Análise Sintática

Sintaxe => Usualmente → Gramática Livre do Contexto (GLC)
→ BNF (Backus-Naur Form)

Gramática Livres de Contexto / Estruturas Recursivas

comando => IF expressao THEN expressao ELSE expressao
expressao => (expressao) OR (expressao) | (expressao) AND (expressao)
expressao => numero
expressao => numero oper_logico numero
oper_logico => < | > | <= | >= | <>

Conceitos utilizados:

* Terminais: IF, >, <, <>, numero, ...
* Não-terminais: comando, expressao, op_logico
* Símbolo Inicial: Um dos não-terminais (S)
* Produções: A => α ou A => $\alpha_1 | \alpha_2 | \alpha_3 | ...$

Árvore de Derivação: é a representação gráfica de uma derivação de uma sentença particular, de acordo com a gramática especificada.

1

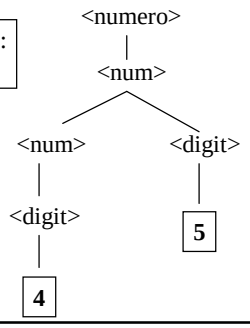
Análise Sintática

Árvore de Derivação: é a representação gráfica de uma derivação de uma sentença particular, de acordo com a gramática especificada.

Exemplo 1:

G = ({<numero>,<num>,<digit>}, {0,1,2,...,9), P, <numero>)
P = { <numero> => <num>
<num> => <num> <digit> | <digit>
<digit> => 0 | 1 | ... | 9 }

Sentença: **45**



Exemplo 2:

G = ({E}, { +, -, *, /, (,) , x }, P, E)
P = {
E => E + E | E - E
E => E * E | E / E
E => (E) | x
}

2

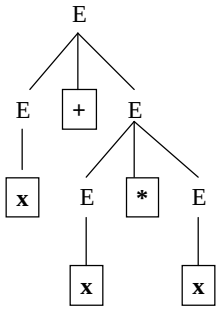
Análise Sintática

Árvore de Derivação x Derivações:

Exemplo 2:

G = ({E}, { +, -, *, /, (,) , x }P, E)
P = {
E => E + E | E - E
E => E * E | E / E
E => (E) | x
}

Sentença:
x + x * x



Derivações:

(a)	**(b)**	**(c)**
E => E + E	**E => E + E**	**E => E + E**
=> x + E	**=> E + E * E**	**=> E + E * E**
=> x + E * E	**=> E + E * x**	**=> x + E * E**
=> x + x * E	**=> E + x * x**	**=> x + x * E**
=> x + x * x	**=> x + x * x**	**=> x + x * x**

3

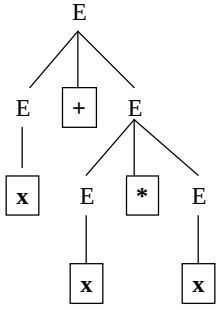
Análise Sintática

Árvore de Derivação x Derivações:

Exemplo 2:

G = ({E}, { +, -, *, /, (,) , x }P, E)
P = {
E => E + E | E - E
E => E * E | E / E
E => (E) | x
}

Sentença:
x + x * x



Derivações:

(a)	**(b)**	**(c)**
E => E + E	**E => E + E**	**E => E + E**
=> x + E	**=> E + E * E**	**=> E + E * E**
=> x + E * E	**=> E + E * x**	**=> x + E * E**
=> x + x * E	**=> E + x * x**	**=> x + x * E**
=> x + x * x	**=> x + x * x**	**=> x + x * x**

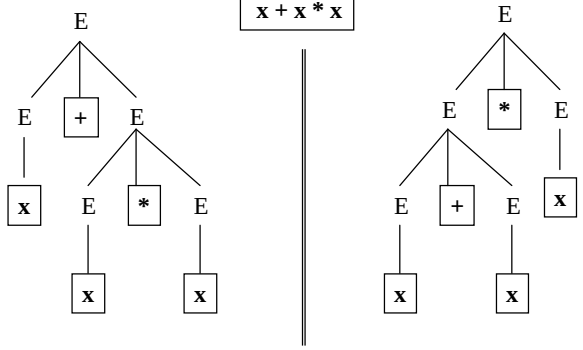
4

Análise Sintática

Gramáticas: Características e Propriedades

* Gramáticas Ambíguas: permite construir mais de uma árvore de derivação

Sentença:
x + x * x



Qual das duas árvores deve ser derivada?

