UNIVERSIDADE FEDERAL DE MINAS GERAIS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

UM COMPILADOR PARA A LINGUAGEM TIGER - ANALISADOR SINTÁTICO

Filipe Rodrigues Batista de Oliveira - 2018055091
Heitor de Paula Santos Damasceno - 2019006671

1 Resumo do Projeto

Nessa parte do trabalho continuou-se a implementação, referente a primeira parte, de um compilador para a linguagem TIGER, sendo desenvolvido a fase de análise sintática. Para este fim foi utilizado o programa CUP, por meio da linguagem JAVA.

Conforme foi comunicado à professora, a primeira parte do trabalho foi convertida para Java, e encontra-se no arquivo **Lexer.lex**, pois julgou-se que seria mais fácil fazer a segunda parte com a tecnologia CUP.

2 Analisador Sintático

Esta etapa da compilação de um programa tem como função receber um conjunto de tokens e gerar a árvore de derivação que é usada para decidir se o código fonte dado como entrada possui a estrutura gramatical correta.

3 Organização do Projeto

O projeto foi feito com o gerador de analisador léxico JLEX e o gerador de analisador sintático CUP, ambas ferramentas que utilizam a linguagem JAVA. O arquivo referente à lexicografia da linguagem é **Lexer.lex**, enquanto o arquivo referente às especificações sintáticas da linguagem é **Parser.cup**.

3.1 Ferramentas de Apoio

Para gerar o arquivo .java do analisador léxico a partir do arquivo .lex com a especificação das expressões regulares foi utilizado o programa **jflex**.

Para instalar o lex no Ubuntu:

```
sudo apt-install jflex
```

Para gerar o .java do analisador sintático que possui a definição da gramática da linguagem, foi utilizado o gerador de analisador sintático CUP, que recebe o código Parser.cup.

As dependências do Cup: **java-cup-11b.jar** e **java-cup-11b-runtime.jar** estão inclusas no diretório do projeto, mas podem ser obtidos no site do maven.

4 Instruções para Execução do Código

O projeto foi elaborado e compilado no ambiente Linux Ubuntu 20.04. Recomenda-se a utilização de ambiente GNU/Linux para que faça o mesmo.

Para realizar a análise sintática de um código, primeiramente o analisador léxico deve ser gerado com o JFLEX. Para gerar o analisador léxico com o JFLEX utilize o comando:

```
jflex Lexer.lex
```

Para compilar o analisador sintático com o CUP utilize o comando:

```
java -jar java-cup-11b.jar -parser Parser
-symbols Sym Parser.cup
```

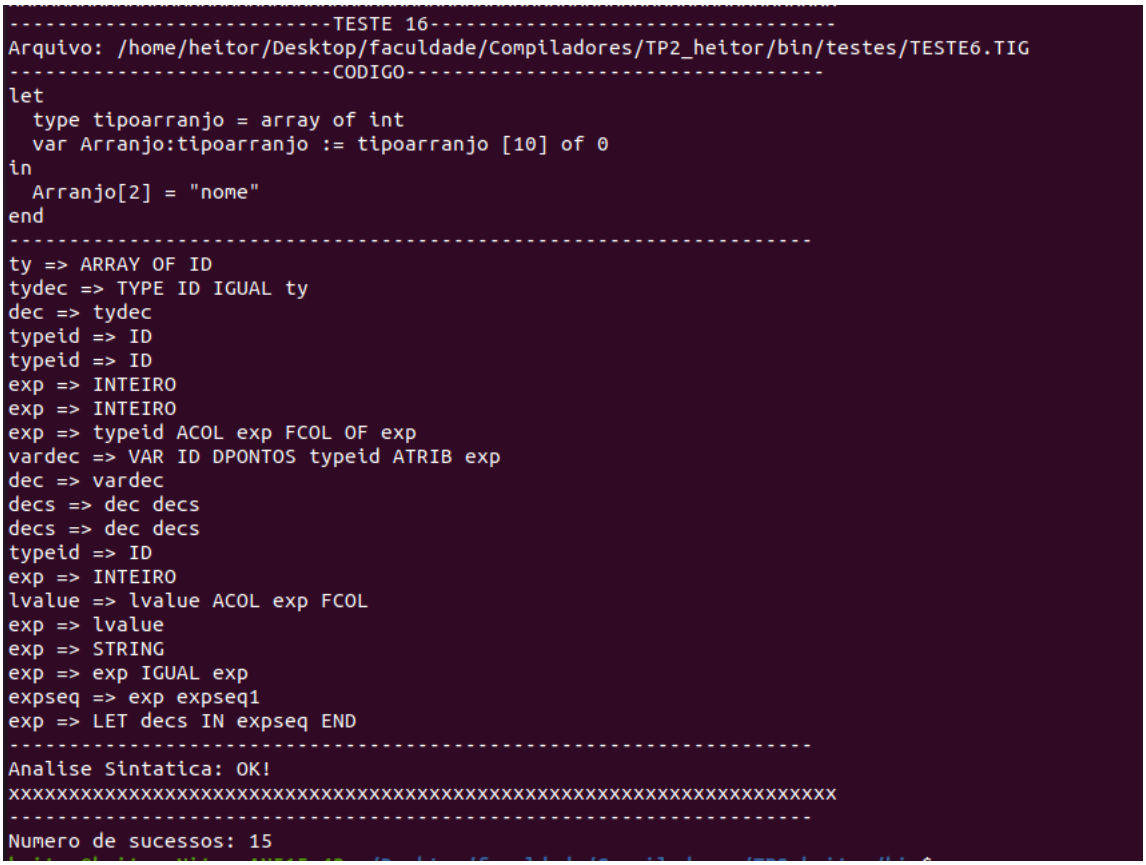
Isso deve gerar os arquivos **Lexer.java**,**Sym.java** e **Parser.java**. Com esses arquivos, é possível compilar o Main do projeto. O seguinte comando compila os arquivos para o diretório **bin**:

```
javac -cp .:./java-cup-11b-runtime.jar:./java-cup-11b.jar -d
bin Main.java
```

