Curso: Engenharia de Computação

Linguagens Formais e Compiladores

Prof. Clayton J A Silva, MSc
clayton.silva@professores.ibmec.edu.br

Análise Sintática

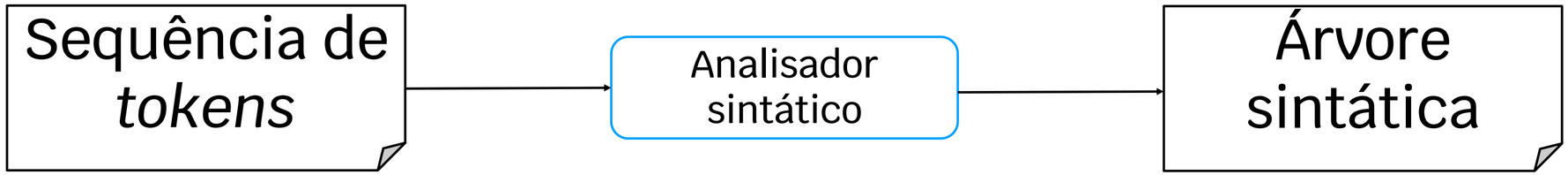
Introdução

- Determina a sintaxe, ou estrutura, de um programa
- A sintaxe de uma LP é definida normalmente pelas **regras gramaticais** de uma **gramática livre de contexto**
- As regras de uma gramática livre de contexto são **recursivas**
- Utiliza como estrutura básica algum tipo de árvore, denominada **árvore de análise sintática** ou simplesmente **árvore sintática**

Introdução

- Existem duas categorias gerais de algoritmos para análise sintática, conforme é construída a análise sintática:
  - **ascendente**
  - **descendente**

Processo de análise sintática



- Em geral, a sequência de tokens não é um parâmetro explícito de entrada, mas ativado pelo analisador por procedimento de varredura, como ***getToken()***

- A etapa de análise sintática do compilador pode se resumir à sua ativação, que retorna o resultado à árvore sintática, por exemplo,

  arvoreSintatica = analisar()

Processo de análise sintática

- Em geral, a estrutura da árvore é definida como uma **estrutura de dados dinâmica**, em que **cada nó é composto por um registro** cujos campos contêm os atributos requeridos para o restante do processo de compilação

- A economia de espaço de armazenamento é sempre levada em consideração em proveito do aumento de desempenho

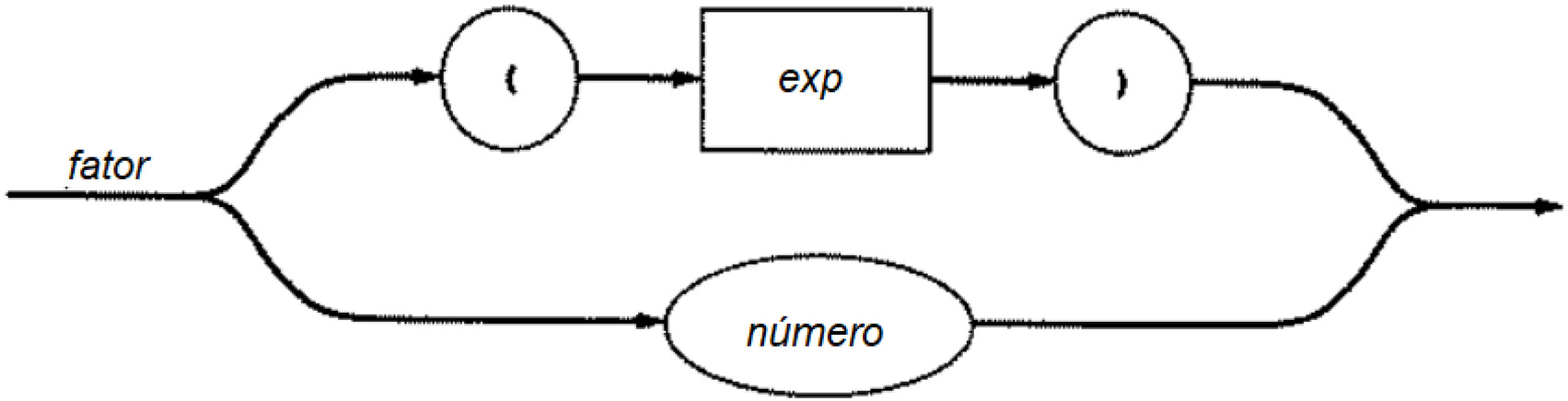
- Tratamento de erros: mais complexo do que a varredura, pois (i) precisa **recuperar** a situação sem erros e **seguir com a análise** para descobrir outros erros; (ii) e pode precisar efetuar a **correção de erros**

