Tema 6

Linguagens livres de contexto

Professora:
Ariane Machado Lima

Vídeo 1

Gramáticas Livres de Contexto

Gramáticas Livres de Contexto

- Relembrar conceitos básicos (vídeo 1 do Tema 2)

Gramáticas

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

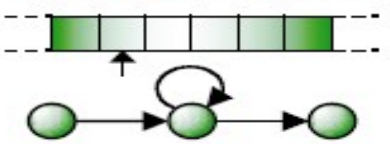
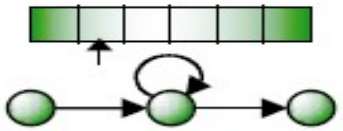
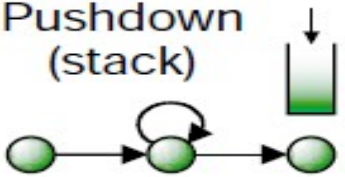
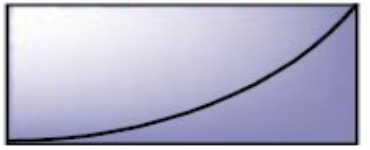
    $(\Sigma \cup V)^* V (\Sigma \cup V)^* \rightarrow (\Sigma \cup V)^*$

Gramáticas

- Definição: uma **gramática livre de contexto** G é uma quádrupla (V, Σ, S, P), na qual
  - V é o conjunto de símbolos não-terminais (ou variáveis)
  - Σ é o conjunto de símbolos terminais
  - S é o símbolo inicial
  - P é o conjunto de produções da forma

    V → (Σ U V)*

Linguagens, dispositivos, gramáticas e complexidades

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 

Linguagens livres de contexto

Dependências encaixadas/aninhadas

Linguagens livres de contexto

Dependências encaixadas/aninhadas

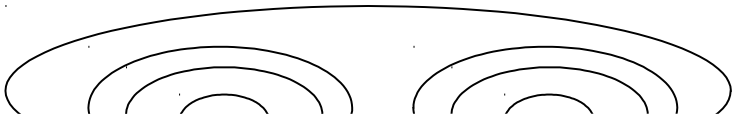
if if if else else else

Linguagens livres de contexto

Dependências encaixadas/aninhadas

Linguagens livres de contexto

Dependências encaixadas/aninhadas



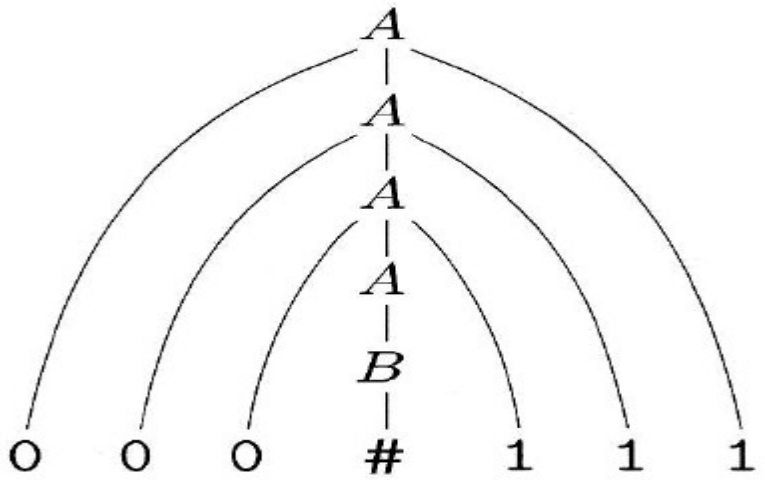
((a+a) + (a+a))

Árvore sintática ou árvore de derivação

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

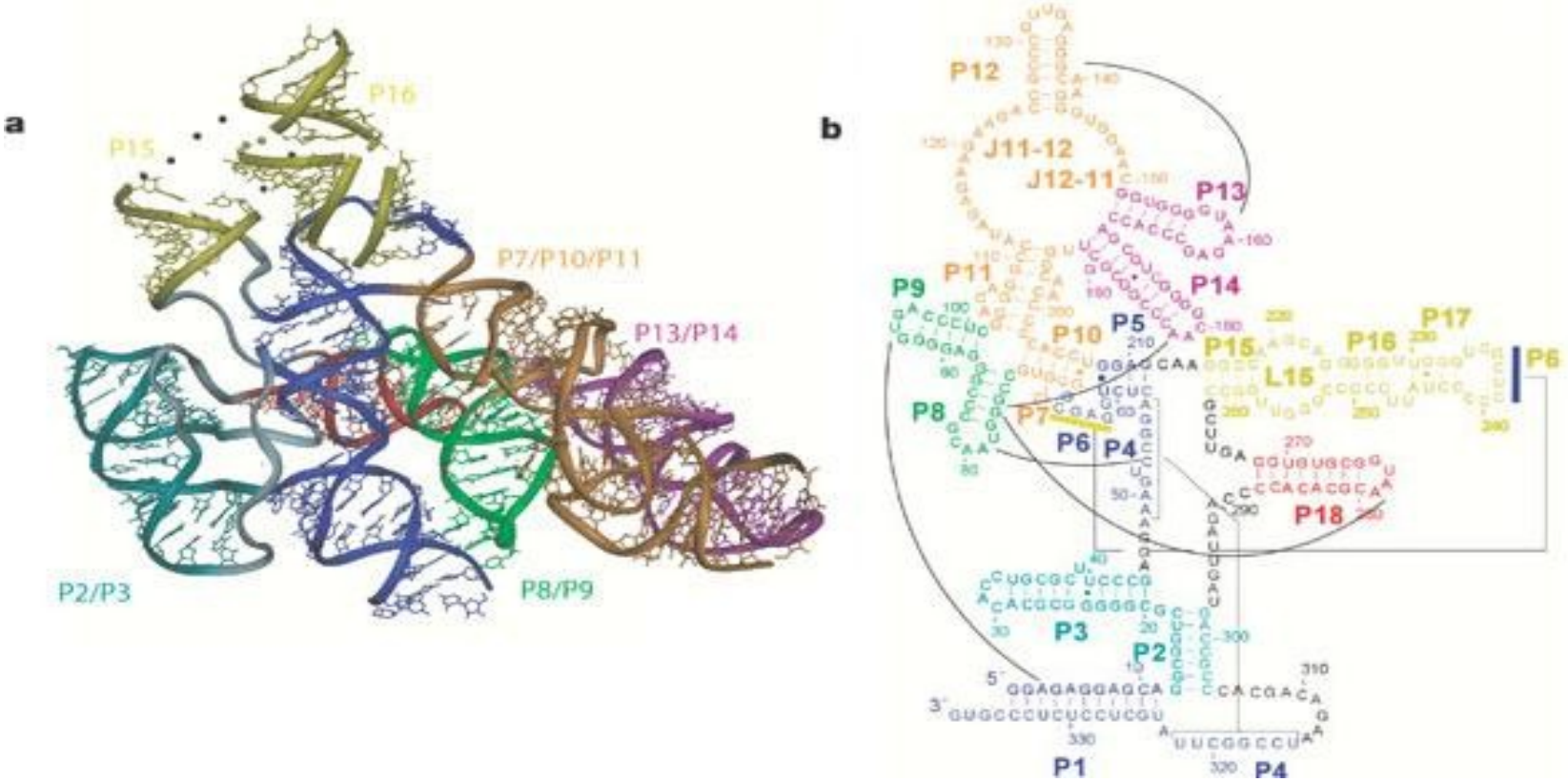
Por exemplo, a gramática G_1 gera a cadeia 000#111.

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



Árvore sintática
ou
Árvore de derivação
=>
Informação estrutural daquele padrão

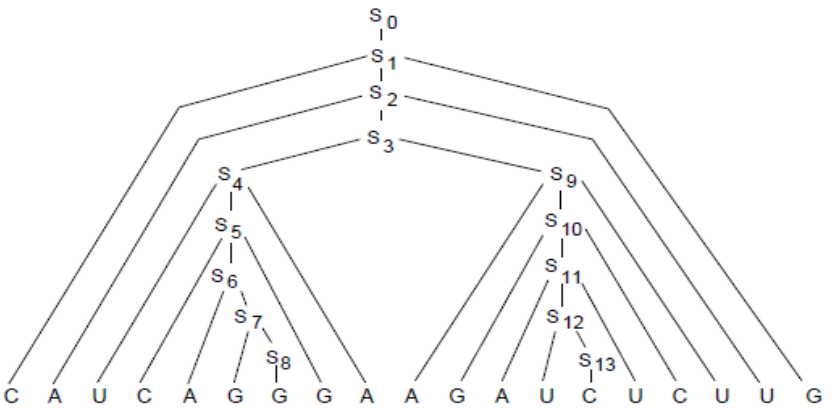
Exemplos
Estrutura secundária de RNAs



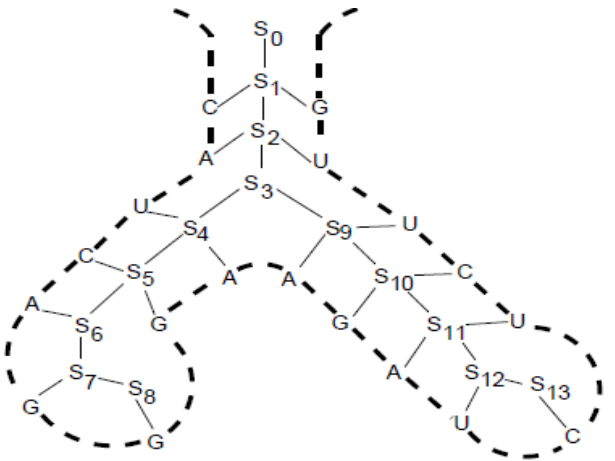
Gramática

$$
\begin{aligned}
P = \{ \quad & S_0 \rightarrow S_1, & & S_7 \rightarrow \mathtt{G}\ S_8, \\
& S_1 \rightarrow \mathtt{C}\ S_2\ \mathtt{G}, & & S_8 \rightarrow \mathtt{G}, \\
& S_1 \rightarrow \mathtt{A}\ S_2\ \mathtt{U}, & & S_8 \rightarrow \mathtt{U}, \\
& S_2 \rightarrow \mathtt{A}\ S_3\ \mathtt{U}, & & S_9 \rightarrow \mathtt{A}\ S_{10}\ \mathtt{U}, \\
& S_3 \rightarrow S_4\ S_9, & & S_{10} \rightarrow \mathtt{C}\ S_{10}\ \mathtt{G}, \\
& S_4 \rightarrow \mathtt{U}\ S_5\ \mathtt{A}, & & S_{10} \rightarrow \mathtt{G}\ S_{11}\ \mathtt{C}, \\
& S_5 \rightarrow \mathtt{C}\ S_6\ \mathtt{G}, & & S_{11} \rightarrow \mathtt{A}\ S_{12}\ \mathtt{U}, \\
& S_6 \rightarrow \mathtt{A}\ S_7, & & S_{12} \rightarrow \mathtt{U}\ S_{13}, \\
& S_7 \rightarrow \mathtt{U}\ S_7, & & S_{13} \rightarrow \mathtt{C} \qquad \}
\end{aligned}
$$

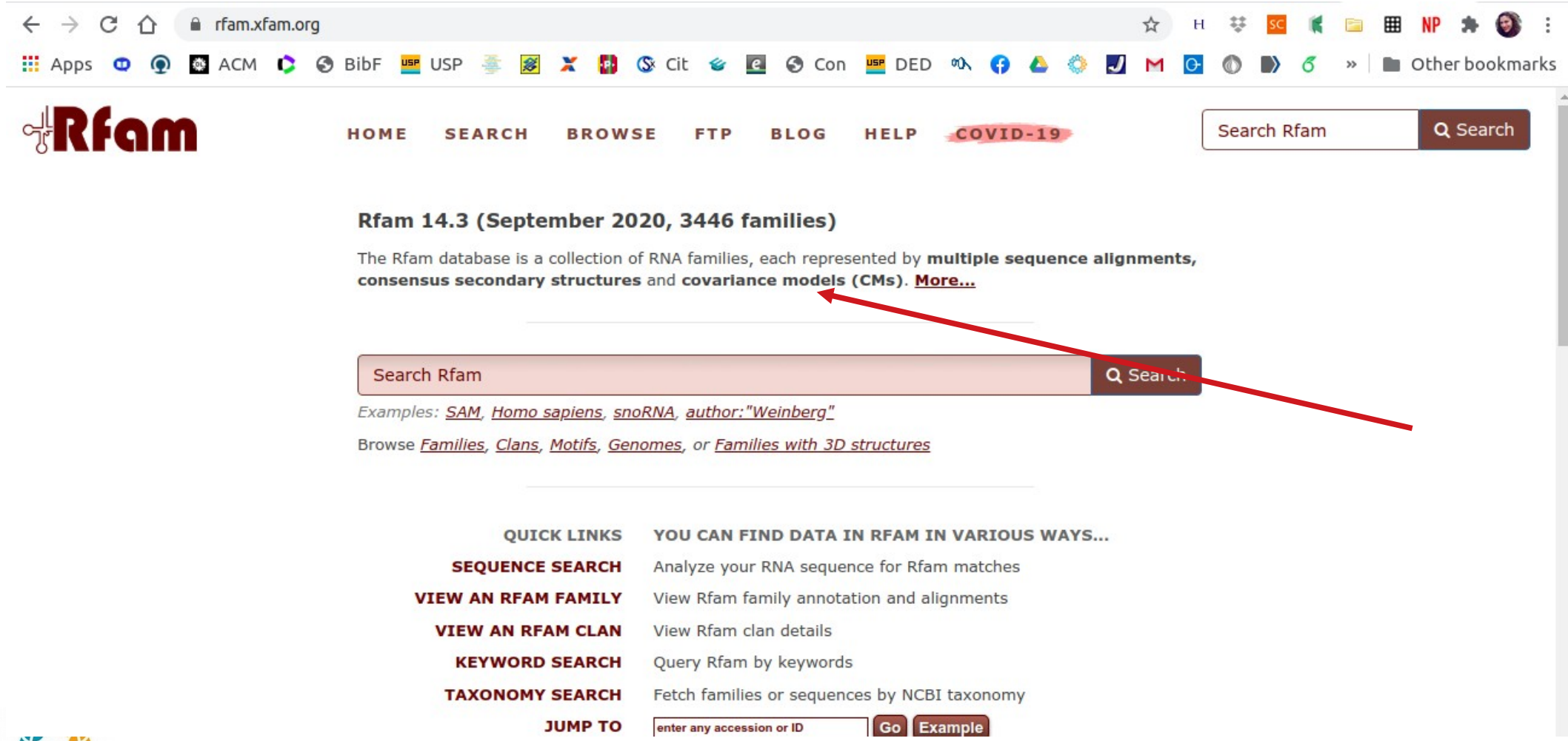
Árvore sintática



Estrutura secundária da molécula

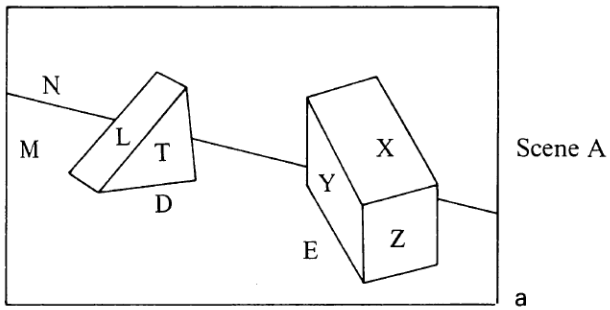


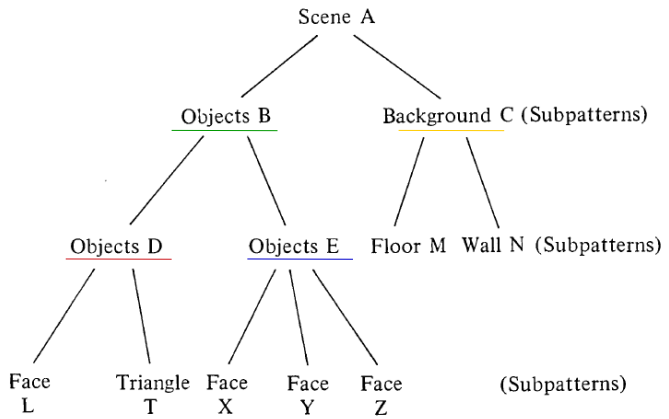
5112–5120 Nucleic Acids Research, 1994, Vol. 22, No. 23


