Construção de Compiladores

Análise sintática

Professor: Luciano Ferreira Silva, Dr.

Análise sintática

- **Análise léxica (uso de gramáticas regulares – tipo 3):**
  - ✓ É adequada para identificar os símbolos básicos que compõem uma linguagem;
  - ✓ **Não** é adequada para identificar a forma como esses símbolos devem ser combinados

Análise sintática

- **Gramáticas livres de contexto (tipo 2) são adequadas para representar as características de sentenças de em linguagens de programação;**

- **Obs.: nem todas as construções de programação são passíveis de representação por esse tipo de gramática;**

  ✓ Faz-se necessário o uso de reconhecedores de sentenças livres de contexto e algumas estratégias heurísticas para automatizar a análise sintática.

Análise sintática

- Considere a expressão:

$$a = a + 2 * b;$$

- O analisador sintático inicia a análise da sentença solicitando o tipo do primeiro token, e segue e segue até encontrar um terminador de sentença, no caso (;)

Análise sintática

- **Uma das tarefas do analisador sintático é reconhecer a estrutura da expressão:**

1. No nível mais alto, este é um comando de atribuição, tem-se algo da forma

na qual o lado esquerdo (lesq) é um identificador (id)

Análise sintática

2. O lado direito (ldir) do comando de atribuição é uma operação de soma, que tem a forma

na qual o operando à esquerda (opere) é um identificador (id).

3. O operando à direita (operd) é uma operação de multiplicação, que tem a forma

na qual o operando à esquerda é uma constante (const) e o operando à direita é um identificador (id).

Reconhecimento de sentenças

- **Reconhecimento de sentenças ou *parsing*:**
  - ✓ É o procedimento que verifica se uma dada sentença pertence à linguagem gerada por uma gramática;
  - ✓ Deve reconhecer expressões do tipo:
    - Declarações;
    - Expressões aritméticas;
    - Construções de controle de controle de execução.

Reconhecimento de sentenças

- Por exemplo: considere uma gramática G que define um subconjunto de expressões aritméticas, apresentada abaixo;

Gramática G para expressões com soma e multiplicação

$E \rightarrow E + E$	Produção 1: contempla a soma
$E \rightarrow E \times E$	Produção 2: propicia a multiplicação
$E \rightarrow (E)$	Produção 3: possibilita o uso de parênteses
$E \rightarrow v$	Produção 4: terminal

Reconhecimento de sentenças

- **Uma sentença é dita válida em uma dada gramática apenas quando existe pelo menos uma seqüência de aplicação de produções que permita obter a sentença a partir do símbolo sentencial;**

- **Há duas formas possíveis de derivar essa seqüência de produções para validar uma sentença:**

  - ✓ O procedimento de reconhecimento descendente *(top-down);*

  - ✓ O procedimento de reconhecimento ascendente *(bottom-up);*

Reconhecimento descendente

- **Segue os seguintes parâmetros:**

1. O ponto de partida é o símbolo sentencial;
2. Seleciona-se uma produção apropriada que aproxime a forma sentencial da sentença;
3. Enquanto há símbolos não-terminais na forma sentencial, produções são selecionadas para levar a forma sentencial até a sentença;
4. Se é possível obter a sentença com estas derivações então ela é reconhecida como válida na gramática;
5. Caso contrário, a sentença não faz parte da gramática.

Reconhecimento ascendente

- **Segue os seguintes parâmetros:**

1. Procura-se produções cujo lado direito combine com os símbolos que estão na sentença;
2. Substitui-se o símbolo ou seqüência de símbolos que combinam com o lado direito pelo símbolo não-terminal do lado esquerdo da produção;
3. A sentença será reconhecida como válida se for possível, ao final dessa sequencia de substituição obter como resultado apenas o símbolo sentencial.

Exemplo de reconhecimento

- Um analisador sintático para a gramática G, com v = x|y|z, pode reconhecer que a expressão (x + y) * z é válida.

