Compilador MiniJava

Trabalho de Compiladores
Ciência da Computação - UFF

Gustavo Dauer
Igor Galvão
Maykom Campos
Vinícius Brum

Índice

Introdução

- Optamos por usar a linguagem Java para o trabalho, devido a maior familiaridade da maior parte do grupo.

- Ferramentas escolhidas: JFlex e Java Cup.
  - Documentação abrangente
  - Bastante difundidas

Introdução

- Utilizamos o JFlex na sua versão 1.4.3 instalado do repositório oficial do Ubuntu 14.04
- E o Java Cup em sua versão 0.11a beta, também instalado do repositório oficial do Ubuntu 14.04
- Utilizando a versão do repositório, fizemos a geração de código por chamadas aos respectivos programas a partir do terminal

JFlex

- O JFlex é um gerador de scanner para Java, escrito em Java.
- Aceita como entrada um conjunto de expressões regulares e ações
- Produz como saída código java para aceitar entradas correspondendo às expressões regulares correspondentes e executar determinadas ações correspondentes à expressão lida.

Geração do Scanner

- Explicação do código de entrada para o JFlex

Geração do Scanner

```
/*-* * LEXICAL FUNCTIONS: */


%cup
%line
%column
%unicode
%class MiniJavaLexer
%{
    private Token createToken(String val) {
        Token tk = new Token(val);
        return tk;
    }
%}
```

Geração do Scanner

/*-* * PATTERN DEFINITIONS: */

comentario = \/[^\/]*\/
letter = [a-zA-Z]
digit = [0-9]
identificador = {letter}({letter}|{digit}|_)*
inteiro = [0-9]+
whitespace = [\n\t\r\f]
newline = [\r\n]

Geração do Scanner

```
/** * LEXICAL RULES: */

""                  { return new Symbol(sym.VOID,createToken(yytext())); }
"{"                  { return new Symbol(sym.ACH,createToken(yytext())); }
"}"                  { return new Symbol(sym.FCH,createToken(yytext())); }
"."                  { return new Symbol(sym.PTO,createToken(yytext())); }
"!"                  { return new Symbol(sym.NEG,createToken(yytext())); }
(...)
class                { return new Symbol(sym.CLA,createToken(yytext())); }
boolean              { return new Symbol(sym.BOO,createToken(yytext())); }
extends              { return new Symbol(sym.EXT,createToken(yytext())); }
public               { return new Symbol(sym.PUB,createToken(yytext())); }
static               { return new Symbol(sym.STA,createToken(yytext())); }
void                 { return new Symbol(sym.VOI,createToken(yytext())); }
main                 { return new Symbol(sym.MAI,createToken(yytext())); }
String               { return new Symbol(sym.STR,createToken(yytext())); }
return               { return new Symbol(sym.RET,createToken(yytext())); }
while                { return new Symbol(sym.WHI,createToken(yytext())); }
System.out.println      { return new Symbol(sym.SOP,createToken(yytext())); }
```

Geração do Scanner

```
length          { return new Symbol(sym.LEN,createToken(yytext())); }
true            { return new Symbol(sym.TRU,createToken(yytext())); }
false           { return new Symbol(sym.FAL,createToken(yytext())); }
this            { return new Symbol(sym.THI,createToken(yytext())); }
new             { return new Symbol(sym.NEW,createToken(yytext())); }
null            { return new Symbol(sym.NUL,createToken(yytext())); }
else            { return new Symbol(sym.ELSE,createToken(yytext())); }
if              { return new Symbol(sym.IF,createToken(yytext())); }

{identificador} { return new Symbol(sym.IDENT,createToken(yytext())); }
{inteiro}       { return new Symbol(sym.INT,createToken(yytext())); }
{comentario}            { /* For this stand-alone lexer, print out comments. */
                          System.out.println("Recognized comment: " + yytext()); }
{whitespace}      { /* Ignore whitespace. */ }
{newline}         { /* Ignore whitespace. */ }
```

Geração do Scanner

- A geração do código a partir do JFlex foi simples:

```
gustavo@gustavo-workstation:~/Downloads/TrabalhoCompiladorFinal/src/trabalhocomp
ilador$ jflex lexer.flex
Picked up JAVA_TOOL_OPTIONS: -javaagent:/usr/share/java/jayatanaag.jar
Reading "lexer.flex"
Constructing NFA : 289 states in NFA
Converting NFA to DFA :
.....................................................................
.....................................................
149 states before minimization, 145 states in minimized DFA
Old file "MiniJavaLexer.java" saved as "MiniJavaLexer.java~"
Writing code to "MiniJavaLexer.java"
gustavo@gustavo-workstation:~/Downloads/TrabalhoCompiladorFinal/src/trabalhocomp
ilador$
```