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

Exemplos

- Sequências que representam imagens que seguem um padrão



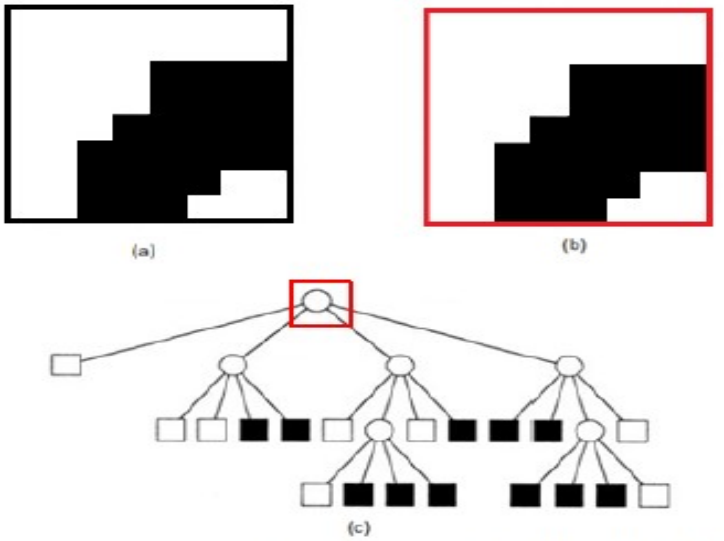


<Scene A> → <Objects B> <Background C>
<Objects B> → <Objects D> <Objects E>
<Objects D> → <Face L> <Triangle T>
<Objects E> → <Face X> <Face Y> <Face Z>
<Background C> → <Floor M> <Wall N>

((LT)(XYZ))(MN))

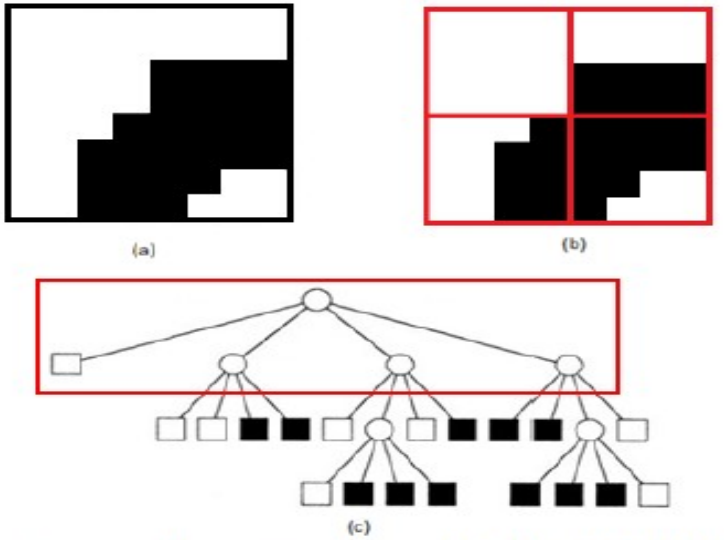
Imagens: representação por quadtrees

Descendo no nível de pixels



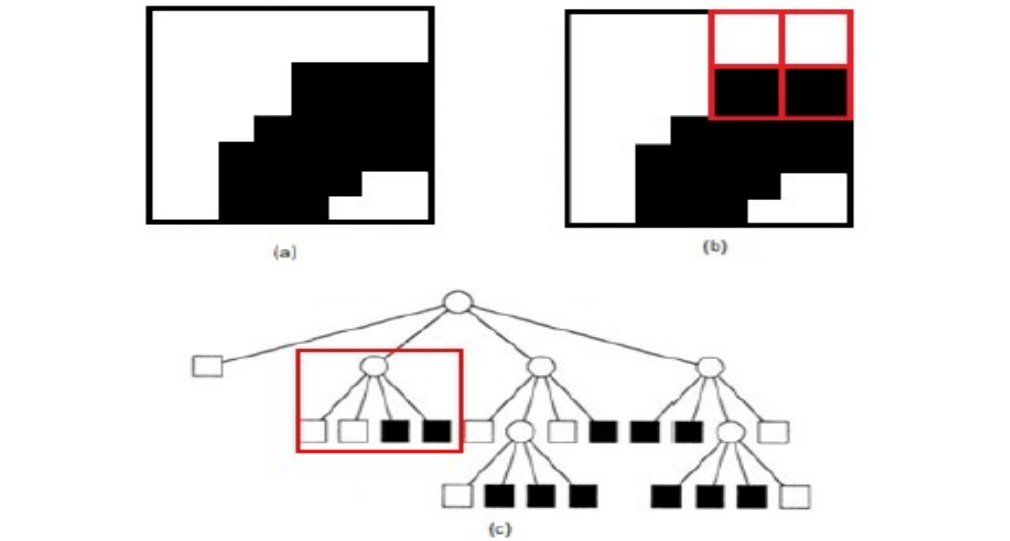
Estrutura de uma quadtree. Baseado em (AIZAWA; NAKAMURA, 1999)

Imagens: representação por quadtrees



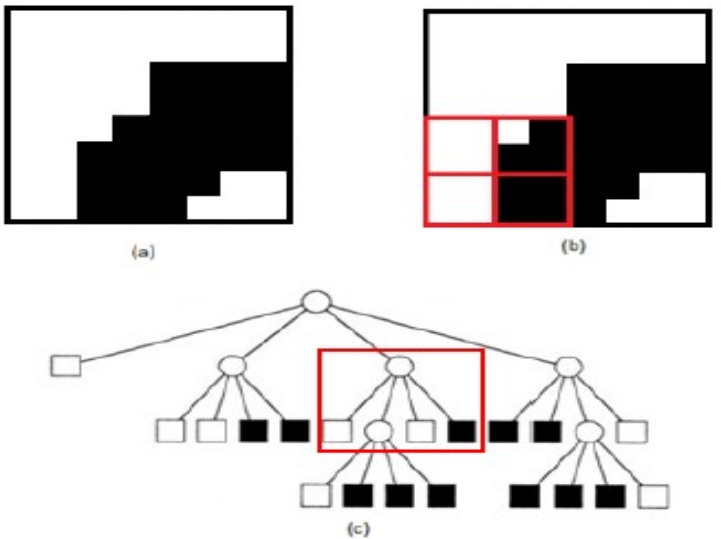
Estrutura de uma quadtree. Baseado em (AIZAWA; NAKAMURA, 1999)

Imagens: representação por quadtrees



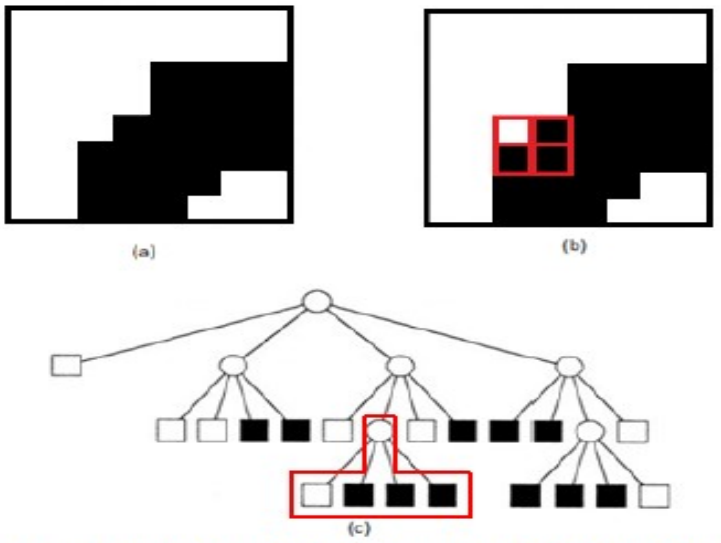
Estrutura de uma quadtree. Baseado em (AIZAWA; NAKAMURA, 1999)

Imagens: representação por quadtrees



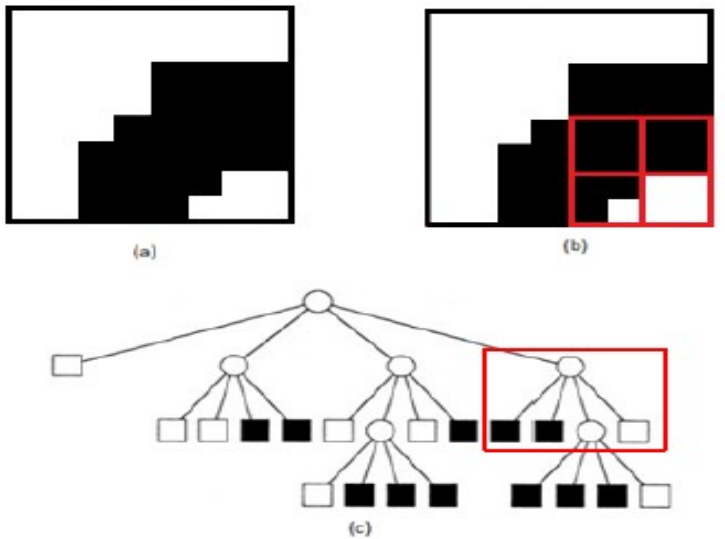
Estrutura de uma quadtree. Baseado em (AIZAWA; NAKAMURA, 1999)

Imagens: representação por quadtrees



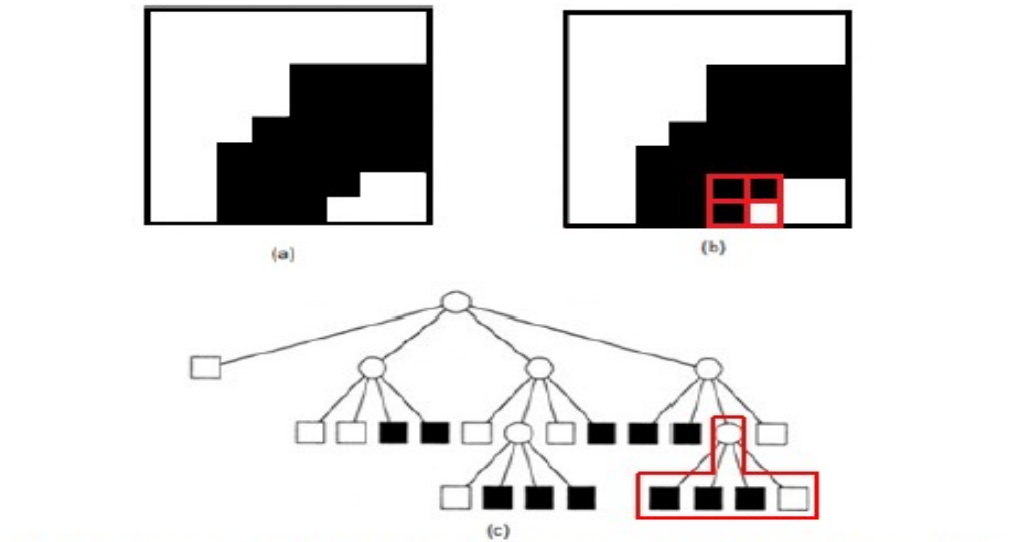
Estrutura de uma quadtree. Baseado em (AIZAWA; NAKAMURA, 1999)

Imagens: representação por quadtrees



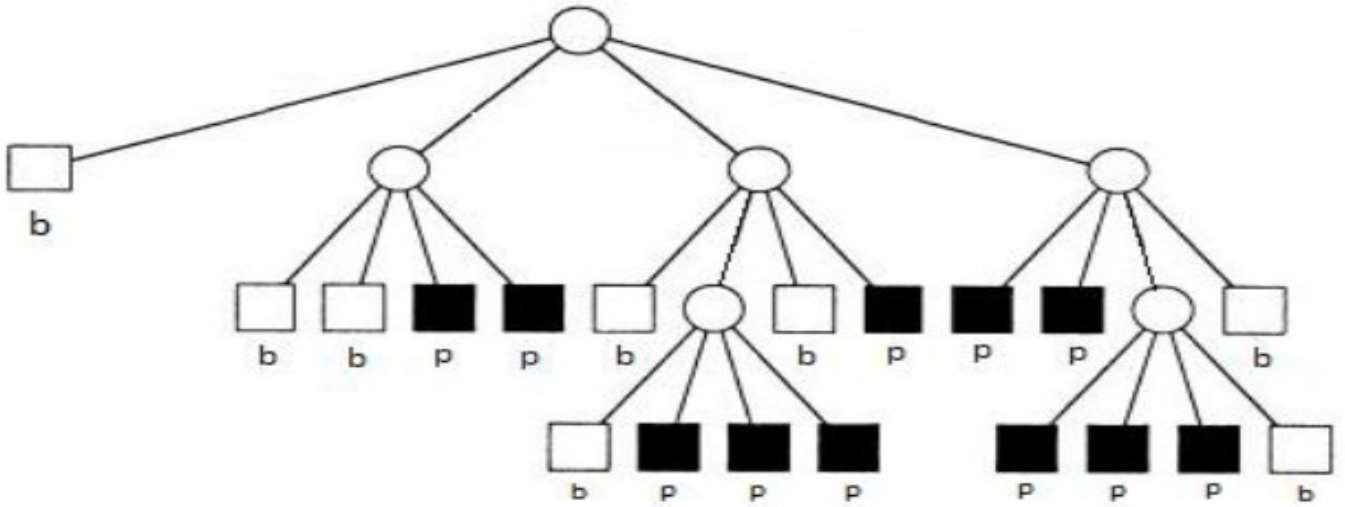
Estrutura de uma quadtree. Baseado em (AIZAWA; NAKAMURA, 1999)