1. O analisador léxico transforma essa expressão em:

Procedimento descendente

- **Primeira possibilidade:**

1. $E \overset{2}{\Rightarrow} E \times E$
2. $E \times E \overset{3}{\Rightarrow} (E) \times E$
3. $(E) \times E \overset{1}{\Rightarrow} (E + E) \times E$
4. $(E + E) \times E \overset{4}{\Rightarrow} (v + E) \times E$
5. $(v + E) \times E \overset{4}{\Rightarrow} (v + v) \times E$
6. $(v + v) \times E \overset{4}{\Rightarrow} (v + v) \times v$

- **Seqüência: 2, 3, 1, 4, 4, 4**
- **Segunda possibilidade:**

1. $E \overset{2}{\Rightarrow} E \times E$
2. $E \times E \overset{3}{\Rightarrow} (E) \times E$
3. $(E) \times E \overset{4}{\Rightarrow} (E) \times v$
4. $(E) \times v \overset{1}{\Rightarrow} (E + E) \times v$

Procedimento descendente

5. $(E + E) \times v \overset{4}{\Rightarrow} (v + E) \times v$
6. $(v + E) \times v \overset{4}{\Rightarrow} (v + v) \times v$

- **Seqüência: 2, 3, 4, 1, 4, 4**
- **Terceira possibilidade:**

1. $E \overset{2}{\Rightarrow} E \times E$
2. $E \times E \overset{4}{\Rightarrow} E \times v$
3. $E \times v \overset{3}{\Rightarrow} (E) \times v$
4. $(E) \times v \overset{1}{\Rightarrow} (E + E) \times v$
5. $(E + E) \times v \overset{4}{\Rightarrow} (E + v) \times v$
6. $(E + v) \times v \overset{4}{\Rightarrow} (v + v) \times v$

- **Seqüência: 2, 4, 3, 1, 4, 4**

Procedimento ascendente

1. $(\underline{v} + v) \times v \overset{4}{\Leftarrow} (E + v) \times v$
2. $(E + \underline{v}) \times v \overset{4}{\Leftarrow} (E + E) \times v$
3. $(E + E) \times \underline{v} \overset{4}{\Leftarrow} (E + E) \times E$
4. $\underline{(E + E)} \times E \overset{1}{\Leftarrow} (E) \times E$
5. $\underline{(E)} \times E \overset{3}{\Leftarrow} E \times E$
6. $\underline{E \times E} \overset{2}{\Leftarrow} E$

- **Seqüência: 4, 4, 4, 1, 3, 2**

Derivações canônicas

- **Problema: dada uma sentença composta de símbolos terminais da linguagem**
  - ✓ Encontrar uma seqüência de derivações, a partir do símbolo sentencial, para indicar a sentença como válida ou inválida na linguagem.
- **Solução: derivações canônicas**
  - ✓ Elas são uma forma sistemática de selecionar qual o símbolo será substituído (qual produção será aplicada).

Derivação canônica

- **Duas possibilidades consideradas:**
  - ✓ Derivação canônica mais à esquerda (*leftmost derivation*)
    - Aplicar uma produção da gramática ao símbolo não terminal que está mais a esquerda;
  - ✓ Seqüência de reconhecimento mais à esquerda (*lefmost parse*): usa apenas a derivação mais à esquerda;
    - Exemplo: seqüência 2, 3, 1, 4, 4, 4
  - ✓ Derivação canônica mais à direita (*rightmost derivation*)
    - Aplicar uma produção da gramática ao símbolo não terminal que está mais a direita;

Derivação canônica

- ✓ Seqüência de reconhecimento mais à direita (*rightmost parse*): usa apenas a derivação mais à direita ;
  - Exemplo: seqüência de produções 2, 4, 3, 1, 4, 4. A seqüência é dada por 4, 4, 1, 3, 4, 2.

Obs.:

- ✓ Uma dada produção é usada o mesmo número de vez em ambas derivações
- ✓ Ao adotar uma das formas canônicas para o reconhecimento o analisador tem uma única estratégia para validar uma sentença;
  - Desprende-se então de escolhas aleatórias.

