COMPILADORES
GRAMÁTICAS

Mariza A S. Bigonha e Roberto S. Bigonha
UFMG

8 agosto de 2011

Gramáticas

Definição 1:
uma gramática regular **contém somente produções da forma $\alpha \rightarrow \beta$, onde $|\alpha| \leq |\beta|$, $\alpha \in V_N$ e β tem a forma aB ou a, onde a $\in V_T$ e B $\in V_N$.**

Exemplo: S $\rightarrow$ aS
S $\rightarrow$ aB
B $\rightarrow$ bC
C $\rightarrow$ aC
C $\rightarrow$ a

A linguagem L(G_1) = $\{a^n b a^m \mid$ n,m $\geq 1 \}$ é gerada pela gramática regular dada.

··· Gramáticas

Definição 2:
uma gramática sensível ao contexto **contém somente produções da forma $\alpha \rightarrow \beta$, onde $|\alpha| \leq |\beta|$, onde $|\alpha|$ denota o comprimento de α.**

Exemplo: S $\rightarrow$ aSBC
S $\rightarrow$ abC
bB $\rightarrow$ bb
bC $\rightarrow$ bc
CB $\rightarrow$ BC
cC $\rightarrow$ cc

A linguagem L(G_2) = $\{a^n b^n c^n \mid$ n $\geq 1 \}$ é gerada pela gramática sensível ao contexto dada.

··· Gramáticas

Definição 3:

uma gramática livre do contexto **contém somente produções da forma $\alpha \rightarrow \beta$, onde $|\alpha| \leq |\beta|$ e $\alpha \in V_N$.**

Exemplo: S $\rightarrow$ aCa
C $\rightarrow$ aCa
C $\rightarrow$ b

A linguagem L(G_3) = $\{a^n b a^n \mid$ n $\geq 1 \}$ é gerada pela gramática livre do contexto dada.

Definição Sintática

Uma Gramática Livre de Contexto **possui 4 componentes:**

1. **Conjunto de símbolos** *terminais*, **às vezes chamados de** *tokens*.
2. **Conjunto de símbolos** *não-terminais*, **às vezes chamados de** *variáveis sintáticas*.
3. **Conjunto de** *produções* **da forma:**
   não-terminal $\rightarrow$ **seqüência de terminais e/ou não-terminais.**
4. **Um não-terminal designado símbolo de partida.**

Linguagens Não Livres-de-Contexto

- Exemplo 1: **Abstração da verificação se identificadores foram declarados.**

  $L_1 =$ { **wcw** | **w em** $(a|b)^*$ }

- Exemplo 2: **Abstração definição/chamada de procedimentos.**

  $L_2 = \{ a^n b^m c^n d^m \mid \mathbf{n \geq 1 \text{ e } m \geq 1} \}$

- Exemplo 3: **Informalmente, autômato finito "não sabe contar"; CFG pode contar até duas qualidades.**

  $L_3 = \{ a^n b^n c^n \mid \mathbf{n \geq 0} \}$ **não é CFL**
  $L_4 = \{ a^n b^n \mid \mathbf{n \geq 0} \}$ **é CFL.**

Linguagens Livres-de-Contexto (CFL)

- **Exemplo 1:** $L_1 = \{ \mathbf{wc}w^R \mid \mathbf{w \text{ em }} (a|b)^* \}$
  S $\rightarrow$ **aSa** | **bSb** | **c**
- **Exemplo 2:** $L_2 = \{ a^n b^m c^m d^n \mid \mathbf{n \geq 1 \text{ e } m \geq 1} \}$
  S $\rightarrow$ **aSd** | **aAd**
  A $\rightarrow$ **bAc** | **bc**
- **Exemplo 3:** $L_2 = \{ a^n b^n c^m d^m \mid \mathbf{n \geq 1 \text{ e } m \geq 1} \}$
  S $\rightarrow$ **AB**
  A $\rightarrow$ **aAb** | **ab**
  B $\rightarrow$ **cBd** | **cd**
- **Exemplo 4:** $L_3 = \{ a^n b^n \mid \mathbf{n \geq 1} \}$
  S $\rightarrow$ **aSb** | **ab**

Mais Definições e Notações

Terminais: **delimitadores, números, letras minúsculas, caracteres entre aspas.**

Não-terminais: **nomes, letras maiúsculas.**

Strings (terminais e não-terminais): **letras gregas minúsculas.**

Exemplo:

A $\rightarrow$ **BC**
B $\rightarrow$ **aBb** | **ab**
C $\rightarrow$ **bCa** | **ba**

Exemplos:

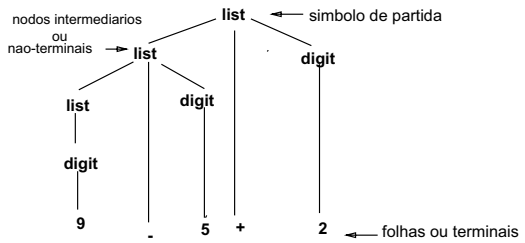
ifstmt → "if" exp "then" stmtl |
"if" exp "then" stmtl "else" stmtl
stmtl → stmt |
stmtl ; stmt
stmt → $\mathcal{E}$ |
assigstmt |
ifstmt |
...

