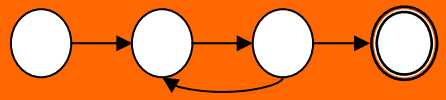
Automata e Linguagens Formais

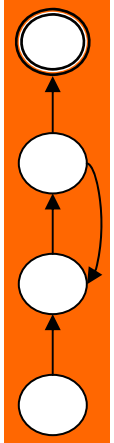
CTC 34



5

Prof. Carlos H. C. Ribeiro
carlos@ita.br

Gramáticas
A Hierarquia de Chomsky
Tipos de gramáticas e linguagens
Pré-normalização de GLCs
Formas Normais: Chomsky e Greibach

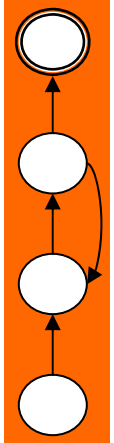


Gramática: Definição

- Uma **gramática** G = (V, Σ, P, S) consiste de:
  - um conjunto finito V de símbolos não-terminais;
  - um conjunto finito Σ de símbolos terminais, onde $V \cap \Sigma = \emptyset$;
  - um subconjunto P de $[(V \cup \Sigma)^* - \Sigma^*] \times (V \cup \Sigma)^*$, chamado conjunto de ***produções*** ou ***regras;***
  - um símbolo inicial $S \in V$.
- Uma produção $(A,B) \in P$ é escrita $A \rightarrow B$, onde $A \in [(V \cup \Sigma)^* - \Sigma^*]$ e $B \in (V \cup \Sigma)^*$.

  Assim, A deve conter ao menos um símbolo não-terminal e B pode conter qualquer combinação de símbolos terminais e não-terminais.

Uma gramática *gera (produz)* as cadeias de uma linguagem.



Exemplos de Gramáticas

Seja a gramática G = (V, Σ, P, S), com:

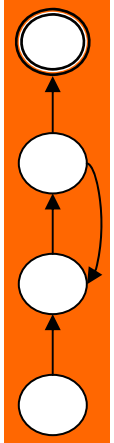
a) V={A,B}, Σ={0,1,#}, P={A→0A1,A→B,B→#},S=A.

b) V={S}, Σ ={a,b}, P={S→Sa, S→b}, S=S.

c) V={A,B}, Σ ={0,1}, P={B→ε, B→A, A→1A, A→0A, A→1, A→0}, S=B.

d) Uma gramática para gerar palíndromos sobre {a,b}:

V={S}, Σ ={a,b}, P={S → a, S → b, S → ε, S → aSa, S → bSb}, S=S.



Derivações

- ***Def.:*** Seja a gramática G = (V, Σ, P, S). Se $\alpha \rightarrow \beta$ é uma produção e $x\alpha y \in (V \cup \Sigma)^*$, dizemos que $x\beta y$ é **diretamente derivável** de $x\alpha y$ e escrevemos: $x\alpha y \Rightarrow x\beta y$.

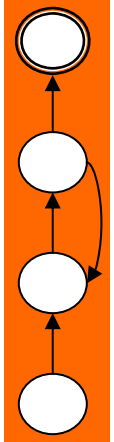
  Se $\alpha_i \in (V \cup \Sigma)^*$ para $i=1,..,n$ e α_{i+1} é diretamente derivável de α_i para $i=1,..,n-1$, dizemos que **α_n é derivável de α_1** e escrevemos: $\alpha_1 \overset{*}{\Rightarrow} \alpha_n$

- Derivação de α_n a partir de α_1: $\alpha_1 \overset{*}{\Rightarrow} \alpha_n$: $\alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow \alpha_n$

Por convenção, qualquer elemento de $(V \cup \Sigma)^*$ é derivável de si mesmo.

- **A linguagem L(G) gerada por G** consiste de todas as cadeias sobre Σ derivadas de S.

As gramáticas G e G' são equivalentes se L(G) = L(G').



