Analisador Sintático para a Linguagem lang

Dupla

- Eduardo Vieira Marques Pereira do Valle - 201665554C
- Matheus Brinati Altomar - 201665564C

Arquivos Fonte

- No.java para servir apenas como um SuperNode para o retorno da função parseFile;
- idTipoErrado.lan arquivo de teste para verificar erro sintático de nome de tipo;
- ControlParser.java controla a chamada do parser e do analisador léxico criados pela ferramenta ANTLR;
- Alteração no arquivo LangCompiler.java para utilizar ControlParser.java;
- lang.g4 arquivo utilizado pela ferramenta para gerar o parser e o analisador léxico;
- Arquivos gerados pela ferramenta ANTLR:
    - lang.interp;
    - lang.tokens;
    - langLexer.interp;
    - langLexer.tokens;
    - langLexer.java;
    - langParser.java.

Expressões Regulares Alteradas

Em virtude da impossibilidade de interligar o analisador léxico desenvolvido com a ferramenta jflex, foi necessário desenvolver o analisador léxico na ferramenta ANTLR, que esta também foi utilizada para desenvolver o analisador sintático. Dessa forma, não criamos dois estados léxicos como era preciso no jflex e alteramos a expressão regular do comentário de múltiplas linhas para: '{-' .*? '-}' de acordo com as regras léxicas do ANTLR, além disso o ANTLR trata de ambiguidade léxica da mesma maneira que o jflex, portanto os identificadores foram os últimos tokens a serem definidos.

Gramática

Para definir a gramática da linguagem lang utilizando a ferramenta ANTLR, não necessitou de grandes alterações na gramática, já que ANTLR aceita o formato EBNF

(repetição através do uso desta simbologia ()* e para caso de opcional ()?) e consegue lidar com recursão a esquerda direta. Além de alterar a produção de data e btype para aceitarem apenas o nome de tipo(token IDtype, identificador que começa com a letra maiúscula, para testar isso criamos o arquivo teste idTipoErrado.lan na pasta errado), e todas as outras ocorrências de identificador aceitarem tanto o nome de tipo quanto identificador(token ID, identificador que começa com a letra minúscula).

ANTLR

Utilizamos ANTLR versão 4.8 disponível no link https://www.antlr.org/ , pois é simples de utilizar e aprender, por ter uma documentação boa e uma comunidade bem desenvolvida; pensamos em utilizar a versão 3.5.2 por poder desenvolver a árvore AST direto pelo código do ANTLR, mas a documentação não era bem estruturada como o da versão mais recente(4.8). ANTLR é um gerador de analisador sintático que pode ser usado para ler, processar, executar ou traduzir texto estruturado ou arquivos binários. Os analisadores ANTLR usam uma nova tecnologia de análise chamada Adaptive LL (*) ou ALL (*) ("all star") que é uma extensão do LL (*) da versão 3 que realiza análises gramaticais dinamicamente em tempo de execução, em vez de estaticamente, antes que o analisador gerado seja executado. Como os analisadores ALL (*) têm acesso às sequências de entrada reais, eles sempre podem descobrir como reconhecer as sequências entrelaçando de maneira apropriada a gramática.

Compilação

Todos os comandos a seguir devem ser executados dentro do terminal dentro da pasta lang. O uso do ANTLR adiciona um comando para gerar os arquivos: lang.interp, lang.tokens, langLexer.interp, langLexer.tokens, langLexer.java e langParser.java a partir do arquivo lang.g4, entretanto este comando gera mais arquivos desnecessários que foram removidos, devido a isso enviamos aqueles arquivos ao invés de enviar apenas o lang.g4 e ter mais uma etapa de execução de comandos.

O primeiro comando para compilar todos os arquivos .java encontrados na pasta do programa seria este:

- javac -cp .:antlr-4.8-complete.jar ast/*.java parser/*.java LangCompiler.java -d .

Por fim, basta usar o comando abaixo para executar os arquivos gerados pelo comando anterior, a fim de testar os arquivos encontrados na pasta errado:

- java -classpath .:antlr-4.8-complete.jar lang.LangCompiler -bs