Notação abreviada:

ifstmt → "if" exp "then" stmtl ["else" stmtl]
stmtl → stmt { ; stmt }

Mais Exemplos

list → list + digit |
list - digit |
digit
digit → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9



Mais Conceitos:

- "Reconhecimento" (Parsing)
- Árvore de "reconhecimento" ou Árvore de "derivação" (parsing)
- String "gerado por" ou "derivado de" uma árvore de "derivação" . Também chamado produto ou yield da árvore.
- Gramática ambigua
- Sentença

Derivações

Uma gramática deriva cadeias começando com o símbolo inicial e substituindo repetidamente um não-terminal pelo corpo de uma produção para esse não-terminal.

As cadeias de terminais que podem ser derivadas do símbolo inicial formam a *linguagem* definida pela gramática.

A linguagem definida pela gramática dada consiste em listas (`list`) de dígitos (`digit`) separados por sinais de adição e subtração.

1. list → list + digit |
2. list - digit |
3. digit
4. digit → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

As dez produções para o não-terminal `digit` permitem que ele represente qualquer um dos terminais 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9.

· · · Derivações

1. list → list + digit |
2. list - digit |
3. digit
4. digit → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Pela produção (3), um único dígito por si só é uma lista.

As produções (1) e (2) expressam a regra de que qualquer lista seguida por um sinal de adição ou subtração e depois outro dígito compõe uma nova lista.

As produções de (1) a (4) são tudo aquilo de que precisamos para definir a linguagem desejada.

· · · Derivações

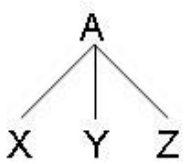
Exemplo: é possível deduzir que $9-5+2$ é uma lista da seguinte maneira:

- 9 é uma *lista* pela produção (3), pois 9 é um *dígito*.
- 9-5 é uma *lista* pela produção (2), pois 9 é uma *lista* e 5 é um *dígito*.
- 9-5+2 é uma *lista* pela produção (1), pois 9-5 é uma *lista* e 2 é um *dígito*.

1. list → list + digit |
2. list - digit |
3. digit
4. digit → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Árvores de derivação

Uma árvore de derivação mostra de forma representativa como o símbolo inicial de uma gramática deriva uma cadeia na linguagem.



Se o não-terminal A possui uma produção $A \rightarrow XYZ$, uma árvore de derivação pode ter um nó interior rotulado com A, com três filhos chamados X, Y e Z, da esquerda para a direita.

Árvores de derivação

Dada uma gramática livre de contexto, uma árvore de derivação de acordo com a gramática é uma árvore com as seguintes propriedades:

- A raiz é rotulada pelo símbolo inicial.
- Cada folha é rotulada por um terminal ou por ϵ.
- Cada nó interior é rotulado por um não-terminal.
- Se A é o não-terminal rotulando algum nó interior e X_1, X_2, ..., X_n, são os rótulos dos filhos desse nó da esquerda para a direita, deve haver uma produção $A \rightarrow X_1 X_2 \ldots X_n$. Cada X_1, X_2, ..., X_n, representa um símbolo que é um terminal ou um não-terminal.

  Caso Especial se $A \rightarrow$ é uma produção, um nó rotulado com A pode ter um único filho rotulado com ϵ.

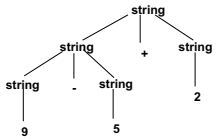
Gramática Ambigua

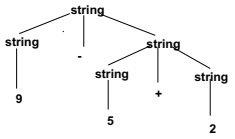
Uma gramática pode ter mais de uma árvore de derivação gerando determinada cadeia de terminais. Essa gramática é considerada ambígua.

string → **string + string** |
string − string | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**

· · · Gramática Ambigua

string → **string + string** |
string − string | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**





Associatividade de Operadores

7 + 3 − 2 = (7 + 3) − 2 ≡ esquerda

7 ** 2 ** 3 = 7 **(2 ** 3) ≡ direita

Precedência de Operadores
1 + 3 * 5 = 1 + (3 * 5) *** >> +**
(1 + 3) * 5

Associatividade de "+" e "*" nao resolve a ambiguidade

Associatividade e prioridade na gramática

Criar tabela de precedência:

↓ **(1) + − associa esquerda** *expr*
(2) * / associa esquerda *term*
(3) ** associa direita `expexp factor`

- Definir produções para unidades básicas

factor → **"digit"** |
(expr)

- Definir produções por níveis de prioridade crescente, respeitando a associatividade.

expr → **expr + term** |
expr − term |
term
term → **term * expexp** |
term / expexp |
expexp
expexp → **factor ** expexp** |
factor

FIM