ACH2043
INTRODUÇÃO À TEORIA DA COMPUTAÇÃO

Cap. 2.1 – Gramáticas Livres de Contexto

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

Introdução a gramáticas livres de contexto

Aulas 7 a 10

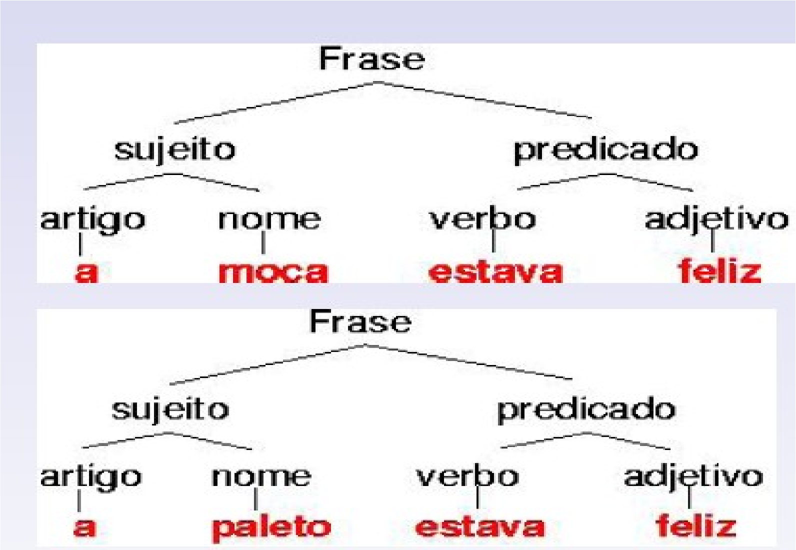
- Aula 7: conceitos básicos de gramáticas e hierarquia de Chomsky

Gramáticas

conjunto de produções

símbolo inicial

Frase	→	sujeito	predicado
sujeito	→	artigo	nome
artigo	→	a	
artigo	→	o	
nome	→	paletó	
nome	→	moça	
nome	→	dia	
predicado	→	verbo	adjetivo
verbo	→	é	
verbo	→	estava	
adjectivo	→	feliz	
adjectivo	→	azul	



símbolos não-terminais

símbolos terminais

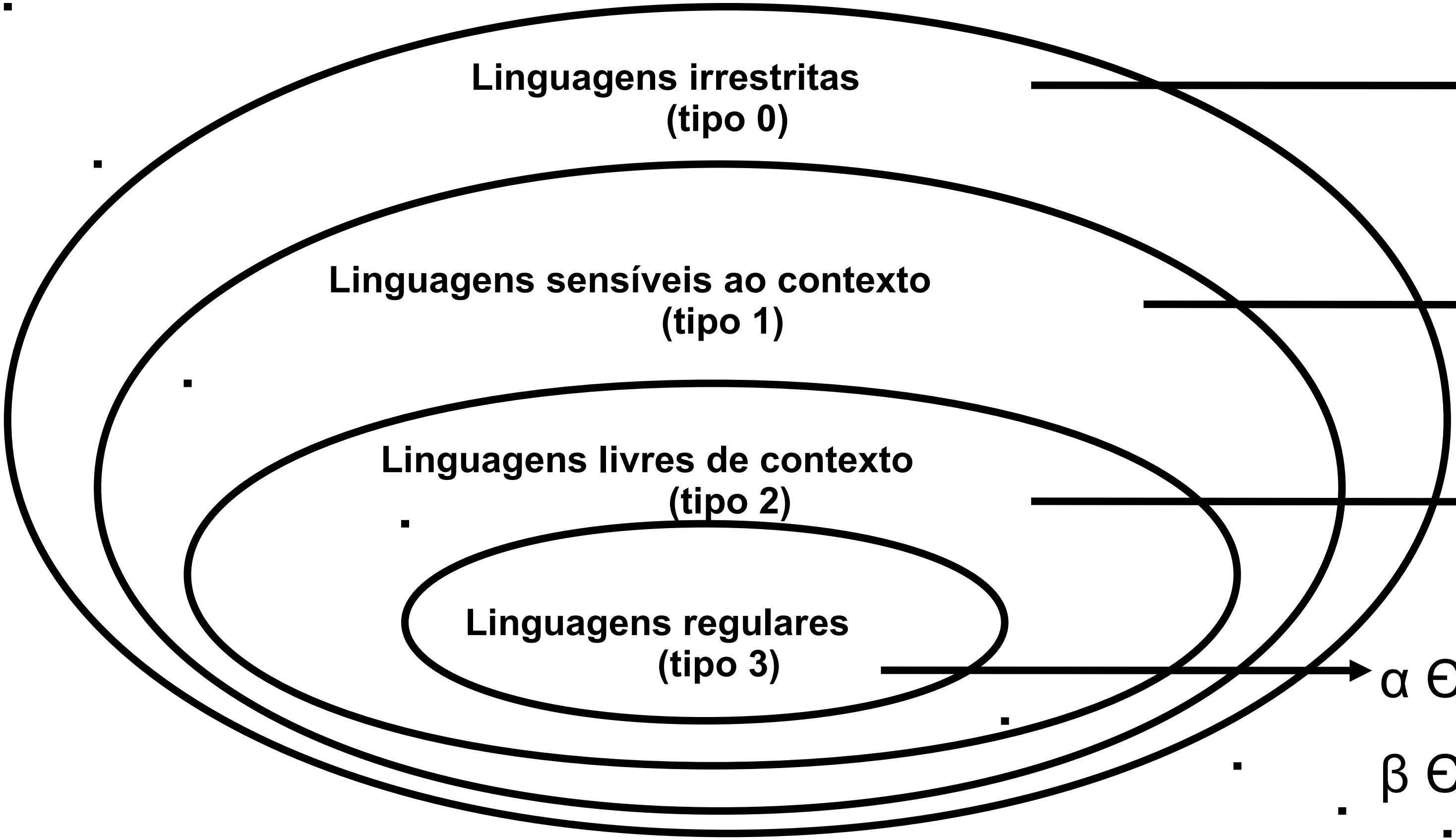
Gramáticas

- Definição: uma gramática G é uma quádrupla (V, Σ, S, P), onde
  - V é o conjunto de símbolos não-terminais (variáveis)
  - Σ é o conjunto de símbolos terminais
  - S é o símbolo inicial
  - P é o conjunto de produções da forma

