente transforma o arquivo da linguagem em tokens disponíveis, gerando também algumas funções auxiliares para ajudar aos acessos dos tokens, além de ser muito mais eficiente para a evolução do compilador.

2. Analisador Sintático

Para gerar as classes relacionadas ao Analisador Sintático, precisamos primeiro definir um Analisador Léxico, proposto no trabalho anterior, e a partir dele gerar os tokens da linguagem. Com o analisador Léxico e os tokens em mãos, precisamos apenas definir uma gramática livre de contexto para a linguagem **lang,** para que o Analisador Sintático funcione corretamente. Para isso, foi incluído no arquivo **Lang.g4** a seguinte gramática:

```
prog: (data)* (func)*;

data: DATA TYPE OPEN_CURLY_BRACER (decl)* CLOSE_CURLY_BRACER;

decl: ID DOUBLE_COLON type SEMICOLON;

func: ID OPEN_PARENTESIS (params)? CLOSE_PARENTESIS (COLON type (COMMA type)*)? OPEN_CURLY_BRACER (cmd)* CLOSE_CURLY_BRACER;

params: ID DOUBLE_COLON type (COMMA ID DOUBLE_COLON type)*;

type: type OPEN_BRACKET CLOSE_BRACKET
    | btype;

btype: BTYPE
     | TYPE;

cmd: OPEN_CURLY_BRACER (cmd)* CLOSE_CURLY_BRACER
   | IF OPEN_PARENTESIS exp CLOSE_PARENTESIS cmd
   | IF OPEN_PARENTESIS exp CLOSE_PARENTESIS cmd ELSE cmd
   | ITERATE OPEN_PARENTESIS exp CLOSE_PARENTESIS cmd
   | READ lvalue SEMICOLON
```

```
    | PRINT exp SEMICOLON
    | RETURN exp (COMMA exp)* SEMICOLON
    | lvalue ATTRIBUTION exp SEMICOLON
    | ID OPEN_PARENTESIS (exps)? CLOSE_PARENTESIS (LESS_THAN (COLON)? lvalue
(COMMA lvalue)* (COLON)? MORE_THAN)? SEMICOLON;


exp: exp AND exp
    | rexp ;

rexp:  rexp LESS_THAN aexp
     | rexp MORE_THAN aexp
     | rexp EQUAL aexp
     | rexp NOT_EQUAL aexp
     | aexp;

aexp:  aexp PLUS mexp
     | aexp MINUS mexp
     | mexp;

mexp:  mexp MULT sexp
    | mexp DIV sexp
    | mexp MOD sexp
    | sexp;

sexp:  NOT sexp
    | MINUS sexp
    | TRUE
    | FALSE
    | NULL
    | INT
    | FLOAT
    | CHAR
    | LITERAL
    | pexp;

pexp: lvalue
    | OPEN_PARENTESIS exp CLOSE_PARENTESIS
    | NEW type (OPEN_BRACKET exp CLOSE_BRACKET)?
        | ID OPEN_PARENTESIS (exps)? CLOSE_PARENTESIS (OPEN_BRACKET exp
CLOSE_BRACKET)?;

lvalue: ID
      | lvalue OPEN_BRACKET exp CLOSE_BRACKET
      | lvalue ACCESSOR ID;
```

Com isso feito, podemos realizar a geração do código utilizando a ferramenta citada.

Vale ressaltar que a ferramenta ANTLR 4 reescreve automaticamente a regra para ser não recursivo à esquerda e não ambíguo. O mesmo utiliza uma variação da estratégia de implementação *ALL(*)*, visto que ele reescreve a regra para que não exista recursão à esquerda, o que permite a utilização desta estratégia de implementação. Ao contrário de analisadores *LL(k)* e *LL(*)*, analisadores *ALL(*)* sempre escolhem a primeira alternativa que leva à uma análise válida. Todas as gramáticas não recursivas à esquerda são, portanto, *ALL(*)*. Em vez de confiar na análise gramatical estática, um analisador *ALL(*)* se adapta às sentenças de entrada apresentadas a ele no *parsetime*.

3. Compilação

Para executar o analisador léxico basta entrar na raiz do projeto e rodar o *build.sh* pela linha de comando ou executar os seguintes comandos em sequência:

- *java -jar lang/parser/grammar/antlr-4.8-complete.jar -o lang/parser/grammar/antlr -package lang.parser.grammar.antlr -visitor lang/parser/grammar/Lang.g4 -Xexact-output-dir*

- *javac -cp .:lang/parser/grammar/antlr-4.8-complete.jar lang/LangCompiler.java*

- *java -cp .:lang/parser/grammar/antlr-4.8-complete.jar lang/LangCompiler -bs*

4. Referências

https://www.antlr.org/papers/allstar-techreport.pdf

https://tomassetti.me/antlr-mega-tutorial/