Java Cup

- Java Cup é um gerador de parser para Java, escrito em Java.
- Como entrada recebe uma especificação baseada na gramática correspondente e em conjunto com um scanner capaz de devolver tokens produz o código java de um analisador sintático.
- Versões antigas não aceitavam ambiguidade
- A versão mais recente aceita ambiguidade, uma vez que, declare-se as respectivas precedências

Geração do Parser

- Definição da Gramática

- Geração de código

Gramática

PROG -> MAIN CLASSLIST
MAIN -> class id '{' public static void main (String [ ] id) '{' CMD '}' '}'
CLASSLIST -> CLASSE CLASSLIST
CLASSE -> class id [extends id] VARLIST METODOLIST
VARLIST -> VARLIST VAR
METODOLIST -> METODO METODOLIST
VAR -> TIPO id ;
METODO -> public TIPO id PARAMSMETODO '{' VARLIST CMDLIST return EXP ; '}'
PARAMSMETODO -> '(' PARAMSLIST')'
CMDLIST -> CMD CMDLIST
PARAMSLIST -> PARAMS PARAMSLIST

GRAMATICA

```
TIPO  ->int '[' ']'
        | boolean
        | int
        | id
CMD ->  '{' CMDLIST '}'
        | if '(' EXP ')' CMD
        | if '(' EXP ')' CMD else CMD
        | while '(' EXP ')' CMD
        | System.out.println '(' EXP ')' ;
        | id = EXP ;
        | id '[' EXP ']' = EXP ;
```

GRAMATICA

EXP -> REXP {EXPLINE}
EXPLINE -> && REXP {EXPLINE}
REXP -> AEXP {REXPLINE}
REXPLINE -> < AEXP {EXPLINE}
| == AEXP {REXPLINE}
| != AEXP {REXPLINE}
AEXP -> MEXP {AEXPLINE}
AEXPLINE-> + MEXP {AEXPLINE}
| AEXP - MEXP {AEXPLINE}
MEXP -> SEXP {MEXPLNE}
MEXPLINE -> * SEXP {MEXPLNE}
| / SEXP {MEXPLNE}

GRAMÁTICA

```
SEXP -> ! SEXP
        | - SEXP
        | true
        | false
        | num
        | null
        | new int '[' EXP ']'
        | PEXP . length
        | PEXP '[' EXP ']'
        | PEXP
```

Gramática

PEXP -> id
| this
| new id '(' ')'
| '(' [EXPS] ')' {PEXPLINE}
PEXPLINE -> . id {PEXPEMPTY} {PEXPLINE}
PEXPEMPTY -> '(' [EXPS] ')'
EXPS -> EXP {, EXP}

Geração do Parser

- E a geração do código a partir do Cup também foi simples de ser feita:

```
gustavo@gustavo-workstation:~/Downloads/TrabalhoCompiladorFinal/src/trabalhocomp
iladors cup parser.cup
Picked up JAVA_TOOL_OPTIONS: -javaagent:/usr/share/java/jayatanaag.jar
Warning : Terminal "OR" was declared but never used
Warning : Terminal "GTR" was declared but never used
Warning : Terminal "LESS_EQ" was declared but never used
Warning : Terminal "GTR_EQ" was declared but never used
------- CUP v0.11a beta 20060608 Parser Generation Summary -------
  0 errors and 4 warnings
  47 terminals, 27 non-terminals, and 67 productions declared,
  producing 174 unique parse states.
  4 terminals declared but not used.
  0 non-terminals declared but not used.
  0 productions never reduced.
  0 conflicts detected (0 expected).
  Code written to "parser.java", and "sym.java".
---------------------------------------------------- (v0.11a beta 20060608)
gustavo@gustavo-workstation:~/Downloads/TrabalhoCompiladorFinal/src/trabalhocomp
iladors
```

Geração do Parser

- Conflitos

```
Warning : *** Shift/Reduce conflict found in state #101
  between CMD ::= IFT LPAREN EXP RPAREN CMD (*)
  and     CMD ::= IFT LPAREN EXP RPAREN CMD (*) ELSET CMD
  under symbol ELSET
  Resolved in favor of shifting.

Error : *** More conflicts encountered than expected -- parser generation aborted
------- CUP v0.11a beta 20060608 Parser Generation Summary -------
  1 error and 11 warnings
  53 terminals, 15 non-terminals, and 47 productions declared,
  producing 162 unique parse states.
  10 terminals declared but not used.
  0 non-terminals declared but not used.
  0 productions never reduced.
  1 conflict detected (0 expected).
  No code produced.
------------------------------------------------- (v0.11a beta 20060608)
```

Código Java

```
package trabalhocompilador;

public class Token {
    public String val;
    public Token(String val) {
        this.val = val;
    }
    @Override
    public String toString() {
        return val;
    }
}
```

Código Java

```
String nomeDoArquivo =f.getAbsolutePath()+"/src/trabalhocompilador/program.txt";
String[] argsMiniJavaParser = {nomeDoArquivo};

MiniJavaParser.main(argsMiniJavaParser);
System.out.println("Código Reconhecido com sucesso!");
```