Uma versão já compilada está presente no diretório "bin"do projeto, **main.jar**. Para executá-lo basta entrar no diretório "bin"e utilizar o comando:

```
java -cp .:./../java-cup-11b-runtime.jar Main
```

O **main** lê todos os arquivos da pasta **bin/testes/** e faz a análise sintática deles. Caso queira ler um arquivo específico, basta colocá-lo na pasta **bin/testes/**. Você pode também retirar arquivos da pasta, caso não queira realizar testes com os arquivos lá presentes.


----------TESTE 16----------
Arquivo: /home/heitor/Desktop/faculdade/Compiladores/TP2_heitor/bin/testes/TESTE6.TIG
----------CODIGO----------
let
type tipoarranjo = array of int
var Arranjo:tipoarranjo := tipoarranjo [10] of 0
in
Arranjo[2] = "nome"
end
ty => ARRAY OF ID
tydec => TYPE ID IGUAL ty
dec => tydec
typeid => ID
typeid => ID
exp => INTEIRO
exp => INTEIRO
exp => typeid ACOL exp FCOL OF exp
vardec => VAR ID DPONTOS typeid ATRIB exp
dec => vardec
decs => dec decs
decs => dec decs
typeid => ID
exp => INTEIRO
lvalue => lvalue ACOL exp FCOL
exp => lvalue
exp => STRING
exp => exp IGUAL exp
expseq => exp expseq1
exp => LET decs IN expseq END
Analise Sintatica: OK!
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Numero de sucessos: 15

5 Suposições, Problemas Encontrados e Erros

5.1 Conflitos da Gramática

Durante a implementação da gramática da linguagem TIGER, foram encontradas duas instâncias de conflitos que geraram resultados errados durante a execução dos testes. Esses conflitos, assim como a resolução adotada foram detelhados a seguir.

5.1.1 Conflito 1

```
*** Shift/Reduce conflict found in state #133
  between exp ::= IF exp THEN exp (*)
  and     exp ::= IF exp THEN exp (*) ELSE exp
  under symbol ELSE
```

Esse conflito ocorre pois as construções:
exp ::= IF exp THEN exp
e
exp ::= IF exp THEN exp ELSE exp
Compartilham a mesma construção inicial ”exp ::= IF exp THEN exp”. Ela foi resolvida adicionando precedência à esquerda ao símbolo **ELSE**:

```
precedence left ELSE;
```

Essa resolução é correta pois na linguagem TIGER o ELSE é casado com o IF mais próximo.

5.1.2 Conflito 2

```
*** Reduce/Reduce conflict found in state #11
 between typeid ::= ID (*)
 and     lvalue ::= ID (*)
```

Esse conflito ocorre pois ambas as construções ”lvalue”e ”typeid”podem assumir o lexema **ID**. Para resolvê-lo, limitamos a regra apenas à construção typeid, e duplicamos todas as regras em que lvalue aparece para aceitar também typeid no seu lugar.

6 Testes

Esta etapa do trabalho foi testada com os arquivos de teste disponibilizados pela professora. Os resultados estão na **Tabela 1**.

Testes e Resultados	
Arquivo	**Resultado**
FATORIAL.TIG	SUCESSO
QUEENS.TIG	SUCESSO
TESTE0.TIG	SUCESSO
TESTE1.TIG	SUCESSO
TESTE2.TIG	SUCESSO
TESTE3.TIG	SUCESSO
TESTE4.TIG	SUCESSO
TESTE5.TIG	**ERRO!**
TESTE6.TIG	SUCESSO
teste7.TIG	SUCESSO
teste9.TIG	SUCESSO
TesteSeman1.TIG	SUCESSO
TesteSeman2.TIG	SUCESSO
TesteSeman3.TIG	SUCESSO
TESTEOK.TIG	SUCESSO
FATORIAL.TIG	SUCESSO

Tabela 1: Testes que foram executados e seu resultado.

7 Conclusão

Com este trabalho foi possível entender o funcionamento de analisadores sintáticos, e compreender como se dá a integração entre analisadores léxicos e analisadores sintáticos nos compiladores.
A partir dessa etapa será possível montar, na próxima, a árvore semântica da linguagem TIGER.

8 Bibliografia

Especificações léxicas da linguagem TIGER:
https://www.lrde.epita.fr/ tiger/tiger.split/Lexical-Specifications.html
Arquivos de teste, especificações do trabalho, informações adicionais:
https://homepages.dcc.ufmg.br/˜mariza/Cursos/CompiladoresI/Geral/Projeto2021-2/compProjeto2021-2.html
Tutorial para o uso do CUP com o JFLEX:
https://johnidm.gitbooks.io/compiladores-para-humanos/content/part2/using-jflex-java-cup.html