$$(\Sigma \cup V)^* \, V \, (\Sigma \cup V)^* \rightarrow (\Sigma \cup V)^*$$

Hierarquia de Chomsky

$$\alpha \rightarrow \beta$$



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

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

$\alpha \in V$

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

$\alpha \in V$

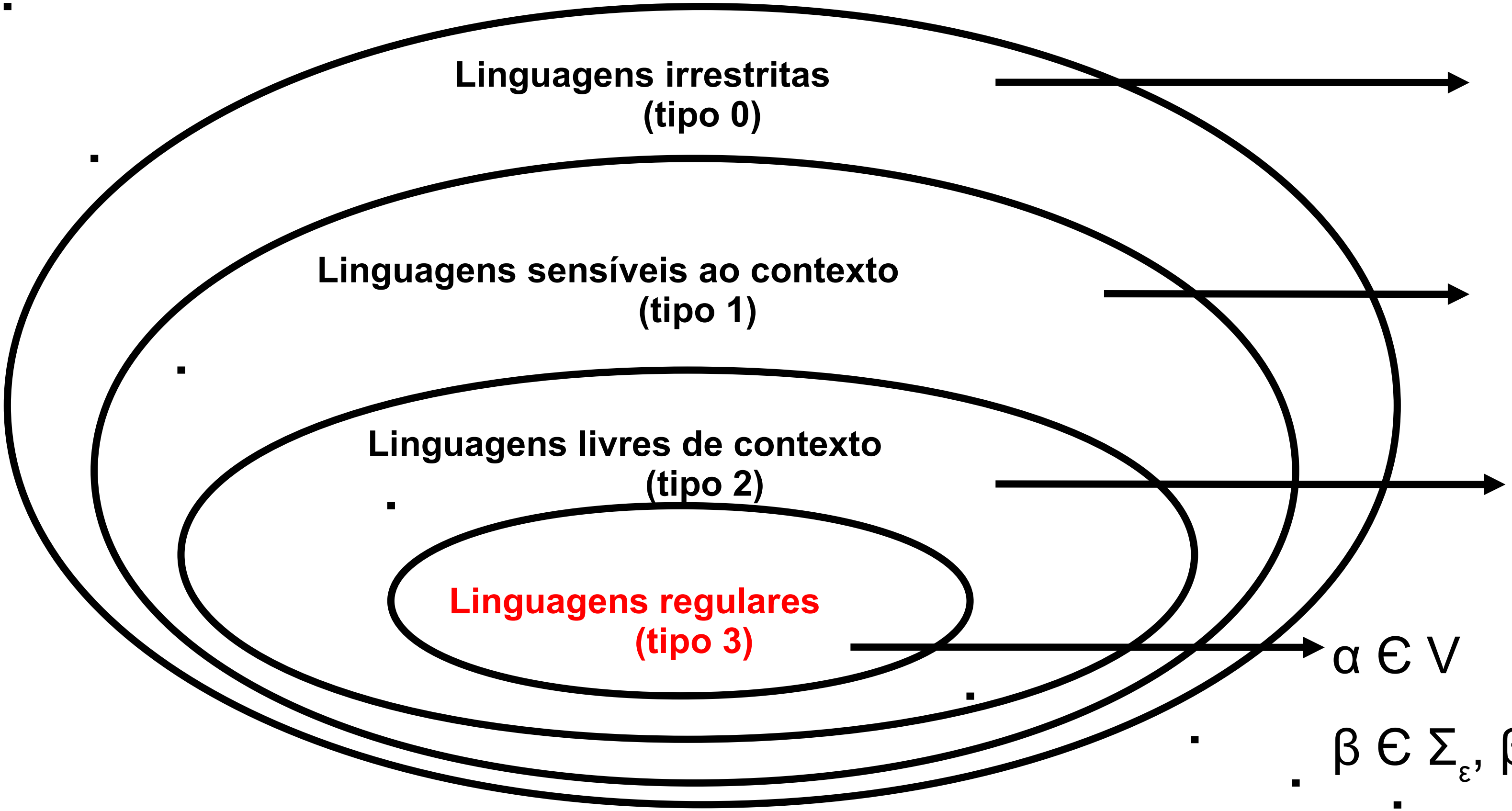
$\beta \in \Sigma_\varepsilon$, $\beta \in V$, $\beta \in (V\Sigma$ ou $\Sigma V)$

Aulas 7 a 10

- Aula 7: conceitos básicos de gramáticas e hierarquia de Chomsky
- Aula 8: Gramáticas regulares e autômatos

Hierarquia de Chomsky

$\alpha \rightarrow \beta$



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

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

$\alpha \in V$

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

$\alpha \in V$

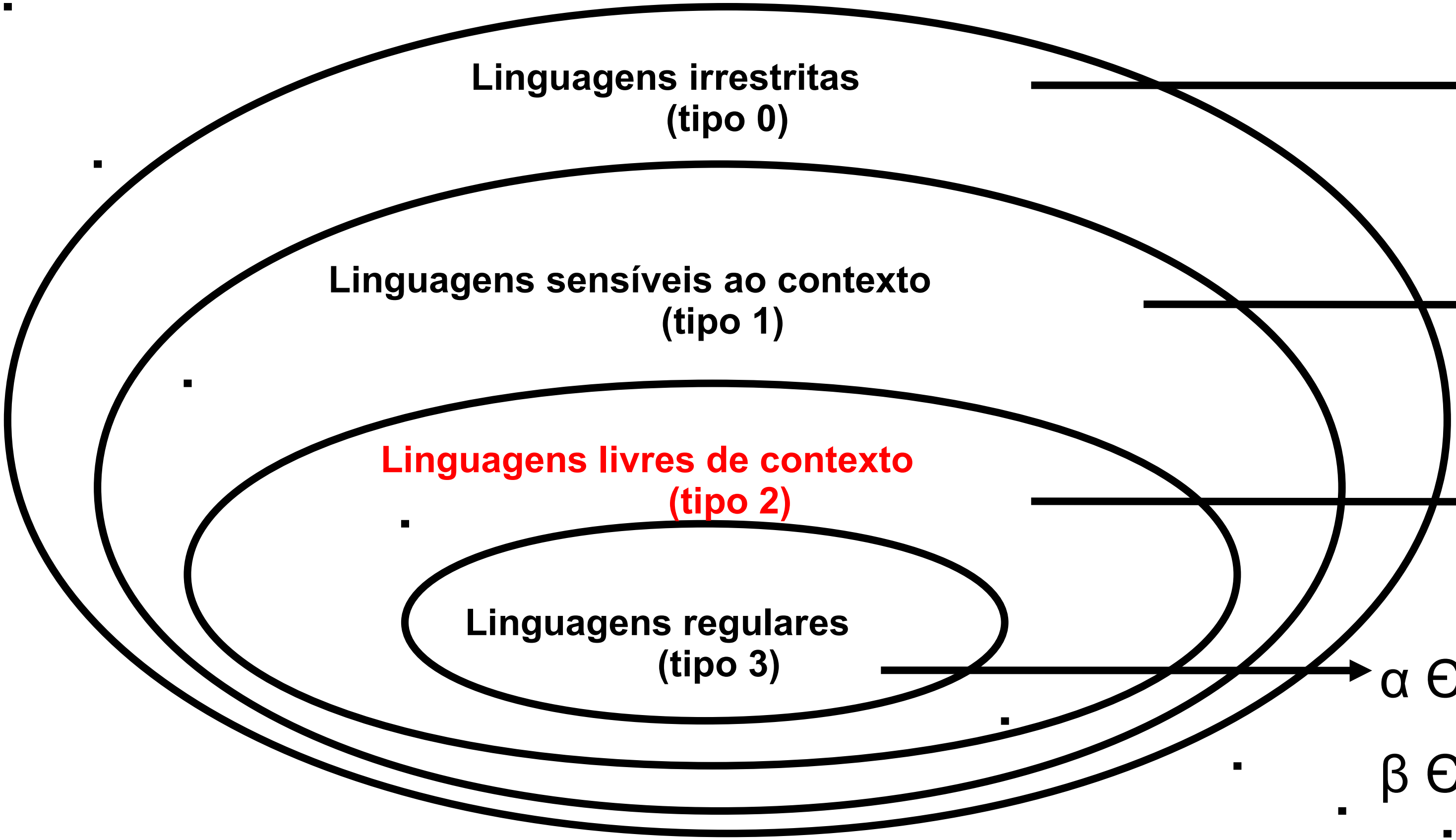
$\beta \in \Sigma_{\varepsilon}$, $\beta \in V$, $\beta \in (V\Sigma$ ou $\Sigma V)$