Derivações: Exemplos, Formas Sentenciais e Sentenças

- Seja a gramática G = (V, Σ, P, S), com

V={S, A}, Σ ={a,b}, P={S→bS, S→aA, A→bA, A→b}

(a) A cadeia abAbb é diretamente derivável de aAbb, escrita como aAbb ⇒ abAbb, usando a produção A→ bA.

(b) A cadeia bbab é derivável de S, escrita $S \overset{*}{\Rightarrow} bbab$. A derivação é: S⇒bS⇒bbS⇒bbaA⇒bbab

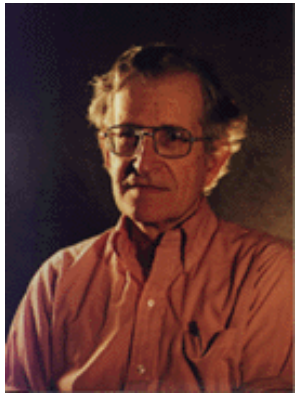
- ***Def.:*** Seja a gramática G = (V, Σ, P, S). Diz-se que $w \in (V \cup \Sigma)^*$ é uma **forma sentencial** de G se existir derivação $S \overset{*}{\Rightarrow} w$ em G.
- ***Def.:*** Uma cadeia $w \in \Sigma^*$ é uma **sentença** de G se existir uma derivação $S \overset{*}{\Rightarrow} w$ em G.

Forma sentencial: qualquer cadeia (incluindo símbolos não-terminais) deriváveis a partir do símbolo inicial S.

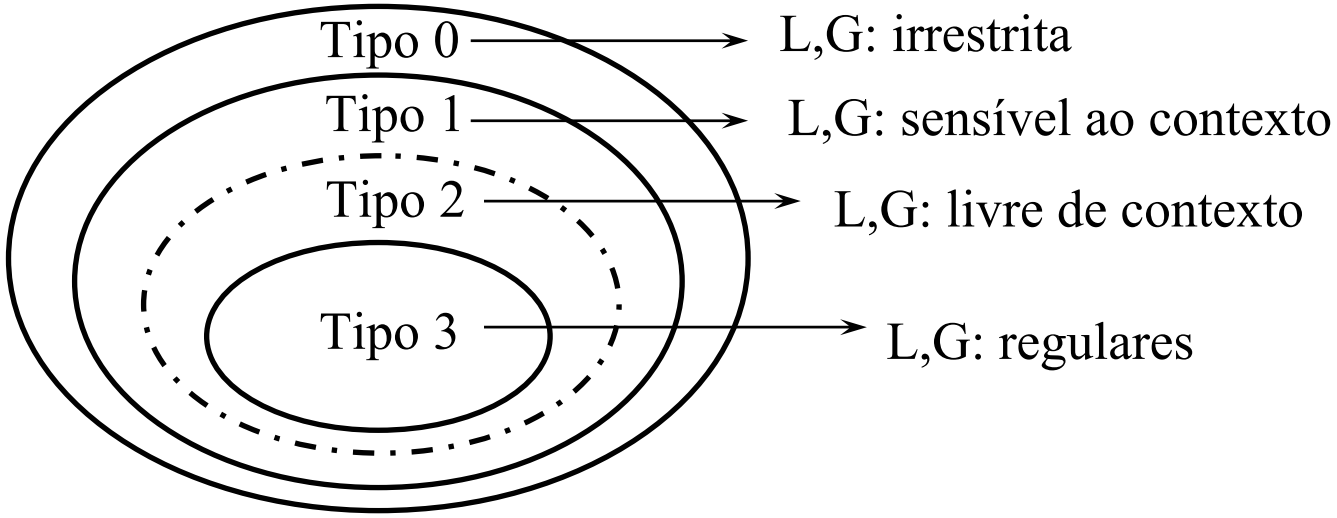
Sentença: forma sentencial apenas com símbolos terminais.

