ACH2043
INTRODUÇÃO À TEORIA DA COMPUTAÇÃO

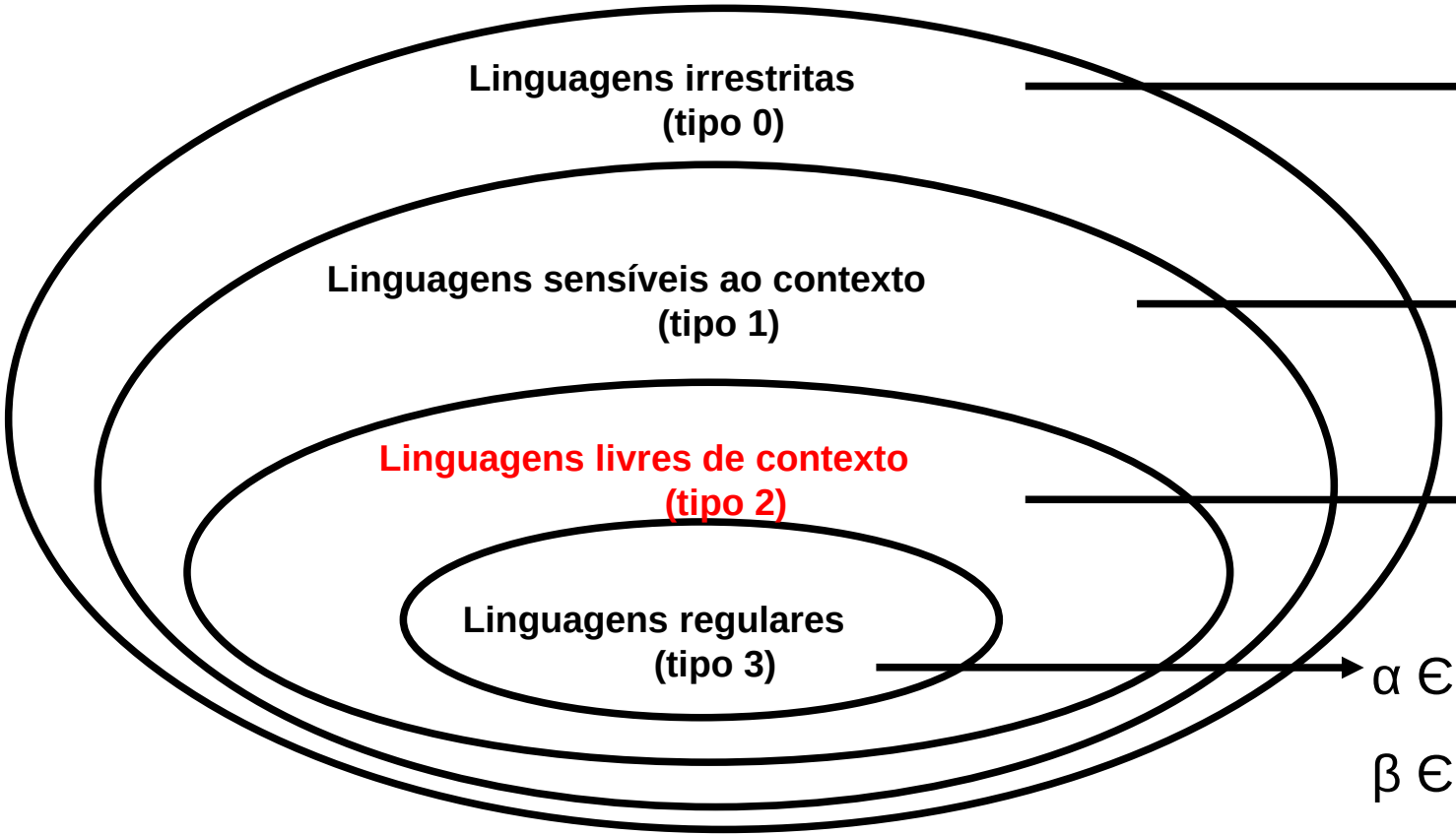
Aula 11
Análise Sintática de
Gramáticas Livres de Contexto

Profa. Ariane Machado Lima
ariane.machado@usp.br

Aula passada

Hierarquia de Chomsky

$\alpha \rightarrow \beta$



$\alpha \in (V \cup \Sigma)^* V (V \cup \Sigma)^*$
$\beta \in (V \cup \Sigma)^*$

$\alpha \in (V \cup \Sigma)^* V (V \cup \Sigma)^*$
$\beta \in (V \cup \Sigma)^*$
$|\alpha| <= |\beta|$

$\alpha \in V$

$\beta \in (V \cup \Sigma)^*$

$\alpha \in V$

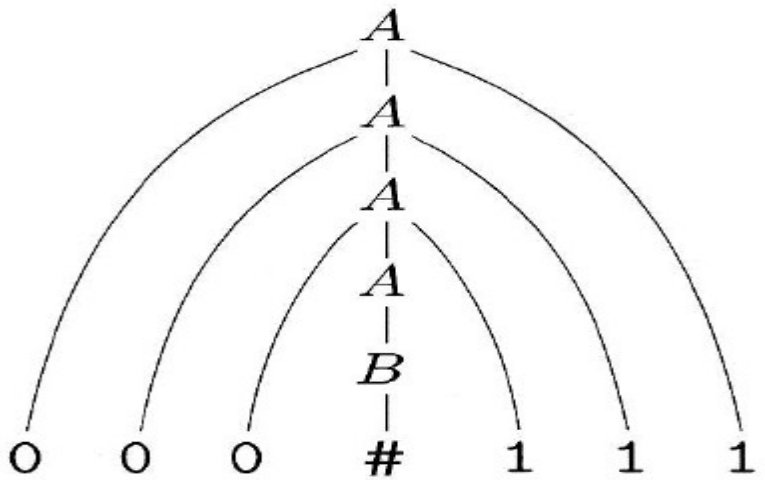
$\beta \in \Sigma_\varepsilon$, $\beta \in V$, $\beta \in (V\Sigma$ ou $\Sigma V)$

Gramáticas Livres de Contexto

$$A \rightarrow 0A1$$
$$A \rightarrow B$$
$$B \rightarrow \#$$

Por exemplo, a gramática G_1 gera a cadeia 000#111.

$$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$$



Árvore sintática
ou
Árvore de derivação

Projetando gramáticas livres de contexto

Lembram-se da linguagem $A = \{0^n1^n \mid n \geq 0\}$?

Era regular?

É livre de contexto!

Qual seria a gramática que a gera?

$S \rightarrow 0\ S\ 1$

$S \rightarrow \varepsilon$

Projetando gramáticas livres de contexto

Lembram-se da linguagem $A = \{0^n1^n \mid n \geq 1\}$?

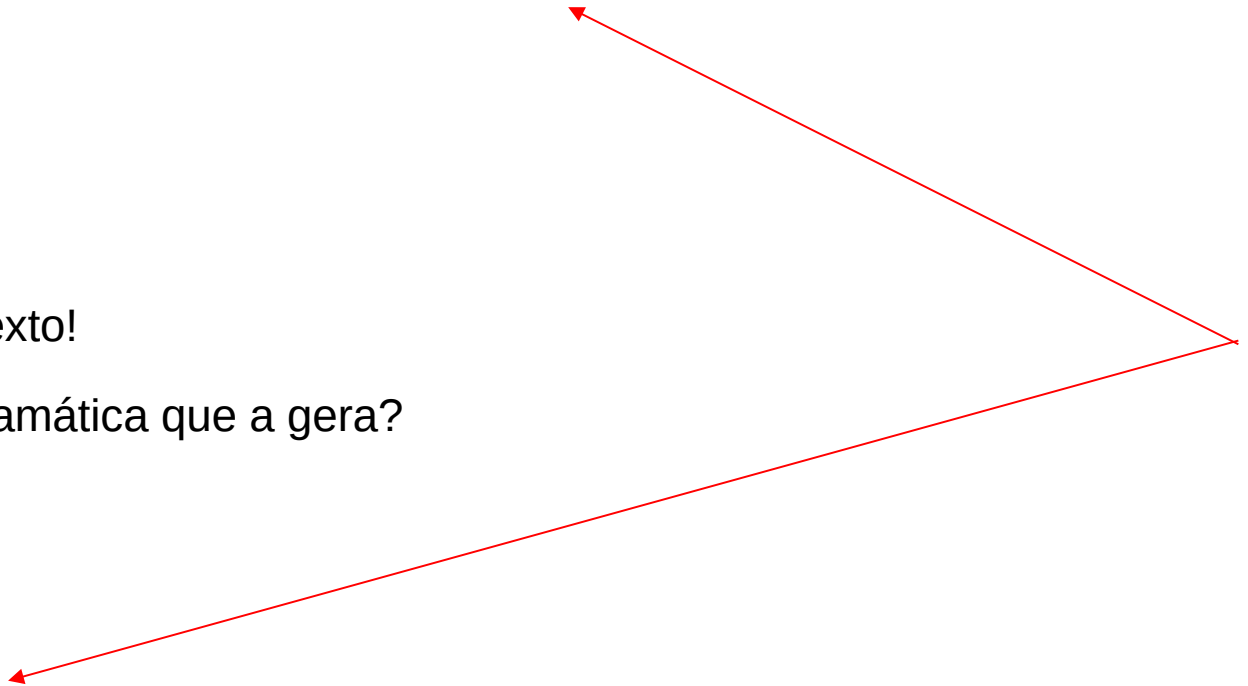
Era regular?

É livre de contexto!

Qual seria a gramática que a gera?

$S \rightarrow 0\ S\ 1$

$S \rightarrow 01$



Dicas para projetar gramáticas livres de contexto

- É a união de linguagens mais simples?

Por exemplo, para obter uma gramática para a linguagem $\{0^n1^n \mid n \geq 0\} \cup \{1^n0^n \mid n \geq 0\}$, primeiro construa a gramática

$$S_1 \rightarrow 0S_1 1 \mid \varepsilon$$

para a linguagem $\{0^n1^n \mid n \geq 0\}$ e a gramática

$$S_2 \rightarrow 1S_2 0 \mid \varepsilon$$

para a linguagem $\{1^n0^n \mid n \geq 0\}$ e então adicione a regra $S \rightarrow S_1 \mid S_2$ para dar a gramática

$$\begin{aligned} S &\rightarrow S_1 \mid S_2 \\ S_1 &\rightarrow 0S_1 1 \mid \varepsilon \\ S_2 &\rightarrow 1S_2 0 \mid \varepsilon \, . \end{aligned}$$

Dicas para projetar gramáticas livres de contexto

- Definições recursivas

EXEMPLO 2.3

Considere a gramática $G_3 = (\{S\}, \{\mathtt{a}, \mathtt{b}\}, R, S)$. O conjunto de regras, R, é

$$S \rightarrow \mathtt{a}S\mathtt{b} \mid SS \mid \varepsilon.$$

Exercício – projeto de gramática (baseado nos ex 4 e 7 do cap 4 do livro de RAMOS)

Contrua gramáticas livres de contexto para as seguintes linguagens:

1) $\{a^i b^{2i} \mid i >= 1\}$

2) $\{ww^R \mid w \in \{a, b, c\}^*\}$, sendo w^R a cadeia w reversa (escrita de trás para a frente)

3) $\{wcw^R \mid w \in \{a, b\}^*\}$, sendo w^R a cadeia w reversa (escrita de trás para a frente)