Aulas 7 a 10

- Aula 7: conceitos básicos de gramáticas e hierarquia de Chomsky
- Aula 8: Gramáticas regulares e autômatos
- Aula 9: Gramáticas estocásticas
- Aula 10: HMMs

Aula de hoje

$$\alpha \rightarrow \beta$$



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

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

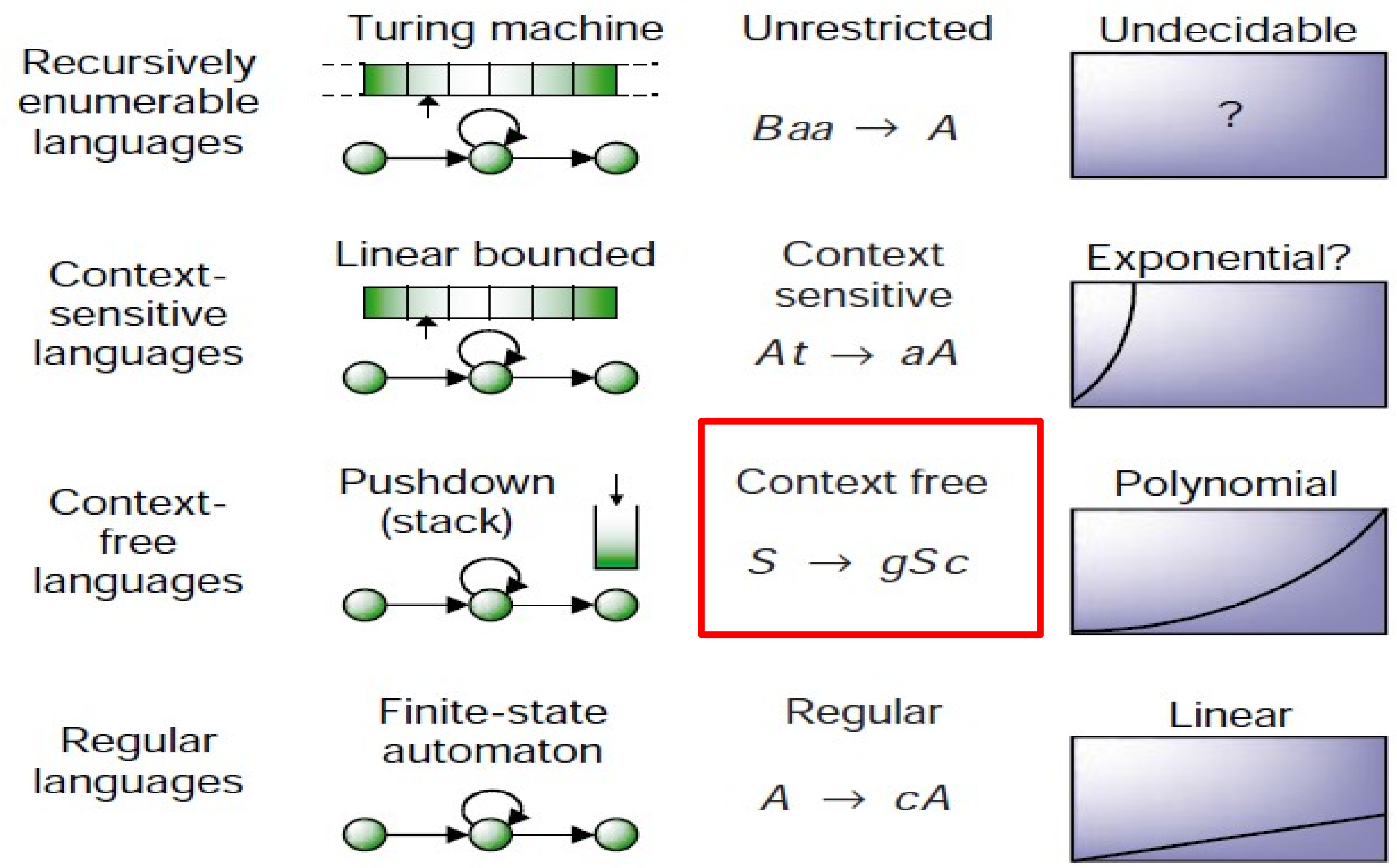
$\alpha \in V$

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

$\alpha \in V$

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

Linguagens, dispositivos, gramáticas e complexidades

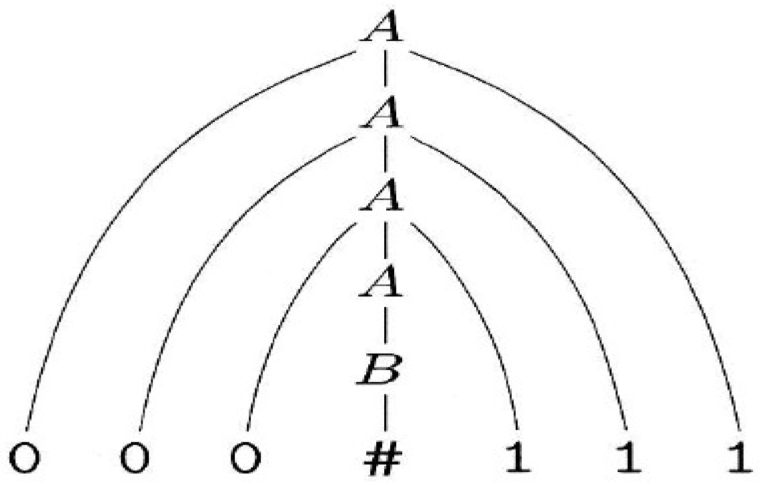


Gramáticas Livres de Contexto

$$
\begin{aligned}
A &\rightarrow 0A1 \\
A &\rightarrow B \\
B &\rightarrow \#
\end{aligned}
$$

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?

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?

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?

$S \rightarrow 0\ S\ 1$

$S \rightarrow \varepsilon$

Projetando gramáticas livres de contexto

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

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

$$S_1 \to 0S_11 \mid \varepsilon$$

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

?

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

Dicas para projetar gramáticas livres de contexto

- É a união de linguagens mais simples?

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

$$S_1 \rightarrow 0S_11 \mid \varepsilon$$

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

$$S_2 \rightarrow 1S_20 \mid \varepsilon$$

para a linguagem $\{1^n0^n | n \geq 0\}$ e então adicione a regra ? para dar a gramática

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

$$S_1 \to 0S_11 \mid \varepsilon$$

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

$$S_2 \to 1S_20 \mid \varepsilon$$

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

$$\begin{aligned} S &\to S_1 \mid S_2 \\ S_1 &\to 0S_11 \mid \varepsilon \\ S_2 &\to 1S_20 \mid \varepsilon\ . \end{aligned}$$

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

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

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

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

Dá para representar RNAs com isso!!!!