Imagens: representação por quadtrees



Estrutura de uma quadtree. Baseado em (AIZAWA; NAKAMURA, 1999)

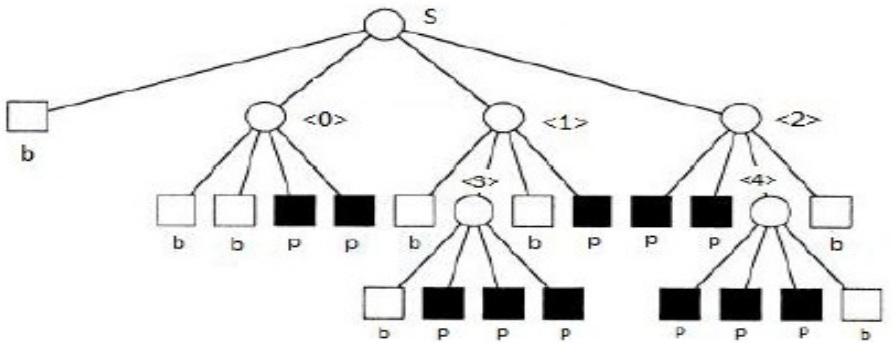
Imagens: representação por quadtrees



- String gerada: (b(bbpp)(b(bppp)bp)(pp(pppb)b));

Note que essa é uma string com **informação estrutural**

Imagens: representação por quadtrees



- Gramática gerada $G = (V_N, V_T, R, S)$
  - S representa o símbolo inicial da gramática;
  - V_N é o conjunto de símbolos não-terminais {S, <0>, <1>, ..., <4>};
  - V_T é o conjunto de símbolos terminais {b, p, (,)};
  - R é o conjunto de regras de produção:
    - S → (b <0> <1> <2>) [1.0]
    - <0> → (b b p p) [1.0]
    - <1> → (b <3> b p) [1.0]
    - <2> → (p p <4> b) [1.0]
    - <3> → (b p p p) [1.0]
    - <4> → (p p p b) [1.0]

Mais exemplos de gramáticas - Detecção de estenoses

Artificial Intelligence in Medicine 26 (2002) 145–159

www.elsevier.com/locate/artmed

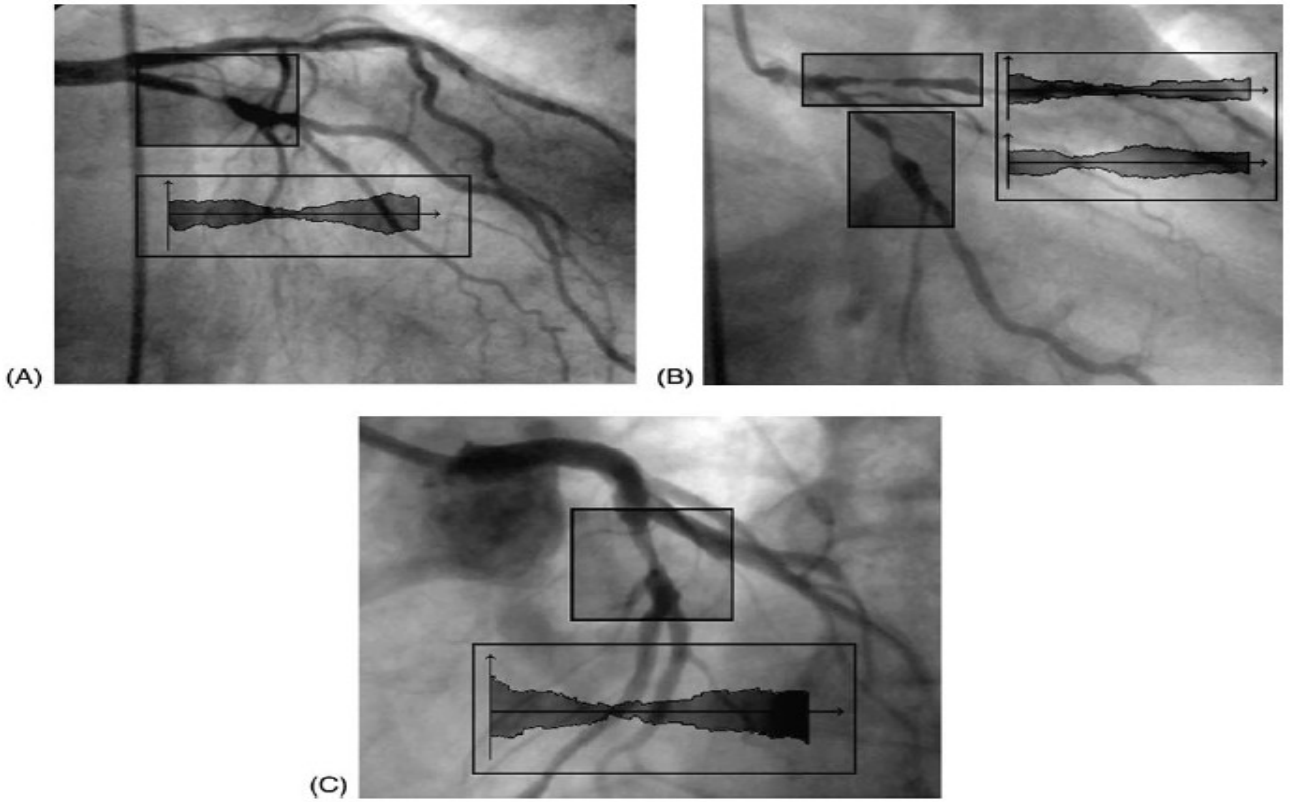
Syntactic reasoning and pattern recognition for analysis of coronary artery images

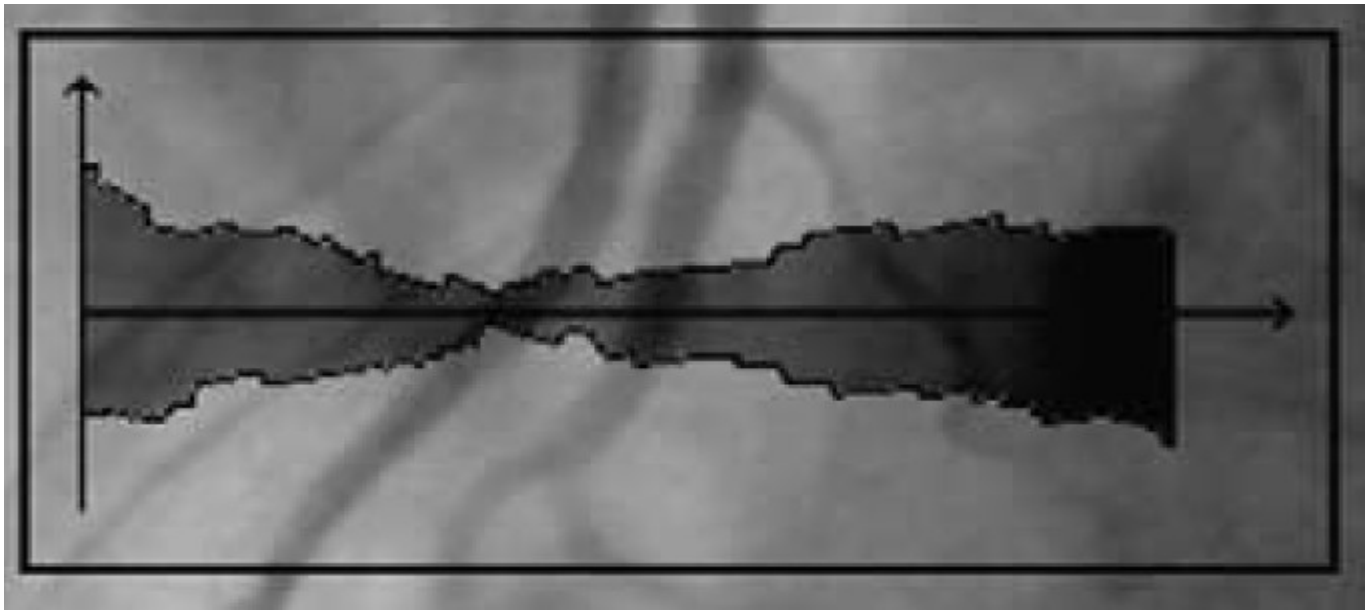
Marek R. Ogiela*, Ryszard Tadeusiewicz

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

Mais exemplos de gramáticas - Detecção de estenoses





$$V_T = \{h, v, nv\} \text{ for } h \in (-10°, 10°), v \in (11°, 90°), nv \in (-11°, -90°)$$

Mais exemplos de gramáticas - Detecção de estenoses

$$G_{CA} = (V_N, V_T, SP, STP)$$

where V_N is the set of non-terminal symbols, V_T the set of terminal symbols, SP the production set, STS the grammar start symbol.

$$V_N = \{\text{SYMPTOM}, \text{STENOSIS}, H, V, \text{NV}\}$$

$$V_T = \{h, v, nv\} \text{ for } h \in (-10°, 10°), v \in (11°, 90°), nv \in (-11°, -90°)$$

STS=SYMPTOM

SP:

1. SYMPTOM → STENOSIS
2. STENOSIS → NV HV
3. STENOSIS → NV *V*|NV *H*
4. *V* → *v*|*V v*
5. NV → nv|NV nv
6. *H* → *h*|*H h*

Revisão de gramáticas em imagens

Using Grammars for Pattern Recognition in Images: A Systematic Review

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

ACM Computing Surveys, Vol. 46, No. 2, Article 26, Publication date: November 2013.

Exemplos

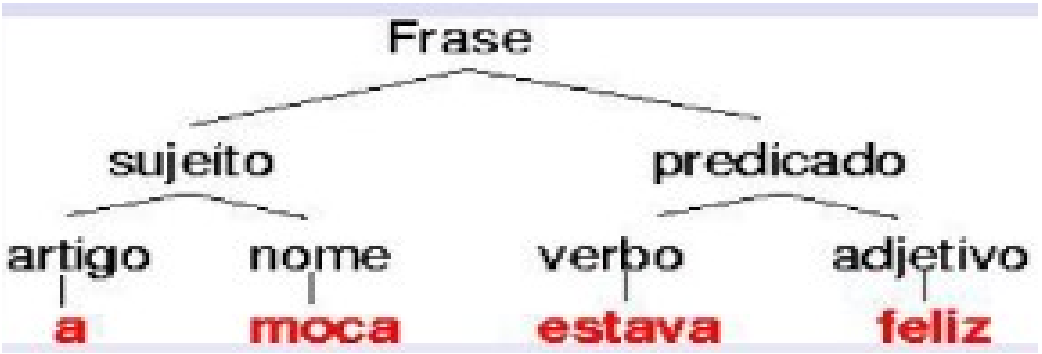
Processamento de texto (ex: XML)

```
(1)  <?xml version="1.0" encoding="UTF-8"?>
(2)  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
(3)  <xs:element name="personal">
(4)      <xs:annotation>
(5)          <xs:documentation>Ein Einfaches Beispiel für ein XML-Schema</xs:documentation>
(6)      </xs:annotation>
(7)      <xs:complexType>
(8)          <xs:sequence minOccurs="0" maxOccurs="unbounded">
(9)              <xs:element name="person">
(10)                 <xs:complexType>
(11)                     <xs:sequence>
(12)                         <xs:element="name" type="xs:string"/>
(13)                         <xs:element="vorname" type="xs:string"/>
(14)                         <xs:element="tel" type="xs:string"/>
(15)                     </xs:sequence>
(16)                 </xs:complexType>
(17)             </xs:element>
(18)         </xs:sequence>
(19)     </xs:complexType>
(20) </xs:element>
(21) </xs:schema>
```

https://www.eworks.de/uploads/pics/2008-05-29_XML_Beispiel2_Schema.GIF

Exemplos

Processamento de linguagem natural



Gramática

- Reconhecimento
- Geração
- Árvore sintática

Fim do vídeo 1

Gramáticas Livres de Contexto

Vídeo 2

Ambiguidade

Análise sintática e ambiguidade

Derivações

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

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

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)`

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

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

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

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

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

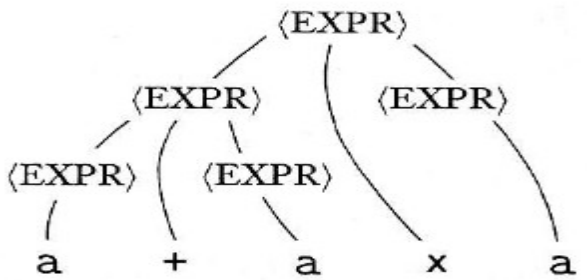
Obs: passo necessário para a compilação de um programa em uma dada linguagem de programação

Ex: gramática para expressões aritméticas simples

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

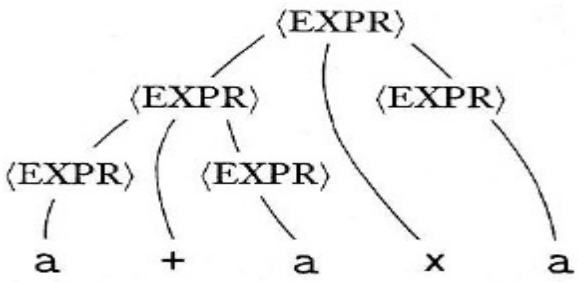
Ex: gramática para expressões aritméticas simples

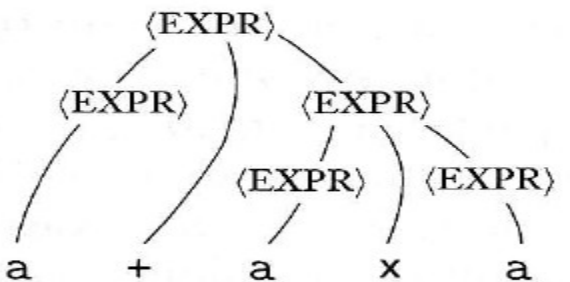
$$\langle \text{EXPR} \rangle \rightarrow \langle \text{EXPR} \rangle + \langle \text{EXPR} \rangle \mid \langle \text{EXPR} \rangle \times \langle \text{EXPR} \rangle \mid (\langle \text{EXPR} \rangle) \mid \text{a}$$