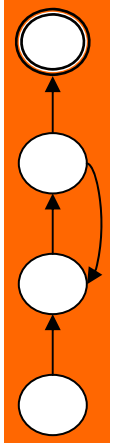
A Hierarquia de Chomsky

- Conforme as restrições impostas ao formato das produções de uma gramática, varia-se a classe de linguagens que tal gramática gera.
- Existem 4 classes de gramáticas, capazes de gerar 4 classes correspondentes de linguagens, de acordo com a denominada ***Hierarquia de Chomsky***, que estabelece uma relação de inclusão entre as gramáticas.



Noam Chomsky

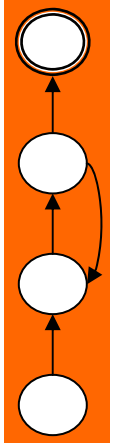




A Hierarquia de Chomsky, Linguagens e Gramáticas

- *Def.:* Uma linguagem L é irrestrita (respectivamente sensível ao contexto, livre de contexto, regular) se existe uma gramática irrestrita (respectivamente sensível ao contexto, livre de contexto, regular) G, com L=L(G).

- **Toda LR é LLC. Nem toda LLC é LR.**
- **Toda LLC que não produz cadeia vazia é LSC. Nem toda LSC é LLC.**
- **Toda LSC é LI. Nem toda LI é LSC.**



Gramáticas Regulares

- ***Definição (inclui itens a,b,c e d):*** *Seja G uma gramática e* ε *a cadeia nula.*

(a) Se toda produção estiver na forma $A \rightarrow a$ ou $A \rightarrow aB$ ou $A \rightarrow \varepsilon$, com $A,B \in V$, $a \in \Sigma$, G é uma **gramática regular** (ou **tipo 3**).

  – Nesta gramática, pode-se substituir um símbolo não terminal por: (i) um símbolo terminal, (ii) um símbolo terminal seguido por um não terminal ou (iii) pela cadeia nula.

Exemplo:

G = (V, Σ, P, S), com Σ ={a,b}, V={S, X}, P={$S \rightarrow bS$, $S \rightarrow aX$, $X \rightarrow bX$, $X \rightarrow b$} é **regular.**

Derivações possíveis: ab, abb, bbbabb,…

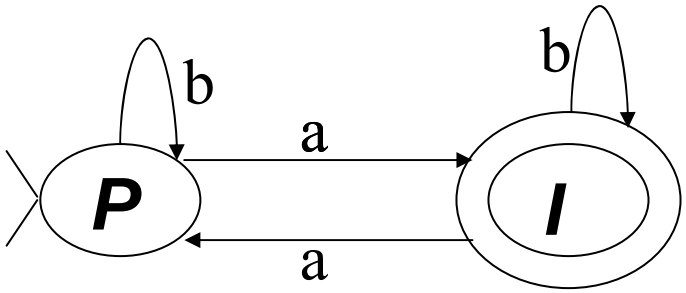
L(G) = $\mathbf{b^*abb^*}$

G é uma gramática regular, e é por isso que a linguagem L(G) que ela gera é uma linguagem regular!

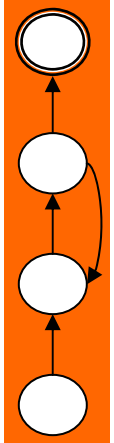
Gramáticas Regulares e Automata Finitos

- Nesta seção mostraremos que gramáticas regulares e automata finitos são essencialmente equivalentes, no sentido em que ambos são especificações de uma linguagem regular:
  - A gramática é **geradora** da linguagem
  - O autômato é **reconhecedor** da linguagem (já vimos)

- Seja o autômato finito abaixo, o qual aceita cadeias sobre {a,b} que contêm um número ímpar de a's.



Como determinar a gramática regular equivalente?



Gramáticas Regulares a partir de AFs

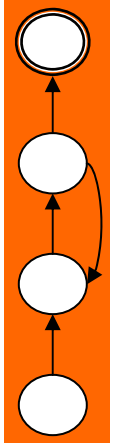
Teorema *(Chomsky e Miller, 1958)*:

Seja *M* um AF de estado inicial S. Seja Σ o conjunto dos símbolos de entrada e V o conjunto de estados de M. Defina produções $A \rightarrow xA'$ se existir um arco rotulado *x* de A para A' e $A \rightarrow \varepsilon$ se A for um estado de aceitação. Então a gramática regular G=(V,Σ,P,S) é tal que L(G)=L(M).