Diagramas sintáticos

- Representações gráficas visuais para regras de BNF estendida.
- Elementos:
  - círculos ou formas ovais indicam os terminais
  - quadrados ou retângulos indicam não terminais
  - linhas direcionadas representam sequências

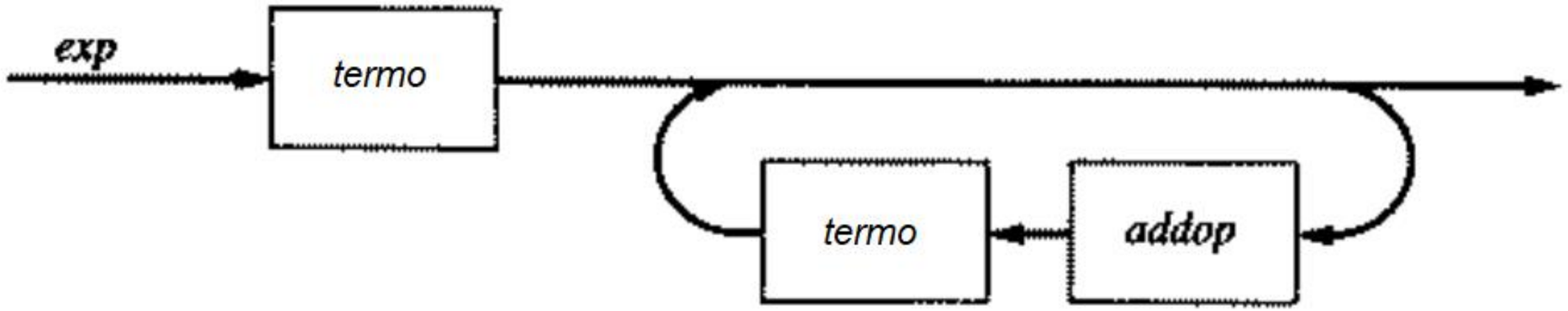
Diagramas sintáticos