4) $\{a^m b^m c^n d^n \mid m >= 0, n >= 1\}$

5) $\{a^m b^n c^n d^m \mid m >= 1, n >= 0\}$

6) $\{aa^* b^i c^j \mid i >= 1, j >= i\}$

Respostas

Exercício – projeto de gramática (baseado nos ex 4 e 7 do cap 4 do livro de RAMOS)

Contrua gramáticas livres de contexto para as seguintes linguagens:

1) $\{a^i b^{2i} \mid i >= 1\}$

S → aSbb

S → abb

Exercício – projeto de gramática (baseado nos ex 4 e 7 do cap 4 do livro de RAMOS)

Contrua gramáticas livres de contexto para as seguintes linguagens:

2) $\{ww^R \mid w \{a, b, c\}^*\}$

$S \rightarrow aSa \mid bSb \mid cSc \mid \varepsilon$

3) $\{wcw^R \mid w \{a, b\}^*\}$

$S \rightarrow aSa \mid bSb \mid c$

Exercício – projeto de gramática (baseado nos ex 4 e 7 do cap 4 do livro de RAMOS)

Contrua gramáticas livres de contexto para as seguintes linguagens:

4) $\{a^m b^m c^n d^n \mid m >= 0, n >= 1\}$

S → XY

X → aXb | ε

Y → cYd | cd

Exercício – projeto de gramática (baseado nos ex 4 e 7 do cap 4 do livro de RAMOS)

Contrua gramáticas livres de contexto para as seguintes linguagens:

5) $\{a^m b^n c^n d^m \mid m >= 1, n >= 0\}$

S → aSd | aYd

Y → bYc | ε

Exercício – projeto de gramática (baseado nos ex 4 e 7 do cap 4 do livro de RAMOS)

Contrua gramáticas livres de contexto para as seguintes linguagens:

6) $\{aa^*b^ic^j \mid i >= 1, j >= i\}$

S → ABC

A → aA | a

B → bBc | bc

C → cC | ε

Aula de hoje

Análise sintática e ambiguidade

Derivações

- É possível que uma mesma cadeia possua mais de uma derivação

Derivações

Gramática:

1. $S \rightarrow S\ ;\ S$
2. $S \rightarrow \text{id}\ :=\ E$
3. $S \rightarrow \text{print}\ (\ L\ )$
4. $E \rightarrow \text{id}$
5. $E \rightarrow \text{num}$
6. $E \rightarrow E\ +\ E$
7. $E \rightarrow (S\ ,\ E\ )$
8. $L \rightarrow E$
9. $L \rightarrow L\ ,\ E$

Cadeia: `id := num; id := id + (id := num + num, id)`

$\underline{S}$
$S\ ;\ \underline{S}$
$\underline{S}\ ;\ \text{id}\ :=\ E$
$\text{id}\ :=\ \underline{E}\ ;\ \text{id}\ :=\ E$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ \underline{E}$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ E\ +\ \underline{E}$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ \underline{E}\ +\ (S\ ,\ E\ )$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ \text{id}\ +\ (\underline{S}\ ,\ E\ )$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ \text{id}\ +\ (\text{id}\ :=\ \underline{E}\ ,\ E\ )$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ \text{id}\ +\ (\text{id}\ :=\ E\ +\ E\ ,\ \underline{E}\ )$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ \text{id}\ +\ (\text{id}\ :=\ \underline{E}\ +\ E\ ,\ \text{id}\ )$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ \text{id}\ +\ (\text{id}\ :=\ \text{num}\ +\ \underline{E}\ ,\ \text{id}\ )$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ \text{id}\ +\ (\text{id}\ :=\ \text{num}\ +\ \text{num}\ ,\ \text{id}\ )$

Uma derivação possível

Derivações

- **Derivação mais à esquerda** (sempre o primeiro símbolo não terminal da forma sentencial é substituído primeiro)
- **Derivação mais à direita** (sempre o último símbolo não terminal da forma sentencial é substituído primeiro)

Derivações

Gramática:

1	$S \rightarrow S\ ;\ S$	4	$E \rightarrow \text{id}$	8	$L \rightarrow E$
2	$S \rightarrow \text{id}\ :=\ E$	5	$E \rightarrow \text{num}$	9	$L \rightarrow L\ ,\ E$
3	$S \rightarrow \text{print}\ (\ L\ )$	6	$E \rightarrow E\ +\ E$		
		7	$E \rightarrow (S\ ,\ E\ )$		

Cadeia: `id := num; id := id + (id := num + num, id)`

$\underline{S}$
$S\ ;\ \underline{S}$
$\underline{S}\ ;\ \text{id}\ :=\ E$
$\text{id}\ :=\ \underline{E}\ ;\ \text{id}\ :=\ E$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ \underline{E}$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ E\ +\ \underline{E}$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ \underline{E}\ +\ (S\ ,\ E\ )$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ \text{id}\ +\ (\underline{S}\ ,\ E\ )$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ \text{id}\ +\ (\text{id}\ :=\ \underline{E}\ ,\ E\ )$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ \text{id}\ +\ (\text{id}\ :=\ E\ +\ E\ ,\ \underline{E}\ )$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ \text{id}\ +\ (\text{id}\ :=\ \underline{E}\ +\ E\ ,\ \text{id}\ )$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ \text{id}\ +\ (\text{id}\ :=\ \text{num}\ +\ \underline{E}\ ,\ \text{id}\ )$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ \text{id}\ +\ (\text{id}\ :=\ \text{num}\ +\ \text{num}\ ,\ \text{id}\ )$

Uma derivação possível

Mais à esquerda ou mais à direita?

Derivações

Gramática:

1	$S \rightarrow S\ ;\ S$	4	$E \rightarrow \text{id}$	8	$L \rightarrow E$
2	$S \rightarrow \text{id}\ :=\ E$	5	$E \rightarrow \text{num}$	9	$L \rightarrow L\ ,\ E$
3	$S \rightarrow \text{print}\ (\ L\ )$	6	$E \rightarrow E\ +\ E$		
		7	$E \rightarrow (S\ ,\ E\ )$		

Cadeia: `id := num; id := id + (id := num + num, id)`

$\underline{S}$
$S\ ;\ \underline{S}$
$\underline{S}\ ;\ \text{id}\ :=\ E$
$\text{id}\ :=\ \underline{E}\ ;\ \text{id}\ :=\ E$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ \underline{E}$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ E\ +\ \underline{E}$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ \underline{E}\ +\ (S\ ,\ E\ )$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ \text{id}\ +\ (\underline{S}\ ,\ E\ )$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ \text{id}\ +\ (\text{id}\ :=\ \underline{E}\ ,\ E\ )$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ \text{id}\ +\ (\text{id}\ :=\ E\ +\ E\ ,\ \underline{E}\ )$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ \text{id}\ +\ (\text{id}\ :=\ \underline{E}\ +\ E\ ,\ \text{id}\ )$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ \text{id}\ +\ (\text{id}\ :=\ \text{num}\ +\ \underline{E}\ ,\ \text{id}\ )$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ \text{id}\ +\ (\text{id}\ :=\ \text{num}\ +\ \text{num}\ ,\ \text{id}\ )$

Uma derivação possível

Mais à esquerda ou mais à direita? Nenhuma das 2

Derivações

Gramática:

1	$S \rightarrow S\ ;\ S$	4	$E \rightarrow \text{id}$	8	$L \rightarrow E$
2	$S \rightarrow \text{id} := E$	5	$E \rightarrow \text{num}$	9	$L \rightarrow L\ ,\ E$
3	$S \rightarrow \text{print}\ (\ L\ )$	6	$E \rightarrow E + E$		
		7	$E \rightarrow (S\ ,\ E\ )$		

Cadeia: `id := num; id := id + (id := num + num, id)`

$\underline{S}$
$S\ ;\ \underline{S}$
$\underline{S}\ ;\ \text{id} := E$
$\text{id} := \underline{E}\ ;\ \text{id} := E$
$\text{id} := \text{num}\ ;\ \text{id} := \underline{E}$
$\text{id} := \text{num}\ ;\ \text{id} := E + \underline{E}$
$\text{id} := \text{num}\ ;\ \text{id} := \underline{E} + (S\ ,\ E\ )$
$\text{id} := \text{num}\ ;\ \text{id} := \text{id} + (\underline{S}\ ,\ E\ )$
$\text{id} := \text{num}\ ;\ \text{id} := \text{id} + (\text{id} := \underline{E}\ ,\ E\ )$
$\text{id} := \text{num}\ ;\ \text{id} := \text{id} + (\text{id} := E + E\ ,\ \underline{E}\ )$
$\text{id} := \text{num}\ ;\ \text{id} := \text{id} + (\text{id} := \underline{E} + E\ ,\ \text{id}\ )$
$\text{id} := \text{num}\ ;\ \text{id} := \text{id} + (\text{id} := \text{num} + \underline{E}\ ,\ \text{id}\ )$
$\text{id} := \text{num}\ ;\ \text{id} := \text{id} + (\text{id} := \text{num} + \text{num}\ ,\ \text{id}\ )$

Uma derivação possível

Mais à esquerda ou mais à direita? Nenhuma das 2

$\underline{S}$
$\underline{S}\ ;\ S$
$\text{id} := \underline{E}\ ;\ S$
$\text{id} := \text{num}\ ;\ \underline{S}$
$\text{id} := \text{num}\ ;\ \text{id} := \underline{E}$
$\text{id} := \text{num}\ ;\ \text{id} := \underline{E} + E$
$\vdots$

Derivações

Gramática:

1	$S \rightarrow S\ ;\ S$	4	$E \rightarrow \text{id}$	8	$L \rightarrow E$
2	$S \rightarrow \text{id} := E$	5	$E \rightarrow \text{num}$	9	$L \rightarrow L\ ,\ E$
3	$S \rightarrow \text{print}\ (\ L\ )$	6	$E \rightarrow E + E$		
		7	$E \rightarrow (S\ ,\ E\ )$		

Cadeia: `id := num; id := id + (id := num + num, id)`

$\underline{S}$
$S\ ;\ \underline{S}$
$\underline{S}\ ;\ \text{id} := E$
$\text{id} := \underline{E}\ ;\ \text{id} := E$
$\text{id} := \text{num}\ ;\ \text{id} := \underline{E}$
$\text{id} := \text{num}\ ;\ \text{id} := E + \underline{E}$
$\text{id} := \text{num}\ ;\ \text{id} := \underline{E} + (S\ ,\ E\ )$
$\text{id} := \text{num}\ ;\ \text{id} := \text{id} + (\underline{S}\ ,\ E\ )$
$\text{id} := \text{num}\ ;\ \text{id} := \text{id} + (\text{id} := \underline{E}\ ,\ E\ )$
$\text{id} := \text{num}\ ;\ \text{id} := \text{id} + (\text{id} := E + E\ ,\ \underline{E}\ )$
$\text{id} := \text{num}\ ;\ \text{id} := \text{id} + (\text{id} := \underline{E} + E\ ,\ \text{id}\ )$
$\text{id} := \text{num}\ ;\ \text{id} := \text{id} + (\text{id} := \text{num} + \underline{E}\ ,\ \text{id}\ )$
$\text{id} := \text{num}\ ;\ \text{id} := \text{id} + (\text{id} := \text{num} + \text{num}\ ,\ \text{id}\ )$