Para o exemplo do slide anterior:

- **AF ⇨ G:** os símbolos de entrada {a,b} do AF são os símbolos terminais de G. Os estados **P** e **I** são os símbolos não-terminais. O estado inicial **P** é o símbolo inicial. Os arcos do AF correspondem às produções de G. Se existir um arco rotulado por *x* de A para A', escreve-se a produção: $A \rightarrow xA'$.

  Temos então: $\boldsymbol{P \rightarrow bP}$, $\boldsymbol{P \rightarrow aI}$, $\boldsymbol{I \rightarrow aP}$, $\boldsymbol{I \rightarrow bI}$
- Além disso, se A for estado de aceitação, inclui-se $A \rightarrow \varepsilon$. *No exemplo:* $\boldsymbol{I \rightarrow \varepsilon}$.
- Assim, a gramática G=(V,Σ,P,**P**), com V={**I**,**P**}, Σ={a,b} e P={$\boldsymbol{P \rightarrow bP}$, $\boldsymbol{P \rightarrow aI}$, $\boldsymbol{I \rightarrow aP}$,$\boldsymbol{I \rightarrow bI}$, $\boldsymbol{I \rightarrow \varepsilon}$} gera a linguagem L(G), que corresponde ao conjunto de cadeias aceitas pelo autômato finito.



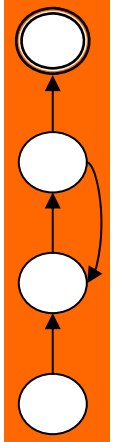
AFs a Partir de Gramáticas Regulares

Teorema *(Chomsky e Miller, 1958)*:

Seja G = (V,Σ,P,S) uma gramática regular. Seja I = Σ, X = V ∪ {F}, onde F∉ V∪ Σ, f(X,x) = {X' | X→xX' ∈ P} ∪ {F | X →x ∈ P}, A = {F} ∪ {X | X→ε ∈ P}. O autômato finito não-determinista *M* = (I, X, f, A, S) aceita precisamente as cadeias de L(G).

Exemplo:

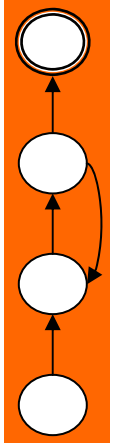
- Seja a gramática regular G=(V,T,P,S), com V={S, C}, T={a,b}, P={S→bS, S→aC, C→bC, C→b}. Determinar o AF correspondente.
- **G ⇨ AF:** Os símbolos não terminais serão os estados. Para cada produção da forma A→*x*A', desenhar uma ligação de A a A', com rótulo *x* (produções S→bS, S→aC, C→bC). A produção C→b equivale a: C→bF, F→ε, sendo F um símbolo não-terminal adicional. A produção F→ε indica que F é um estado de aceitação.



Equivalência AFs ⇔ Gramáticas Regulares

- Vimos então que, se A é um autômato finito, existe uma gramática regular G, com L(G)=L(*M*). Vimos também que, se G é uma gramática regular, existe um autômato finito não-determinista *M*, com L(G)=L(*M*).
- Como já vimos (aula 2) que sempre existe um AFD equivalente a qualquer AFND, concluímos que:

Se G é gramática regular, existe um AFD M, com L(G)=L(M).



Gramáticas Livres de Contexto

(b) Se toda produção estiver na forma $A \rightarrow \delta$, com $A \in V$, $\delta \in (V \cup \Sigma)^*$, G é uma **gramática livre de contexto** (ou **tipo 2**).

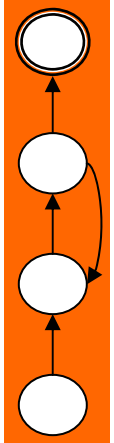
- Nesta gramática, pode-se substituir A (um não-terminal isolado) por δ sempre que se queira, independentemente do contexto em que A esteja inserido.