Fim do vídeo

Introdução a gramáticas livres de contexto

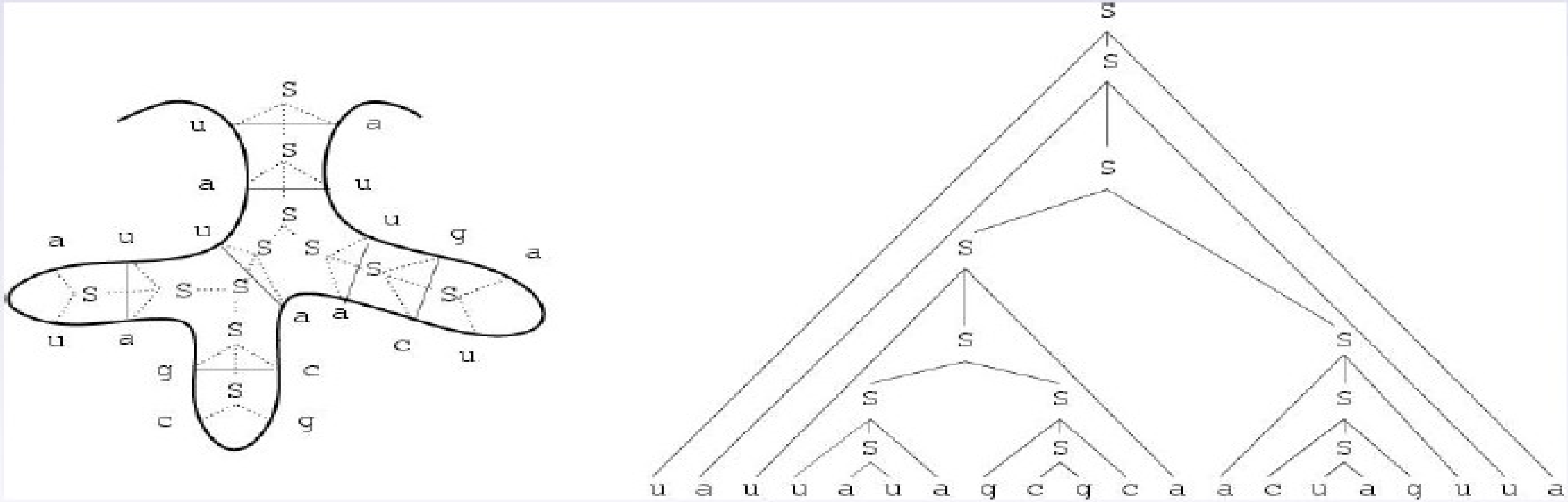
Aplicações de gramáticas livres de contexto

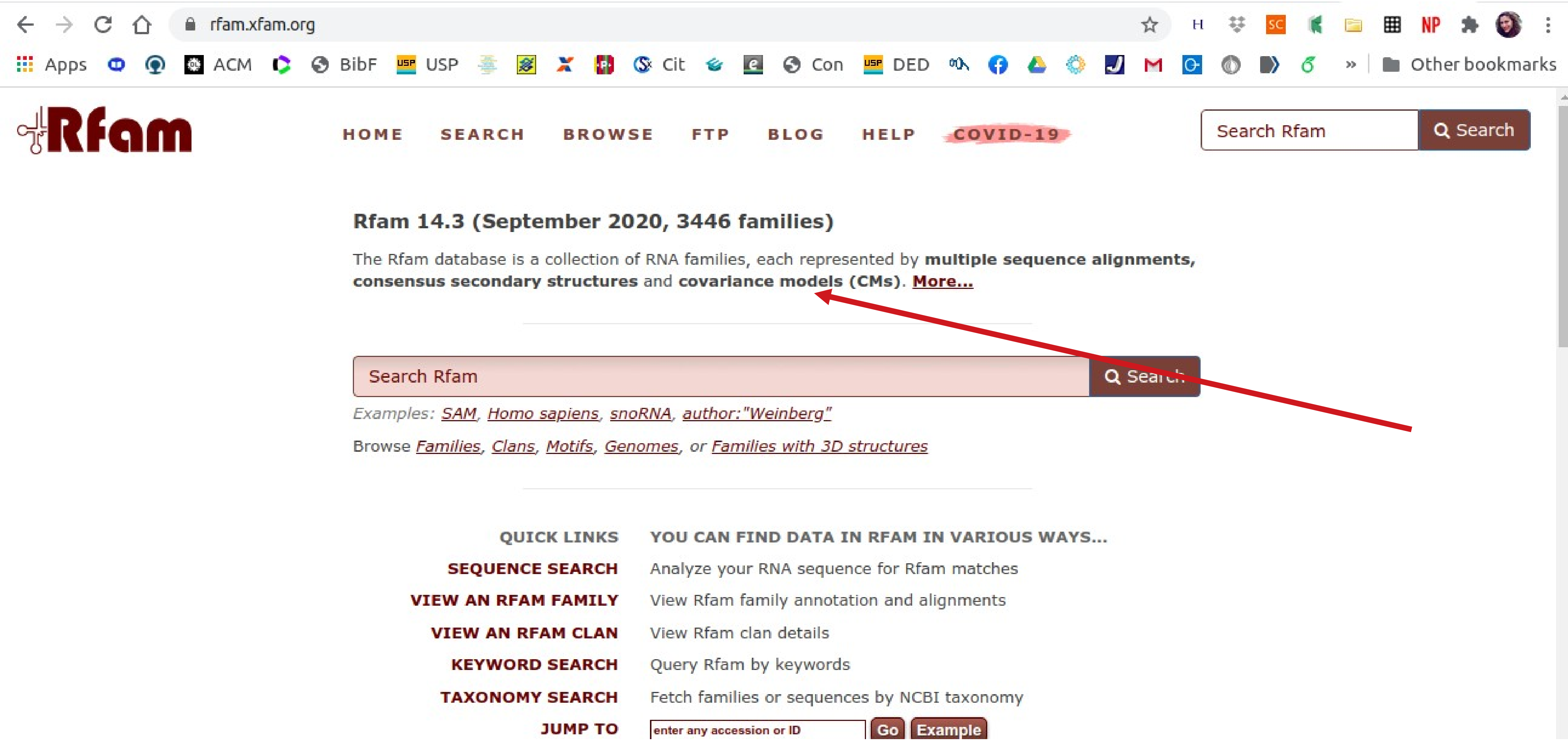
Abrindo parêntesis ….

para exemplificar algumas aplicações importantes de gramáticas livres de contexto….

Estrutura secundária de RNAs

S →	aSu [0.1]	uSa [0.1]	cSg [0.1]	gSc [0.1]	
S →	aS [0.1]	uS [0.1]	cS [0.1]	gS [0.1]	
S →	SS [0.04]	a [0.04]	u [0.04]	c [0.04]	g [0.04]




rfam.xfam.org
Apps
ACM
BibF
USP
Cit
Con
DED
Other bookmarks
Rfam
HOME
SEARCH
BROWSE
FTP
BLOG
HELP
COVID-19
Search Rfam
Search
Rfam 14.3 (September 2020, 3446 families)
The Rfam database is a collection of RNA families, each represented by multiple sequence alignments, consensus secondary structures and covariance models (CMs). More...
Search Rfam
Search
Examples: SAM, Homo sapiens, snoRNA, author:"Weinberg"
Browse Families, Clans, Motifs, Genomes, or Families with 3D structures
QUICK LINKS
YOU CAN FIND DATA IN RFAM IN VARIOUS WAYS...
SEQUENCE SEARCH
Analyze your RNA sequence for Rfam matches
VIEW AN RFAM FAMILY
View Rfam family annotation and alignments
VIEW AN RFAM CLAN
View Rfam clan details
KEYWORD SEARCH
Query Rfam by keywords
TAXONOMY SEARCH
Fetch families or sequences by NCBI taxonomy
JUMP TO
enter any accession or ID
Go
Example