Ex: gramática para expressões aritméticas simples

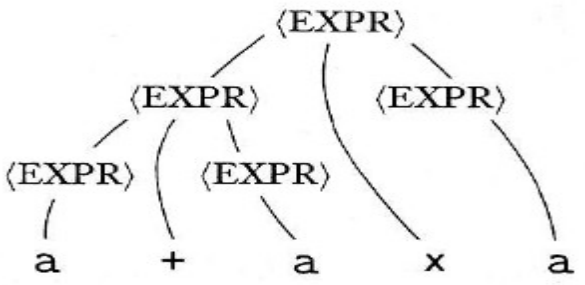
⟨EXPR⟩ → ⟨EXPR⟩+⟨EXPR⟩ | ⟨EXPR⟩x⟨EXPR⟩ | (⟨EXPR⟩) | a

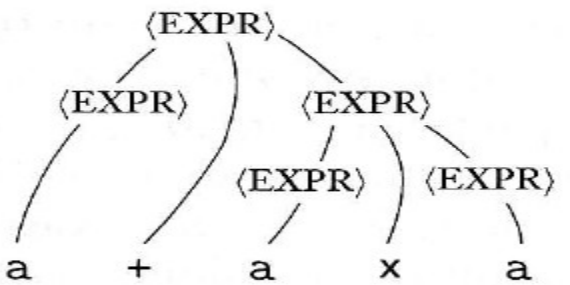




Ex: gramática para expressões aritméticas simples

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

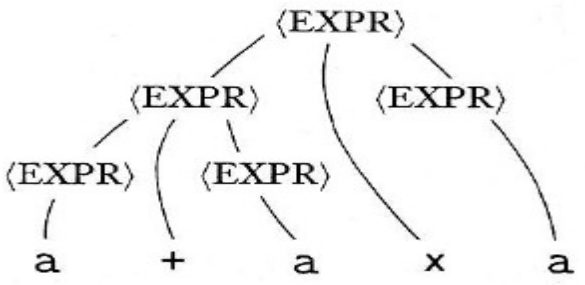


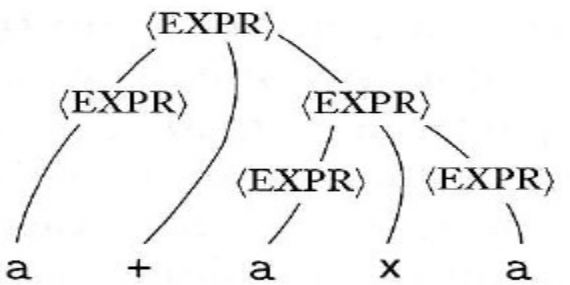


Duas árvores sintáticas distintas para a mesma cadeia!!!!

Ex: gramática para expressões aritméticas simples

⟨EXPR⟩ → ⟨EXPR⟩+⟨EXPR⟩ | ⟨EXPR⟩x⟨EXPR⟩ | (⟨EXPR⟩) | 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 $\{\texttt{a}, \texttt{+}, \texttt{x}, \texttt{(}, \texttt{)}\}$. As regras são

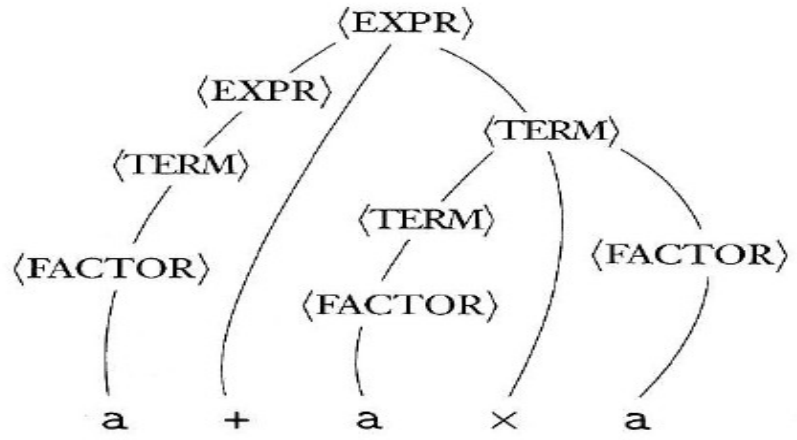
$$\begin{aligned} \langle \text{EXPR} \rangle &\rightarrow \langle \text{EXPR} \rangle \texttt{+} \langle \text{TERM} \rangle \mid \langle \text{TERM} \rangle \\ \langle \text{TERM} \rangle &\rightarrow \langle \text{TERM} \rangle \texttt{x} \langle \text{FACTOR} \rangle \mid \langle \text{FACTOR} \rangle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

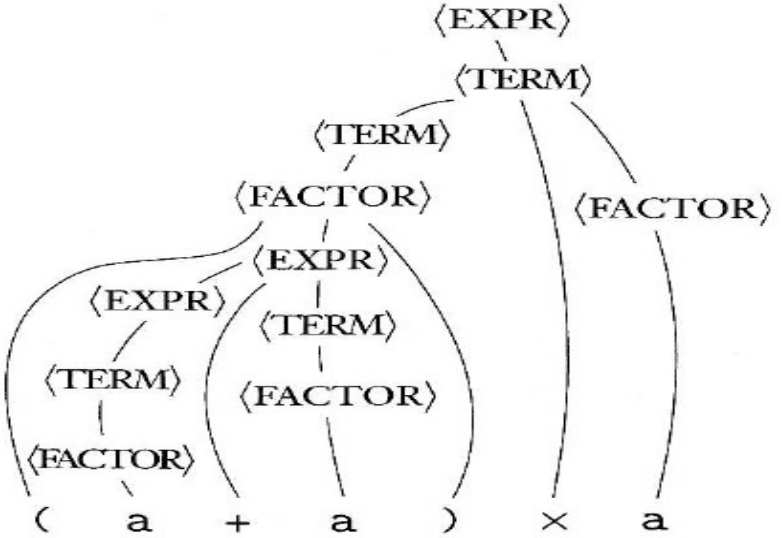
Expressões aritméticas sem ambiguidade

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

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





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 2

Análise sintática e ambiguidade

Vídeos 3 e 4

Mais sobre análise sintática e Forma Normal de Chomsky

(algoritmo CYK)

Análise sintática

Problema: Dada uma gramática G e uma cadeia w, saber se w ϵ 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 → ...)

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)`

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

Derivação mais à esquerda

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)`

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

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

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>

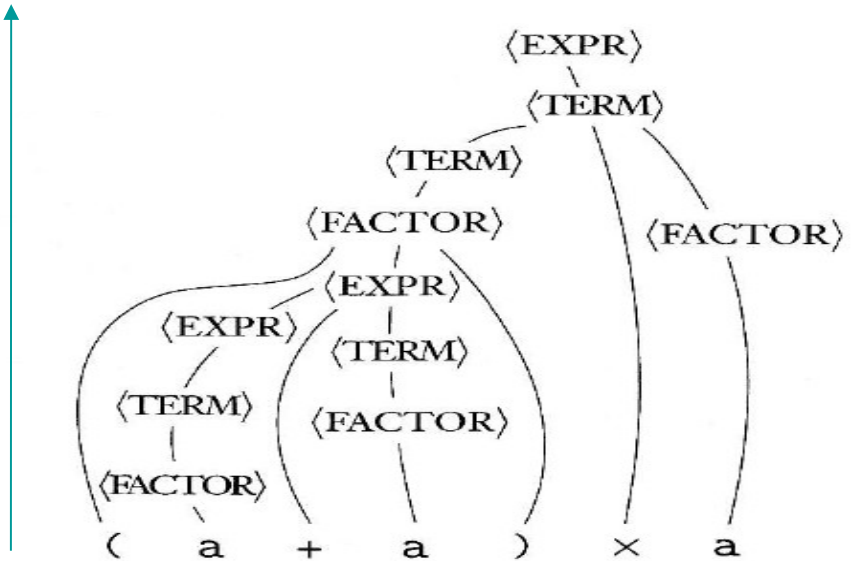
| { <lista_com> }

Fácil de tratar

Análise sintática ascendente

- Bottom-up

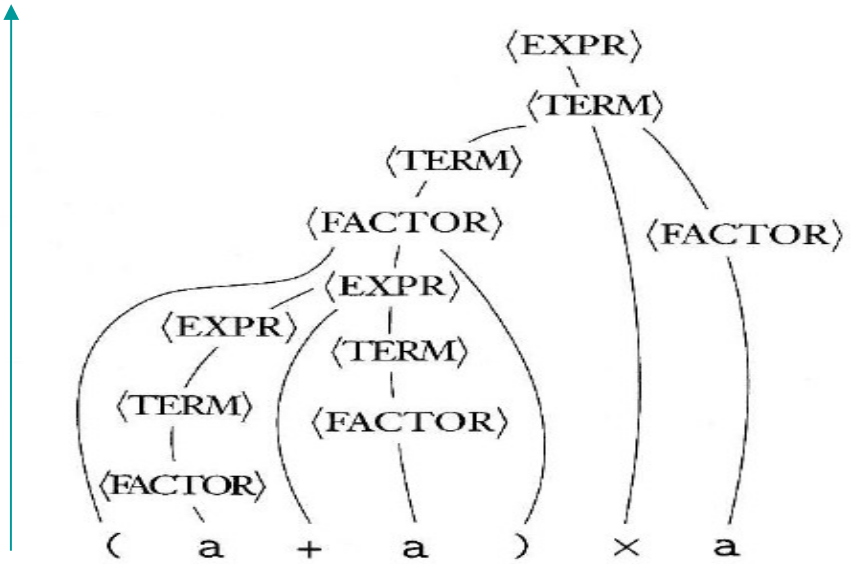
$$\langle \text{EXPR} \rangle \rightarrow \langle \text{EXPR} \rangle + \langle \text{TERM} \rangle \mid \langle \text{TERM} \rangle$$
$$\langle \text{TERM} \rangle \rightarrow \langle \text{TERM} \rangle \times \langle \text{FACTOR} \rangle \mid \langle \text{FACTOR} \rangle$$
$$\langle \text{FACTOR}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



Análise sintática ascendente

- Bottom-up
- 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?
- 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, 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:

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

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

$T \rightarrow SB$

$U \rightarrow SA$

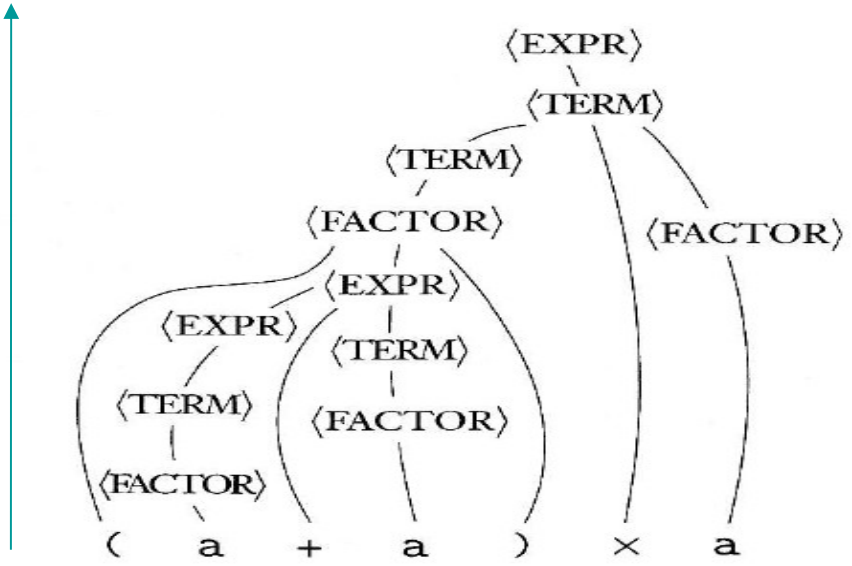
$A \rightarrow a$

$B \rightarrow b$

Algoritmo CYK para análise sintática

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

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



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 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)

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

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

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 $S \to \varepsilon$ 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 $S \to \varepsilon$ 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

Exemplo:

Gramática na FNC:

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

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

$T \rightarrow SB$

$U \rightarrow SA$

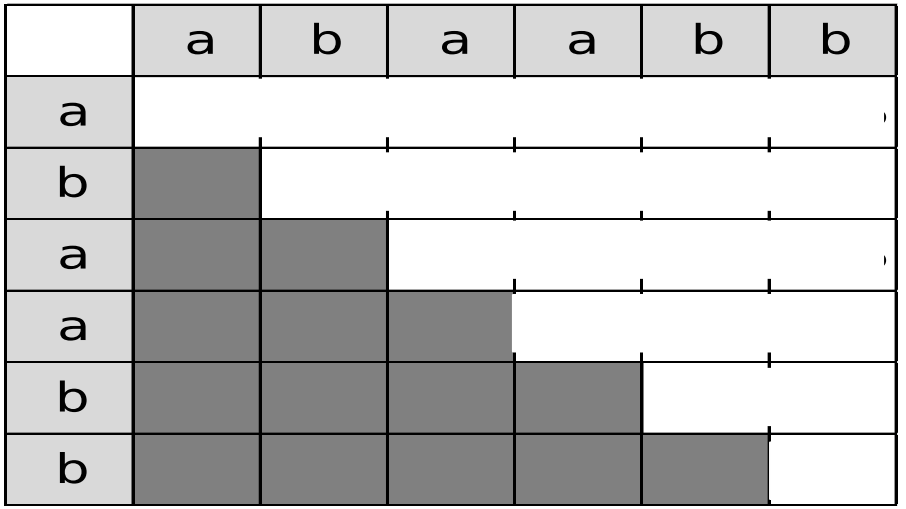
$A \rightarrow a$

$B \rightarrow b$

Cadeia:

a b a a b b

Tabela:



Algoritmo CYK para análise sintática

Exemplo:

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 $S \rightarrow \varepsilon$ 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 $table(i, i)$.

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 → YZ

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:

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

Gramática na FNC:

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

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

$T \rightarrow SB$

$U \rightarrow SA$

$A \rightarrow a$

$B \rightarrow b$

ab é gerado por AB

Cadeia:

a b a a b b

(i,j) = (1,2)

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

Gramática 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

Cadeia:

a b a a b b

(i,j) = (1,2)

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

Gramática 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

Cadeia:

a b a a b b

(i,j) = (1,2)

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 $X \rightarrow YZ$

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:

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

Gramática 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) = 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 $X \rightarrow YZ$

Gramática 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

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

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

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:

(i,j) = (3,4) = 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 $X \rightarrow YZ$

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:

(i,j) = (3,4) = 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 $X \rightarrow YZ$

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:

(i,j) = (4,5) = 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 $X \rightarrow YZ$

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:

(i,j) = (5,6) = 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 2
Por ser tamanho > 1, a variável que a gera a faz por meio de uma produção no formato $X \rightarrow YZ$

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:

(i,j) = (5,6) = 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!

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	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!

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:

(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!

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:

(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!