- Exemplo: *termo* → *(exp)* | *número*



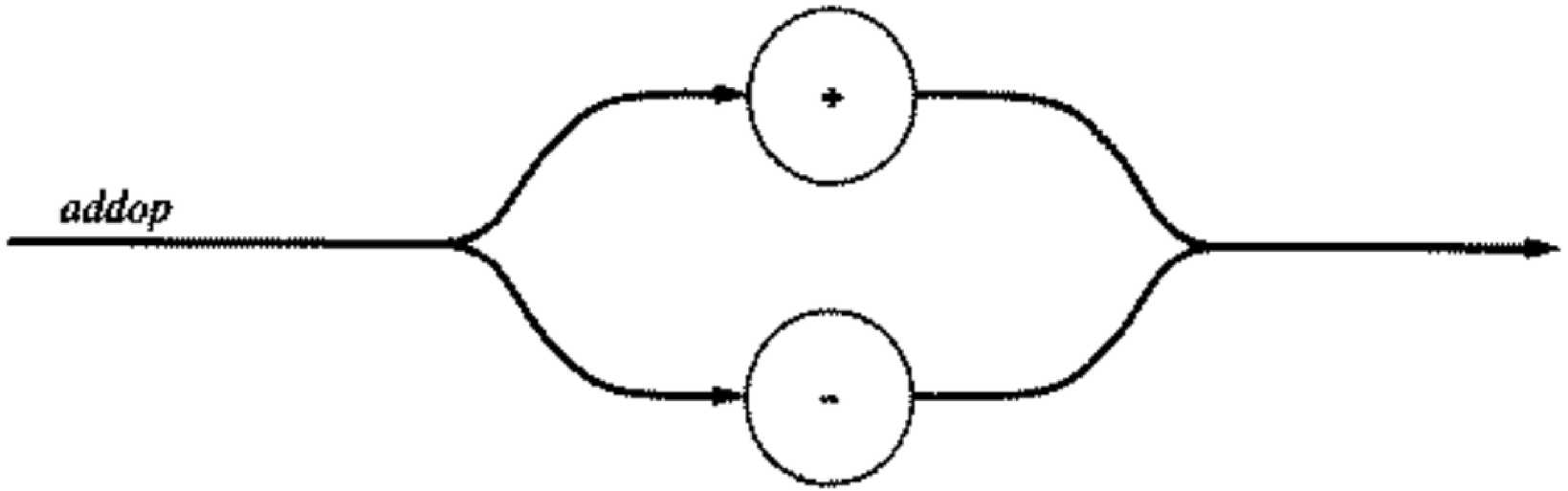
Diagramas sintáticos

- Exemplo: *exp* → *exp addop termo* | *termo*



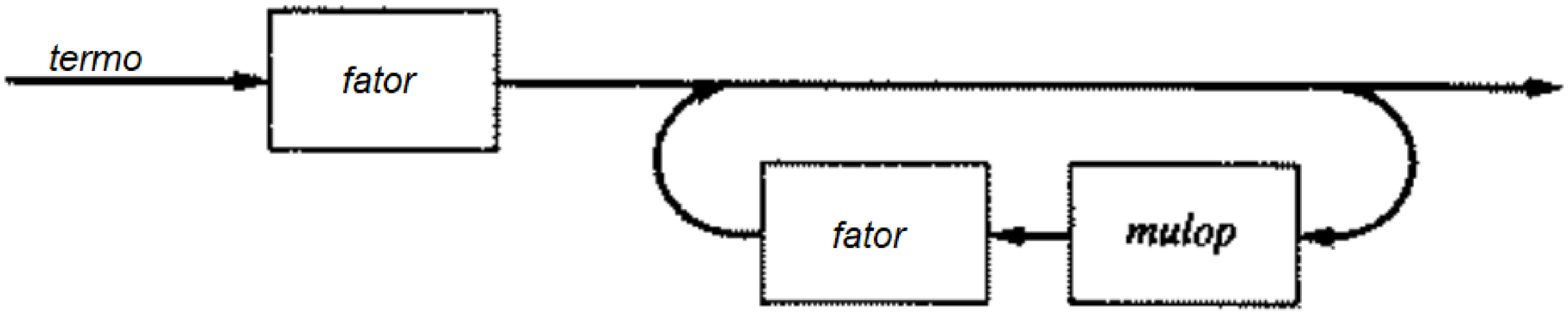
Diagramas sintáticos

- Exemplo: *addop* → *+* | *-*



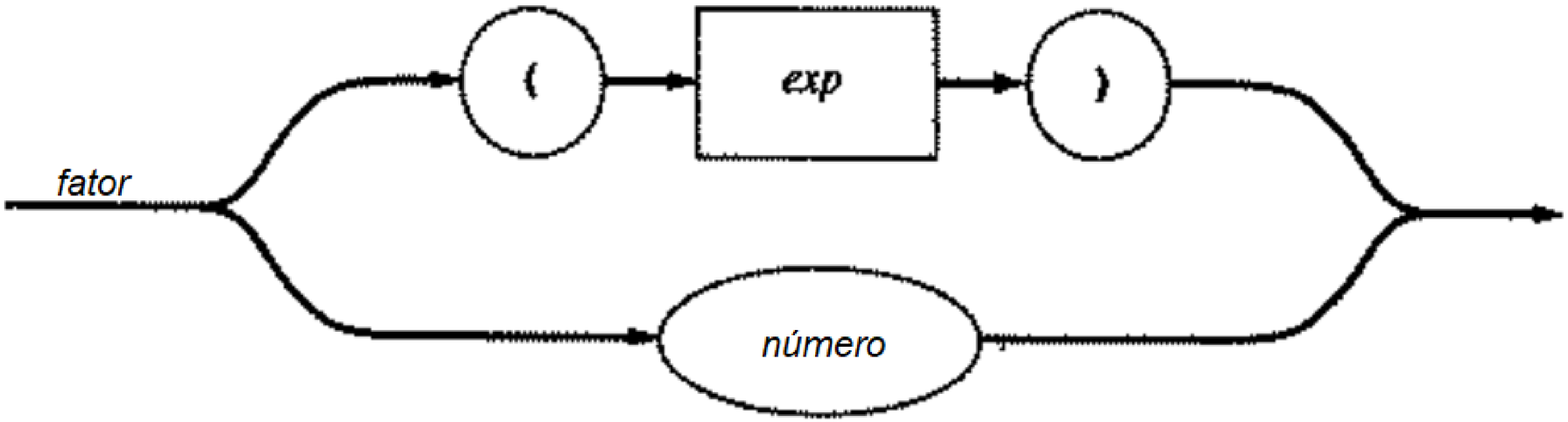
Diagramas sintáticos