ELSEVIER

Artificial Intelligence in Medicine 26 (2002) 145–159

Artificial Intelligence in Medicine

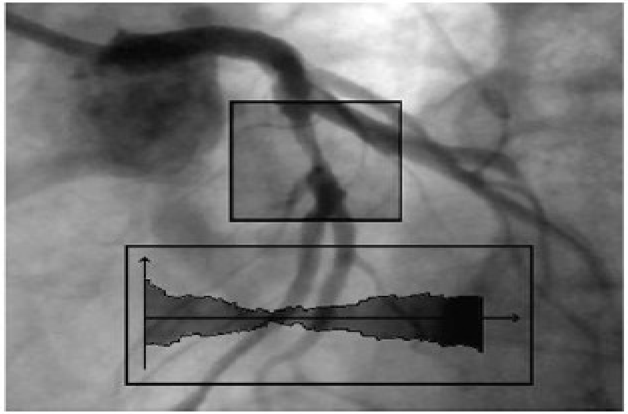
www.elsevier.com/locate/artmed

Syntactic reasoning and pattern recognition for analysis of coronary artery images

Marek R. Ogiela*, Ryszard Tadeusiewicz

Institute of Automatics 30 Mickiewicza Avenue, University of Mining and Metallurgy, PL-30-059, Krakow, Poland

Received 5 March 2002; received in revised form 23 April 2002; accepted 23 April 2002



$$V_{\mathrm{N}} = \{\mathrm{SYMPTOM}, \mathrm{STENOSIS}, H, V, \mathrm{NV}\}$$

$$V_{\mathrm{T}} = \{h, v, nv\} \text{ for } h \in (-10^\circ, 10^\circ), v \in (11^\circ, 90^\circ), nv \in (-11^\circ, -90^\circ)$$

1. $\mathrm{SYMPTOM} \longrightarrow \mathrm{STENOSIS}$
2. $\mathrm{STENOSIS} \longrightarrow \mathrm{NV\ H\ V}$
3. $\mathrm{STENOSIS} \longrightarrow \mathrm{NV}\ \ V|\mathrm{NV}\ \ H$
4. $V \longrightarrow v|V\ v$
5. $\mathrm{NV} \longrightarrow \mathrm{nv}|\mathrm{NV}\ \ \mathrm{nv}$
6. $H \longrightarrow h|H\ h$

Linguagens de programação

- Gramática que define uma linguagem de programação
- Compilador que checa se o programa está de acordo com a gramática (se o programa pertence à linguagem gerada pela 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 $L \rightarrow L \; , \; E$

Outro exemplo:
http://pubs.opengroup.org/onlinepubs/7908799/xcu/awk.html#tag_000_000_108_016

Using Grammars for Pattern Recognition in Images: A Systematic Review

26

RICARDO WANDRÉ DIAS PEDRO, FÁTIMA L. S. NUNES, and ARIANE MACHADO-LIMA, School of Arts, Sciences and Humanities, University of Sao Paulo, Brazil

Grammars are widely used to describe string languages such as programming and natural languages and, more recently, biosequences. Moreover, since the 1980s grammars have been used in computer vision and related areas. Some factors accountable for this increasing use regard its relatively simple understanding and its ability to represent some semantic pattern models found in images, both spatially and temporally. The objective of this article is to present an overview regarding the use of syntactic pattern recognition methods in image representations in several applications. To achieve this purpose, we used a systematic review process to investigate the main digital libraries in the area and to document the phases of the study in order to allow the auditing and further investigation. The results indicated that in some of the studies retrieved, manually created grammars were used to comply with a particular purpose. Other studies performed a learning process of the grammatical rules. In addition, this article also points out still unexplored research opportunities in the literature.

Categories and Subject Descriptors: F.4.2 [**Mathematical Logic and Formal Language**]: Grammars and Other Rewriting Systems—*Decision problems*; I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis; I.4.10 [**Image Processing and Computer Vision**]: Image Representation; I.5.1 [**Pattern Recognition**]: Models—*Structural*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Image grammars, computer vision, image representation, formal languages, syntactic methods, pattern recognition

ACM Reference Format:
Pedro, R. W. D., Nunes, F. L. S., and Machado-Lima, A. 2013. Using grammars for pattern recognition in images: A systematic review. ACM Comput. Surv. 46, 2, Article 26 (November 2013), 34 pages.
DOI: http://dx.doi.org/10.1145/2543581.2543593

Received: 28 September 2019 | Revised: 25 February 2020 | Accepted: 10 March 2020

DOI: 10.1111/coin.12320

SPECIAL ISSUE ARTICLE

Computational Intelligence WILEY

Towards an approach using grammars for automatic classification of masses in mammograms

Ricardo Wandré Dias Pedro[1] | Ariane Machado-Lima[2] | Fátima L. S. Nunes[1,2]

… fechando parêntesis.

Fim do vídeo

Aplicações de gramáticas livres de contexto

Análise sintática e ambiguidade

Derivações

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

Derivações

Gramática:

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

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

$$
\begin{array}{l}
\underline{S} \\
S\ ;\ \underline{S} \\
\underline{S}\ ;\ \mathrm{id}\ :=\ E \\
\mathrm{id}\ :=\ \underline{E}\ ;\ \mathrm{id}\ :=\ E \\
\mathrm{id}\ :=\ \mathrm{num}\ ;\ \mathrm{id}\ :=\ \underline{E} \\
\mathrm{id}\ :=\ \mathrm{num}\ ;\ \mathrm{id}\ :=\ E + \underline{E} \\
\mathrm{id}\ :=\ \mathrm{num}\ ;\ \mathrm{id}\ :=\ \underline{E} + (S\ ,\ E\ ) \\
\mathrm{id}\ :=\ \mathrm{num}\ ;\ \mathrm{id}\ :=\ \mathrm{id} + (\underline{S}\ ,\ E\ ) \\
\mathrm{id}\ :=\ \mathrm{num}\ ;\ \mathrm{id}\ :=\ \mathrm{id} + (\mathrm{id}\ :=\ \underline{E}\ ,\ E\ ) \\
\mathrm{id}\ :=\ \mathrm{num}\ ;\ \mathrm{id}\ :=\ \mathrm{id} + (\mathrm{id}\ :=\ E + E\ ,\ \underline{E}\ ) \\
\mathrm{id}\ :=\ \mathrm{num}\ ;\ \mathrm{id}\ :=\ \mathrm{id} + (\mathrm{id}\ :=\ \underline{E} + E\ ,\ \mathrm{id}\ ) \\
\mathrm{id}\ :=\ \mathrm{num}\ ;\ \mathrm{id}\ :=\ \mathrm{id} + (\mathrm{id}\ :=\ \mathrm{num} + \underline{E}\ ,\ \mathrm{id}\ ) \\
\mathrm{id}\ :=\ \mathrm{num}\ ;\ \mathrm{id}\ :=\ \mathrm{id} + (\mathrm{id}\ :=\ \mathrm{num} + \mathrm{num}\ ,\ \mathrm{id}\ )
\end{array}
$$

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 $S \rightarrow S \; ; \; S$
2 $S \rightarrow \mathrm{id} := E$
3 $S \rightarrow \mathrm{print} \; (\; L \;)$
4 $E \rightarrow \mathrm{id}$
5 $E \rightarrow \mathrm{num}$
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)`

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

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

Uma derivação possível
Mais à esquerda ou mais à direita? Nenhuma das 2

Derivações

Gramática:

1. $S \rightarrow S \; ; \; S$
2. $S \rightarrow \mathrm{id} := E$
3. $S \rightarrow \mathrm{print} \; (\, L \,)$
4. $E \rightarrow \mathrm{id}$
5. $E \rightarrow \mathrm{num}$
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)`

$$
\begin{array}{l}
\underline{S} \\
S \; ; \; \underline{S} \\
\underline{S} \; ; \; \mathrm{id} := E \\
\mathrm{id} := \underline{E} \; ; \; \mathrm{id} := E \\
\mathrm{id} := \mathrm{num} \; ; \; \mathrm{id} := \underline{E} \\
\mathrm{id} := \mathrm{num} \; ; \; \mathrm{id} := E + \underline{E} \\
\mathrm{id} := \mathrm{num} \; ; \; \mathrm{id} := \underline{E} + (S \, , \, E \,) \\
\mathrm{id} := \mathrm{num} \; ; \; \mathrm{id} := \mathrm{id} + (\underline{S} \, , \, E \,) \\
\mathrm{id} := \mathrm{num} \; ; \; \mathrm{id} := \mathrm{id} + (\mathrm{id} := \underline{E} \, , \, E \,) \\
\mathrm{id} := \mathrm{num} \; ; \; \mathrm{id} := \mathrm{id} + (\mathrm{id} := E + E \, , \, \underline{E} \,) \\
\mathrm{id} := \mathrm{num} \; ; \; \mathrm{id} := \mathrm{id} + (\mathrm{id} := \underline{E} + E \, , \, \mathrm{id} \,) \\
\mathrm{id} := \mathrm{num} \; ; \; \mathrm{id} := \mathrm{id} + (\mathrm{id} := \mathrm{num} + \underline{E} \, , \, \mathrm{id} \,) \\
\mathrm{id} := \mathrm{num} \; ; \; \mathrm{id} := \mathrm{id} + (\mathrm{id} := \mathrm{num} + \mathrm{num} \, , \, \mathrm{id} \,)
\end{array}
$$

Uma derivação possível
Mais à esquerda ou mais à direita? Nenhuma das 2

$$
\begin{array}{l}
\underline{S} \\
\underline{S} \; ; \; S \\
\mathrm{id} := \underline{E} \; ; \; S \\
\mathrm{id} := \mathrm{num} \; ; \; \underline{S} \\
\mathrm{id} := \mathrm{num} \; ; \; \mathrm{id} := \underline{E} \\
\mathrm{id} := \mathrm{num} \; ; \; \mathrm{id} := \underline{E} + E \\
\quad \vdots
\end{array}
$$

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

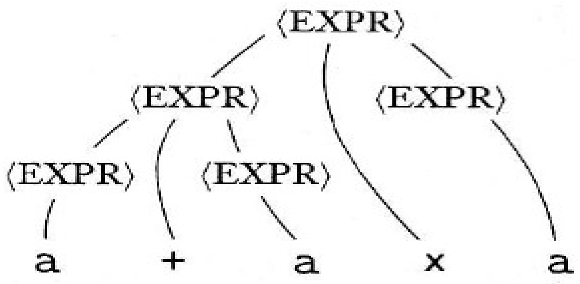
Obs: passo necessário para a compilação de um programa em uma dada linguagem de programação

Ex: gramática para expressões aritméticas simples

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

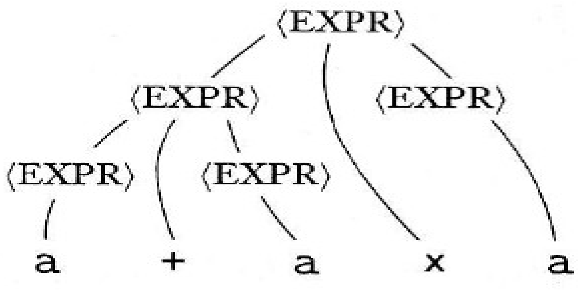
Ex: gramática para expressões aritméticas simples

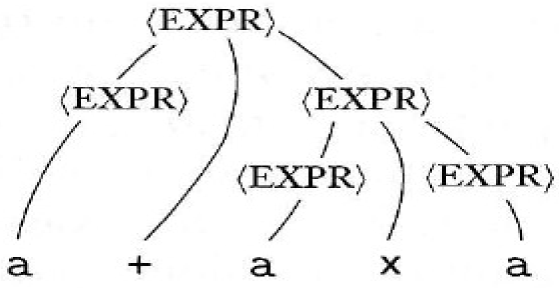
$$\langle \mathrm{EXPR} \rangle \rightarrow \langle \mathrm{EXPR} \rangle \mathtt{+} \langle \mathrm{EXPR} \rangle \mid \langle \mathrm{EXPR} \rangle \mathtt{x} \langle \mathrm{EXPR} \rangle \mid \mathtt{(}\, \langle \mathrm{EXPR} \rangle \,\mathtt{)} \mid \mathtt{a}$$



Ex: gramática para expressões aritméticas simples

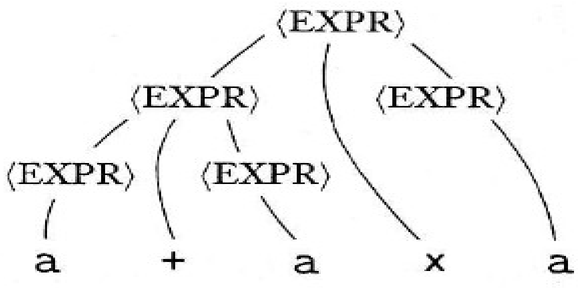
$$\langle \mathrm{EXPR} \rangle \rightarrow \langle \mathrm{EXPR} \rangle \mathtt{+} \langle \mathrm{EXPR} \rangle \mid \langle \mathrm{EXPR} \rangle \mathtt{x} \langle \mathrm{EXPR} \rangle \mid \mathtt{(} \langle \mathrm{EXPR} \rangle \mathtt{)} \mid \mathtt{a}$$

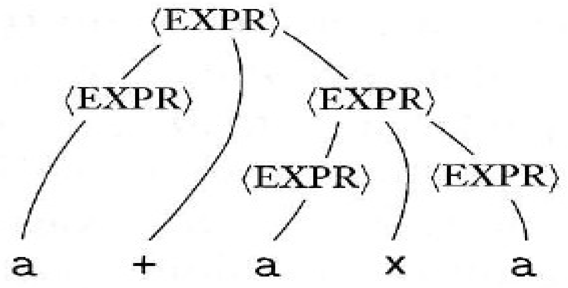




Ex: gramática para expressões aritméticas simples

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





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

Ex: gramática para expressões aritméticas simples

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

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

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.

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.

- Ambiguidade é às vezes indesejável, por exemplo em linguagens de programação
- Algumas gramáticas ambíguas podem ser convertidas em não-ambíguas
- Algumas linguagens são inerentemente ambíguas (só podem ser descritas por gramáticas ambíguas)
  - Eu vi o menino com uma luneta

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 $\{\mathtt{a}, +, \mathtt{x}, (,)\}$. As regras são