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

$S_0 ? \stackrel{*}{\Rightarrow} aba ?$

$S ? \stackrel{*}{\Rightarrow} aba ?$

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!

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

$S_0 A \stackrel{*}{\Rightarrow} aba$?

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

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!

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

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

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

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!

Gramática na FNC:

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

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

$T \rightarrow SB$

$U \rightarrow SA$

$A \rightarrow a$

$B \rightarrow b$

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

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

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

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!

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

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

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

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

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!

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

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

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

Não, então fica só o U mesmo...

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

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!

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:

(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!

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	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!

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

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!

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

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!

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

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!

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

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....

terceiro! Tenho que testar as TRÊS possibilidades!

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	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?

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

	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...

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

	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*."

Fim do vídeo 4

Algoritmo CYK

Para quem quiser exercícios...

- Exercícios do livro do Sipser (cap 2): 2.1, 2.3, 2.4, 2.6, 2.9
- Obs: lembrem-se que o símbolo inicial de uma gramática é o símbolo do lado esquerdo da primeira produção da gramática

Forma Normal de Chomsky

- Lembrando que, para o uso do algoritmo CYK, a gramática precisa estar na forma normal de Chomsky
- A boa notícia é que há um teorema que diz que toda GLC pode ser convertida na Forma Normal de Chomsky (prova no fim do Cap 2.1 do livro de Sipser)
- E quem ainda quiser fazer um exercício sobre isso, faça o 2.14 do Sipser.

Vídeo 5

Algoritmo de Earley

Analisador sintático CYK

- Uso universal (algoritmo independe da gramática)
- Estratégia ascendente
- Complexidade $O(n^3)$ (SEMPRE)
- Exige gramática na forma normal de Chomsky:

$$A \rightarrow BC, A \in N, B \in N, C \in N$$

$$A \rightarrow a, A \in N, a \in \Sigma$$

Analisador sintático de Earley

- Uso universal (algoritmo independe da gramática)
- Estratégia descendente, mas preenche uma tabela de forma ascendente
- Gramática não precisa estar em uma forma específica
- Identifica diferentes árvores de derivação
- Complexidade $O(n^3)$ para gramáticas ambíguas, mas $O(n^2)$ ou $O(n)$ para certas gramáticas

Analisador sintático de Earley

- Uso de um símbolo especial indicando fim de cadeia (ex: "|-" → cadeia a+a*a|-)
- Para uma cadeia de entrada de tamanho n (n+1 contando com o símbolo |-), uma tabela de n+2 colunas será criada, cada uma correspondendo a um ponto da análise da cadeia

$E \rightarrow T \mid E+T$
$T \rightarrow P \mid T*P$
$P \rightarrow a$

input string = a+a*a⊣

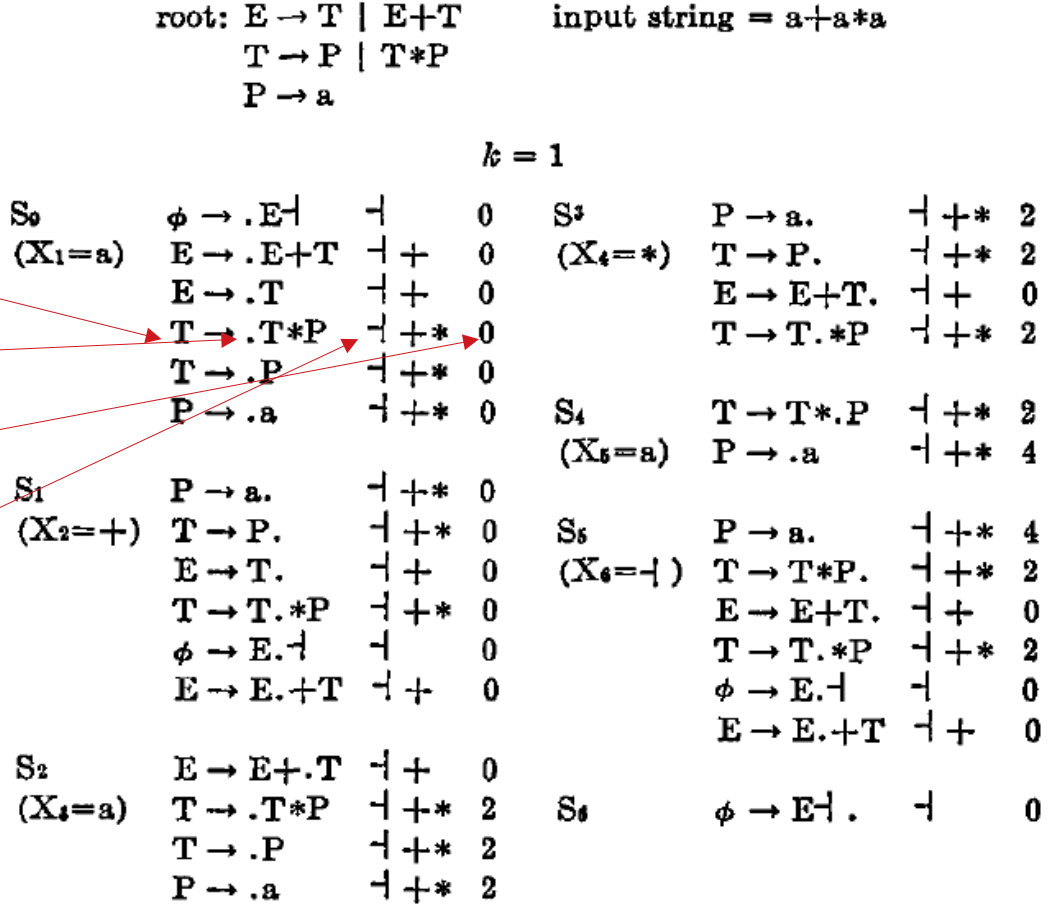
$k = 1$

S_0 (a)	S_1 (+)	S_2 (a)	S_3 (*)	S_4 (a)	S_5 (|-)	S_6
...	...	...	...	...	...	...

Analisador sintático de Earley

- Cada coluna terá vários estados da análise, cada um correspondendo a:

- Uma produção da gramática
- A posição desta produção em que se encontra a análise
- A coluna que originou aquela produção
- Cadeia (de tamanho k) de símbolos look-ahead (terminais que sucedem aquela produção)



Exemplo:

$E \rightarrow T \mid E+T$
$T \rightarrow P \mid T*P$
$P \rightarrow a$

input string = a+a*a

$k = 1$

Inicialização

$E \rightarrow T \mid E+T$
$T \rightarrow P \mid T*P$
$P \rightarrow a$

input string = a+a*a⊣

$k = 1$

S_0 (a)	S_1 (+)	S_2 (a)	S_3 (*)	S_4 (a)	S_5 (⊣-)	S_6
...	...	...	...	...	...	...

Inicialização

$E \rightarrow T \mid E+T$
$T \rightarrow P \mid T*P$
$P \rightarrow a$

input string = a+a*a⊣

$k = 1$

S_0 (a)	S_1 (+)	S_2 (a)	S_3 (*)	S_4 (a)	S_5 (|-)	S_6
$\phi \rightarrow .E\dashv$ ⊣ 0	...	...	...	...	...	...

$\phi \rightarrow .E\dashv$ ⊣ 0

produção lookahead Coluna de origem

Ponto de análise da produção

Para cada coluna j = 0 até n+1
  Para cada estado (linha) s
    Predictor, completer ou scanner

root: E → T | E+T input string = a+a*a
T → P | T*P
P → a

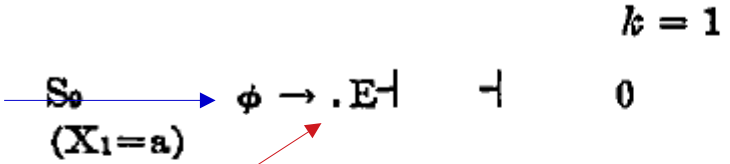
$k = 1$

S_0 $(X_1=a)$	$\phi \rightarrow .E\dashv$	$\dashv$	0

Para cada coluna j = 0 até n+1
  Para cada estado (linha) s
    Predictor, completer ou scanner

root: E → T | E+T
T → P | T*P
P → a

input string = a+a*a



Predictor:
Depois do ponto vem um não terminal, então preciso “abri-lo”, trazendo suas produções para essa mesma coluna

Para cada coluna j = 0 até n+1
  Para cada estado (linha) s
    Predictor, completer ou scanner

root: E → T | E+T input string = a+a*a
T → P | T*P
P → a

$k = 1$

S_0 →	$\phi \rightarrow .E$	⊣	0
$(X_1 = a)$	$E \rightarrow .E+T$	⊣	0
	$E \rightarrow .T$	⊣	0

Predictor:
Depois do ponto vem um não terminal, então preciso “abri-lo”, trazendo suas produções para essa mesma coluna

Para cada coluna j = 0 até n+1
  Para cada estado (linha) s
    Predictor, completer ou scanner

root: E → T | E+T input string = a+a*a
T → P | T*P
P → a

$k = 1$

S_0	$\phi \rightarrow .E$	⊣	0
$(X_1 = a)$ →	$E \rightarrow .E+T$	⊣	0
	$E \rightarrow .T$	⊣	0

Predictor:
Depois do ponto vem um não terminal, então preciso “abri-lo”, trazendo suas produções para essa mesma coluna

Para cada coluna j = 0 até n+1
  Para cada estado (linha) s
    Predictor, completer ou scanner

root: E → T | E+T input string = a+a*a
T → P | T*P
P → a

$k = 1$

S_0	$\phi \rightarrow .E$	⊣	0
$(X_1=a)$	$E \rightarrow .E+T$	⊣ +	0
→	$E \rightarrow .T$	⊣ +	0

Predictor:
Depois do ponto vem um não terminal, então preciso “abri-lo”, trazendo suas produções para essa mesma coluna

Para cada coluna j = 0 até n+1
  Para cada estado (linha) s
    Predictor, completer ou scanner

root: E → T | E+T input string = a+a*a
T → P | T*P
P → a

$k = 1$

S_0	$\phi \rightarrow .E$	⊣	0
$(X_1=a)$	$E \rightarrow .E+T$	⊣ +	0
→	$E \rightarrow .T$	⊣ +	0
	$T \rightarrow .T*P$	⊣ +	0
	$T \rightarrow .P$	⊣ +	0

Predictor:
Depois do ponto vem um não terminal, então preciso “abri-lo”, trazendo suas produções para essa mesma coluna

Para cada coluna j = 0 até n+1
  Para cada estado (linha) s
    Predictor, completer ou scanner

root: E → T | E+T input string = a+a*a
T → P | T*P
P → a

$k = 1$

S_0	$\phi \rightarrow .E$	⊣	0
$(X_1=a)$	$E \rightarrow .E+T$	⊣ +	0
	$E \rightarrow .T$	⊣ +	0
	$T \rightarrow .T*P$	⊣ +	0
	$T \rightarrow .P$	⊣ +	0

Predictor:
Depois do ponto vem um não terminal, então preciso “abri-lo”, trazendo suas produções para essa mesma coluna

Para cada coluna j = 0 até n+1
  Para cada estado (linha) s
    Predictor, completer ou scanner

root: E → T | E+T input string = a+a*a
T → P | T*P
P → a

$k = 1$

S_0	$\phi \rightarrow .E$	⊣	0
$(X_1=a)$	$E \rightarrow .E+T$	⊣ +	0
	$E \rightarrow .T$	⊣ +	0
	$T \rightarrow .T*P$	⊣ +*	0
	$T \rightarrow .P$	⊣ +*	0

Predictor:
Depois do ponto vem um não terminal, então preciso “abri-lo”, trazendo suas produções para essa mesma coluna

Para cada coluna j = 0 até n+1
  Para cada estado (linha) s
    Predictor, completer ou scanner

root: E → T | E+T input string = a+a*a
T → P | T*P
P → a

$k = 1$

S_0	$\phi \rightarrow .E$	⊣	0
$(X_1=a)$	$E \rightarrow .E+T$	⊣ +	0
	$E \rightarrow .T$	⊣ +	0
	$T \rightarrow .T*P$	⊣ +*	0
→	$T \rightarrow .P$	⊣ +*	0

Predictor:
Depois do ponto vem um não terminal, então preciso “abri-lo”, trazendo suas produções para essa mesma coluna

Para cada coluna j = 0 até n+1
  Para cada estado (linha) s
    Predictor, completer ou scanner

root: E → T | E+T input string = a+a*a
T → P | T*P
P → a

$k = 1$

S_0	$\phi \rightarrow .E\dashv$	$\dashv$	0
$(X_1=a)$	$E \rightarrow .E+T$	$\dashv +$	0
	$E \rightarrow .T$	$\dashv +$	0
	$T \rightarrow .T*P$	$\dashv +*$	0
	$T \rightarrow .P$	$\dashv +*$	0
	$P \rightarrow .a$	$\dashv +*$	0

Predictor:
Depois do ponto vem um não terminal, então preciso “abri-lo”, trazendo suas produções para essa mesma coluna

Para cada coluna j = 0 até n+1
  Para cada estado (linha) s
    Predictor, completer ou scanner

root: E → T | E+T input string = a+a*a
T → P | T*P
P → a

$k = 1$

S_0	$\phi \rightarrow .E\dashv$	$\dashv$	0
$(X_1=a)$	$E \rightarrow .E+T$	$\dashv +$	0
	$E \rightarrow .T$	$\dashv +$	0
	$T \rightarrow .T*P$	$\dashv +*$	0
	$T \rightarrow .P$	$\dashv +*$	0
→	$P \rightarrow .a$	$\dashv +*$	0

Para cada coluna j = 0 até n+1
  Para cada estado (linha) s
    Predictor, completer ou scanner

root: E → T | E+T input string = a+a*a
T → P | T*P
P → a

$k = 1$

S_0	$\phi \rightarrow .E\dashv$	$\dashv$	0
$(X_1=a)$	$E \rightarrow .E+T$	$\dashv +$	0
	$E \rightarrow .T$	$\dashv +$	0
	$T \rightarrow .T*P$	$\dashv +*$	0
	$T \rightarrow .P$	$\dashv +*$	0
→	$P \rightarrow .a$	$\dashv +*$	0

Scanner:
Depois do ponto vem um terminal. Se ele bater com o símbolo atual da cadeia de entrada então posso evoluir na análise:
- avanço o ponto da produção e a levo para a próxima coluna

Para cada coluna j = 0 até n+1
  Para cada estado (linha) s
    Predictor, completer ou scanner

root: E → T | E+T input string = a+a*a
T → P | T*P
P → a

$k = 1$

S_0	$\phi \rightarrow .E$	⊣	0
(X_1=a)	$E \rightarrow .E+T$	⊣ +	0
	$E \rightarrow .T$	⊣ +	0
	$T \rightarrow .T*P$	⊣ +*	0
	$T \rightarrow .P$	⊣ +*	0
	$P \rightarrow .a$	⊣ +*	0
S_1	$P \rightarrow a.$	⊣ +*	0
(X_2=+)			

