Avaliação LFC - parte 1 – João D. Mayer

1- Exemplo:

IF a THEN IF b THEN x ELSE y

O exemplo apresentado segue a regra proposta pela gramática do exercício.

A ambiguidade nesse exemplo ocorre pois não é possível determinar se a expressão ELSE refere-se ao primeiro ou ao segundo IF. Dessa forma é possivel que y aconteça através de dois caminhos diferentes, sendo eles: (a=verdadeiro, b=falso) ou (a=falso). Confirmando assim a ambiguidade.

Uma forma de se resolver essa ambiguidade seria adicionando um terminal a regra, esse terminal teria o papel de delimitar o fim de um IF. Fazendo isso a regra ficaria da seguinte forma:

<cmd> ::= if <Lexpr> then <cmdList> endif
<cmd> ::= if <Lexpr> then <cmdList> else <cmdList> endif

O mesmo exemplo agora seguindo a nova regra:

IF a THEN IF b THEN x ENDIF ELSE y ENDIF
ou
IF a THEN IF b THEN x ELSE y ENDIF ENDIF

Dessa forma não é possivel que dois caminhos diferentes levem ao mesmo resultado em uma mesma expressão.

Na primeira expressão, para que o resultado seja y, é preciso de (a=falso).

Na segunda expressão, para que o resultado seja y, é preciso de (a=verdadeiro, b=falso).

Nota-se então que o terminal cumpriu seu papel solucionando a ambiguidade da expressão.

2- Gramáticas LL(k) são análises sintáticas descendentes recursivas, os caracteres de entrada são lidos da esquerda para a direita e há uma derivação mais à esquerda. O termo k refere-se à quantidade de tokens posteriores ao símbolo atual ainda não consumidos da entrada que são usados para tomar decisões na análise.

A recursão à esquerda muitas vezes apresenta problemas para os analisadores, seja porque os leva à recursão infinita (como no caso da maioria dos analisadores descendentes) ou porque eles esperam regras de uma forma normal que a proíbem. Portanto, uma gramática é frequentemente pré-processada para eliminar a recursão à esquerda.

O algoritmo de Paull é o mais conhecido quando o assunto é eliminar recursão a esquerda:

Assign an ordering A1,…,An to the nonterminals of the grammar.

```
for i:=1 to n do begin
    for j:=1 to i−1 do begin
        for each production of the form Ai→Ajα do begin
            remove Ai→Ajα from the grammar
            for each production of the form Aj→β do begin
                add Ai→βα to the grammar
            end
        end
    end
    transform the Ai-productions to eliminate direct left recursion
end
```

3- Um erro de verificação de tipo ocorre quando um compilador não consegue classificar corretamente um operando.

A tabela de símbolos é uma estrutura, normalmente árvore ou tabela hash, utilizada por todas as fases de compilação pois nela são armazenados os identificadores das constantes, funções, variáveis e tipos de dados da linguagem. Um compilador usa uma tabela de símbolos para guardar informações sobre os nomes declarados em um programa. A tabela de símbolos é pesquisada cada vez que um nome é encontrado no programa fonte. Alterações são feitas na tabela de símbolos sempre que um novo nome ou nova informação sobre um nome já existente é obtida. Com isso é possível concluir que a tabela de símbolos serve como um banco de dados para o processo de compilação.

Seu principal conteúdo são informações sobre tipos e atributosde cada nome definido pelo usuário no programa. Essas informações são colocadas na tabelade símbolos pelos analisadores léxico e sintáticoe usadas pelo analisador semântico e pelo gerador de código.

Essa estrutura de chave-valor permite que a pesquisa de tipos pelo compilador seja feita de forma muito mais eficiente, pois a busca por um valor é feita de imediato, desde que se saiba a chave correspondente ao mesmo.

4- Não, pois ao realizar a derivação do símbolo não terminal <cmd> o próximo token seria o "if", que é um simbolo em comum entre as regras tornando impossível determinar qual delas usar. Logo, a gramática não é LL(1).