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

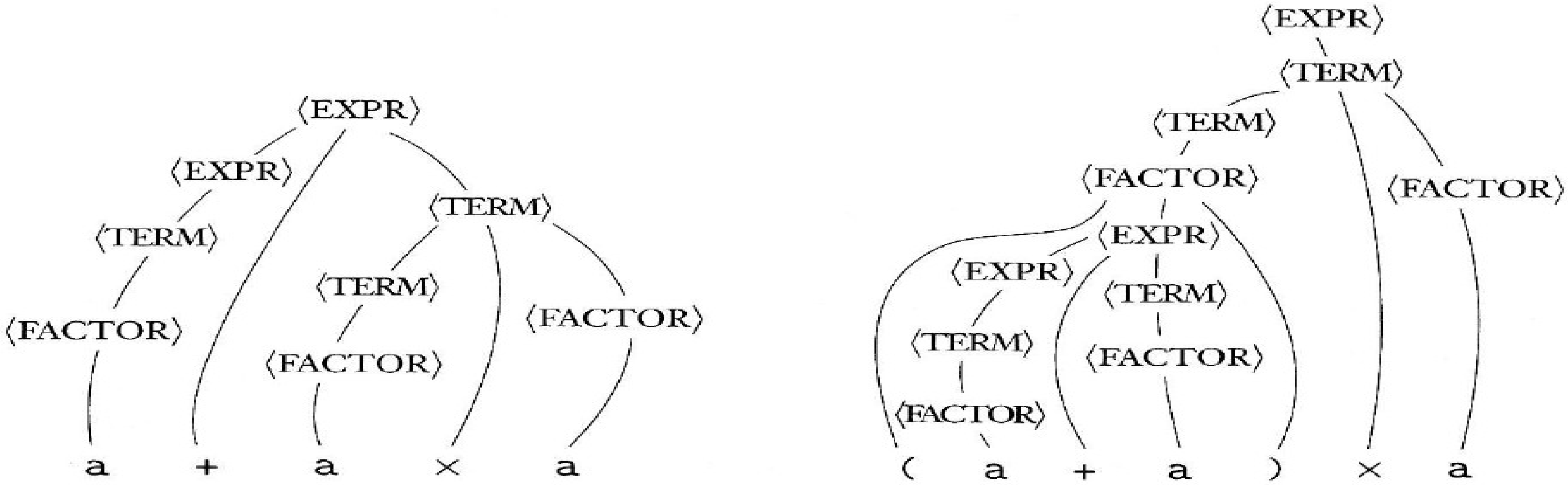
Expressões aritméticas sem ambiguidade

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 $\{\texttt{a}, +, \texttt{x}, (,)\}$. As regras são

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



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 eles – 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

Fim do vídeo

Análise sintática e ambiguidade

Vídeo

Mais sobre análise sintática, Forma Normal de Chomsky

Análise sintática

Problema: Dada uma gramática G e uma cadeia w, saber se w є L(G) (isto é, encontrar ao menos uma derivação 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… 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 ou mais complexas
- Estratégias dependentes das gramáticas (subclasses de livres-de-contexto)
- Tema da disciplina de construção de compiladores

Análise sintática descendente

- Top-down
- Cada não terminal A teria uma sub-rotina associada para tratar todas as possibilidades de produção que o tenha do lado esquerdo (A $\rightarrow$ ...)

Exemplo

Gramática:

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

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

$\underline{S}$

$\underline{S} \; ; \; S$

$\mathrm{id} := \underline{E} \; ; \; S$

$\mathrm{id} := \mathrm{num} \; ; \; \underline{S}$

$\mathrm{id} := \mathrm{num} \; ; \; \mathrm{id} := \underline{E}$

$\mathrm{id} := \mathrm{num} \; ; \; \mathrm{id} := \underline{E} + E$

$\vdots$

Derivação mais à esquerda

Exemplo

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)`

$$
\begin{array}{l}
\underline{S} \\
\underline{S}\ ;\ S \\
\text{id} := \underline{E}\ ;\ S \\
\text{id} := \text{num}\ ;\ \underline{S} \\
\text{id} := \text{num}\ ;\ \text{id} := \underline{E} \\
\text{id} := \text{num}\ ;\ \text{id} := \underline{E} + E \\
\quad \vdots
\end{array}
$$

"Recursão à esquerda": quando eu páro de chamar recursivamente S?

Derivação mais à esquerda

Exemplo 2

<com> → if (<exp>) <com> else <com>

| while (<exp>) <com>

| { <lista_com> }

Exemplo 2

<com> → if (<exp>) <com> else <com>

| while (<exp>) <com>

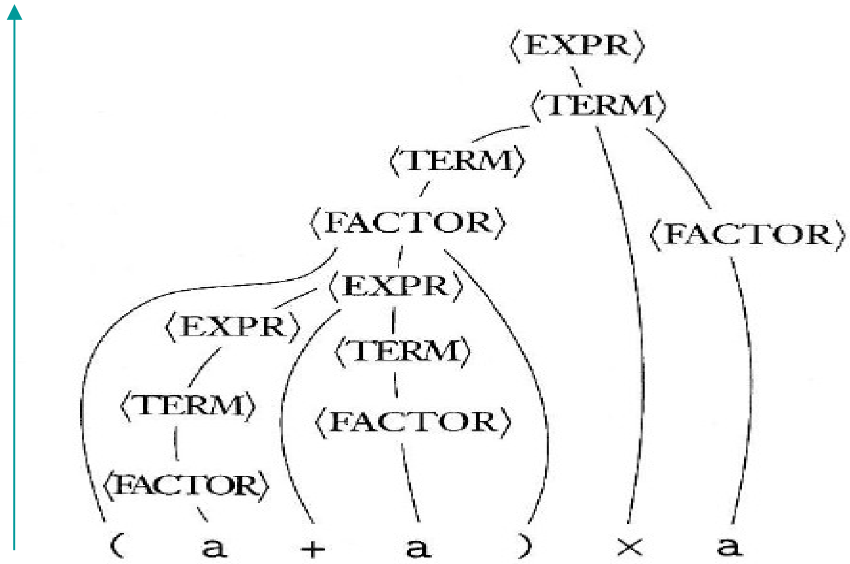
| { <lista_com> }

Fácil de tratar

Análise sintática ascendente

- Bottom-up

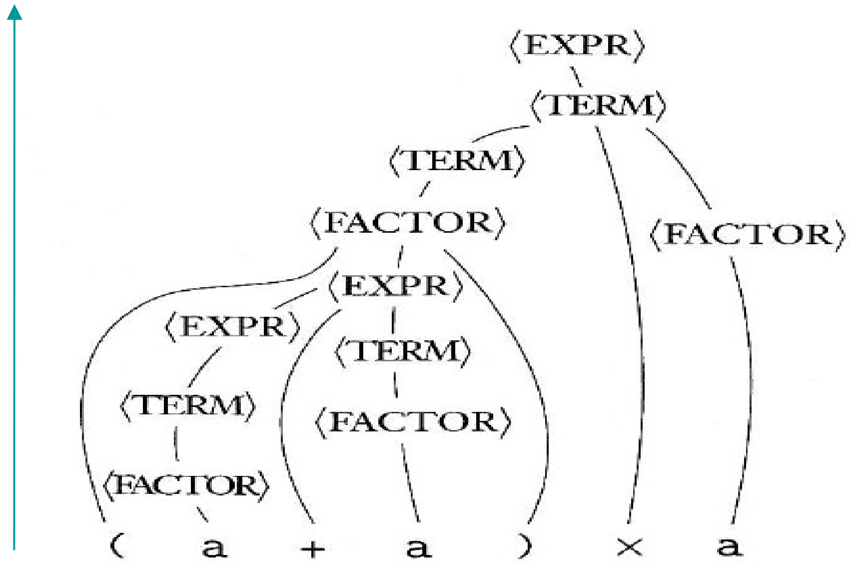
$$\langle \mathrm{EXPR} \rangle \rightarrow \langle \mathrm{EXPR} \rangle + \langle \mathrm{TERM} \rangle \mid \langle \mathrm{TERM} \rangle$$
$$\langle \mathrm{TERM} \rangle \rightarrow \langle \mathrm{TERM} \rangle \times \langle \mathrm{FACTOR} \rangle \mid \langle \mathrm{FACTOR} \rangle$$
$$\langle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



Análise sintática ascendente

- Bottom-up
- Umas das estratégias: algoritmo CYK

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



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?
- 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