Scanner:
Depois do ponto vem um terminal. Se ele bater com o símbolo atual da cadeia de entrada então posso evoluir na análise:
- avanço o ponto da produção e a levo para a próxima coluna

Para cada coluna j = 0 até n+1
  Para cada estado (linha) s
    Predictor, completer ou scanner

root: E → T | E+T input string = a+a*a
T → P | T*P
P → a

$k = 1$

S_0	$\phi \rightarrow .E$	⊣	0
(X_1=a)	$E \rightarrow .E+T$	⊣ +	0
	$E \rightarrow .T$	⊣ +	0
	$T \rightarrow .T*P$	⊣ +*	0
	$T \rightarrow .P$	⊣ +*	0
	$P \rightarrow .a$	⊣ +*	0
S_1	$P \rightarrow a.$	⊣ +*	0
(X_2=+)			

Para cada coluna j = 0 até n+1
  Para cada estado (linha) s
    Predictor, completer ou scanner

root: E → T | E+T input string = a+a*a
T → P | T*P
P → a

$k = 1$

S_0	$\phi \rightarrow .E$	⊣	0
(X_1=a)	$E \rightarrow .E+T$	⊣ +	0
	$E \rightarrow .T$	⊣ +	0
	$T \rightarrow .T*P$	⊣ +*	0
	$T \rightarrow .P$	⊣ +*	0
	$P \rightarrow .a$	⊣ +*	0
S_1	$P \rightarrow a.$	⊣ +*	0
(X_2=+)			

Completer:
O ponto está no final da produção. Então posso evoluir na análise do não terminal reconhecido (lado esquerdo da produção) :
- trago para essa coluna as produções (daquele estado de origem em que aquele não-terminal está logo depois do ponto)

Para cada coluna j = 0 até n+1
  Para cada estado (linha) s
    Predictor, completer ou scanner

root: E → T | E+T input string = a+a*a
T → P | T*P
P → a

$k = 1$

S_0	$\phi \rightarrow .E$	⊣	0
(X_1=a)	$E \rightarrow .E+T$	⊣ +	0
	$E \rightarrow .T$	⊣ +	0
	$T \rightarrow .T*P$	⊣ +*	0
	$T \rightarrow .P$	⊣ +*	0
	$P \rightarrow .a$	⊣ +*	0
S_1	$P \rightarrow a.$	⊣ +*	0
(X_2=+)			

Completer:
O ponto está no final da produção. Então posso evoluir na análise do não terminal reconhecido (lado esquerdo da produção) :
- trago para essa coluna as produções (daquele estado de origem em que aquele não-terminal está logo depois do ponto)

Para cada coluna j = 0 até n+1
  Para cada estado (linha) s
    Predictor, completer ou scanner

root: E → T | E+T input string = a+a*a
T → P | T*P
P → a

$k = 1$

S_0	$\phi \rightarrow .E$	⊣	0
(X_1=a)	$E \rightarrow .E+T$	⊣ +	0
	$E \rightarrow .T$	⊣ +	0
	$T \rightarrow .T*P$	⊣ +*	0
	$T \rightarrow .P$	⊣ +*	0
	$P \rightarrow .a$	⊣ +*	0
S_1	$P \rightarrow a.$	⊣ +*	0
(X_2=+)	$T \rightarrow P.$	⊣ +*	0

Completer:
O ponto está no final da produção. Então posso evoluir na análise do não terminal reconhecido (lado esquerdo da produção) :
- trago para essa coluna as produções (daquele estado de origem em que aquele não-terminal está logo depois do ponto)

Para cada coluna j = 0 até n+1
  Para cada estado (linha) s
    Predictor, completer ou scanner

root: E → T | E+T input string = a+a*a
T → P | T*P
P → a

$k = 1$

S_0	$\phi \rightarrow .E$	⊣	0
$(X_1=a)$	$E \rightarrow .E+T$	⊣ +	0
	$E \rightarrow .T$	⊣ +	0
	$T \rightarrow .T*P$	⊣ +*	0
	$T \rightarrow .P$	⊣ +*	0
	$P \rightarrow .a$	⊣ +*	0
S_1	$P \rightarrow a.$	⊣ +*	0
$(X_2=+)$	$T \rightarrow P.$	⊣ +*	0

Completer:
O ponto está no final da produção. Então posso evoluir na análise do não terminal reconhecido (lado esquerdo da produção) :
- trago para essa coluna as produções (daquele estado de origem em que aquele não-terminal está logo depois do ponto)

Para cada coluna j = 0 até n+1
  Para cada estado (linha) s
    Predictor, completer ou scanner

root: E → T | E+T input string = a+a*a
T → P | T*P
P → a

$k = 1$

S_0	$\phi \rightarrow .E$	⊣	0
$(X_1=a)$	$E \rightarrow .E+T$	⊣ +	0
	$E \rightarrow .T$	⊣ +	0
	$T \rightarrow .T*P$	⊣ +*	0
	$T \rightarrow .P$	⊣ +*	0
	$P \rightarrow .a$	⊣ +*	0
S_1	$P \rightarrow a.$	⊣ +*	0
$(X_2=+)$	$T \rightarrow P.$	⊣ +*	0
	$E \rightarrow T.$	⊣ +	0
	$T \rightarrow T.*P$	⊣ +*	0

Completer:
O ponto está no final da produção. Então posso evoluir na análise do não terminal reconhecido (lado esquerdo da produção) :
- trago para essa coluna as produções (daquele estado de origem em que aquele não-terminal está logo depois do ponto)

Para cada coluna j = 0 até n+1
  Para cada estado (linha) s
    Predictor, completer ou scanner

root: E → T | E+T input string = a+a*a
T → P | T*P
P → a

$k = 1$

S_0	$\phi \rightarrow .E$	⊣	0
(X_1=a)	$E \rightarrow .E+T$	⊣ +	0
	$E \rightarrow .T$	⊣ +	0
	$T \rightarrow .T*P$	⊣ +*	0
	$T \rightarrow .P$	⊣ +*	0
	$P \rightarrow .a$	⊣ +*	0
S_1	$P \rightarrow a.$	⊣ +*	0
(X_2=+)	$T \rightarrow P.$	⊣ +*	0
→	$E \rightarrow T.$	⊣ +	0
	$T \rightarrow T.*P$	⊣ +*	0

Completer:
O ponto está no final da produção. Então posso evoluir na análise do não terminal reconhecido (lado esquerdo da produção) :
- trago para essa coluna as produções (daquele estado de origem em que aquele não-terminal está logo depois do ponto)

Para cada coluna j = 0 até n+1
  Para cada estado (linha) s
    Predictor, completer ou scanner

root: E → T | E+T input string = a+a*a
T → P | T*P
P → a

$k = 1$

S_0	$\phi \rightarrow .E\dashv$	$\dashv$	0
$(X_1=a)$	$E \rightarrow .E+T$	$\dashv +$	0
	$E \rightarrow .T$	$\dashv +$	0
	$T \rightarrow .T*P$	$\dashv +*$	0
	$T \rightarrow .P$	$\dashv +*$	0
	$P \rightarrow .a$	$\dashv +*$	0
S_1	$P \rightarrow a.$	$\dashv +*$	0
$(X_2=+)$	$T \rightarrow P.$	$\dashv +*$	0
→	$E \rightarrow T.$	$\dashv +$	0
	$T \rightarrow T.*P$	$\dashv +*$	0
	$\phi \rightarrow E.\dashv$	$\dashv$	0
	$E \rightarrow E.+T$	$\dashv +$	0

Completer:
O ponto está no final da produção. Então posso evoluir na análise do não terminal reconhecido (lado esquerdo da produção) :
- trago para essa coluna as produções (daquele estado de origem em que aquele não-terminal está logo depois do ponto)

Para cada coluna j = 0 até n+1
  Para cada estado (linha) s
    Predictor, completer ou scanner

root: E → T | E+T input string = a+a*a
T → P | T*P
P → a

$k = 1$

S_0	$\phi \rightarrow .E\dashv$	$\dashv$	0
(X_1=a)	$E \rightarrow .E+T$	$\dashv +$	0
	$E \rightarrow .T$	$\dashv +$	0
	$T \rightarrow .T*P$	$\dashv +*$	0
	$T \rightarrow .P$	$\dashv +*$	0
	$P \rightarrow .a$	$\dashv +*$	0
S_1	$P \rightarrow a.$	$\dashv +*$	0
(X_2=+)	$T \rightarrow P.$	$\dashv +*$	0
	$E \rightarrow T.$	$\dashv +$	0
	$T \rightarrow T.*P$	$\dashv +*$	0
	$\phi \rightarrow E.\dashv$	$\dashv$	0
	$E \rightarrow E.+T$	$\dashv +$	0

Scanner:
Depois do ponto vem um terminal. Se ele bater com o símbolo atual da cadeia de entrada então posso evoluir na análise:
- avanço o ponto da produção e a levo para a próxima coluna

Para cada coluna j = 0 até n+1

Para cada estado (linha) s

Predictor, completer ou scanner

root: E → T | E+T input string = a+a*a

T → P | T*P

P → a

$k = 1$

S_0	$\phi \rightarrow .E\dashv$	$\dashv$	0
(X_1=a)	$E \rightarrow .E+T$	$\dashv +$	0
	$E \rightarrow .T$	$\dashv +$	0
	$T \rightarrow .T*P$	$\dashv +*$	0
	$T \rightarrow .P$	$\dashv +*$	0
	$P \rightarrow .a$	$\dashv +*$	0
S_1	$P \rightarrow a.$	$\dashv +*$	0
(X_2=+)	$T \rightarrow P.$	$\dashv +*$	0
	$E \rightarrow T.$	$\dashv +$	0
	$T \rightarrow T.*P$	$\dashv +*$	0
→	$\phi \rightarrow E.\dashv$	$\dashv$	0
	$E \rightarrow E.+T$	$\dashv +$	0

Para cada coluna j = 0 até n+1
  Para cada estado (linha) s
    Predictor, completer ou scanner

root: E → T | E+T input string = a+a*a
T → P | T*P
P → a

$k = 1$

S_0	$\phi \rightarrow .E\dashv$	$\dashv$	0
(X_1=a)	$E \rightarrow .E+T$	$\dashv +$	0
	$E \rightarrow .T$	$\dashv +$	0
	$T \rightarrow .T*P$	$\dashv +*$	0
	$T \rightarrow .P$	$\dashv +*$	0
	$P \rightarrow .a$	$\dashv +*$	0
S_1	$P \rightarrow a.$	$\dashv +*$	0
(X_2=+)	$T \rightarrow P.$	$\dashv +*$	0
	$E \rightarrow T.$	$\dashv +$	0
	$T \rightarrow T.*P$	$\dashv +*$	0
	$\phi \rightarrow E.\dashv$	$\dashv$	0
→	$E \rightarrow E.+T$	$\dashv +$	0

Scanner:
Depois do ponto vem um terminal. Se ele bater com o símbolo atual da cadeia de entrada então posso evoluir na análise:
- avanço o ponto da produção e a levo para a próxima coluna

Para cada coluna j = 0 até n+1
  Para cada estado (linha) s
    Predictor, completer ou scanner

root: E → T | E+T input string = a+a*a
T → P | T*P
P → a

			k = 1
S_0	$\phi \rightarrow .E$	⊣	0
(X_1=a)	$E \rightarrow .E+T$	⊣ +	0
	$E \rightarrow .T$	⊣ +	0
	$T \rightarrow .T*P$	⊣ +*	0
	$T \rightarrow .P$	⊣ +*	0
	$P \rightarrow .a$	⊣ +*	0
S_1	$P \rightarrow a.$	⊣ +*	0
(X_2=+)	$T \rightarrow P.$	⊣ +*	0
	$E \rightarrow T.$	⊣ +	0
	$T \rightarrow T.*P$	⊣ +*	0
	$\phi \rightarrow E.$⊣	⊣	0
→	$E \rightarrow E.+T$	⊣ +	0
S_2	$E \rightarrow E+.T$	⊣ +	0
(X_4=a)			

Scanner:
Depois do ponto vem um terminal. Se ele bater com o símbolo atual da cadeia de entrada então posso evoluir na análise:
- avanço o ponto da produção e a levo para a próxima coluna

Para cada coluna j = 0 até n+1
  Para cada estado (linha) s
    Predictor, completer ou scanner

E assim sucessivamente...

root: E → T | E+T input string = a+a*a
T → P | T*P
P → a

$k = 1$

S_0	$\phi \rightarrow .E\dashv$	$\dashv$	0
(X_1=a)	$E \rightarrow .E+T$	$\dashv +$	0
	$E \rightarrow .T$	$\dashv +$	0
	$T \rightarrow .T*P$	$\dashv +*$	0
	$T \rightarrow .P$	$\dashv +*$	0
	$P \rightarrow .a$	$\dashv +*$	0
S_1	$P \rightarrow a.$	$\dashv +*$	0
(X_2=+)	$T \rightarrow P.$	$\dashv +*$	0
	$E \rightarrow T.$	$\dashv +$	0
	$T \rightarrow T.*P$	$\dashv +*$	0
	$\phi \rightarrow E.\dashv$	$\dashv$	0
	$E \rightarrow E.+T$	$\dashv +$	0
S_2	$E \rightarrow E+.T$	$\dashv +$	0
(X_3=a)			

Para cada coluna j = 0 até n+1

Para cada estado (linha) s

Predictor, completer ou scanner

Se na coluna n+2 está o estado

$\phi \rightarrow E\dashv .\quad \dashv \quad 0$

então aceite

root: $E \rightarrow T \mid E+T$
$T \rightarrow P \mid T*P$
$P \rightarrow a$

input string = a+a*a

$k = 1$

S_0	$\phi \rightarrow .E\dashv$	$\dashv$	0	S_3	$P \rightarrow a.$	$\dashv +*$	2
(X_1=a)	$E \rightarrow .E+T$	$\dashv +$	0	(X_4=*)	$T \rightarrow P.$	$\dashv +*$	2
	$E \rightarrow .T$	$\dashv +$	0		$E \rightarrow E+T.$	$\dashv +$	0
	$T \rightarrow .T*P$	$\dashv +*$	0		$T \rightarrow T.*P$	$\dashv +*$	2
	$T \rightarrow .P$	$\dashv +*$	0				
	$P \rightarrow .a$	$\dashv +*$	0	S_4	$T \rightarrow T*.P$	$\dashv +*$	2
				(X_5=a)	$P \rightarrow .a$	$\dashv +*$	4
S_1	$P \rightarrow a.$	$\dashv +*$	0				
(X_2=+)	$T \rightarrow P.$	$\dashv +*$	0	S_5	$P \rightarrow a.$	$\dashv +*$	4
	$E \rightarrow T.$	$\dashv +$	0	(X_6=⊣)	$T \rightarrow T*P.$	$\dashv +*$	2
	$T \rightarrow T.*P$	$\dashv +*$	0		$E \rightarrow E+T.$	$\dashv +$	0
	$\phi \rightarrow E.\dashv$	$\dashv$	0		$T \rightarrow T.*P$	$\dashv +*$	2
	$E \rightarrow E.+T$	$\dashv +$	0		$\phi \rightarrow E.\dashv$	$\dashv$	0
					$E \rightarrow E.+T$	$\dashv +$	0
S_2	$E \rightarrow E+.T$	$\dashv +$	0				
(X_3=a)	$T \rightarrow .T*P$	$\dashv +*$	2	S_6	$\phi \rightarrow E\dashv .$	$\dashv$	0
	$T \rightarrow .P$	$\dashv +*$	2				
	$P \rightarrow .a$	$\dashv +*$	2				

