Centro Universitário de Belo Horizonte – Uni-BH
Curso: Ciência da Computação
Disciplina: Compiladores
Professor: Gustavo Alves Fernandes

Trabalho Prático II – Análise Sintática

Descrição do trabalho

Nesta etapa, você deverá implementar um analisador sintático descendente (top-down) para a linguagem *PasC*, cuja descrição encontra-se no enunciado do trabalho prático I.

Seu compilador deverá ser um analisador de uma única passada. Dessa forma, ele deverá interagir com o analisador léxico para obter os tokens do arquivo-fonte. Você deve implementar seu analisador sintático utilizando o algoritmo de Parser Preditivo Recursivo (Procedimentos para cada Não-terminal) ou o algorimto de Parser Preditivo Não-Recursivo (Pilha).

O analisador sintático deverá reportar possíveis erros ocorridos no programa-fonte. O analisador deverá informar qual o erro encontrado (informar que token era espearado e qual token apareceu) e sua localização no arquivo-fonte. Não haverá recuperação de erro para a análise sintática, logo que um erro sintático for encontrado, o processo de compilação deverá ser abortado. A identificação dos erros Léxicos continuam de acordo com o TP1, isto é, deverão ser identificados, sinalizados e com recuperação de erro funcional.

Para implementar o analisador sintático, você deverá modificar a estrutura gramatical da linguagem. Você deverá adequá-la e eliminar a recursividade à esquerda e fatorar a gramática, ou seja, a gramática PasC ainda não é LL(1). Portanto, você deverá verificar as regras que infringem as restrições das gramáticas LL(1) e adaptá-las para tornar a gramática LL(1).

Cronograma e Valor

O trabalho vale 30 pontos no total. Ele deverá ser entregue por etapas, sendo a segunda etapa correspondendo ao Analisador Sintático, conforme consta na tabela abaixo.

Etapa	**Data de entrega**	**Valor**	**Multa por atraso**
Analisador Léxico e Tabela de símbolos	11/04/2017	10 pontos	2pts/dia
Analisador Sintático	03/06/2017	10 pontos	2pts/dia
Analisador Semântico	A definir	10 pontos	2pts/dia

O que fazer?

1. Fatorar a gramática para as regras “id-list”, “if-stmt”, “expression”
2. Eliminar a recursão a esquerda para as regras “simple-expr”, “term”
3. Implementar os algoritmos de Parser Preditivo Recursivo ou Não-Recursivo

O que entregar?

1. A nova versão da gramática;
2. Apresentar o cálculo do FIRST, FOLLOW e Tabela Preditiva.
3. Programa com todos os arquivos-fonte;
4. Relatório contendo testes realizados com programas (de acordo com a gramática) corretos e errados (no mínimo, 3 certos e 3 errados), e também deverá conter a descrição de cada função/método do Parser.

Regras:

- O trabalho poderá ser realizado individualmente ou em dupla.
- Não é permitido o uso de ferramentas para geração do analisador sintático.
- A implementação deverá ser realizada em uma das linguagens C, C++, C#, Java, Python ou Ruby.
- Trabalhos total ou parcialmente iguais receberão avaliação nula.
- Se o seu programa não compilar/executar, a avaliação será nula.
- Ultrapassados cinco (5) dias, após a data definida para entrega, nenhum trabalho será recebido.

Pontuação extra (3 pontos)

A recuperação de erros comum para um analisador sintático é o Modo Pânico. Se um token aparece em um momento que não é esperado, este deve ser ignorado. Ou seja, todos os tokens não esperados deverão ser ignorados (com mensagem de erro, é claro) até que o token esperado (sincronizante) apareça. Sendo assim, é possível tratar o modo pânico na recuperação de erros sintáticos da seguinte maneira:

- Parser Preditivo Não-Recursivo: `skip()` e `synch()` como foi visto em sala;
- Parser Preditivo Recursivo: utilizar o Follow(A), sendo A o não terminal da produção corrente, para descobrir os tokens sincronizantes. Enquanto o token na entrada não for um token sincronizante, então aponte o erro sintático e avance entrada. Ao encontrar o token sincronizante, volte ao ponto corrente na recursão.

Você deverá construir esses métodos para que o modo pânico gere “menos confusão” ao Parser e tente fazer uma maior varredura no código. Contudo, se o número de erros sintáticos ultrapassar o limite de 5 erros, o compilador deverá abortar a análise.

Em anexo (pasta: parser_*exemplo*) segue um exemplo de uma Gramática, AFD, programas de exemplo, e a saída dos Tokens. **ATENÇÂO**: a gramática do exemplo não tem relação com a gramática do *PasC*.