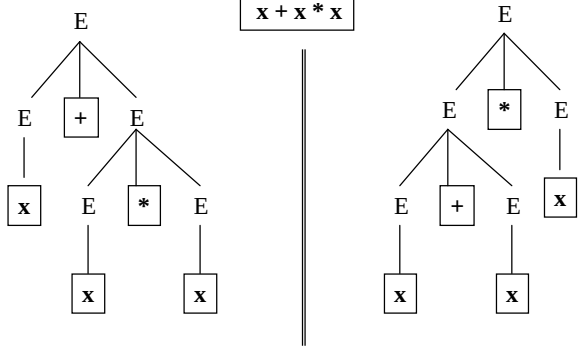
5

Análise Sintática

Gramáticas: Características e Propriedades

* Gramáticas Ambíguas: permite construir mais de uma árvore de derivação

Sentença:
x + x * x



Qual das duas árvores deve ser derivada?
A gramática não permite determinar...
A precedência de operadores poderia ser usada neste caso (árvore à esquerda)...

6

Análise Sintática

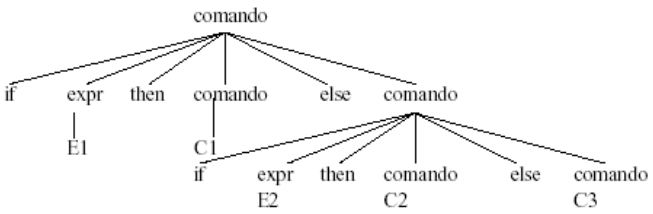
Gramáticas: Características e Propriedades

* Gramáticas Ambíguas: permite construir mais de uma árvore de derivação

Outro exemplo: comando => IF expressao THEN comando |
IF expressao THEN comando ELSE comando |
outro_comando

if E1 then C1 else if E2 then C2 else C3

tem a árvore de derivação apresentada a seguir:



7

Análise Sintática

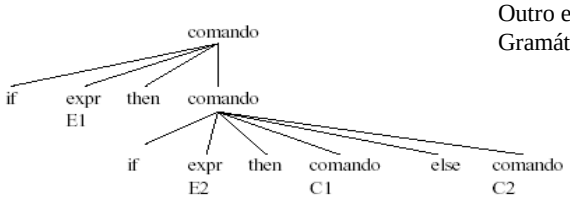
Gramáticas: Características e Propriedades

* Gramáticas Ambíguas: permite construir mais de uma árvore de derivação

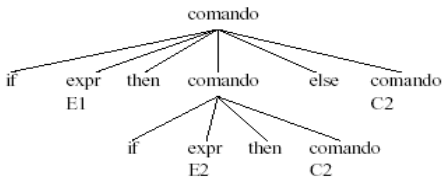
if E1 then if E2 then C1 else C2

tem duas árvores de derivação associadas:

Outro exemplo...
Gramática ambígua!



ou:



8

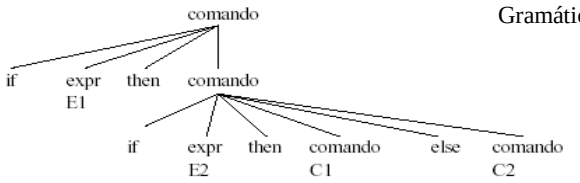
Análise Sintática

Gramáticas: Características e Propriedades

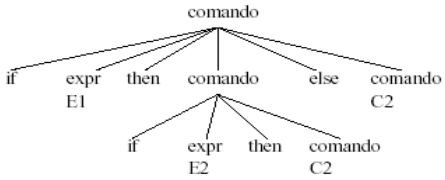
* Gramáticas Ambíguas: permite construir mais de uma árvore de derivação

if E1 then if E2 then C1 else C2

tem duas árvores de derivação associadas:



ou:



Outro exemplo...
Gramática ambígua!