Uma derivação possível

Mais à esquerda ou mais à direita? Nenhuma das 2

$\underline{S}$
$\underline{S}\ ;\ S$
$\text{id} := \underline{E}\ ;\ S$
$\text{id} := \text{num}\ ;\ \underline{S}$
$\text{id} := \text{num}\ ;\ \text{id} := \underline{E}$
$\text{id} := \text{num}\ ;\ \text{id} := \underline{E} + E$
$\vdots$

Derivação mais à esquerda

Análise sintática

Um algoritmo de análise sintática (ou **analisador sintático**) é tal que, dada uma cadeia e uma gramática, encontra uma derivação (ou alternativamente uma árvore sintática) para a cadeia segundo aquela gramática

Obs: passo necessário para a compilação de um programa em uma dada linguagem de programação

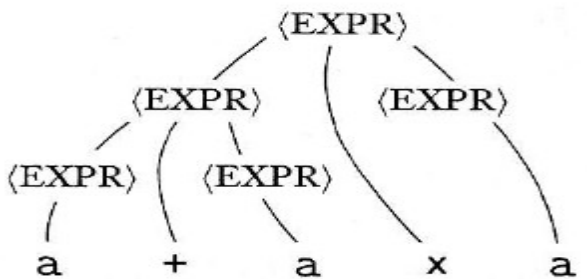
Antes de falar sobre análise sintática, vamos falar sobre um conceito importante: **ambiguidade**

Ex: gramática para expressões aritméticas simples

$$\langle \text{EXPR} \rangle \rightarrow \langle \text{EXPR} \rangle + \langle \text{EXPR} \rangle \mid \langle \text{EXPR} \rangle \times \langle \text{EXPR} \rangle \mid (\langle \text{EXPR} \rangle) \mid \mathtt{a}$$

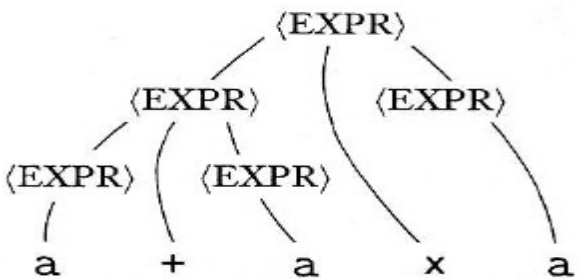
Ex: gramática para expressões aritméticas simples

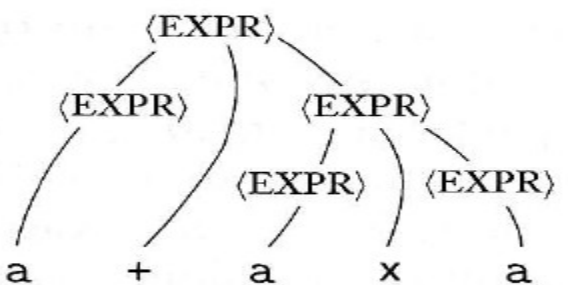
$$\langle \text{EXPR} \rangle \rightarrow \langle \text{EXPR} \rangle + \langle \text{EXPR} \rangle \mid \langle \text{EXPR} \rangle \times \langle \text{EXPR} \rangle \mid (\langle \text{EXPR} \rangle) \mid \texttt{a}$$



Ex: gramática para expressões aritméticas simples

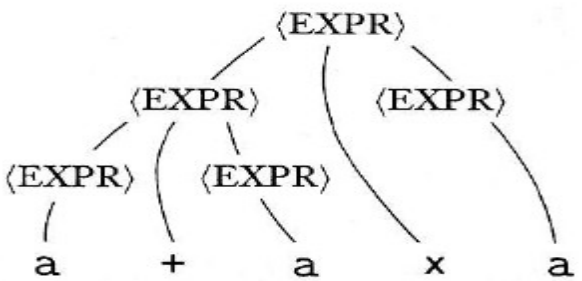
$$\langle \text{EXPR} \rangle \rightarrow \langle \text{EXPR} \rangle + \langle \text{EXPR} \rangle \mid \langle \text{EXPR} \rangle \times \langle \text{EXPR} \rangle \mid (\langle \text{EXPR} \rangle) \mid \text{a}$$

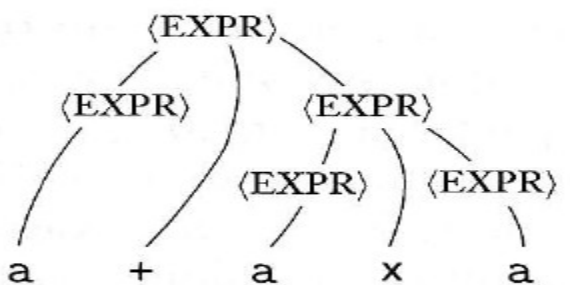




Ex: gramática para expressões aritméticas simples

$$\langle \text{EXPR} \rangle \rightarrow \langle \text{EXPR} \rangle + \langle \text{EXPR} \rangle \mid \langle \text{EXPR} \rangle \times \langle \text{EXPR} \rangle \mid (\langle \text{EXPR} \rangle) \mid \text{a}$$

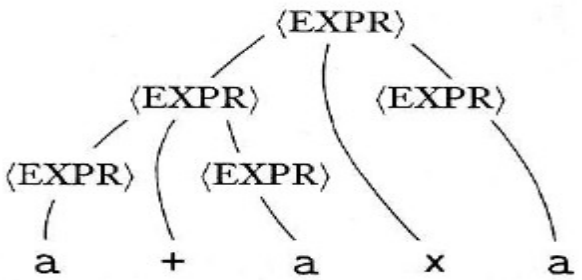


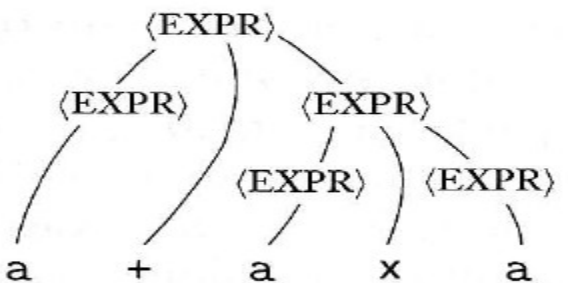


Duas árvores sintáticas distintas para a mesma cadeia!!!!

Ex: gramática para expressões aritméticas simples

$$\langle \text{EXPR} \rangle \rightarrow \langle \text{EXPR} \rangle + \langle \text{EXPR} \rangle \mid \langle \text{EXPR} \rangle \times \langle \text{EXPR} \rangle \mid (\langle \text{EXPR} \rangle) \mid \text{a}$$

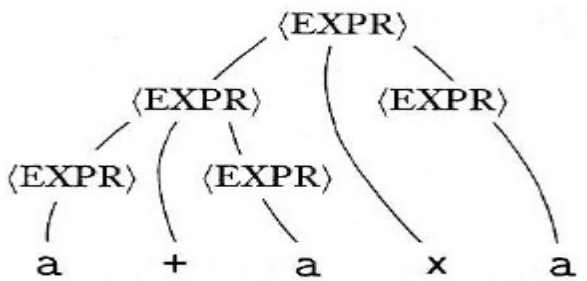


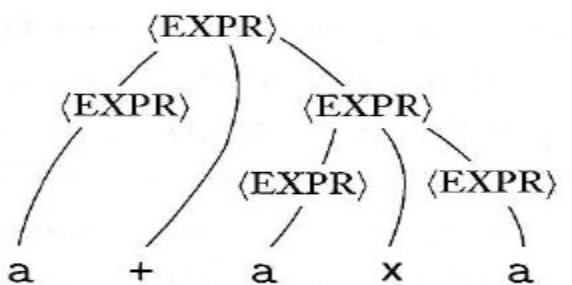


Duas árvores sintáticas distintas para a mesma cadeia!!!!
Logo, dizemos que essa gramática é AMBÍGUA

Ex: gramática para expressões aritméticas simples

$$\langle \text{EXPR} \rangle \rightarrow \langle \text{EXPR} \rangle + \langle \text{EXPR} \rangle \mid \langle \text{EXPR} \rangle \times \langle \text{EXPR} \rangle \mid (\langle \text{EXPR} \rangle) \mid \text{a}$$

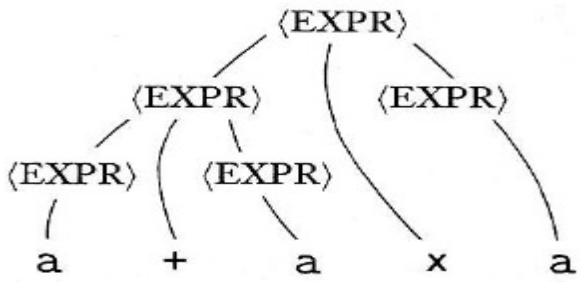




Uma outra forma de verificar se há ambiguidade é olhar as derivações...

Ex: gramática para expressões aritméticas simples

$$\langle \text{EXPR} \rangle \rightarrow \langle \text{EXPR} \rangle + \langle \text{EXPR} \rangle \mid \langle \text{EXPR} \rangle \times \langle \text{EXPR} \rangle \mid (\langle \text{EXPR} \rangle) \mid \text{a}$$



Várias derivações possíveis dariam essa mesma árvore, ex:

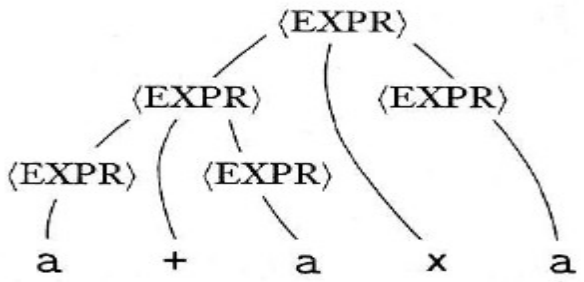
E => E x **E** => **E** x a => E + **E** x a => **E** + a x a => a + a x a

incluindo uma derivação mais à esquerda:

E => **E** x E => **E** + E x E => a + **E** x E => a + a x **E** => a + a x a

Ex: gramática para expressões aritméticas simples

$$\langle EXPR \rangle \rightarrow \langle EXPR \rangle + \langle EXPR \rangle \mid \langle EXPR \rangle \times \langle EXPR \rangle \mid (\langle EXPR \rangle) \mid a$$



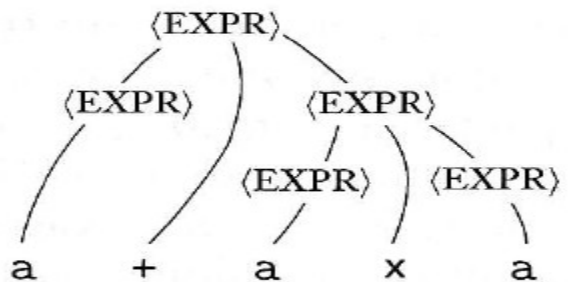
Várias derivações possíveis dariam essa mesma árvore, ex:

E => E x **E** => **E** x a => E + **E** x a => **E** + a x a => a + a x a

incluindo uma derivação mais à esquerda:

E => **E** x E => **E** + E x E => a + **E** x E => a + a x **E** => a + a x a

A mesma cadeia, outra árvore!



