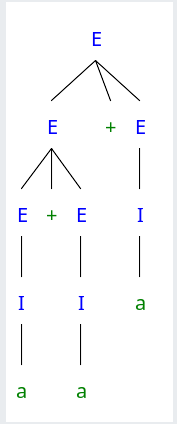
Preparation Activity PA07 – Context-Free Grammars (CFGs)

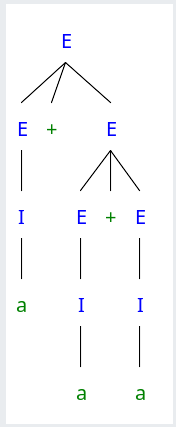
1. Tendo em consideração a CFG G1:  
   E → I | E+E | E×E | (E)   
   I → a | b | Ia | Ib | I0 | I1  
     
   a) – A CFG apresenta ambiguidade, pois existem casos em que se pode obter duas arvores de derivação diferentes para a mesma *string*, havendo mais que uma forma alternativa de derivar uma dada *string* com as produções fornecidas. Isto deve-se ao facto de existirem produções como “E → E+E”, que apresentam recursividade à esquerda e à direita.

Um exemplo seria a *string* “a+a+a” que apresenta duas arvores de derivação:

Primeira Derivação Possível



Segunda Derivação Possível



b) – Uma derivação *leftmost* para “((a)+(a×b))” possível seria:   
  
E => (E) => (E+E) => ((E) + E) => ((I) + E) => ((a) + E) => ((a) + (E)) => ((a) + (E×E)) => ((a) + (I×E)) => ((a) + (a×E)) => ((a) + (a×I)) => ((a) + (a×b))

1. a) – Considerando a seguinte gramática G2:

E → I | E+I | E×I | (E)   
I → a | b | Ia | Ib | I0 | I1  
  
G2 não é ambígua, mas, por outro lado, não representa a mesma linguagem que G1, pois em G2 não é possível ter expressões do tipo “E+(E)” ou “E×(E)”, isto porque o lado direito das somas e multiplicações em G2 não tem possibilidade de produzir “E” (devido a uma tentativa de remover ambiguidade, certamente) e, sendo assim, do lado direito apenas se poderão ter identificadores sem parêntesis à volta (que são as produções de “I”), estando apenas disponível derivar “E+I” ou “E×I” em G2.

b) – Considerando a gramática G3:

E → F | E+F | E×F   
F → I | (E)   
I → a | b | Ia | Ib | I0 | I1

G3 não é ambígua, sendo que, removeu-se a recursividade de “E” no operando do lado direito das somas e multiplicações, criando apenas uma derivação possível para essas *strings*. Adicionalmente, representa a mesma linguagem que G1, pois resolve o problema da gramática anterior (G2), criando uma variável intermédia “F” que contempla a possibilidade de o operando direito das expressões ser tanto um identificador como uma expressão entre parêntesis, tal como está estabelecido na gramática original.

c) – Considerando a gramática G4:  
E → J | E×J   
J → I | J+I   
I → a | b | Ia | Ib | I0 | I1 | (E)

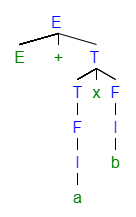
O problema de G4 é que admite que um identificador, representado na gramática pelo símbolo não terminal “I“, pode ser representado tanto por expressões formadas por “a”, “b”, “0” e “1” sem parêntesis como por uma expressão entre parêntesis. Assim sendo, pode-se considerar que é uma extensão da linguagem G1, pois admite mais casos possíveis para identificadores do que a gramática original, sendo que, G1 apenas admite identificadores constituídos pelos símbolos “a”, “b”, “0” ou “1” e sem parêntesis.

d) – Considerando a gramática G5:

E → T | E+T   
T → F | T×F   
F → I | (E)   
I → a | b | Ia | Ib | I0 | I1

A seguinte gramática respeita a precedência aritmética da multiplicação sobre a soma, pois, encontrando uma expressão que contenha ambas, tal como se poderia ver pela arvore de derivação resultante, calcular-se-ia primeiro uma multiplicação e apenas a seguir a soma.

Logo, tendo um exemplo do gênero “a + a × b”, a arvore de derivação seria:



Pela forma como a gramática está escrita, os operadores com menos precedência são produzidos sempre ao nível mais baixo da árvore de derivação, mais próximo da raiz. Assim, vê-se que a produção “E+T” seria sempre a primeira a ser produzida por estar no início (está logo na variável de início da gramática), depois, num diferente símbolo não terminal que se segue ao símbolo inicial “E”, que neste caso é “T”, está a regra correspondente à operação de maior precedência, assim, a produção “T×F” estará sempre mais longe da raiz da árvore, num nível superior, demarcando a sua prioridade sobre a soma.