Exemplo: A gramática G = (V, Σ, P, S), com Σ ={a,b}, V={S}, P={S$\rightarrow$aSb, S$\rightarrow$ab} é **livre de contexto.**

Derivações possíveis: ab, aabb, aaabbb,…

L(G)={$a^n b^n \mid n=1,2,\ldots$} $\Rightarrow$ L é uma linguagem livre de contexto, mas não é uma linguagem regular!

Linguagens livres de contexto permitem, além das operações permitidas para uma linguagem regular, operações de aninhamento.



Forma Backus-Naur

- BNF ("**B**ackus-**N**aur **F**orm"): modo alternativo de descrever gramáticas livres de contexto.

  – símbolos não terminais: inclusos em <..>
  – produção $A \rightarrow T$, escrita: A::=T
  – produções da forma: $A::=T_1$, $A::=T_2$, …, $A::=T_n$ podem ser combinadas em
    $A::= T_1 | T_2 | \ldots | T_n$. *(lê-se "ou" para "|")*

Um exemplo:

Gramática para inteiros - um inteiro é definido como uma cadeia contendo um sinal opcional (+ ou -), seguido por uma cadeia de dígitos (0 a 9).

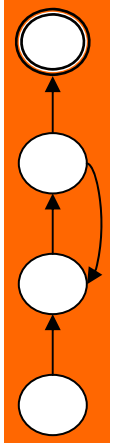
Símbolo inicial: <inteiro>

<digito>::= 0 |1| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<inteiro>::= <inteiro com sinal> | <inteiro sem sinal>

<inteiro com sinal>::= + < inteiro sem sinal> | - <inteiro sem sinal>

<inteiro sem sinal>::= <digito> | <digito> <inteiro sem sinal>



BNF: Exemplo

derivação do inteiro -901

<inteiro> ⇒ <inteiro com sinal>

⇒ - <inteiro sem sinal>

⇒ - <digito> <inteiro sem sinal>

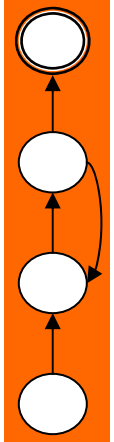
⇒ - <digito> <digito> <inteiro sem sinal>

⇒ - <digito> <digito> <digito>

⇒ - 9 <digito> <digito>

⇒ - 90 <digito>

⇒ - 901.



A gramática do exemplo anterior é LC, mas se mudarmos as produções para:

<digitos>::= 0<digitos> |1<digitos> |... | 9<digitos> | ε

<inteiro>::= + <inteiro sem sinal> | - <inteiro sem sinal> | 0<digitos> |1<digitos> |... | 9 <digitos>

<inteiro sem sinal>::= 0<digitos> |1<digitos> |... | 9<digitos>

resultará numa gramática G regular. Como a linguagem L=L(G) gerada não foi modificada, concluímos que L é definida mais precisamente como uma linguagem regular.

Árvores de Derivação

Forma de representar derivações em uma gramática livre de contexto.