Várias derivações possíveis dariam essa mesma árvore, ex:

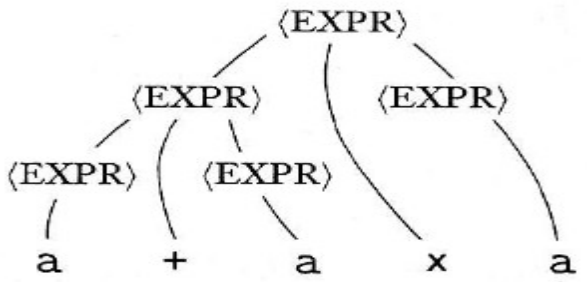
E => E + **E** => E + **E** x E => E + a x **E** => **E** + a x a => a + a x a

incluindo uma OUTRA derivação mais à esquerda:

E => **E** + E => a + **E** => a + **E** x E => a + a x **E** => a + a x a

Ex: gramática para expressões aritméticas simples

⟨EXPR⟩ → ⟨EXPR⟩+⟨EXPR⟩ | ⟨EXPR⟩x⟨EXPR⟩ | (⟨EXPR⟩) | a



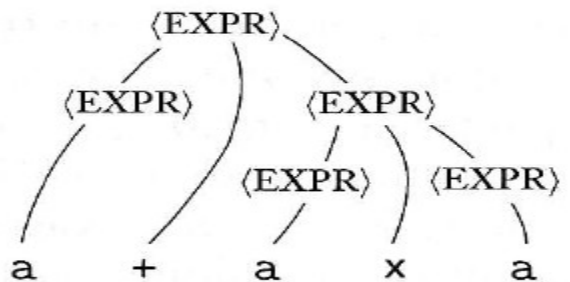
Várias derivações possíveis dariam essa mesma árvore, ex:

E => E x **E** => **E** x a => E + **E** x a => **E** + a x a => a + a x a

incluindo uma derivação mais à esquerda:

E => **E** x E => **E** + E x E => a + **E** x E => a + a x **E** => a + a x a

A mesma cadeia, outra árvore!



Várias derivações possíveis dariam essa mesma árvore, ex:

E => E + **E** => E + **E** x E => E + a x **E** => **E** + a x a => a + a x a

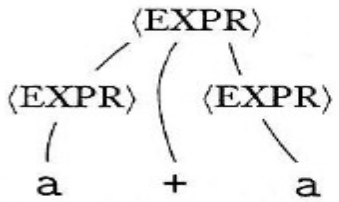
incluindo uma OUTRA derivação mais à esquerda:

E => **E** + E => a + **E** => a + **E** x E => a + a x **E** => a + a x a

Na verdade cada árvore sintática corresponde a uma derivação mais à esquerda distinta

Ex: gramática para expressões aritméticas simples

$$\langle \text{EXPR} \rangle \rightarrow \langle \text{EXPR} \rangle + \langle \text{EXPR} \rangle \mid \langle \text{EXPR} \rangle \times \langle \text{EXPR} \rangle \mid (\langle \text{EXPR} \rangle) \mid \text{a}$$



Duas derivações possíveis dariam essa mesma árvore, ex:

E => E + **E** => **E** + a => a + a

incluindo uma derivação mais à esquerda:

E => **E** + E => a + **E** => a + a

Muitas cadeias:
- possuem só uma árvore sintática,
- portanto só uma derivação mais à esquerda,
- embora possam possuir outras derivações

Árvores sintáticas x derivações (moral da história)

- Uma cadeia normalmente possui várias derivações possíveis
- Uma cadeia possui ao menos uma derivação mais à esquerda, podendo possuir mais de uma
- Uma cadeia possui ao menos uma árvore sintática, podendo possuir mais de uma
- Cada árvore sintática corresponde a exatamente uma derivação mais à esquerda (que é distinta da derivação mais à esquerda correspondente a qualquer outra árvore sintática)
- Logo...

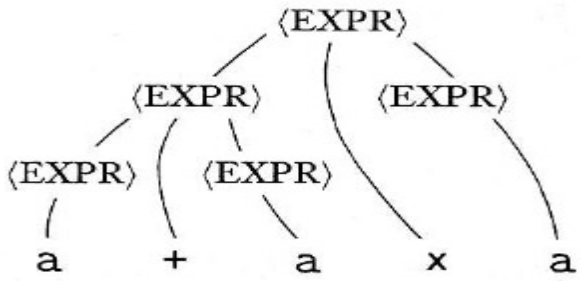
Ambiguidade

DEFINIÇÃO 2.7

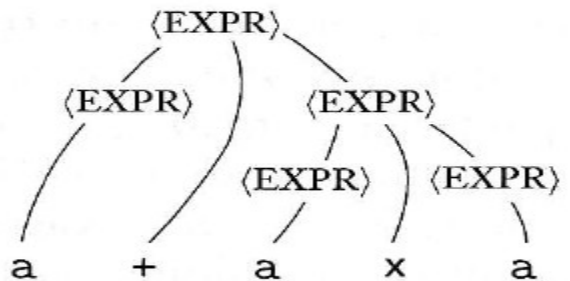
Uma cadeia w é derivada ***ambiguamente*** na gramática livre-do-contexto G se ela tem duas ou mais derivações mais à esquerda diferentes. A gramática G é ***ambígua*** se ela gera alguma cadeia ambiguamente.

Logo, esta gramática é ambígua porque ”a + a x a” é derivada ambiguamente por ela

⟨EXPR⟩ → ⟨EXPR⟩+⟨EXPR⟩ | ⟨EXPR⟩x⟨EXPR⟩ | (⟨EXPR⟩) | a



A mesma cadeia, outra árvore!



Várias derivações possíveis dariam essa mesma árvore, ex:

E => E x **E** => **E** x a => E + **E** x a => **E** + a x a => a + a x a

incluindo uma derivação mais à esquerda:

E => **E** x E => **E** + E x E => a + **E** x E => a + a x **E** => a + a x a

Várias derivações possíveis dariam essa mesma árvore, ex:

E => E + **E** => E + **E** x E => E + a x **E** => **E** + a x a => a + a x a

incluindo uma OUTRA derivação mais à esquerda:

E => **E** + E => a + **E** => a + **E** x E => a + a x **E** => a + a x a

Na verdade cada árvore sintática corresponde a uma derivação mais à esquerda distinta

Ambiguidade

- Ambiguidade é às vezes indesejável, por exemplo em linguagens de programação
- Algumas gramáticas ambíguas podem ser convertidas em não-ambíguas
- Dizemos que uma linguagem é inerentemente ambígua se só pode ser descrita por gramáticas ambíguas. Ex: a Língua Portuguesa
  - Eu vi o menino com uma luneta

Como remover ambiguidade?

- Convertendo a gramática em não ambígua (se possível)

Ex: expressões aritméticas sem ambiguidade

EXEMPLO 2.4

Considere a gramática $G_4 = (V, \Sigma, R, \langle \text{EXPR} \rangle)$.
V é $\{\langle \text{EXPR} \rangle, \langle \text{TERM} \rangle, \langle \text{FACTOR} \rangle\}$ e Σ é $\{\texttt{a}, \texttt{+}, \texttt{x}, \texttt{(}, \texttt{)}\}$. As regras são

$$\begin{aligned} \langle \text{EXPR} \rangle &\rightarrow \langle \text{EXPR} \rangle \texttt{+} \langle \text{TERM} \rangle \mid \langle \text{TERM} \rangle \\ \langle \text{TERM} \rangle &\rightarrow \langle \text{TERM} \rangle \texttt{x} \langle \text{FACTOR} \rangle \mid \langle \text{FACTOR} \rangle \\ \langle \text{FACTOR} \rangle &\rightarrow \texttt{(} \langle \text{EXPR} \rangle \texttt{)} \mid \texttt{a} \end{aligned}$$

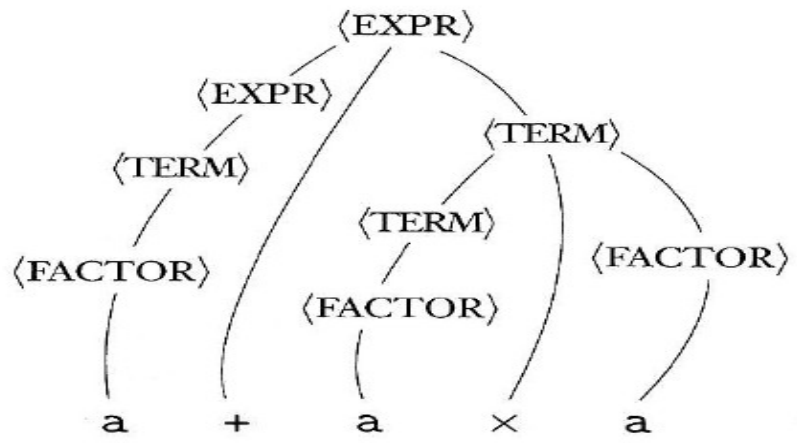
Ex: expressões aritméticas sem ambiguidade

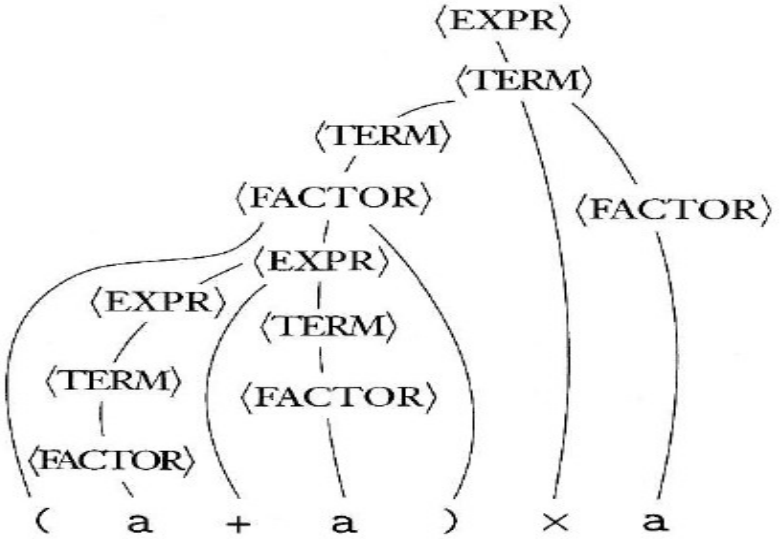
EXEMPLO 2.4

Considere a gramática $G_4 = (V, \Sigma, R, \langle \text{EXPR} \rangle)$.
V é $\{\langle \text{EXPR} \rangle, \langle \text{TERM} \rangle, \langle \text{FACTOR} \rangle\}$ e Σ é $\{\texttt{a}, +, \times, (,)\}$. As regras são

$$\begin{aligned} \langle \text{EXPR} \rangle &\rightarrow \langle \text{EXPR} \rangle + \langle \text{TERM} \rangle \mid \langle \text{TERM} \rangle \\ \langle \text{TERM} \rangle &\rightarrow \langle \text{TERM} \rangle \times \langle \text{FACTOR} \rangle \mid \langle \text{FACTOR} \rangle \\ \langle \text{FACTOR} \rangle &\rightarrow (\langle \text{EXPR} \rangle) \mid \texttt{a} \end{aligned}$$

Esta gramática estabelece uma precedência entre os operadores!





O caso if then else

<prog> → … <com>...

<com> → ...

<com> → <cond>

<cond> → if <exp> then <com>

<cond> → if <exp> then <com> else <com>

<exp> → …

if <exp> then if <com> then <com> else <com>

O caso if then else

<prog> → … <com>...

<com> → ...