Para cada coluna j = 0 até n+1

Para cada estado (linha) s

Predictor, completer ou scanner

Se na coluna n+2 está o estado

$\phi \rightarrow E\dashv\ .\quad \dashv\quad 0$

então aceite

E para reconstruir a árvore sintática?

root: $E \rightarrow T \mid E+T$
$T \rightarrow P \mid T*P$
$P \rightarrow a$

input string = a+a*a

$k = 1$

S_0	$\phi \rightarrow .E\dashv$	$\dashv$	0	S_3	$P \rightarrow a.$	$\dashv +*$	2
$(X_1=a)$	$E \rightarrow .E+T$	$\dashv +$	0	$(X_4=*)$	$T \rightarrow P.$	$\dashv +*$	2
	$E \rightarrow .T$	$\dashv +$	0		$E \rightarrow E+T.$	$\dashv +$	0
	$T \rightarrow .T*P$	$\dashv +*$	0		$T \rightarrow T.*P$	$\dashv +*$	2
	$T \rightarrow .P$	$\dashv +*$	0				
	$P \rightarrow .a$	$\dashv +*$	0	S_4	$T \rightarrow T*.P$	$\dashv +*$	2
				$(X_5=a)$	$P \rightarrow .a$	$\dashv +*$	4
S_1	$P \rightarrow a.$	$\dashv +*$	0				
$(X_2=+)$	$T \rightarrow P.$	$\dashv +*$	0	S_5	$P \rightarrow a.$	$\dashv +*$	4
	$E \rightarrow T.$	$\dashv +$	0	$(X_6=\dashv)$	$T \rightarrow T*P.$	$\dashv +*$	2
	$T \rightarrow T.*P$	$\dashv +*$	0		$E \rightarrow E+T.$	$\dashv +$	0
	$\phi \rightarrow E.\dashv$	$\dashv$	0		$T \rightarrow T.*P$	$\dashv +*$	2
	$E \rightarrow E.+T$	$\dashv +$	0		$\phi \rightarrow E.\dashv$	$\dashv$	0
					$E \rightarrow E.+T$	$\dashv +$	0
S_2	$E \rightarrow E+.T$	$\dashv +$	0				
$(X_3=a)$	$T \rightarrow .T*P$	$\dashv +*$	2	S_6	$\phi \rightarrow E\dashv\ .$	$\dashv$	0
	$T \rightarrow .P$	$\dashv +*$	2				
	$P \rightarrow .a$	$\dashv +*$	2				

Para cada coluna j = 0 até n+1

Para cada estado (linha) s

Predictor, completer ou scanner

Se na coluna n+2 está o estado

$\phi \rightarrow E\dashv .\quad \dashv \quad 0$

então aceite

E para reconstruir
a árvore sintática?
Completer deve deixar rastro

root: $E \rightarrow T \mid E+T$
$T \rightarrow P \mid T*P$
$P \rightarrow a$

input string = a+a*a

$k = 1$

S_0	$\phi \rightarrow .E\dashv$	$\dashv$	0	S_3	$P \rightarrow a.$	$\dashv +*$	2
$(X_1=a)$	$E \rightarrow .E+T$	$\dashv +$	0	$(X_4=*)$	$T \rightarrow P.$	$\dashv +*$	2
	$E \rightarrow .T$	$\dashv +$	0		$E \rightarrow E+T.$	$\dashv +$	0
	$T \rightarrow .T*P$	$\dashv +*$	0		$T \rightarrow T.*P$	$\dashv +*$	2
	$T \rightarrow .P$	$\dashv +*$	0				
	$P \rightarrow .a$	$\dashv +*$	0	S_4	$T \rightarrow T*.P$	$\dashv +*$	2
				$(X_5=a)$	$P \rightarrow .a$	$\dashv +*$	4
S_1	$P \rightarrow a.$	$\dashv +*$	0				
$(X_2=+)$	$T \rightarrow P.$	$\dashv +*$	0	S_5	$P \rightarrow a.$	$\dashv +*$	4
	$E \rightarrow T.$	$\dashv +$	0	$(X_6=\dashv)$	$T \rightarrow T*P.$	$\dashv +*$	2
	$T \rightarrow T.*P$	$\dashv +*$	0		$E \rightarrow E+T.$	$\dashv +$	0
	$\phi \rightarrow E.\dashv$	$\dashv$	0		$T \rightarrow T.*P$	$\dashv +*$	2
	$E \rightarrow E.+T$	$\dashv +$	0		$\phi \rightarrow E.\dashv$	$\dashv$	0
					$E \rightarrow E.+T$	$\dashv +$	0
S_2	$E \rightarrow E+.T$	$\dashv +$	0				
$(X_3=a)$	$T \rightarrow .T*P$	$\dashv +*$	2	S_6	$\phi \rightarrow E\dashv .$	$\dashv$	0
	$T \rightarrow .P$	$\dashv +*$	2				
	$P \rightarrow .a$	$\dashv +*$	2				

Para cada coluna j = 0 até n+1

Para cada estado (linha) s

Predictor, completer ou scanner

Se na coluna n+2 está o estado

$\phi \rightarrow E\dashv .\quad \dashv \quad 0$

então aceite

E para reconstruir
a árvore sintática?
Completer deve deixar rastro
Basta seguir
o processo contrário dos completers

root: $E \rightarrow T \mid E+T$
$T \rightarrow P \mid T*P$
$P \rightarrow a$

input string = a+a*a

$k = 1$

S_0	$\phi \rightarrow .E\dashv$	$\dashv$	0	S_3	$P \rightarrow a.$	$\dashv +*$	2
$(X_1=a)$	$E \rightarrow .E+T$	$\dashv +$	0	$(X_4=*)$	$T \rightarrow P.$	$\dashv +*$	2
	$E \rightarrow .T$	$\dashv +$	0		$E \rightarrow E+T.$	$\dashv +$	0
	$T \rightarrow .T*P$	$\dashv +*$	0		$T \rightarrow T.*P$	$\dashv +*$	2
	$T \rightarrow .P$	$\dashv +*$	0				
	$P \rightarrow .a$	$\dashv +*$	0	S_4	$T \rightarrow T*.P$	$\dashv +*$	2
				$(X_5=a)$	$P \rightarrow .a$	$\dashv +*$	4
S_1	$P \rightarrow a.$	$\dashv +*$	0				
$(X_2=+)$	$T \rightarrow P.$	$\dashv +*$	0	S_5	$P \rightarrow a.$	$\dashv +*$	4
	$E \rightarrow T.$	$\dashv +$	0	$(X_6=\dashv)$	$T \rightarrow T*P.$	$\dashv +*$	2
	$T \rightarrow T.*P$	$\dashv +*$	0		$E \rightarrow E+T.$	$\dashv +$	0
	$\phi \rightarrow E.\dashv$	$\dashv$	0		$T \rightarrow T.*P$	$\dashv +*$	2
	$E \rightarrow E.+T$	$\dashv +$	0		$\phi \rightarrow E.\dashv$	$\dashv$	0
					$E \rightarrow E.+T$	$\dashv +$	0
S_2	$E \rightarrow E+.T$	$\dashv +$	0				
$(X_3=a)$	$T \rightarrow .T*P$	$\dashv +*$	2	S_6	$\phi \rightarrow E\dashv .$	$\dashv$	0
	$T \rightarrow .P$	$\dashv +*$	2				
	$P \rightarrow .a$	$\dashv +*$	2				

Para cada coluna j = 0 até n+1

Para cada estado (linha) s

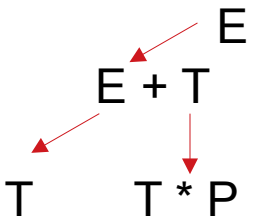
Predictor, completer ou scanner

Se na coluna n+2 está o estado

$\phi \rightarrow E\dashv .\quad \dashv \quad 0$

então aceite

E para reconstruir
a árvore sintática?
Completer deve deixar rastro
Basta seguir
o processo contrário dos completers



root: $E \rightarrow T \mid E+T$
$T \rightarrow P \mid T*P$
$P \rightarrow a$

input string = a+a*a

$k = 1$

S_0	$\phi \rightarrow .E\dashv$	$\dashv$	0	S_3	$P \rightarrow a.$	$\dashv +*$	2
$(X_1=a)$	$E \rightarrow .E+T$	$\dashv +$	0	$(X_4=*)$	$T \rightarrow P.$	$\dashv +*$	2
	$E \rightarrow .T$	$\dashv +$	0		$E \rightarrow E+T.$	$\dashv +$	0
	$T \rightarrow .T*P$	$\dashv +*$	0		$T \rightarrow T.*P$	$\dashv +*$	2
	$T \rightarrow .P$	$\dashv +*$	0				
	$P \rightarrow .a$	$\dashv +*$	0	S_4	$T \rightarrow T*.P$	$\dashv +*$	2
				$(X_5=a)$	$P \rightarrow .a$	$\dashv +*$	4
S_1	$P \rightarrow a.$	$\dashv +*$	0				
$(X_2=+)$	$T \rightarrow P.$	$\dashv +*$	0	S_5	$P \rightarrow a.$	$\dashv +*$	4
	$E \rightarrow T.$	$\dashv +$	0	$(X_6=\dashv)$	$T \rightarrow T*P.$	$\dashv +*$	2
	$T \rightarrow T.*P$	$\dashv +*$	0		$E \rightarrow E+T.$	$\dashv +$	0
	$\phi \rightarrow E.\dashv$	$\dashv$	0		$T \rightarrow T.*P$	$\dashv +*$	2
	$E \rightarrow E.+T$	$\dashv +$	0		$\phi \rightarrow E.\dashv$	$\dashv$	0
					$E \rightarrow E.+T$	$\dashv +$	0
S_2	$E \rightarrow E+.T$	$\dashv +$	0				
$(X_3=a)$	$T \rightarrow .T*P$	$\dashv +*$	2	S_6	$\phi \rightarrow E\dashv .$	$\dashv$	0
	$T \rightarrow .P$	$\dashv +*$	2				
	$P \rightarrow .a$	$\dashv +*$	2				

Para cada coluna j = 0 até n+1

Para cada estado (linha) s

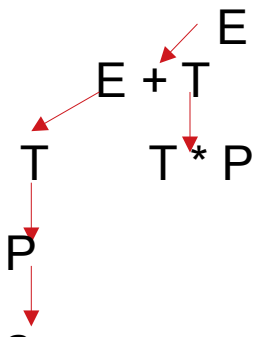
Predictor, completer ou scanner

Se na coluna n+2 está o estado

$\phi \rightarrow E\dashv .\quad \dashv \quad 0$

então aceite

E para reconstruir
a árvore sintática?
Completer deve deixar rastro
Basta seguir
o processo contrário dos completers



root: $E \rightarrow T \mid E+T$
$T \rightarrow P \mid T*P$
$P \rightarrow a$

input string = a+a*a

$k = 1$

S_0	$\phi \rightarrow .E\dashv$	$\dashv$	0	S_3	$P \rightarrow a.$	$\dashv +*$	2
$(X_1=a)$	$E \rightarrow .E+T$	$\dashv +$	0	$(X_4=*)$	$T \rightarrow P.$	$\dashv +*$	2
	$E \rightarrow .T$	$\dashv +$	0		$E \rightarrow E+T.$	$\dashv +$	0
	$T \rightarrow .T*P$	$\dashv +*$	0		$T \rightarrow T.*P$	$\dashv +*$	2
	$T \rightarrow .P$	$\dashv +*$	0				
	$P \rightarrow .a$	$\dashv +*$	0	S_4	$T \rightarrow T*.P$	$\dashv +*$	2
				$(X_5=a)$	$P \rightarrow .a$	$\dashv +*$	4
S_1	$P \rightarrow a.$	$\dashv +*$	0				
$(X_2=+)$	$T \rightarrow P.$	$\dashv +*$	0	S_5	$P \rightarrow a.$	$\dashv +*$	4
	$E \rightarrow T.$	$\dashv +$	0	$(X_6=\dashv)$	$T \rightarrow T*P.$	$\dashv +*$	2
	$T \rightarrow T.*P$	$\dashv +*$	0		$E \rightarrow E+T.$	$\dashv +$	0
	$\phi \rightarrow E.\dashv$	$\dashv$	0		$T \rightarrow T.*P$	$\dashv +*$	2
	$E \rightarrow E.+T$	$\dashv +$	0		$\phi \rightarrow E.\dashv$	$\dashv$	0
					$E \rightarrow E.+T$	$\dashv +$	0
S_2	$E \rightarrow E+.T$	$\dashv +$	0				
$(X_3=a)$	$T \rightarrow .T*P$	$\dashv +*$	2	S_6	$\phi \rightarrow E\dashv .$	$\dashv$	0
	$T \rightarrow .P$	$\dashv +*$	2				
	$P \rightarrow .a$	$\dashv +*$	2				