- Exemplo: *termo* → *termo mulop fator* | *fator*



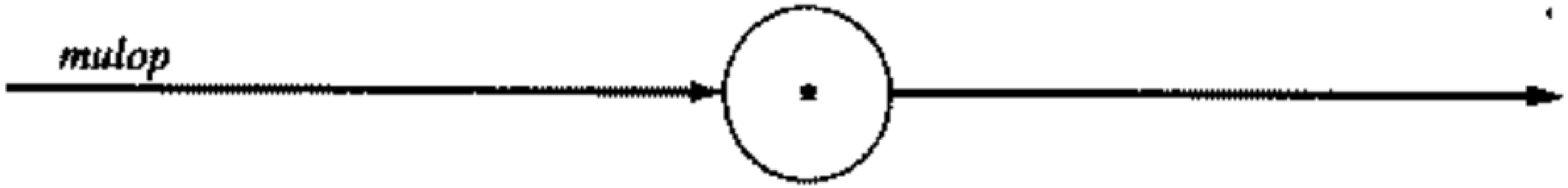
Diagramas sintáticos

- Exemplo: *fator* → *(exp)* | *número*



Diagramas sintáticos

- Exemplo: *mulop* → *



Árvores binárias

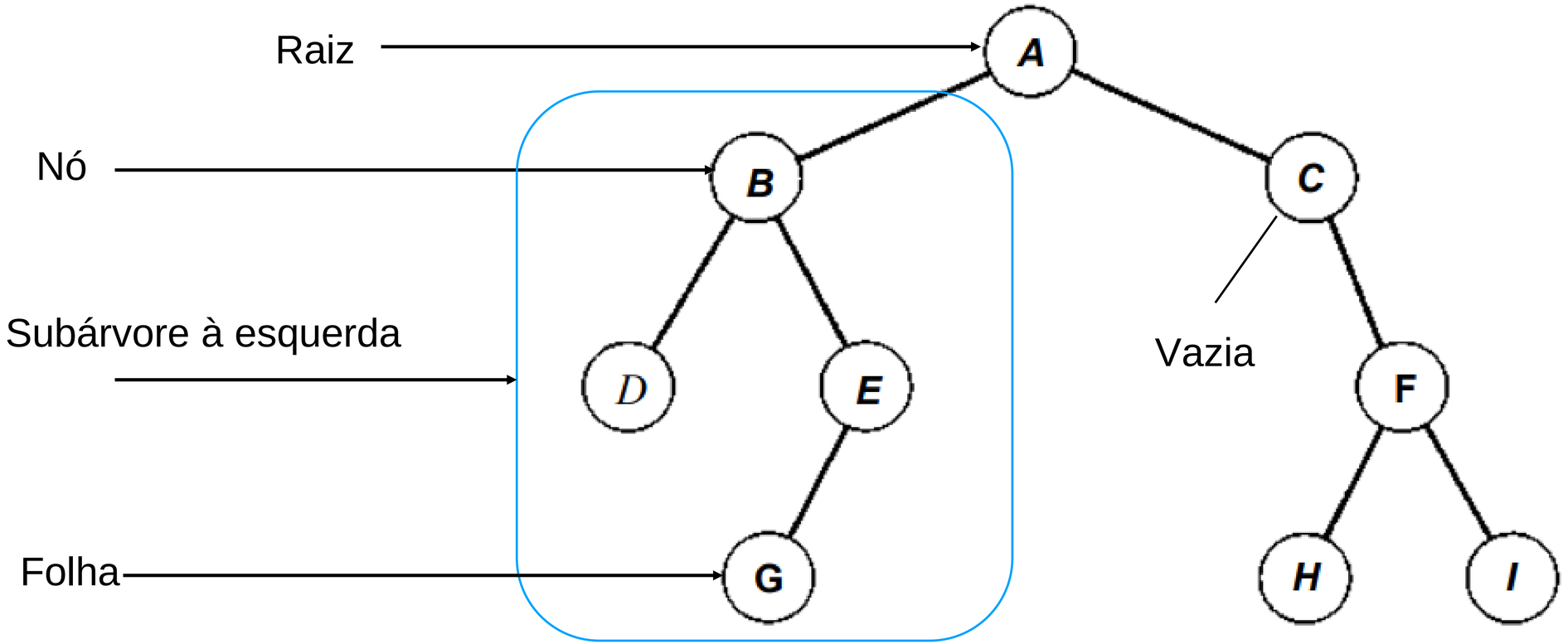


Figura 5.1.1 Uma árvore binaria.