<com> → <cond>

<cond> → if <exp> then <com>

<cond> → if <exp> then <com> else <com>

<exp> → …

if <exp> then if <com> then <com> else <com>

Com qual *if* o *else* faz “par”?

O caso if then else

<prog> → … <com>...

<com> → ...

<com> → <cond>

<cond> → if <exp> then <com>

<cond> → if <exp> then <com> else <com>

<exp> → …

if <exp> then if <com> then <com> else <com>

AMBIGUIDADE!!!

O caso if then else

Solução 1: “manter a ambiguidade” sintática, e resolvê-la por meio de uma convenção: o *else* deve “fazer par” com o último *if* (solução adotada por muitas linguagens de programação)

Solução 2: resolver a ambiguidade sintaticamente, tornando a gramática não ambígua

O caso if then else – Solução 2

<prog> → … <com>...

<com> → ...

<com> → <cond>

<cond> → if <exp> then <com> endif

<cond> → if <exp> then <com> else <com> endif

<exp> → …

if <exp> then if <com> then <com> endif else <com> endif

ou

if <exp> then if <com> then <com> else <com> endif endif

SEM AMBIGUIDADE!!!

Resolução de ambiguidade

- Opção 1: Convencionar a forma de desambiguar, e “programar o analisador sintático” para seguir esse convenção
  - Como feito na maioria das linguagens de programação para tratar o caso if/then/else (que casa o else com o if imediatamente anterior)
  - Obs: isso não tira a ambiguidade da gramática (simplesmente o analisador sintático se satisfaz com uma árvore ao invés de calcular todas)
- Opção 2: tirar a ambiguidade da gramática
  - Como feito nas expressões aritméticas
  - Como feito no caso if/then/else com inclusão da palavra-chave endif

Mais sobre análise sintática, Forma Normal de Chomsky

Análise sintática

Problema: Dada uma gramática G e uma cadeia w, saber se w ϵ L(G) (isto é, encontrar ao menos uma derivação de w a partir do símbolo inicial de G).

Para linguagens regulares: o AFD é um reconhecedor eficiente

Para linguagens livres de contexto: até existe uma máquina de estados equivalente (autômatos a pilha que veremos adiante), mas eles não são tão eficientes no caso geral… dá para fazer melhor com gramáticas

Análise sintática

- Esse não é um tema de nossa disciplina, mas é importante entender o que está envolvido para compreendermos alguns tópicos do curso
- Há diferentes estratégias de se programar um analisador sintático, algumas mais simples outras mais complexas
- Existem estratégias dependentes das gramáticas (subclasses de livre-de-contexto)
- Tema da disciplina de construção de compiladores

Análise sintática descendente

- Top-down (do símbolo inicial aos símbolos terminais)
- Cada não terminal A teria uma sub-rotina associada para tratar todas as possibilidades de produção que o tenha do lado esquerdo (A → ...)

Exemplo:

<com> → if (<exp>) <com> else <com>

| while (<exp>) <com>

| { <lista_com> }

Análise sintática descendente - problema

Gramática:

1	$S \rightarrow S\ ;\ S$	4	$E \rightarrow \text{id}$	8	$L \rightarrow E$
2	$S \rightarrow \text{id}\ :=\ E$	5	$E \rightarrow \text{num}$	9	$L \rightarrow L\ ,\ E$
3	$S \rightarrow \text{print}\ (\ L\ )$	6	$E \rightarrow E\ +\ E$		
		7	$E \rightarrow (S\ ,\ E\ )$		

Cadeia: `id := num; id := id + (id := num + num, id)`

$\underline{S}$
$\underline{S}\ ;\ S$
$\text{id}\ :=\ \underline{E}\ ;\ S$
$\text{id}\ :=\ \text{num}\ ;\ \underline{S}$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ \underline{E}$
$\text{id}\ :=\ \text{num}\ ;\ \text{id}\ :=\ \underline{E}\ +\ E$
$\vdots$

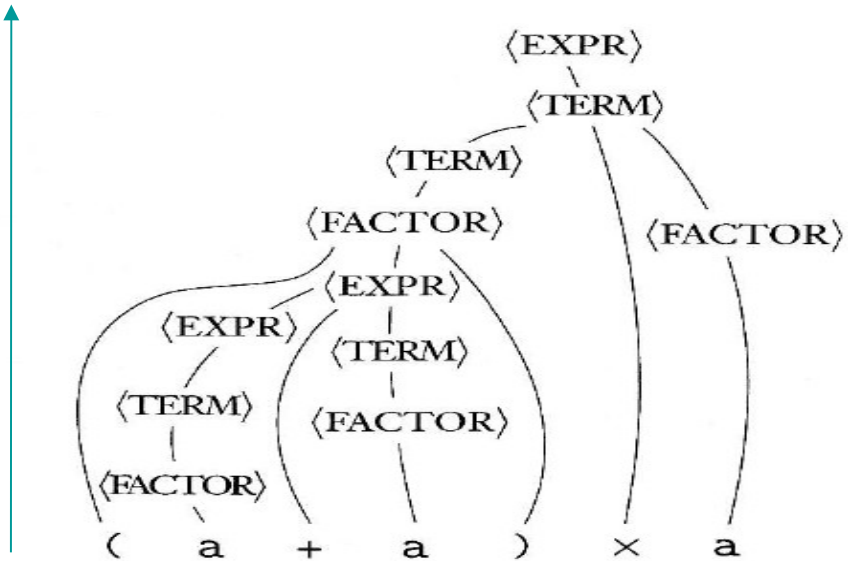
“Recursão à esquerda”: quando eu páro de chamar recursivamente S? (problema… precisamos de um analisador mais poderoso..)

Derivação mais à esquerda

Análise sintática ascendente

- Bottom-up (parte dos símbolos terminais)

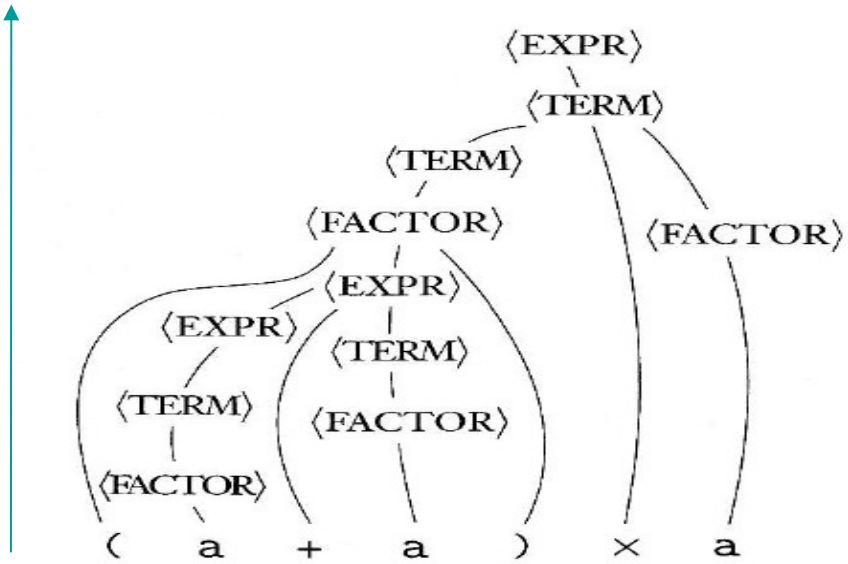
$$\langle \text{EXPR} \rangle \rightarrow \langle \text{EXPR} \rangle + \langle \text{TERM} \rangle \mid \langle \text{TERM} \rangle$$
$$\langle \text{TERM} \rangle \rightarrow \langle \text{TERM} \rangle \times \langle \text{FACTOR} \rangle \mid \langle \text{FACTOR} \rangle$$
$$\langle \text{FACTOR} \rangle \rightarrow (\langle \text{EXPR} \rangle) \mid \texttt{a}$$



Análise sintática ascendente

- Bottom-up (parte dos símbolos terminais)
- Umas das estratégias: algoritmo CYK

$$\langle \text{EXPR} \rangle \rightarrow \langle \text{EXPR} \rangle + \langle \text{TERM} \rangle \mid \langle \text{TERM} \rangle$$
$$\langle \text{TERM} \rangle \rightarrow \langle \text{TERM} \rangle \times \langle \text{FACTOR} \rangle \mid \langle \text{FACTOR} \rangle$$
$$\langle \text{FACTOR} \rangle \rightarrow (\langle \text{EXPR} \rangle) \mid \text{a}$$



Algoritmo CYK para análise sintática

Algoritmo CYK (Cocke–Younger–Kasami)

- Complexidade: Polinomial - $O(n^3m)$ onde n é o tamanho da cadeia e m é o número de regras de G
- G deve estar na Forma Normal de Chomsky
- Por que vamos ver esse algoritmo?
  - Para termos um exemplo de um algoritmo de análise sintática
  - Como motivação para estudarmos a Formal Normal de Chomsky, e o teorema de que QUALQUER gramática livre de contexto pode ser convertida para a forma normal de Chomsky

Forma Normal de Chomsky

Uma GLC está na Forma Normal de Chomsky se:

a) Toda regra de produção é da forma

$$A \rightarrow BC \quad \text{ou } A \rightarrow a$$

sendo B,C variáveis (símbolos não terminais), *a* um símbolo terminal

b) A variável inicial S não pode aparecer no lado direito de nenhuma regra

c) Somente a variável inicial pode ter a regra

$$S \rightarrow \varepsilon$$

Algoritmo CYK para análise sintática

Exemplo:

Gramática original:

$S \rightarrow aSb \mid bSa \mid SS \mid \varepsilon$

- Conversão para FNC (veremos adiante como fazer):

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid AB \mid BA$
$S \rightarrow AT \mid BU \mid SS \mid AB \mid BA$
$T \rightarrow SB$
$U \rightarrow SA$
$A \rightarrow a$
$B \rightarrow b$

Algoritmo CYK para análise sintática

$D =$ "On input $w = w_1 \cdots w_n$:

1. If $w = \varepsilon$ and $S \to \varepsilon$ is a rule, *accept*. [[handle $w = \varepsilon$ case]]

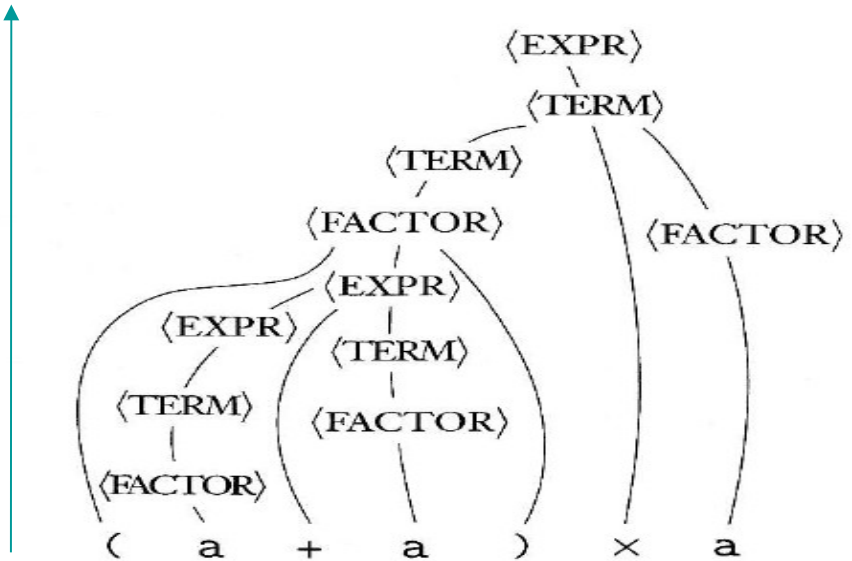
Algoritmo CYK para análise sintática

Programação dinâmica: uso de soluções de subproblemas menores para resolver subproblemas maiores (até chegar à solução do problema original)

