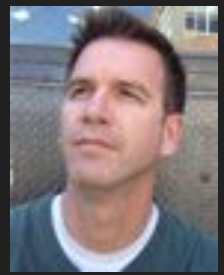
Analizador Léxico

René Nolio
Ricardo Boing

Ferramenta utilizada: ANTLR4

O ANTLR (ANother Tool for Language Recognition), pronunciado *antler*, é um poderoso gerador de analisador léxico e sintático LL(k). É amplamente usado para construir linguagens, ferramentas e frameworks. A partir de uma gramática, o ANTLR gera um analisador sintático que pode construir e analisar árvores sintáticas. A última versão estável foi lançada em 2017. É o sucessor do PCCTS (Purdue Compiler Construction Tool Set), desenvolvido em 1989. Foi escrito em java e é *cross-plataform*.

Foi desenvolvida por Terence Parr, professor de ciências da computação da Universidade de São Francisco.



Arquivo.g4

- CLASS : 'class';
- EXTENDS : 'extends';
- ABRCHAVE : '{';
- PTVIR : ';';
- FECHCHAVE : '}';
- INT : 'int';
- ….

Arquivo.g4

- fragment LOWERCASE: [a-z];
- fragment UPPERCASE: [A-Z];
- fragment DIGIT: [0-9];
- ….

Arquivo.g4

- fragment LOWERCASE: [a-z];
- fragment UPPERCASE: [A-Z];
- fragment DIGIT: [0-9];
- ….

- INTCONSTANT : (DIGIT)+;
- STRINGCONSTANT : ["](LOWERCASE | UPPERCASE | DIGIT | ESPECIAL)*["];

Arquivo.g4

- fragment EXEMPLO:
- (
- [\u0060]
- | [\u005C]
- | [\u00C0-\u00C5]
-);

ANTLR

- XccLexer lexer = new XccLexer(input);
- CommonTokenStream tokens = new CommonTokenStream(lexer);
- Vocabulary vocabularioAntlr = lexer.getVocabulary();

- List<Token> listaDeTokens = tokens.getTokens();
- Token token = listaDeTokens.get(c);
- nomeDoLexema = token.getText();
- linhaDaOcorrencia = tokenAntlr.getLine();
- colunaDaOcorrencia = tokenAntlr.getCharPositionInLine();
- nomeDoToken = vocabularioAntlr.getSymbolicName(token.getType());

Tarefa AS1: Questão 1

A gramática X++ é recursiva à esquerda?

- P’ = φ, conjunto de não terminais (Vn) sem recursão à esquerda
- $\forall$ A, se $\forall$ P são da forma $A \rightarrow \beta\alpha$, $\alpha \in V^*$, $\beta \in Vt$, $A \in Vn$ então $P' = P' \cup \{A\}$
- Demais produções: se todas as produções de A são do tipo $A \rightarrow C\alpha$, onde $C \in P'$, então $P' = P' \cup \{A\}$
- Demais produções: se $\nexists\, A \rightarrow B\alpha$, então $P' = P' \cup \{B\}$ (pode $\exists\, B \rightarrow \varepsilon$, seria necessário um passo adicional)
- Neste ponto, P’ = P, então não existe recursão à esquerda em nenhuma produção.

Tarefa AS1: Questão 2

A gramática X++ está fatorada à esquerda?

- Sejam f1, f2, ... fn os conjuntos FIRST das produções de A. G está fatorada se $\forall$ A, f1 $\cap$ f2 $\cap$ ... $\cap$ fn = φ, A $\in$ Vn
- Computamos os conjuntos first para todas as produções:
- FIRST(statement) = {INT, STRING, IDENT} $\cup$ {IDENT} $\cup$ …

  statement → (INT | STRING | IDENT) IDENT (ABRCOL FECHCOL)* (VIRG IDENT (ABRCOL FECHCOL)*)* PTVIR | IDENT (ABRCOL expression FECHCOL | PONTO IDENT (ABRPAR arglist FECHPAR)?)* ATR (expression | alocexpression) PTVIR

Fatoramos:

statement → (INT | STRING) IDENT (ABRCOL FECHCOL)* (VIRG IDENT (ABRCOL FECHCOL)*)* PTVIR | IDENT statementaux

statementaux → IDENT (ABRCOL FECHCOL)* (VIRG IDENT (ABRCOL FECHCOL)*)* PTVIR | (ABRCOL expression FECHCOL | PONTO IDENT (ABRPAR arglist FECHPAR)?)* ATR (expression | alocexpression) PTVIR

- As demais produções estão fatoradas

Tarefa AS1: Questão 3

A gramática X++ está em LL(3). Por que o 3?

- G está fatorada se qualquer palavra puder ser derivada de forma determinística, isto é, $\exists !$ produção $A \rightarrow \beta\alpha$, $\forall$ A $\in$ Vn. Ou seja, consultamos um símbolo à frente. LL(1)
- Para dois símbolos se qualquer palavra puder ser derivada de forma determinística, mas consultando dois símbolos à frente, isto é, $\exists !$ produção $A \rightarrow \beta\delta\alpha$, $\forall$ A $\in$ Vn; mas podem existir outras produções $A \rightarrow \beta\theta\alpha$, $\theta \neq \delta$; $\beta, \delta, \theta \in$ Vt. LL(2)
- Não é LL(1): vide questão 2. Podemos demonstrar que não é LL(2). Portanto, é LL(3)

- Como G é LL(3), é necessário verificar três símbolos à frente para decidir deterministicamente qual produção utilizar para derivar uma palavra.
- LL(k) significa que faz o análise da esquerda pra direita (L = Left to right) e constroi a derivação mais à esquerda (L = leftmost derivation), o k indica quantos símbolos à frente o analisador irá verificar.

Referências bibliográficas

- https://www.antlr.org/
- https://unicode-table.com/pt/#control-character
- https://tomassetti.me/antlr-mega-tutorial/
- DELAMARO, Márcio Eduardo. Como Construir um compilador. São Paulo, Novatec, 2004.