Para cada coluna j = 0 até n+1

Para cada estado (linha) s

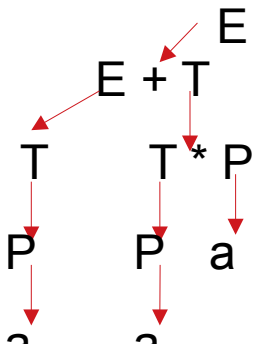
Predictor, completer ou scanner

Se na coluna n+2 está o estado

$\phi \rightarrow E\dashv .\quad \dashv \quad 0$

então aceite

E para reconstruir
a árvore sintática?
Completer deve deixar rastro
Basta seguir
o processo contrário dos completers



root: $E \rightarrow T \mid E+T$
$T \rightarrow P \mid T*P$
$P \rightarrow a$

input string = a+a*a

$k = 1$

S_0	$\phi \rightarrow .E\dashv$	$\dashv$	0	S_3	$P \rightarrow a.$	$\dashv +*$	2
$(X_1=a)$	$E \rightarrow .E+T$	$\dashv +$	0	$(X_4=*)$	$T \rightarrow P.$	$\dashv +*$	2
	$E \rightarrow .T$	$\dashv +$	0		$E \rightarrow E+T.$	$\dashv +$	0
	$T \rightarrow .T*P$	$\dashv +*$	0		$T \rightarrow T.*P$	$\dashv +*$	2
	$T \rightarrow .P$	$\dashv +*$	0				
	$P \rightarrow .a$	$\dashv +*$	0	S_4	$T \rightarrow T*.P$	$\dashv +*$	2
				$(X_5=a)$	$P \rightarrow .a$	$\dashv +*$	4
S_1	$P \rightarrow a.$	$\dashv +*$	0				
$(X_2=+)$	$T \rightarrow P.$	$\dashv +*$	0	S_5	$P \rightarrow a.$	$\dashv +*$	4
	$E \rightarrow T.$	$\dashv +$	0	$(X_6=\dashv)$	$T \rightarrow T*P.$	$\dashv +*$	2
	$T \rightarrow T.*P$	$\dashv +*$	0		$E \rightarrow E+T.$	$\dashv +$	0
	$\phi \rightarrow E.\dashv$	$\dashv$	0		$T \rightarrow T.*P$	$\dashv +*$	2
	$E \rightarrow E.+T$	$\dashv +$	0		$\phi \rightarrow E.\dashv$	$\dashv$	0
					$E \rightarrow E.+T$	$\dashv +$	0
S_2	$E \rightarrow E+.T$	$\dashv +$	0				
$(X_3=a)$	$T \rightarrow .T*P$	$\dashv +*$	2	S_6	$\phi \rightarrow E\dashv .$	$\dashv$	0
	$T \rightarrow .P$	$\dashv +*$	2				
	$P \rightarrow .a$	$\dashv +*$	2				

Fim do vídeo 5

Algoritmo de Earley

Vídeo 6

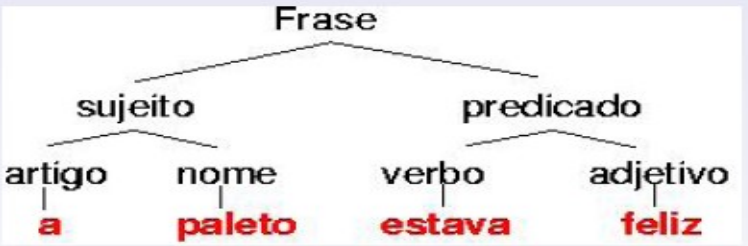
Gramáticas estocásticas

Gramáticas estocásticas

Frase	→	sujeito	predicado	[1.0]
sujeito	→	artigo	nome	[1.0]
artigo	→	a		[0.3]
artigo	→	o		[0.7]
nome	→	paletó		[0.2]
nome	→	moça		[0.3]
nome	→	dia		[0.5]
predicado	→	verbo	adjetivo	[1.0]
verbo	→	é		[0.7]
verbo	→	estava		[0.3]
adjectivo	→	feliz		[0.6]
adjectivo	→	azul		[0.4]

Gramáticas estocásticas

Frase	→	sujeito	predicado	[1.0]
sujeito	→	artigo	nome	[1.0]
artigo	→	a		[0.3]
artigo	→	o		[0.7]
nome	→	paleto		[0.2]
nome	→	moça		[0.3]
nome	→	dia		[0.5]
predicado	→	verbo	adjetivo	[1.0]
verbo	→	é		[0.7]
verbo	→	estava		[0.3]
adjectivo	→	feliz		[0.6]
adjectivo	→	azul		[0.4]

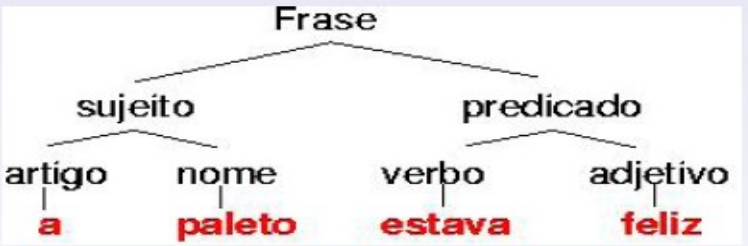


Gramáticas estocásticas

Frase	→	sujeito	predicado	[1.0]
sujeito	→	artigo	nome	[1.0]
artigo	→	a		[0.3]
artigo	→	o		[0.7]
nome	→	paletó		[0.2]
nome	→	moça		[0.3]
nome	→	dia		[0.5]
predicado	→	verbo	adjetivo	[1.0]
verbo	→	é		[0.7]
verbo	→	estava		[0.3]
adjectivo	→	feliz		[0.6]
adjectivo	→	azul		[0.4]

Gramáticas estocásticas

Frase	→	sujeito	predicado	[1.0]
sujeito	→	artigo	nome	[1.0]
artigo	→	a		[0.3]
artigo	→	o		[0.7]
nome	→	paletó		[0.2]
nome	→	moça		[0.3]
nome	→	dia		[0.5]
predicado	→	verbo	adjetivo	[1.0]
verbo	→	é		[0.7]
verbo	→	estava		[0.3]
adjectivo	→	feliz		[0.6]
adjectivo	→	azul		[0.4]

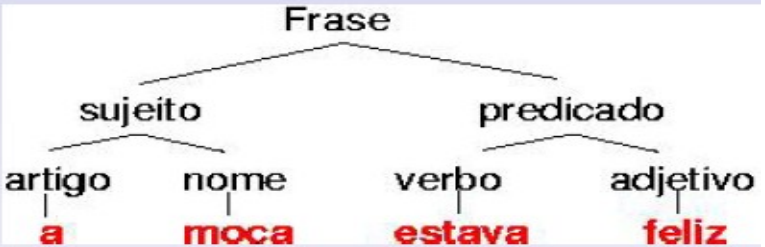


Gramáticas estocásticas

Frase	→	sujeito	predicado	[1.0]
sujeito	→	artigo	nome	[1.0]
artigo	→	a		[0.3]
artigo	→	o		[0.7]
nome	→	paletó		[0.2]
nome	→	moça		[0.3]
nome	→	dia		[0.5]
predicado	→	verbo	adjetivo	[1.0]
verbo	→	é		[0.7]
verbo	→	estava		[0.3]
adjectivo	→	feliz		[0.6]
adjectivo	→	azul		[0.4]

Gramáticas estocásticas

Frase	→	sujeito	predicado	[1.0]
sujeito	→	artigo	nome	[1.0]
artigo	→	a		[0.3]
artigo	→	o		[0.7]
nome	→	paletó		[0.2]
nome	→	moça		[0.3]
nome	→	dia		[0.5]
predicado	→	verbo	adjetivo	[1.0]
verbo	→	é		[0.7]
verbo	→	estava		[0.3]
adjectivo	→	feliz		[0.6]
adjectivo	→	azul		[0.4]



P(x, t | G) = produto das probabilidades das produções usadas na derivação (árvore sintática) t de x (dadas pela gramática G)

$P(x \mid G) = \sum_i P(x, t_i \mid G)$ se a gramática for ambígua

Gramáticas Estocásticas

- Definição: uma gramática estocática G é uma quíntupla (V, Σ, S, P, ρ), 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

    $(\Sigma \cup V)^* V (\Sigma \cup V)^* \rightarrow (\Sigma \cup V)^*$
  - ρ é o conjunto de distribuições de probabilidades sobre as produções de mesmo lado esquerdo

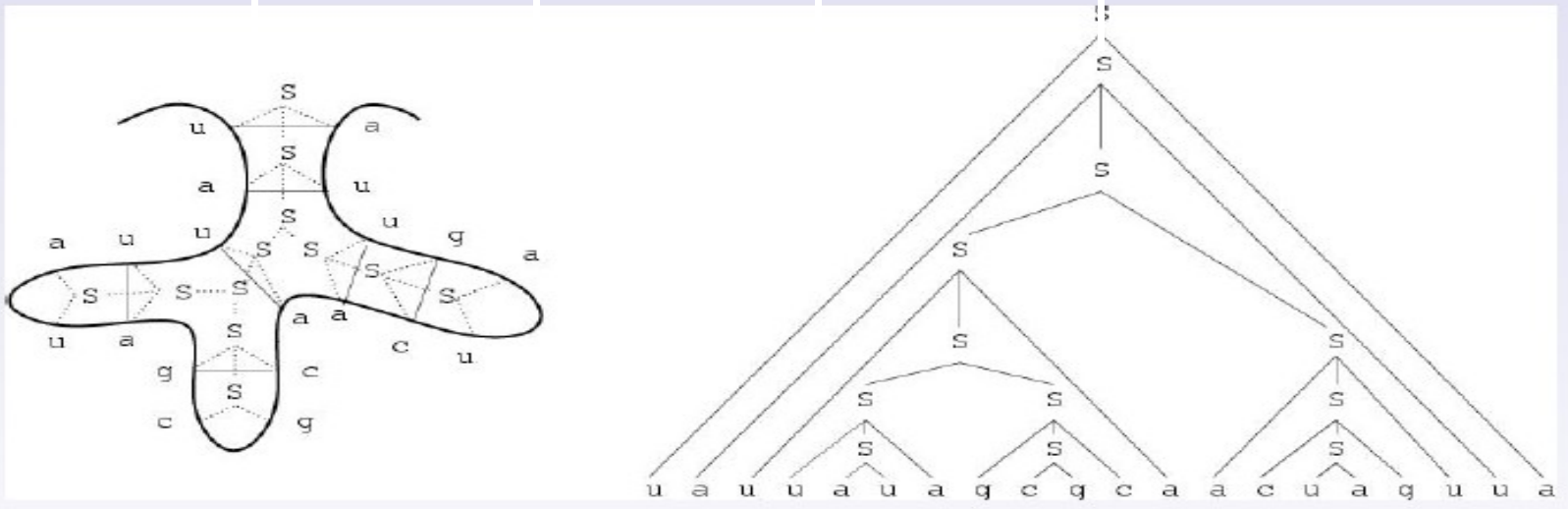
    $\sum_i \rho(\alpha \rightarrow \beta_i) = 1$

Análise sintática

- Analisadores sintáticos: programas que, dados uma gramática G e uma sequência s, solta:
  - Se a sequência s é ou não reconhecida pela gramática G (determinístico - não estocástico)
  - Qual a probabilidade da sequência s dada G (estocástico)
  - Qual(is) a(s) árvore(s) sintática(s) de s dada G (uma, só a mais provável ou todas)

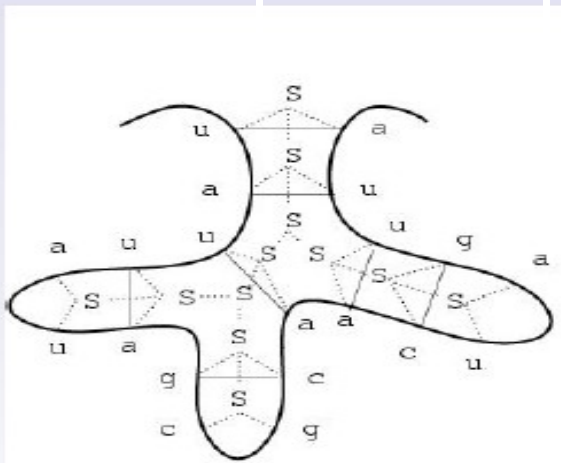
Estrutura secundária e gramáticas

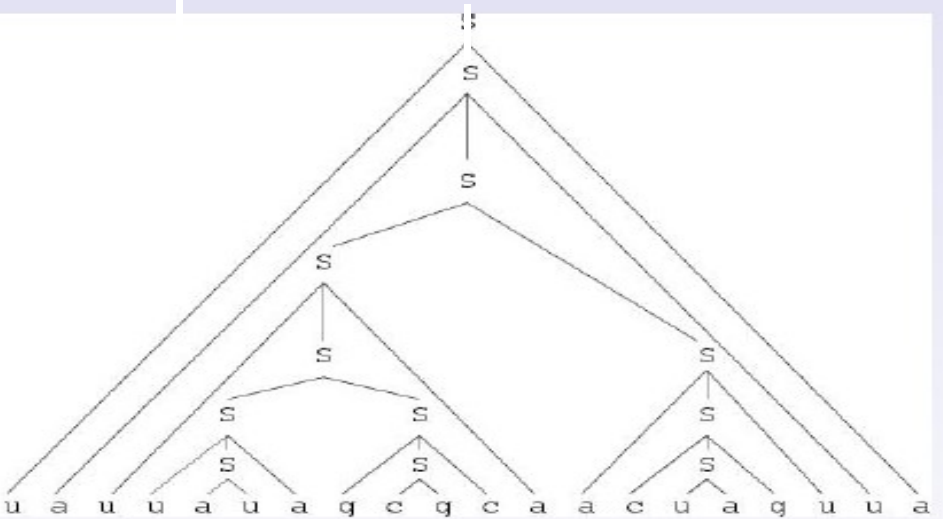
S →	aSu [0.1]	uSa [0.1]	cSg [0.1]	gSc [0.1]
S →	aS [0.1]	uS [0.1]	cS [0.1]	gS [0.1]
S →	SS [0.04]	a [0.04]	u [0.04]	c [0.04] g [0.04]



Estrutura secundária e gramáticas

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]





Mas note que a Forma Normal de Chomsky nos atrapalha nesse caso!

Software para análise sintática (não probabilística)

- Geradores de analisadores sintáticos gerais:
  - Descendentes, gramáticas LL(k): ANTLR (Another Tool for Language Recognition) e JavaCC
  - Ascendentes, como o YACC (Yet Another Compiler-Compiler): gramáticas LALR
- Geradores mais simples: pyparsing
- Dedicadas a processamento de linguagem natural (rotinas extras para esse fim): NLTK, Spacy, Stanford CoreNLP Python, TextBlob, Gensim, Pattern, Polyglot, PyNLPl, Vocabulary

Análise sintática probabilística

- Implementações disponibilizadas, por ex no GitHub (ex: CYK, Earley)
- Na próxima aula discutiremos esses algoritmos, assim como algoritmos de aprendizados de GLC Estocásticas (estimação de probabilidades e inferência das regras)

Fim do vídeo 6

Gramáticas estocásticas

Referências

DURBIN, R.; EDDY, S. R.; KROGH, A. **Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.** Cambridge University Press, 2002. Cap 9

EARLEY, J. An Efficient Context-Free Parsing Algorithm. **Communications of the ACM**, vol 13, n.2, p 94-102, 1970.

RAMOS, M. V. M.; NETO, J. J.; VEJA, I. S. **Linguagens Formais: Teoria, Modelagem e Implementação**. Ed. Bookman, 2009.

SIPSER, M. **Introdução à Teoria da Computação**. Ed. Thomson, 2007