UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE CRATEÚS
CURSO: CIÊNCIA DA COMPUTAÇÃO
DISCIPLINA: COMPILADORES

JOÃO PAULO DE ARAÚJO
MARCUS VINÍCIUS MARTINS MELO
HÊNIO TIERRA SAMPAIO

ANALISADOR SINTÁTICO

Trabalho apresentado à Disciplina de Compiladores, ministrada pelo Prof. Roberto Cabral Rabelo Filho, para obtenção parcial de créditos e título de Bacharel em Ciência da Computação.

Maio / 2018
CRATEÚS-CE

Sumário

1 INTRODUÇÃO

Segundo o site wikipedia, **análise sintática** (também conhecida pelo termo em inglês ***parsing***) é o processo de analisar uma sequência de entrada (lida de um arquivo de computador ou do teclado, por exemplo) para determinar sua estrutura gramatical segundo uma determinada gramática formal. Essa análise faz parte de um compilador, junto com a análise lexical e análise semântica.

A análise sintática transforma um texto na entrada em uma estrutura de dados, em geral uma árvore, o que é conveniente para processamento posterior e captura a hierarquia implícita desta entrada. Através da análise lexical é obtido um grupo de tokens, para que o analisador sintático use um conjunto de regras para construir uma árvore sintática da estrutura.

2 OBJETIVO DO TRABALHO

A proposta deste trabalho se objetiva em analisar uma gramática LR(1) qualquer e realizar o reconhecimento sintático de uma palavra, analisando suas regras de acordo com as regras da gramática.

3 METODOLOGIA

Para a elaboração deste trabalho, as atividades necessárias podem ser divididas em termos de fases temporais, baseadas nas sequências propostas na descrição do trabalho.

A divisão em fases gera as etapas: planejamento, implementação de algoritmos para calcular o **NULLABLE**, **FIRST** e **FOLLOW,** montar o autômato que auxilia na construção da tabela do parser e, eventualmente, a tabela do parser, bem como um algoritmo capaz de percorrer a tabela e analisar as entradas do analisador. A iniciação do projeto constituiu-se no planejamento. Nesta fase a linguagem para o desenvolvimento da aplicação foi escolhida, sendo esta a linguagem Python, assim como um estudo detalhado de como programar as funções necessárias para o funcionamento do analisador sintático. Nesta etapa, optou-se por dividir o algoritmo sugerido, este responsável por calcular *NULLABLE*, *FIRST* e *FOLLOW*, em três algoritmos, com o intuito de facilitar a implementação.

Vencida a etapa de planejamento, a etapa seguinte foi a divisão e a implementação dos algoritmos. Nesta etapa, os conhecimentos adqueridos em sala de aula foram de extrema importância, dado que a implementação das três funções foram inteiramente fundamentadas nestes conhecimentos. A abordagem inicia-se na sequência dada pelo calculo, primeiramente do *NULLABLE*, seguido dos cálculos do *FIRST* e do *FOLLOW*.

A etapa seguinte definiu-se pela construção do autômato, este de extrema importância para o funcionamento do analisador. Nesta etapa, optou-se por não seguir o pseudocódigo sugerido, dodo o nível de abstração encontrado no mesmo, o que fez desta uma das etapas que mais demandou tempo.

Vencida a etapa de construção do autômato, a etapa seguinte foi a construção da tabela parser. Esta etapa foi a mais simples, dado que sua conclusão dependia inteiramente da conclusão da etapa anterior, sendo inseridas operações simples na etapa anterior, possibilitando êxito na construção da tabela do parser.

Por último, a implementação da função responsável por percorrer o parse gerado não foi implementada com total certeza de sucesso. Os motivos que ocasionaram a não obtenção de êxito completo nesta etapa são expostas no tópico seguinte.

4 DIFICULDADES ENCONTRADAS

Durante a elaboração do trabalho, três dificuldades demonstraram-se bastante relevantes. A primeira delas foi o nível de abstração nos pseudocódigos sugeridos para elaborar as principais funções da implementação, sendo de extrema importância os conhecimentos adqueridos em sala de aula para a obtenção de êxito ao enfrentar essa dificuldade.

A segunda caracterizou-se pela falta de exemplos diversos para a realização de testes na implementação, pois não foi fácil encontrar exemplos de linguagens com autômatos já construídos para comparar com os resultados gerados pela implementação, demandando bastante tempo na validação dos testes e prejudicando o andamento do trabalho. Esta dificuldade foi contornada como auxílio da ferramenta LR(1) Parser Generator, disponível em http://jsmachines.sourceforge.net/machines/lr1.html.

A terceira dificuldade deu-se pelo fato de que os membros da equipe não compreenderam ao certo como percorrer a tabela gerada para analisar as entradas dadas à implementação. Pelo fato desta dificuldade não ter sido contornada com êxito, a implementação foi entregue sem a concreta certeza de que está funcionando corretamente.

5 CONCLUSÃO

A partir do trabalho, foi possível perceber, de início, a complexidade por trás de um analisador sintático, dado o número de operações que o mesmo realiza. Contudo, a maior conclusão se da ao fato da percepção, ao longo do trabalho, no grau de complexidade na criação de uma linguagem de programação, levando em conta que o trabalho realizado é uma pequena parcela do que realmente se trata um compilador.

6 REFERÊNCIAS

APPE A. ; **Modern Compiler Implementation in Java**. Second Edition. October 2002;

LR(1) Parser Generator; Disponível em:http://jsmachines.sourceforge.net/machines/lr1.html; Acessado em 17/05/2018;

Parsing V LR(1) Parsers; Disponível em : https://people.cs.umass.edu/~moss/610-slides/11.pdf; Acessado em 17/05/2018.