Árvores

- Conjunto finito e não-vazio de elementos, no qual um elemento é chamado raiz e os elementos restantes são particionados em m >= 0 subconjuntos disjuntos, cada um dos quais sendo uma árvore em si mesmo.

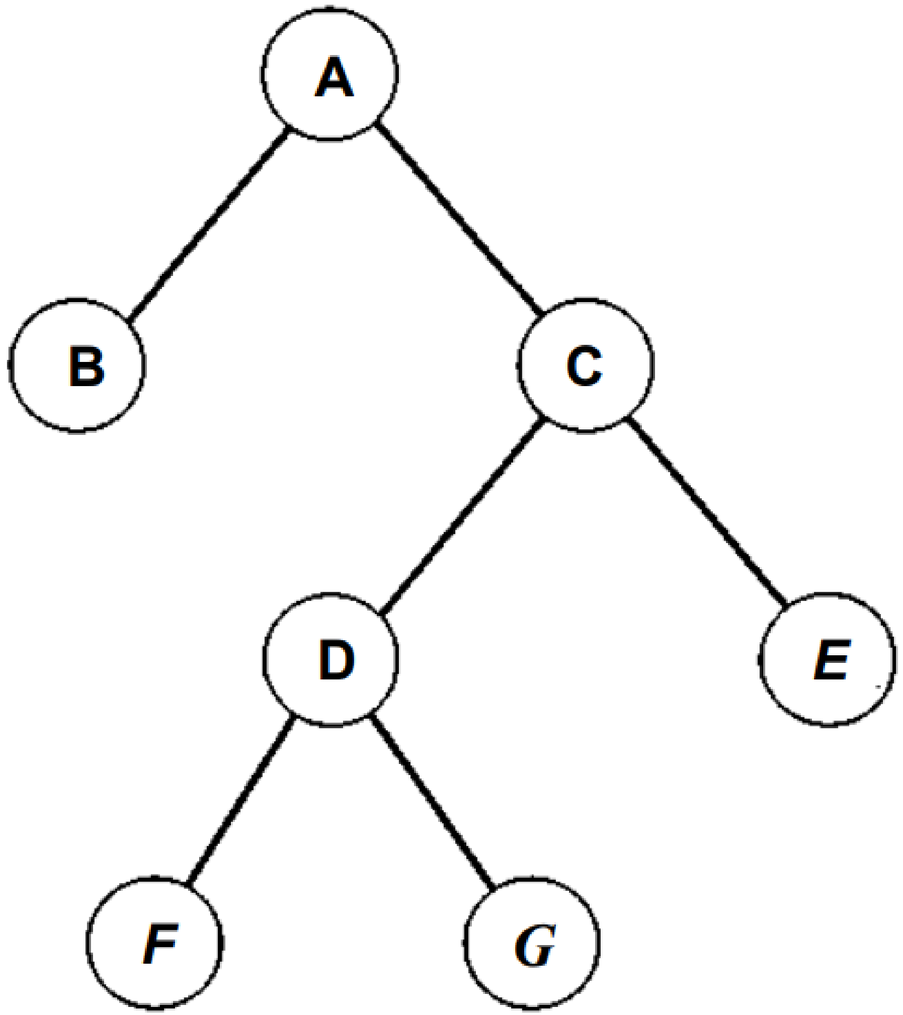
- Cada elemento numa árvore é chamado nó da árvore.

- Cada nó pode ser a raiz de uma árvore com zero ou mais subárvores. Um nó sem subárvores é uma folha. Usamos os termos/mi, filho, irmão, ancestral, descendente, nível e profundidade com a mesma conotação utilizada para as árvores binárias

Árvores

- Se todo nó que não é folha numa árvore binária tiver subárvores esquerda e direita não-vazias, a árvore será considerada uma **árvore estritamente binária**

- O nível de um nó numa árvore binária é definido como segue: **a raiz da árvore tem nível 0**, e o nível de qualquer **outro nó** na árvore é **um nível a mais que o nível de seu pai**.