Algoritmo CYK para análise sintática

Programação dinâmica: uso de soluções de subproblemas menores para resolver subproblemas maiores (até chegar à solução do problema original)

$$\langle \text{EXPR} \rangle \rightarrow \langle \text{EXPR} \rangle + \langle \text{TERM} \rangle \mid \langle \text{TERM} \rangle$$
$$\langle \text{TERM} \rangle \rightarrow \langle \text{TERM} \rangle \times \langle \text{FACTOR} \rangle \mid \langle \text{FACTOR} \rangle$$
$$\langle \text{FACTOR} \rangle \rightarrow (\langle \text{EXPR} \rangle) \mid \text{a}$$



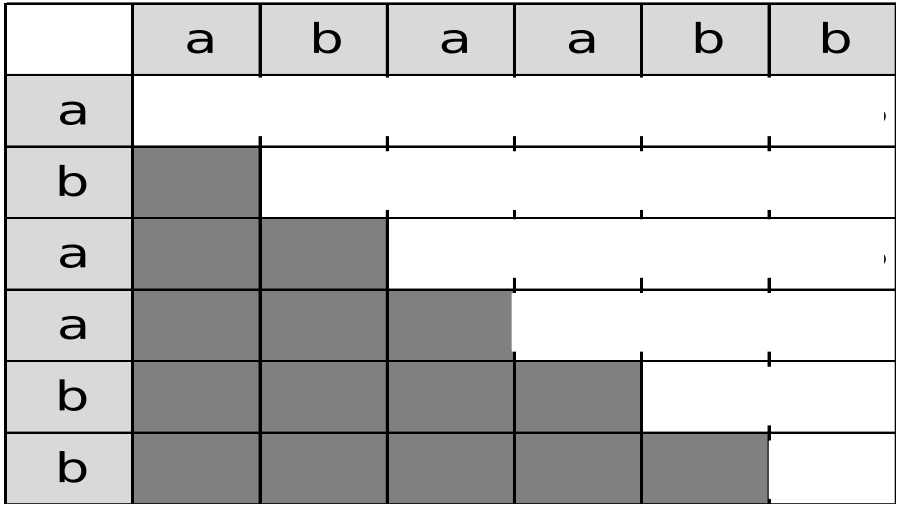
Algoritmo CYK para análise sintática

Programação dinâmica: uso de soluções de subproblemas menores para resolver subproblemas maiores (até chegar à solução do problema original)

Tabela n×n (para uma cadeia de tamanho n):

- Para i ≤ j, a entrada (i,j) da tabela contém todas as variáveis que geram a subcadeia $w_i\, w_{i+1} \ldots w_j$
- Tratam-se subcadeias de tamanhos crescentes (começando de 1)

Tabela:



Algoritmo CYK para análise sintática

Exemplo:

table[i,j] conterá os símbolos não terminais capazes de gerar a substring $w_i...w_j$

Começamos com substrings de tamanho 1...

Gramática na FNC:

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid AB \mid BA$
$S \rightarrow AT \mid BU \mid SS \mid AB \mid BA$
$T \rightarrow SB$
$U \rightarrow SA$
$A \rightarrow a$
$B \rightarrow b$

Cadeia:

a b a a b b

Tabela:

	a	b	a	a	b	b
a						
b						
a						
a						
b						
b						

Algoritmo CYK para análise sintática

Exemplo:

table[i,j] conterá os símbolos não terminais capazes de gerar a substring $w_i...w_j$

Começamos com substrings de tamanho 1...

Gramática na FNC:

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid AB \mid BA$

$S \rightarrow AT \mid BU \mid SS \mid AB \mid BA$

$T \rightarrow SB$

$U \rightarrow SA$

$A \rightarrow a$

$B \rightarrow b$

Cadeia:

a b a a b b

Tabela:

	a	b	a	a	b	b
a	A					
b		B				
a			A			
a				A		
b					B	
b						B

Algoritmo CYK para análise sintática

$D =$ "On input $w = w_1 \cdots w_n$:

1. If $w = \varepsilon$ and $S \to \varepsilon$ is a rule, *accept*. [[handle $w = \varepsilon$ case]]
2. For $i = 1$ to n: [[examine each substring of length 1]]
3. For each variable A:
4. Test whether $A \to \mathtt{b}$ is a rule, where $\mathtt{b} = w_i$.
5. If so, place A in $table(i, i)$.

Algoritmo CYK para análise sintática

$D =$ "On input $w = w_1 \cdots w_n$:

1. If $w = \varepsilon$ and $S \to \varepsilon$ is a rule, *accept*. [[handle $w = \varepsilon$ case]]
2. For $i = 1$ to n: [[examine each substring of length 1]]
3. For each variable A:
4. Test whether $A \to \mathtt{b}$ is a rule, where $\mathtt{b} = w_i$.
5. If so, place A in $table(i, i)$.

E o restante?

Vamos pensar...

Algoritmo CYK para análise sintática

Exemplo:

Vamos analisar agora substrings de tamanho 2.
Por ter tamanho > 1, a variável que a gera a faz por meio de uma produção no formato X → YZ

Grámática na FNC:

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid AB \mid BA$
$S \rightarrow AT \mid BU \mid SS \mid AB \mid BA$
$T \rightarrow SB$
$U \rightarrow SA$
$A \rightarrow a$
$B \rightarrow b$

Cadeia:

a b a a b b

Tabela:

(i,j) = (1,2)

	a	b	a	a	b	b
a	A					
b		B				
a			A			
a				A		
b					B	
b						B

Algoritmo CYK para análise sintática

Exemplo:

Vamos analisar agora substrings de tamanho 2
Por ter tamanho > 1, a variável que a gera a faz por meio de uma produção no formato $X \rightarrow YZ$

Grámática na FNC:

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid AB \mid BA$
$S \rightarrow AT \mid BU \mid SS \mid AB \mid BA$
$T \rightarrow SB$
$U \rightarrow SA$
$A \rightarrow a$
$B \rightarrow b$

Cadeia:

a b a a b b

Tabela:

(i,j) = (1,2) = (1,1).(2,2)

ab é gerado por AB

	a	b	a	a	b	b
a	A					
b		B				
a			A			
a				A		
b					B	
b						B

Algoritmo CYK para análise sintática

Exemplo:

Vamos analisar agora substrings de tamanho 2
Por ter tamanho > 1, a variável que a gera a faz por meio de uma produção no formato $X \rightarrow YZ$

(i,j) = (1,2) = (1,1).(2,2)

Grámática na FNC:

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid \mathbf{AB} \mid BA$
$S \rightarrow AT \mid BU \mid SS \mid \mathbf{AB} \mid BA$
$T \rightarrow SB$
$U \rightarrow SA$
$A \rightarrow a$
$B \rightarrow b$

ab é gerado por AB
EBA! Existem variáveis que derivam diretamente AB!

Tabela:

	a	b	a	a	b	b
a	A					
b		B				
a			A			
a				A		
b					B	
b						B

Cadeia:

a b a a b b

Algoritmo CYK para análise sintática

Exemplo:

Vamos analisar agora substrings de tamanho 2
Por ter tamanho > 1, a variável que a gera a faz por meio de uma produção no formato $X \rightarrow YZ$

Grámática na FNC:

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid \mathbf{AB} \mid BA$
$S \rightarrow AT \mid BU \mid SS \mid \mathbf{AB} \mid BA$
$T \rightarrow SB$
$U \rightarrow SA$
$A \rightarrow a$
$B \rightarrow b$

Cadeia:

a b a a b b

ab é gerado por AB
EBA! Existem variáveis que derivam diretamente AB!

Tabela:

(i,j) = (1,2) = (1,1).(2,2)

	a	b	a	a	b	b
a	A	S_0, S				
b		B				
a			A			
a				A		
b					B	
b						B

Algoritmo CYK para análise sintática

Exemplo:

Vamos analisar agora substrings de tamanho 2
Por ser tamanho > 1, a variável que a gera a faz por meio de uma produção no formato $X \rightarrow YZ$

(i,j) = (2,3) = (2,2).(3,3) = ba

Grámática na FNC:

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid AB \mid BA$
$S \rightarrow AT \mid BU \mid SS \mid AB \mid BA$
$T \rightarrow SB$
$U \rightarrow SA$
$A \rightarrow a$
$B \rightarrow b$

Cadeia:

a b a a b b

Tabela:

	a	b	a	a	b	b
a	A	S_0, S				
b		B				
a			A			
a				A		
b					B	
b						B

Algoritmo CYK para análise sintática

Exemplo:

Vamos analisar agora substrings de tamanho 2
Por ser tamanho > 1, a variável que a gera a faz por meio de uma produção no formato $X \rightarrow YZ$

Grámática na FNC:

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid AB \mid \mathbf{BA}$
$S \rightarrow AT \mid BU \mid SS \mid AB \mid \mathbf{BA}$
$T \rightarrow SB$
$U \rightarrow SA$
$A \rightarrow a$
$B \rightarrow b$

ba é gerado por BA

Cadeia:

a b a a b b

Tabela:

(i,j) = (2,3) = (2,2).(3,3) = ba

	a	b	a	a	b	b
a	A	S_0, S				
b		B				
a			A			
a				A		
b					B	
b						B

Algoritmo CYK para análise sintática

Exemplo:

Vamos analisar agora substrings de tamanho 2
Por ser tamanho > 1, a variável que a gera a faz por meio de uma produção no formato $X \rightarrow YZ$

Grámática na FNC:

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid AB \mid \mathbf{BA}$
$S \rightarrow AT \mid BU \mid SS \mid AB \mid \mathbf{BA}$
$T \rightarrow SB$
$U \rightarrow SA$
$A \rightarrow a$
$B \rightarrow b$

ba é gerado por BA

Cadeia:

a b a a b b

Tabela:

(i,j) = (2,3) = (2,2).(3,3) = ba

	a	b	a	a	b	b
a	A	S_0, S				
b		B	S_0, S			
a			A			
a				A		
b					B	
b						B

Algoritmo CYK para análise sintática

Exemplo:

Vamos analisar agora substrings de tamanho 2
Por ser tamanho > 1, a variável que a gera a faz por meio de uma produção no formato $X \rightarrow YZ$

Grámática na FNC:

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid AB \mid BA$
$S \rightarrow AT \mid BU \mid SS \mid AB \mid BA$
$T \rightarrow SB$
$U \rightarrow SA$
$A \rightarrow a$
$B \rightarrow b$

Cadeia:

a b a a b b

Tabela:

(i,j) = (3,4) = (3,3).(4,4) = aa

	a	b	a	a	b	b
a	A	S_0, S				
b		B	S_0, S			
a			A			
a				A		
b					B	
b						B

Algoritmo CYK para análise sintática

Exemplo:

Vamos analisar agora substrings de tamanho 2
Por ser tamanho > 1, a variável que a gera a faz por meio de uma produção no formato $X \rightarrow YZ$

Grámática na FNC:

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid AB \mid BA$
$S \rightarrow AT \mid BU \mid SS \mid AB \mid BA$
$T \rightarrow SB$
$U \rightarrow SA$
$A \rightarrow a$
$B \rightarrow b$