$A \rightarrow BC$ ou $A \rightarrow a$

sendo B,C variáveis, 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:

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

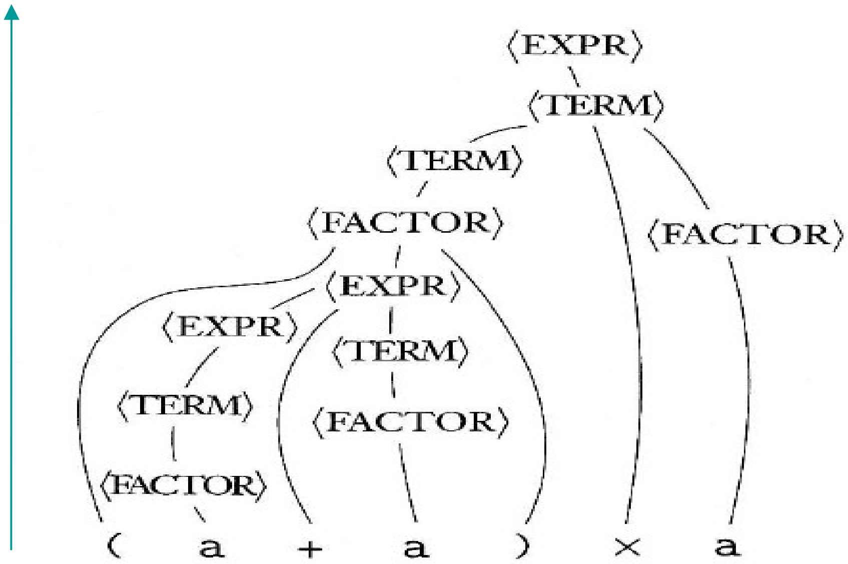
- Conversão para FNC (veremos na próxima aula):

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

Algoritmo CYK para análise sintática

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

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



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 $n \times n$:
  - Para $i \leq j$, 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)

Algoritmo CYK para análise sintática

Exemplo:

Grámática na FNC:

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

Cadeia:

a b a a b b

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

Tabela:

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

Algoritmo CYK para análise sintática

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

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

Algoritmo CYK para análise sintática

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

1. If $w = \varepsilon$ and $S \rightarrow \varepsilon$ 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 $A \rightarrow \mathtt{b}$ is a rule, where $\mathtt{b} = w_i$.
5. If so, place A in $table(i, i)$.

Algoritmo CYK para análise sintática

Exemplo:

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						
b						
a						
a						
b						
b						

Algoritmo CYK para análise sintática

Exemplo:

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					
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 $S \rightarrow \varepsilon$ 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 $A \rightarrow$ b is a rule, where 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 ser tamanho > 1, a variável que a gera a faz por meio de uma produção no formato $X \rightarrow YZ$

(i,j) = (1,2)

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					
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 → YZ

Grámática na FNC:

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

ab é gerado por AB

Tabela:

(i,j) = (1,2)

	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 ser tamanho > 1, a variável que a gera a faz por meio de uma produção no formato $X \rightarrow YZ$

(i,j) = (1,2)

Grámática na FNC:

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

ab é gerado por 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 ser tamanho > 1, a variável que a gera a faz por meio de uma produção no formato $X \rightarrow YZ$

Grámática na FNC:

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

ab é gerado por AB

Tabela:

(i,j) = (1,2)

	a	b	a	a	b	b
a	A	S_0, S				
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 ser tamanho > 1, a variável que a gera a faz por meio de uma produção no formato $X \rightarrow YZ$

(i,j) = (2,3) = 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 $X \rightarrow YZ$

(i,j) = (2,3) = 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**
$T \rightarrow SB$
$U \rightarrow SA$
$A \rightarrow a$
$B \rightarrow b$

ba é gerado por BA

Tabela:

	a	b	a	a	b	b
a	A	S_0, S				
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 ser tamanho > 1, a variável que a gera a faz por meio de uma produção no formato $X \rightarrow YZ$

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

Grámática na FNC:

$S_0 \rightarrow \varepsilon$ | AT | BU | SS | AB | **BA**

$S \rightarrow$ AT | BU | SS | AB | **BA**

$T \rightarrow SB$

$U \rightarrow SA$

$A \rightarrow a$

$B \rightarrow b$

ba é gerado por BA

Tabela:

	a	b	a	a	b	b
a	A	S_0, S				
b		B	S_0, S			
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 ser tamanho > 1, a variável que a gera a faz por meio de uma produção no formato $X \rightarrow YZ$

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

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		
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) = (3,4) = aa

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		
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 $X \rightarrow YZ$

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

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	$\varnothing$		
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) = bb

Grámática na FNC:

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

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	$\varnothing$		
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) = bb

Grámática na FNC:

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

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 (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

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 $X \rightarrow YZ$

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

Exemplo:

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

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

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

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

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:

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

$S_0? \overset{*}{\Rightarrow} aba\ ?$
$S? \overset{*}{\Rightarrow} aba\ ?$

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:

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

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

$T \rightarrow SB$

$U \rightarrow SA$

$A \rightarrow a$

$B \rightarrow b$

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

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

Cadeia:

a b a a b b

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

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:

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

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

Cadeia:
a b a a b b

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

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:

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

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

$T \rightarrow SB$

$U \rightarrow SA$

$A \rightarrow a$

$B \rightarrow b$

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

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

Cadeia:

a b a a b b

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

Tabela:

	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!

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

Grámática na FNC:

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

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

$T \rightarrow SB$

$U \rightarrow SA$

$A \rightarrow a$

$B \rightarrow b$

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

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

Tabela:

	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:

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

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

$T \rightarrow SB$

$U \rightarrow SA$

$A \rightarrow a$

$B \rightarrow b$

Cadeia:

a b a a b b

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

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

Não, então fica só o U mesmo...

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

Tabela:

	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:

- $S_0 \rightarrow \varepsilon$ | AT | BU | SS | AB | BA
- S → AT | BU | SS | AB | BA
- T → SB
- U → SA
- A → a
- B → 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:

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

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

$T \rightarrow SB$

$U \rightarrow SA$

$A \rightarrow a$

$B \rightarrow b$

Cadeia:

a b a a b 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

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 $X \rightarrow YZ$
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:

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

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

$T \to SB$

$U \to SA$

$A \to a$

$B \to b$

Possibilidade 1

AU 

w1...w4 =

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

Cadeia:

a b a a b 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

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 $X \rightarrow YZ$
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:

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

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

$T \rightarrow SB$

$U \rightarrow SA$

$A \rightarrow a$

$B \rightarrow b$

Possibilidade 2

Nada gera aa

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

Cadeia:

a b a a b 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

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 $X \rightarrow YZ$
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:

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

Possibilidade 3

UA

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

Cadeia:
a b a a b 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

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 $S \rightarrow \varepsilon$ 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 $A \rightarrow \mathtt{b}$ 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 $A \rightarrow BC$:
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?

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

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

$T \rightarrow SB$

$U \rightarrow SA$

$A \rightarrow a$

$B \rightarrow b$

	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

Cadeia:

a b a a 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 $S \rightarrow \varepsilon$ 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 $A \rightarrow \mathtt{b}$ 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 $A \rightarrow BC$:
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*."

Exercícios

- Antes de continuarmos, é importante fixar os conceitos aprendidos. Para isso, façam PELO MENOS os seguintes exercícios do livro do Sipser (cap 2): 2.1, 2.3 e 2.4
- Obs: lembrem-se que o símbolo inicial de uma gramática é o símbolo do lado esquerdo da primeira produção da gramática