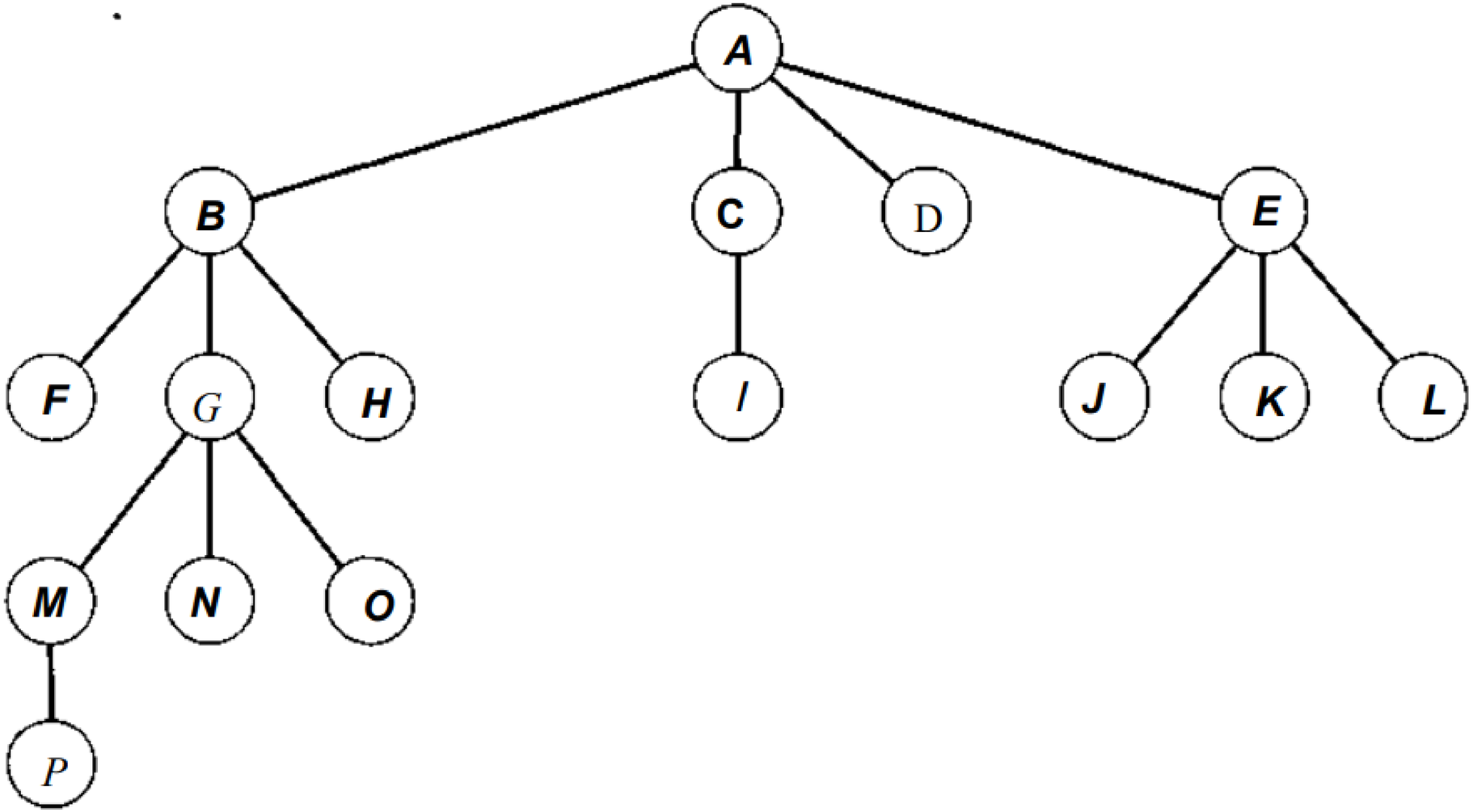


Árvores

- A **profundidade** de uma árvore binária significa o nível máximo de qualquer folha na árvore.

- Uma **árvore binária completa de profundidade *d*** é a árvore estritamente binária em que todas as folhas estejam no nível *d*.

Árvores



IBMEC.BR

/IBMEC

IBMEC

@IBMEC_OFICIAL

@IBMEC

Ibmec