Cadeia:

a b a a b b

Tabela:

(i,j) = (3,4) = (3,3).(4,4) = aa

	a	b	a	a	b	b
a	A	S_0, S				
b		B	S_0, S			
a			A	⊘		
a				A		
b					B	
b						B

Algoritmo CYK para análise sintática

Exemplo:

Vamos analisar agora substrings de tamanho 2
Por ser tamanho > 1, a variável que a gera a faz por meio de uma produção no formato $X \rightarrow YZ$

Grámática na FNC:

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid \mathbf{AB} \mid BA$
$S \rightarrow AT \mid BU \mid SS \mid \mathbf{AB} \mid BA$
$T \rightarrow SB$
$U \rightarrow SA$
$A \rightarrow a$
$B \rightarrow b$

Cadeia:

a b a a b b

Tabela:

(i,j) = (4,5) = (4,4).(5,5) = ab

	a	b	a	a	b	b
a	A	S_0, S				
b		B	S_0, S			
a			A	⊘		
a				A	S_0, S	
b					B	
b						B

Algoritmo CYK para análise sintática

Exemplo:

Vamos analisar agora substrings de tamanho 2
Por ser tamanho > 1, a variável que a gera a faz por meio de uma produção no formato $X \rightarrow YZ$

(i,j) = (5,6) = (5,5).(6,6) = bb

Grámática na FNC:

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid AB \mid BA$
$S \rightarrow AT \mid BU \mid SS \mid AB \mid BA$
$T \rightarrow SB$
$U \rightarrow SA$
$A \rightarrow a$
$B \rightarrow b$

Tabela:

	a	b	a	a	b	b
a	A	S_0, S				
b		B	S_0, S			
a			A	⊘		
a				A	S_0, S	
b					B	
b						B

Cadeia:

a b a a b b

Algoritmo CYK para análise sintática

Exemplo:

Vamos analisar agora substrings de tamanho 2
Por ser tamanho > 1, a variável que a gera a faz por meio de uma produção no formato $X \rightarrow YZ$

Grámática na FNC:

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid AB \mid BA$
$S \rightarrow AT \mid BU \mid SS \mid AB \mid BA$
$T \rightarrow SB$
$U \rightarrow SA$
$A \rightarrow a$
$B \rightarrow b$

Cadeia:

a b a a b b

Tabela:

(i,j) = (5,6) = (5,5).(6,6) = bb

	a	b	a	a	b	b
a	A	S_0, S				
b		B	S_0, S			
a			A	⊘		
a				A	S_0, S	
b					B	⊘
b						B

Algoritmo CYK para análise sintática

Exemplo:

Vamos analisar agora substrings de tamanho **3 (seta verde)**

Por ser tamanho > 1, a variável que a gera a faz por meio de uma produção no formato X → YZ

Mas agora os dois primeiros símbolos devem ser gerados por Y, ou os dos últimos por Z. Tenho que testar as DUAS possibilidades!

Grámática na FNC:

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid AB \mid BA$

$S \rightarrow AT \mid BU \mid SS \mid AB \mid BA$

$T \rightarrow SB$

$U \rightarrow SA$

$A \rightarrow a$

$B \rightarrow b$

Cadeia:

a b a a b b

Tabela:

	a	b	a	a	b	b
a	A	S_0, S				
b		B	S_0, S			
a			A	⊘		
a				A	S_0, S	
b					B	⊘
b						B

Algoritmo CYK para análise sintática

Exemplo:

Vamos analisar agora substrings de tamanho **3**

Por ser tamanho > 1, a variável que a gera a faz por meio de uma produção no formato X → YZ

Mas agora os dois primeiros símbolos devem ser gerados por Y, ou os dos últimos por Z. Tenho que testar as DUAS possibilidades!

Grámática na FNC:

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid AB \mid BA$

$S \rightarrow AT \mid BU \mid SS \mid AB \mid BA$

$T \rightarrow SB$

$U \rightarrow SA$

$A \rightarrow a$

$B \rightarrow b$

Cadeia:

a b a a b b

Tabela:

(i,j) = (1,3) = aba

	a	b	a	a	b	b
a	A	S_0, S				
b		B	S_0, S			
a			A	⊘		
a				A	S_0, S	
b					B	⊘
b						B

Algoritmo CYK para análise sintática

Exemplo:

Vamos analisar agora substrings de tamanho **3**

Por ser tamanho > 1, a variável que a gera a faz por meio de uma produção no formato X → YZ

Mas agora **os dois primeiros símbolos devem ser gerados por Y**, ou os dos últimos por Z. Tenho que testar as DUAS possibilidades!

Grámática na FNC:

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid AB \mid BA$

$S \rightarrow AT \mid BU \mid SS \mid AB \mid BA$

$T \rightarrow SB$

$U \rightarrow SA$

$A \rightarrow a$? => $S_0A \overset{*}{\Rightarrow} aba$

$B \rightarrow b$? => $SA \overset{*}{\Rightarrow} aba$

Cadeia:

a b a a b b

Tabela:

(i,j) = (1,3) = aba = (1,2).(3,3)

	a	b	a	a	b	b
a	A	S_0, S				
b		B	S_0, S			
a			A	⊘		
a				A	S_0, S	
b					B	⊘
b						B

Algoritmo CYK para análise sintática

Exemplo:

Vamos analisar agora substrings de tamanho **3**

Por ser tamanho > 1, a variável que a gera a faz por meio de uma produção no formato X → YZ

Mas agora **os dois primeiros símbolos devem ser gerados por Y**, ou os dos últimos por Z. Tenho que testar as DUAS possibilidades!

Grámática na FNC:

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid AB \mid BA$

$S \rightarrow AT \mid BU \mid SS \mid AB \mid BA$

$T \rightarrow SB$

$U \rightarrow SA$

$A \rightarrow a$

$B \rightarrow b$

$? \Rightarrow S_0A \overset{*}{\Rightarrow} aba$

$U \Rightarrow SA \overset{*}{\Rightarrow} aba$

Tabela:

(i,j) = (1,3) = aba = (1,2).(3,3)

	a	b	a	a	b	b
a	A	S_0, S				
b		B	S_0, S			
a			A	⊘		
a				A	S_0, S	
b					B	⊘
b						B

Cadeia:

a b a a b b

Algoritmo CYK para análise sintática

Exemplo:

Vamos analisar agora substrings de tamanho **3**

Por ser tamanho > 1, a variável que a gera a faz por meio de uma produção no formato X → YZ

Mas agora **os dois primeiros símbolos devem ser gerados por Y**, ou os dos últimos por Z. Tenho que testar as DUAS possibilidades!

Grámática na FNC:

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid AB \mid BA$

$S \rightarrow AT \mid BU \mid SS \mid AB \mid BA$

$T \rightarrow SB$

$U \rightarrow SA$

$A \rightarrow a$

$B \rightarrow b$

$? \Rightarrow S_0A \overset{*}{\Rightarrow} aba$

$U \Rightarrow SA \overset{*}{\Rightarrow} aba$

Tabela:

(i,j) = (1,3) = aba = (1,2).(3,3)

	a	b	a	a	b	b
a	A	S_0, S	U			
b		B	S_0, S			
a			A	⊘		
a				A	S_0, S	
b					B	⊘
b						B

Cadeia:

a b a a b b

Algoritmo CYK para análise sintática

Exemplo:

Vamos analisar agora substrings de tamanho **3**

Por ser tamanho > 1, a variável que a gera a faz por meio de uma produção no formato X → YZ

Mas agora os dois primeiros símbolos devem ser gerados por Y, ou **os dos últimos por Z**. Tenho que testar as DUAS possibilidades!

Grámática na FNC:

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid AB \mid BA$

$S \rightarrow AT \mid BU \mid SS \mid AB \mid BA$

$T \rightarrow SB$

$U \rightarrow SA$

$A \rightarrow a$? => $AS_0 \overset{*}{\Rightarrow} aba$

$B \rightarrow b$? => $AS \overset{*}{\Rightarrow} aba$

Cadeia:

a b a a b b

Tabela:

(i,j) = (1,3) = aba = (1,1).(2,3)

	a	b	a	a	b	b
a	A	S_0, S	U			
b		B	S_0, S			
a			A	⊘		
a				A	S_0, S	
b					B	⊘
b						B

Algoritmo CYK para análise sintática

Exemplo:

Vamos analisar agora substrings de tamanho **3**

Por ser tamanho > 1, a variável que a gera a faz por meio de uma produção no formato X → YZ

Mas agora os dois primeiros símbolos devem ser gerados por Y, ou **os dos últimos por Z**. Tenho que testar as DUAS possibilidades!

Grámática na FNC:

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid AB \mid BA$

$S \rightarrow AT \mid BU \mid SS \mid AB \mid BA$

$T \rightarrow SB$

$U \rightarrow SA$

$A \rightarrow a$

$B \rightarrow b$

$? \Rightarrow AS_0 \overset{*}{\Rightarrow} aba$

$? \Rightarrow AS \overset{*}{\Rightarrow} aba$

Não, então fica só o U mesmo...

Cadeia:

a b a a b b

Tabela:

(i,j) = (1,3) = aba = (1,1).(2,3)

	a	b	a	a	b	b
a	A	S_0, S	U			
b		B	S_0, S			
a			A	⊘		
a				A	S_0, S	
b					B	⊘
b						B

Algoritmo CYK para análise sintática

Exemplo:

Vamos analisar agora substrings de tamanho **3**

Por ser tamanho > 1, a variável que a gera a faz por meio de uma produção no formato X → YZ

Mas agora os dois primeiros símbolos devem ser gerados por Y, ou **os dos últimos por Z**. Tenho que testar as DUAS possibilidades!

Grámática na FNC:

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid AB \mid BA$

$S \rightarrow AT \mid BU \mid SS \mid AB \mid BA$

$T \rightarrow SB$

$U \rightarrow SA$

$A \rightarrow a$

$B \rightarrow b$

Cadeia:

a b a a b b

Tabela:

(i,j) = (1,3) = aba

	a	b	a	a	b	b
a	A	S_0, S	U			
b		B	S_0, S			
a			A	⊘		
a				A	S_0, S	
b					B	⊘
b						B

w1...w3 = w1...w2 . w3...w3 ou w1...w1 . w2...w3

Algoritmo CYK para análise sintática

Exemplo:

Vamos analisar agora substrings de tamanho **3**

Por ser tamanho > 1, a variável que a gera a faz por meio de uma produção no formato X → YZ

Mas agora os dois primeiros símbolos devem ser gerados por Y, ou os dos últimos por Z. Tenho que testar as DUAS possibilidades!

Grámática na FNC:

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid AB \mid BA$

$S \rightarrow AT \mid BU \mid SS \mid AB \mid BA$

$T \rightarrow SB$

$U \rightarrow SA$

$A \rightarrow a$

$B \rightarrow b$

Tabela:

	a	b	a	a	b	b
a	A	S_0, S	U			
b		B	S_0, S	U		
a			A	⊘	⊘	
a				A	S_0, S	T
b					B	⊘
b						B

Cadeia:

a b a a b b

Algoritmo CYK para análise sintática

Exemplo:

Vamos analisar agora substrings de tamanho **4 (seta verde)**

Por ser tamanho > 1, a variável que a gera a faz por meio de uma produção no formato X → YZ

Mas agora a partição em YZ pode ocorrer após o primeiro símbolo, o segundo ou terceiro! Tenho que testar as TRÊS possibilidades!