O uso de árvores

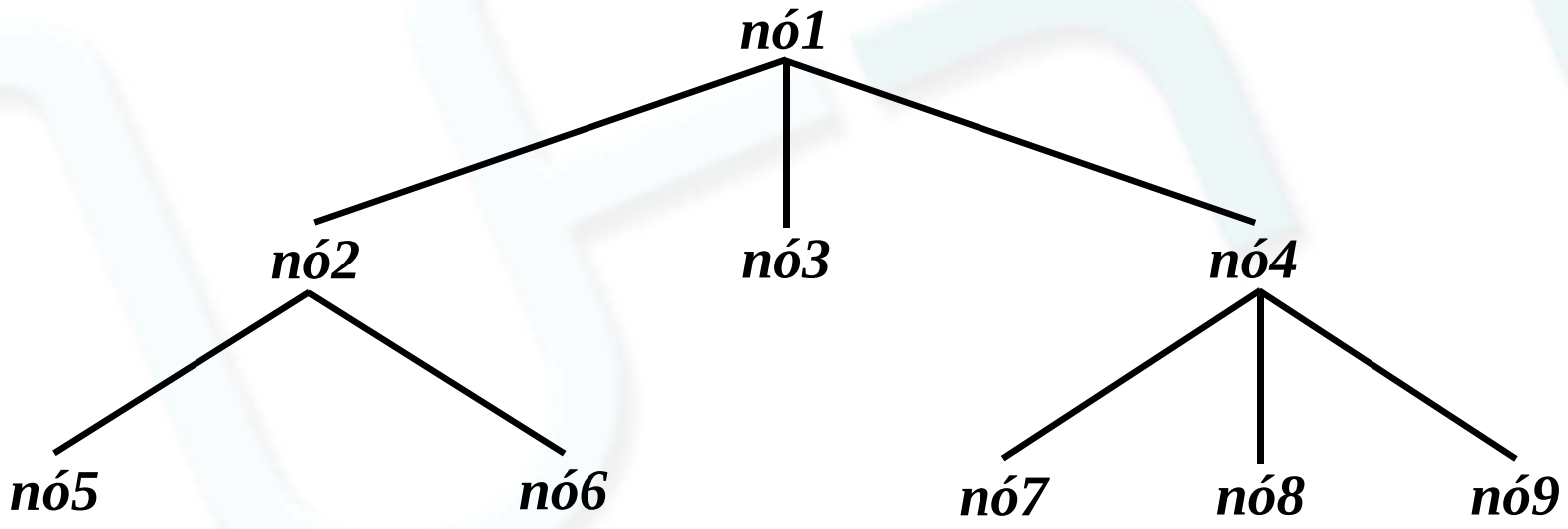
- O processo de reconhecimento de uma sentença é apenas parte das tarefas de um compilador;
- O resultado do reconhecimento precisa ser recuperado para que, em um momento posterior, o compilador possa produzir o código equivalente na linguagem alvo;
- Para tanto, é preciso preservar a informação sobre a seqüência de reconhecimento em uma estrutura de dados apropriada – nesse caso uma estrutura tipo árvore.

Árvores

- **Uma árvore é uma estrutura de dados que:**
  - ✓ Possui um conjunto finito de elemento denominados nós;
  - ✓ Um desses nós é especialmente designado como nó raiz;
    - Ele é o ponto de entrada para obter todos os elementos da estrutura;
  - ✓ Subordinado a um nó podem estar associados subconjuntos disjuntos de nós;
    - Cada um desses conjuntos é organizado na forma de uma árvore, denominada subárvore.

Árvores

- A representação gráfica usualmente utilizada para árvores posiciona a raiz no topo do diagrama, com as subárvores abaixo dos nós.



Árvores

- **O número de subárvores de um nó determina o grau do nó:**
  - ✓ O nó *nó1* tem grau 3; o nó *nó2* tem grau 2; o nó *nó3* tem grau 0
- **O grau da árvore é o maior grau de todos os nós: no caso anterior 3;**
- **Os nós de grau 0 são denominados folhas da árvore: *nó3, nó5, nó6, nó7, nó8, nó9;***
- **Usa-se os nomes pais e filhos para definir hierarquias:**
  - ✓ Os nós *nó5* e *nó6* são filho do *nó2*;
  - ✓ O nó *no4* é pai dos nós *nó7*, *nó8*, *nó9*;
  - ✓ Todo tem apenas um pai e ele é único, com exceção do nó raiz que não tem um nó pai;