Solução...
* Associar o ELSE com o THEN mais próximo!
* Incorporar isto na gramática.

9

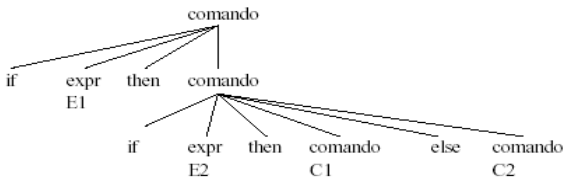
Análise Sintática

Gramáticas: Características e Propriedades

* Gramáticas Ambíguas: permite construir mais de uma árvore de derivação

if E1 then if E2 then C1 else C2

tem duas árvores de derivação associadas:



comando → comando_não_marcado
| comando_marcado

comando_marcado → if expr then comando_marcado else comando_marcado
| outro

comando_não_marcado → if expr then comando
| if expr then comando_marcado else comando_não_marcado

10

Análise Sintática

Gramáticas: Características e Propriedades

* Gramáticas sem ciclos: é uma GLC que não possui derivações da forma
  $A \Rightarrow^{+} A$ para algum símbolo A não-terminal

* Gramáticas ε-Livre: é uma GLC que não possui produções do tipo
  $A \Rightarrow \varepsilon$ (exceto talvez para a produção inicial, $S \Rightarrow \varepsilon$)

* Gramática recursiva à esquerda:é uma GLC que possui produções do tipo
  $A \Rightarrow^{+} A\alpha$ para algum símbolo A não-terminal

Alguns tipos de reconhecedores (top-down) não aceitam este tipo de gramáticas, exigindo que seja eliminada a recursão a esquerda.

$A \rightarrow A\,\alpha$
$\quad / \beta$

pode-se substituir as produções pelas seguintes, sem recursividade e sem perda de significado:

$A \rightarrow \beta\,A'$
$A' \rightarrow \alpha\,A'$
$\quad / \varepsilon$

11

Análise Sintática

Gramáticas: Características e Propriedades

* Gramáticas fatorada à esquerda: é uma GLC que não apresenta produções do tipo
  $A \Rightarrow \alpha\beta_1 \mid \alpha\beta_2$

Fatorando à esquerda a gramática acima, obtém-se: (necessário para análise top-down)
  $A \Rightarrow \alpha A'$
  $A' \Rightarrow \beta_1 \mid \beta_2$

Exemplo de gramática que pode ser fatorada à esquerda:

comando => IF expr THEN comando ELSE comando |
IF expr THEN comando

Ficaria...

comando => IF expr THEN comando resto_do_if
resto_do_if => ELSE comando | ε

> Fatoração é necessária para análise top-down do tipo “descendente preditivo”

12

Análise Sintática

Tipos de Analisadores Gramáticais:

* Analisadores TOP-DOWN:
  Árvore de derivação começa pela raiz indo para as folhas => Análise Descendente
  Tipo LL(1): Left to right / Leftmost derivation / 1 symbol each time - lookahead
  - Recursivo com Retrocesso (Backtracking)
  - Recursivo Preditivo
  - Preditivo Tabular (não recursivo - pilha + tabela)

* Analisadores BOTTON-UP:
  Árvore de derivação começa pelas folhas indo para a raiz => Análise Ascendente
  Shift / Reduce <= Análise Redutiva
  - LR(k) => i) SLR (simple)
  ii) LR canônicos
  iii) LALR (lookahead LR)
  Tipo LALR(1) => Yacc / Bison - Left to right / Rightmost derivation / 1 each time