Grámática na FNC:

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid AB \mid BA$

$S \rightarrow AT \mid BU \mid SS \mid AB \mid BA$

$T \rightarrow SB$

$U \rightarrow SA$

$A \rightarrow a$

$B \rightarrow b$

Cadeia:

a b a a b b

w1...w4 =

w1...w1 . w2...w4 ou

w1...w2 . w3...w4 ou

w1...w3 . w4...w4

Tabela:

	a	b	a	a	b	b
a	A	S_0, S	U			
b		B	S_0, S	U		
a			A	⊘	⊘	
a				A	S_0, S	T
b					B	⊘
b						B

Algoritmo CYK para análise sintática

Exemplo:

Vamos analisar agora substrings de tamanho **4 (seta verde)**

Por ser tamanho > 1, a variável que a gera a faz por meio de uma produção no formato X → YZ

Mas agora a partição em YZ pode ocorrer após o primeiro símbolo, o segundo ou terceiro! Tenho que testar as TRÊS possibilidades!

Grámática na FNC:

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid AB \mid BA$

$S \rightarrow AT \mid BU \mid SS \mid AB \mid BA$

$T \rightarrow SB$

$U \rightarrow SA$

$A \rightarrow a$

$B \rightarrow b$

Cadeia:

a b a a b b

Possibilidade 1

AU

w1...w4 =

w1...w1 . w2...w4 ou

w1...w2 . w3...w4 ou

w1...w3 . w4...w4

Tabela:

	a	b	a	a	b	b
a	A	S_0, S	U			
b		B	S_0, S	U		
a			A	⊘	⊘	
a				A	S_0, S	T
b					B	⊘
b						B

Algoritmo CYK para análise sintática

Exemplo:

Vamos analisar agora substrings de tamanho **4 (seta verde)**

Por ser tamanho > 1, a variável que a gera a faz por meio de uma produção no formato X → YZ

Mas agora a partição em YZ pode ocorrer após o primeiro símbolo, o segundo ou terceiro! Tenho que testar as TRÊS possibilidades!

Grámática na FNC:

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid AB \mid BA$

$S \rightarrow AT \mid BU \mid SS \mid AB \mid BA$

$T \rightarrow SB$

$U \rightarrow SA$

$A \rightarrow a$

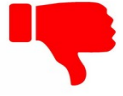
$B \rightarrow b$

Cadeia:

a b a a b b

Possibilidade 2

Nada gera aa



w1...w4 =

w1...w1 . w2...w4 ou

w1...w2 . w3...w4 ou

w1...w3 . w4...w4

Tabela:

	a	b	a	a	b	b
a	A	S_0, S	U			
b		B	S_0, S	U		
a			A	⊘	⊘	
a				A	S_0, S	T
b					B	⊘
b						B

Algoritmo CYK para análise sintática

Exemplo:

Vamos analisar agora substrings de tamanho **4 (seta verde)**

Por ser tamanho > 1, a variável que a gera a faz por meio de uma produção no formato X → YZ

Mas agora a partição em YZ pode ocorrer após o primeiro símbolo, o segundo ou terceiro! Tenho que testar as TRÊS possibilidades!

Grámática na FNC:

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid AB \mid BA$

$S \rightarrow AT \mid BU \mid SS \mid AB \mid BA$

$T \rightarrow SB$

$U \rightarrow SA$

$A \rightarrow a$

$B \rightarrow b$

Cadeia:

a b a a b b

Possibilidade 3

UA

w1...w4 =

w1...w1 . w2...w4 ou

w1...w2 . w3...w4 ou

w1...w3 . w4...w4

Tabela:

	a	b	a	a	b	b
a	A	S_0, S	U	⊘		
b		B	S_0, S	U		
a			A	⊘	⊘	
a				A	S_0, S	T
b					B	⊘
b						B

Algoritmo CYK para análise sintática

E ASSIM POR DIANTE....

Grámática na FNC:

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid AB \mid BA$

$S \rightarrow AT \mid BU \mid SS \mid AB \mid BA$

$T \rightarrow SB$

$U \rightarrow SA$

$A \rightarrow a$

$B \rightarrow b$

Cadeia:

a b a a b b

Tabela:

	a	b	a	a	b	b
a	A	S_0, S	U	⊘	⊘	S_0, S
b		B	S_0, S	U	S_0, S	T
a			A	⊘	⊘	S_0, S
a				A	S_0, S	T
b					B	⊘
b						B

Algoritmo CYK para análise sintática

$D =$ "On input $w = w_1 \cdots w_n$:

1. If $w = \varepsilon$ and $S \to \varepsilon$ is a rule, *accept*. [[handle $w = \varepsilon$ case]]
2. For $i = 1$ to n: [[examine each substring of length 1]]
3. For each variable A:
4. Test whether $A \to \mathtt{b}$ is a rule, where $\mathtt{b} = w_i$.
5. If so, place A in $table(i, i)$.
6. For $l = 2$ to n: [[l is the length of the substring]]
7. For $i = 1$ to $n - l + 1$: [[i is the start position of the substring]]
8. Let $j = i + l - 1$, [[j is the end position of the substring]]
9. For $k = i$ to $j - 1$: [[k is the split position]]
10. For each rule $A \to BC$:
11. If $table(i, k)$ contains B and $table(k + 1, j)$ contains C, put A in $table(i, j)$.

Algoritmo CYK para análise sintática

E aí, o que eu faço quando terminar de preencher a tabela?

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid AB \mid BA$
$S \rightarrow AT \mid BU \mid SS \mid AB \mid BA$
$T \rightarrow SB$
$U \rightarrow SA$
$A \rightarrow a$
$B \rightarrow b$

Cadeia:

a b a a b b

	a	b	a	a	b	b
a	A	S_0, S	U	⊘	⊘	S_0, S
b		B	S_0, S	U	S_0, S	T
a			A	⊘	⊘	S_0, S
a				A	S_0, S	T
b					B	⊘
b						B

Algoritmo CYK para análise sintática

E aí, o que eu faço quando terminar de preencher a tabela?

Se (1,n) contiver o símbolo inicial, então a cadeia é gerada pela gramática...

$S_0 \rightarrow \varepsilon \mid AT \mid BU \mid SS \mid AB \mid BA$
$S \rightarrow AT \mid BU \mid SS \mid AB \mid BA$
$T \rightarrow SB$
$U \rightarrow SA$
$A \rightarrow a$
$B \rightarrow b$

Cadeia:

a b a a b b

	a	b	a	a	b	b
a	A	S_0, S	U	⊘	⊘	S_0, S
b		B	S_0, S	U	S_0, S	T
a			A	⊘	⊘	S_0, S
a				A	S_0, S	T
b					B	⊘
b						B

Algoritmo CYK para análise sintática

$D =$ "On input $w = w_1 \cdots w_n$:

1. If $w = \varepsilon$ and $S \to \varepsilon$ is a rule, *accept*. [[handle $w = \varepsilon$ case]]
2. For $i = 1$ to n: [[examine each substring of length 1]]
3. For each variable A:
4. Test whether $A \to \mathtt{b}$ is a rule, where $\mathtt{b} = w_i$.
5. If so, place A in $table(i, i)$.
6. For $l = 2$ to n: [[l is the length of the substring]]
7. For $i = 1$ to $n - l + 1$: [[i is the start position of the substring]]
8. Let $j = i + l - 1$, [[j is the end position of the substring]]
9. For $k = i$ to $j - 1$: [[k is the split position]]
10. For each rule $A \to BC$:
11. If $table(i, k)$ contains B and $table(k + 1, j)$ contains C, put A in $table(i, j)$.
12. If S is in $table(1, n)$, *accept*. Otherwise, *reject*."

Algoritmo CYK para análise sintática

$D =$ "On input $w = w_1 \cdots w_n$:

1. If $w = \varepsilon$ and $S \to \varepsilon$ is a rule, *accept*. [[handle $w = \varepsilon$ case]]
2. For $i = 1$ to n: [[examine each substring of length 1]]
3. For each variable A:
4. Test whether $A \to \mathtt{b}$ is a rule, where $\mathtt{b} = w_i$.
5. If so, place A in $table(i, i)$.
6. For $l = 2$ to n: [[l is the length of the substring]]
7. For $i = 1$ to $n - l + 1$: [[i is the start position of the substring]]
8. Let $j = i + l - 1$, [[j is the end position of the substring]]
9. For $k = i$ to $j - 1$: [[k is the split position]]
10. For each rule $A \to BC$:
11. If $table(i, k)$ contains B and $table(k + 1, j)$ contains C, put A in $table(i, j)$.
12. If S is in $table(1, n)$, *accept*. Otherwise, *reject*."

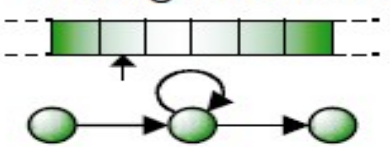
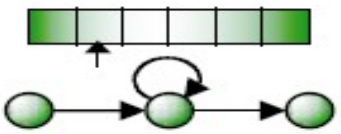
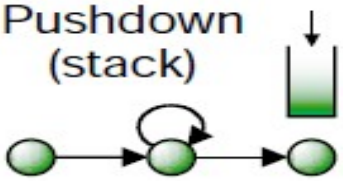
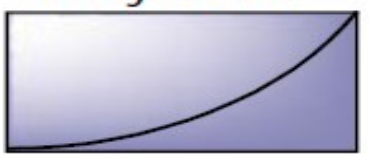
Complexidade: ?

Algoritmo CYK para análise sintática

D = "On input $w = w_1 \cdots w_n$:

1. If $w = \varepsilon$ and $S \to \varepsilon$ is a rule, *accept*. [[handle $w = \varepsilon$ case]]
2. For $i = 1$ to n: [[examine each substring of length 1]]
3. For each variable A:
4. Test whether $A \to \mathtt{b}$ is a rule, where $\mathtt{b} = w_i$.
5. If so, place A in $table(i, i)$.
6. For $l = 2$ to n: [[l is the length of the substring]]
7. For $i = 1$ to $n - l + 1$: [[i is the start position of the substring]]
8. Let $j = i + l - 1$, [[j is the end position of the substring]]
9. For $k = i$ to $j - 1$: [[k is the split position]]
10. For each rule $A \to BC$:
11. If $table(i, k)$ contains B and $table(k + 1, j)$ contains C, put A in $table(i, j)$.
12. If S is in $table(1, n)$, *accept*. Otherwise, *reject*."

Complexidade: $O(n^3 m)$, sendo n = tamanho da cadeia, m = nr de prod de G

Recursively enumerable languages	Turing machine 	Unrestricted $Baa \rightarrow A$	Undecidable 
Context-sensitive languages	Linear bounded 	Context sensitive $At \rightarrow aA$	Exponential? 
Context-free languages	Pushdown (stack) 	Context free $S \rightarrow gSc$	Polynomial 
Regular languages	Finite-state automaton 	Regular $A \rightarrow cA$	Linear 

Caso geral

Exercícios

- Sipser (cap 2): 2.1, 2.3, 2.4, 2.6 (a e b), 2.8, 2.9, 2.16, 2.17
- Use o algoritmo CYK para fazer a análise sintática de baab
- Obs: lembrem-se que o símbolo inicial de uma gramática é o símbolo do lado esquerdo da primeira produção da gramática