Aula 13 – Linguagens Livre de Contexto

O estudo da Classe das Linguagens Livres do Contexto ou Linguagem Tipo 2 é de fundamental importância na Computação, pois:

- Compreende um universo mais amplo de Linguagens, tratando, adequadamente, questões como parênteses balanceados, construções blocos-estruturados, entre outras particularidades típicas de Linguagens de Programação como Pascal, C, Java, entre outras;

- Os algoritmos reconhecedores e geradores que implementam as Linguagens Livres do Contexto são relativamente simples e possuem uma eficiência razoável;

- Os exemplos típicos de aplicações dos conceitos e resultados referentes às Linguagens Livres do Contexto são centrados nas Linguagens de Programação. Em particular, destacam-se analisadores sintáticos, tradutores de linguagens e processadores de textos em geral.

Abordagem

O estudo das Linguagens Livres do Contexto é abordado, usando os seguintes formalismos:

a) Gramática Livre do Contexto: Trata-se de um formalismo axiomático ou gerador, no qual, como o nome indica, é uma Gramática, mas com restrições na forma das regras de produção. Tais restrições são definidas mais livremente que na Gramática Regular;

b) Autômato com Pilha: Trata-se de um formalismo operacional ou reconhecedor, cuja estrutura básica é análoga à do Autômato Finito Não-Determinístico, adicionada de uma memória auxiliar do tipo Pilha (a qual pode ser lida ou gravada).

Aplicações

Relativamente nas Gramáticas Livres do Contexto, os seguintes tópicos são desenvolvidos e são importantes tanto para desenvolvedores como para a otimização de algoritmos reconhecedores, como na prova (demonstração) de teoremas:

a) Árvores de Derivação: Representa a derivação de uma palavra na forma de árvore, partindo do símbolo inicial com raiz, e terminando em símbolos terminais como folhas;

b) Simplificação de Gramática: Simplifica as produções sem reduzir o poder de geração da Gramática;

c) Forma Normal: Estabelece restrições rígidas na forma das produções, sem reduzir o poder de geração da Gramática.

A construção de um Autômato com pilha a partir de uma Gramática Livre de Contexto qualquer, permite estabelecer as seguintes conclusões:

- A construção de um reconhecedor para uma Linguagem Livre do Contexto a partir de sua Gramática é simples e imediata;

- Qualquer Linguagem Livre do Contexto pode ser reconhecida por um autômato com pilha com somente um estado de controle lógico, o que implica que a estrutura de pilha é suficiente como única memória, não sendo necessário usar os estados para "memorizar" informações passadas.

Gramáticas Livres do Contexto

As Linguagens Livres de Contexto ou Linguagem Tipo 2 são definidas a partir das Gramáticas Livres do Contexto.

Definição – Gramática Livre do Contexto

Uma Gramática Livre do Contexto **G** é uma Gramática **G = (V, T, P, S)** com a restrição de que qualquer regra de produção de **P** é da forma:

$\mathbf{A} \rightarrow \alpha$, onde **A** é uma variável de **V**, e uma palavra de $\mathbf{(V \cup T)^*}$.

BNF: Backus Naur Form:

Em Informática uma maneira usual de representar uma Gramática Livre do Contexto é o uso da **Forma Backus Naur** ou simplesmente **BNF** (do Inglês Backus Naur Form) – esta notação é a utilizada nas disciplinas de Compiladores, por ser a mais próxima da implementação e do formalismo.

Em BNF, vale que:

- as variáveis (símbolos não-terminais) são palavras delimitadas pelos símbolos **<** e **>**

- uma regra de produção $\mathbf{A} \rightarrow \alpha$ é representada por: **A ::=** α

No caso de **P = {S** $\rightarrow$ **aSb |** ε**}** teríamos

P = {<S> ::= a<S>b | ε**}**

Exemplo: Identificador em Linguagem C

Suponha que se deseja definir uma BNF capaz de gerar qualquer identificador válido na Linguagem de Programação C, ou seja, na qual:

- toda letra é um identificador, assim como o símbolo "_";

- se S é um identificador, então a concatenação à direita de S com qualquer letra ou dígito também é um identificador;

- O único caractere especial válido é o "_".

Uma BNF é como segue (a variável <identificador> é o símbolo inicial):

<identificador> ::=

Prove por derivação que os seguintes identificadores são válidos:

a) _nro9

b) a

c) 9nro_

d) n99_

Árvore de Derivação

Em aplicações como compiladores e processadores de textos, frequentemente é conveniente representar a derivação de palavras na forma de árvore, partindo do símbolo inicial como a "raiz", e terminando em símbolos terminais como folhas.