* Recuperação de Erros
* Tradução dirigida pela sintaxe

13

Análise Sintática

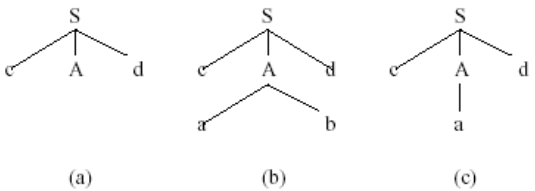
Tipos de Analisadores Gramáticais:

* Analisadores TOP-DOWN:
  Árvore de derivação começa pela raiz indo para as folhas => Análise Descendente
  Tipo LL(1): Left to right / Leftmost derivation / 1 symbol each time - lookahead
  - Recursivo com Retrocesso (Backtracking)
  - Recursivo Preditivo
  - Preditivo Tabular (não recursivo - pilha + tabela)

Exemplo:

$S \rightarrow cAd$
$A \rightarrow ab$
$\mid a$

Sentença: **cad**



14

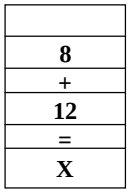
Análise Sintática

Tipos de Analisadores Gramáticais:

* Analisadores BOTTON-UP:
  Árvore de derivação começa pelas folhas indo para a raiz => Análise Ascendente
  Shift / Reduce <= Análise Redutiva
  Tipo LALR(1) => Yacc / Bison - Left to right / Rightmost derivation / 1 each time

Exemplo: Shift (empilha) / Reduce (Busca chegar no S - símbolo inicial)

Sentença:
X = 12 + 8



P = { statement => name = expr |
expr
expr => number + number |
number - number }

P = { statement => name = expr |
expr
expr => number
expr => expr + number |
expr - number }

Empilha: shifting tokens on to the internal stack

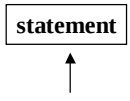
Análise Sintática

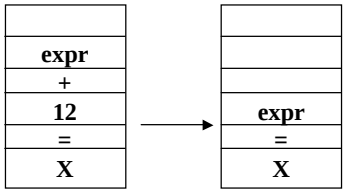
Tipos de Analisadores Gramáticais:

* Analisador Tipo LALR(1) => Yacc / Bison - Shift / Reduce

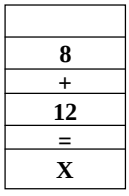
P = { statement => name = expr |
expr
expr => number
expr => expr + number |
expr - number }

Válido:





Sentença:
X = 12 + 8



Reduce: reduz em direção a raiz

Pop(8) : expr => number
Push(expr)
Pop(expr+12) : expr => expr + number
Push(expr)
Pop(X=expr) : statement => name = expr
Push(statement)
Finish => Initial Symbol

Empilha: shifting tokens on to the internal stack

Análise Sintática

Tipos de Analisadores Gramáticais:

* Analisador Tipo LALR(1) => Yacc / Bison - Shift / Reduce

Problemas... "HORSE AND CART"

```
phrase        => cart_animal AND CART |
                 work_animal AND PLOW
cart_animal   => HORSE | GOAT
work_animal   => HORSE | OX
```

Yacc: Gramáticas ambíguas / lookahead de 2 símbolos adiante

Praticando com o YACC / BISON...

```
P = {
      statement  => name = expr   |
                    expr
      expr  => number + number  |
               number - number

   }
```

```
P = {
      statement  => name = expr  |
                    expr
       expr  => number
       expr  => expr + number    |
                expr - number
    }
```

17

Análise Sintática

Tipos de Analisadores Gramáticais:

* Recuperação de Erros

> Modo pânico: "descarta tokens até re-sincronizar"
Exemplo: em Pascal procura pelo próximo ";" ou "end"

> Recuperação local baseada em expressões

> Produção de erros: "gramática com produções para detectar erros"

* Tradução dirigida pela sintaxe

Tradução dirigida por sintaxe é uma técnica que permite realizar tradução (geração de código) concomitantemente com a análise sintática.
Ações semânticas são associadas às regras de produção da gramática.

Exemplo: avaliação de expressões numéricas

18