Definição – Árvore de Derivação

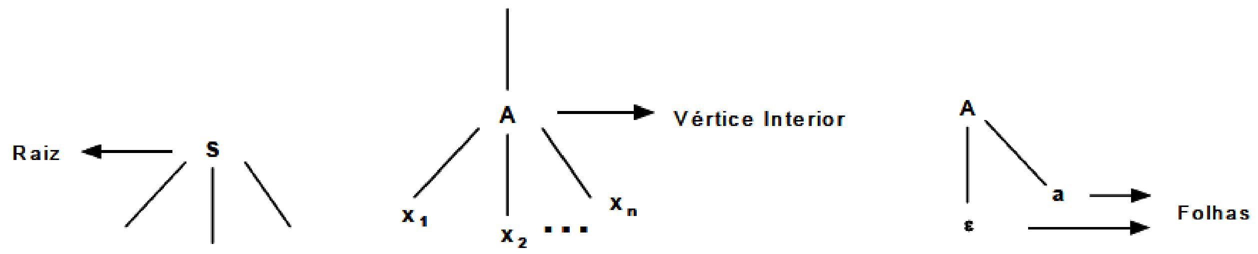
Para uma determinada Gramática Livre do Contexto, a representação da derivação de palavras na forma de árvore, denominada Árvore de Derivação, é como segue:

a) A "raiz" é o símbolo inicial da Gramática.

b) Os vértices interiores obrigatoriamente são variáveis. Se **A** é um vértice interior e $\mathbf{x_1, x_2, \ldots, x_n}$ são os "filhos" de **A**, então:

- $\mathbf{A \rightarrow x_1, x_2, \ldots, x_n}$ é uma produção da Gramática;
- Os vértices $\mathbf{x_1, x_2, \ldots, x_n}$ são ordenados da esquerda para a direita.

c) Um "vértice folha", ou simplesmente folha, é um símbolo terminal, ou o símbolo vazio. Neste caso, o vazio é o único "filho" de seu "pai" $(\mathbf{A \rightarrow \varepsilon})$.



Exemplo: Considere a expressão **x + x * x** gerada por **G = ({E},{+, *, x}, P, E)**,
P = {<E> → <E> + <E> | <E> * <E> | [<E>] | x}.

Mostre a geração da expressão através das regras de produção **P**.

x + x * x

Primeira Solução (Esquerda)	
E ⇒	E → E + E
E + E ⇒	E → x
x + E ⇒	E → E * E
x + E * E ⇒	E → x
x + x * E ⇒	E → x
x + x * x	

Segunda Solução (Direita)	
E ⇒	E → E * E
E * E ⇒	E → E + E
E + E * E ⇒	E → x
E + E * x ⇒	E → x
E + x * x ⇒	E → x
x + x * x	

Observação:

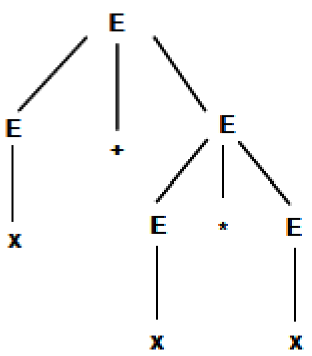
- Na Primeira solução em cada passo de derivação, sempre é derivada a variável mais à esquerda. De forma análoga para a variável mais à direita na segunda solução.

Definição – Derivação mais à Esquerda - Derivação mais à Direita

Dada uma árvore de derivação, uma derivação mais à esquerda (respectivamente, derivação mais a direita) de uma palavra é a sequência de produções aplicadas sempre à variável mais à esquerda (respectivamente, mais à direita) da palavra, em cada passo de derivação.

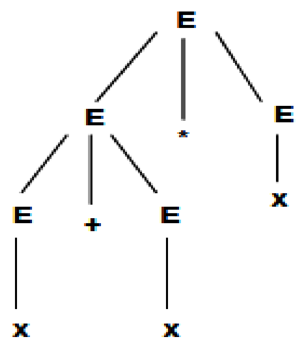
Exemplo: Para as soluções do exemplo acima, construa as árvores de derivação e classifique quanto a derivação para à esquerda ou mais à direita.

A árvore para a Primeira Solução:



Derivação mais à esquerda

Árvore para a Segunda Solução:



Derivação mais à direita

Simplificação de Gramática Livre do Contexto

É possível simplificar alguns tipos de produções sem reduzir o poder de geração das Gramáticas Livres do Contexto. Em geral as simplificações de gramática são usadas na construção e na otimização de algoritmos e na demonstração de teoremas.

Pode-se simplificar:

a) Símbolos Inúteis: exclusão de variáveis ou terminais não usados para gerar palavras;
b) Produções Vazias: exclusão de produções da forma $\mathbf{A} \rightarrow \varepsilon$ (se a palavra vazia pertence à linguagem, é incluída uma produção vazia especifica para tal fim);
c) Produções que Substituem Variáveis: exclusão de produções da forma $\mathbf{A} \rightarrow \mathbf{B}$, ou seja, que simplesmente substituem uma variável por outra e, consequentemente, não adiciona qualquer informação de geração de palavras.