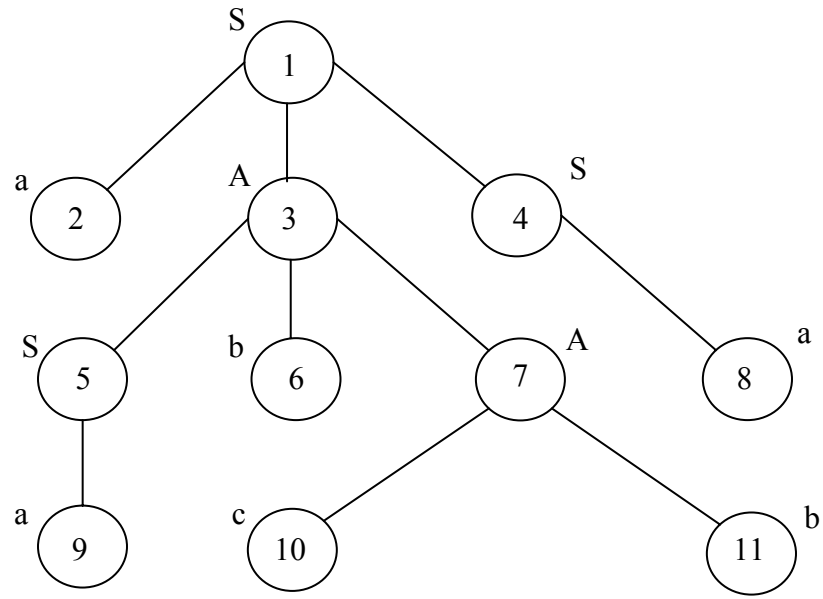
- Nós: símbolos terminais ou não-terminais
- Se existir um vértice A com sucessores X_1, X_2, ... , X_k, então $A \rightarrow X_1X_2...X_k$ é uma produção da gramática. A recíproca não é necessariamente verdadeira.
- O nó-raiz corresponde a um símbolo inicial S.

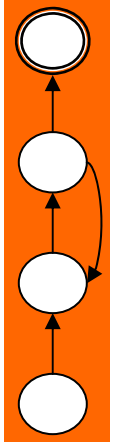
Exemplo G = ({S,A}, {a,b,c}, P, S}

P: S → aAS | a, A → SbA | SS | cb

O produto (***yield***) de uma árvore de derivação é a string formada pela leitura sequencial das folhas da árvore.

Neste exemplo: *produto*= aabcba





Relação entre Derivações e Árvores de Derivação

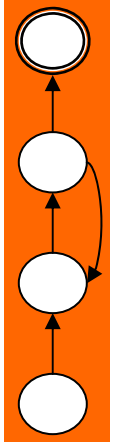
Teorema: Seja G = (V, Σ,P,S) uma GLC. Então $S \overset{*}{\Rightarrow} \alpha$ sss existir uma árvore de derivação em G com produto α.

Ou seja:

a) Dada uma derivação, existe uma árvore correspondente.

b) Dada uma árvore de derivação, seu produto corresponde a alguma derivação em G.

Prova: Hopcroft / Ullman, pp. 85-86



Derivações à Direita e à Esquerda

Derivação à Direita: cada passo de derivação aplicado à variável mais à direita.

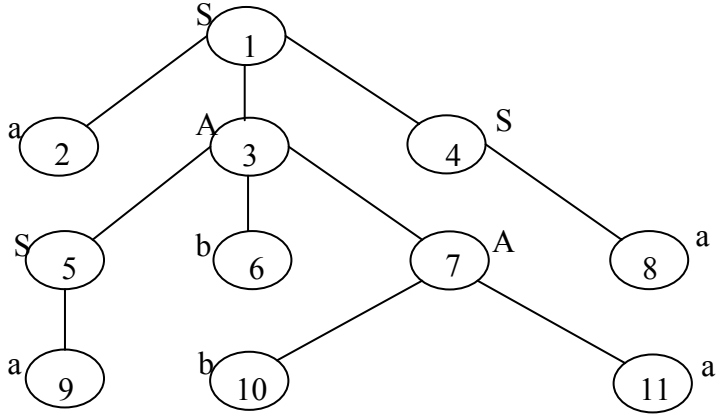
Derivação à Esquerda: cada passo de derivação aplicado à variável mais à esquerda.

Exemplo

G = ({S,A}, {a,b}, P, S}

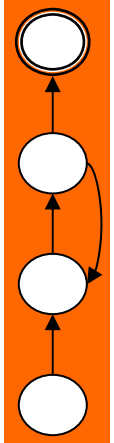
P: $S \rightarrow aAS \mid a$

$A \rightarrow SbA \mid SS \mid ba$



À direita: $S \Rightarrow aAS \Rightarrow aAa \Rightarrow aSbAa \Rightarrow aSbbaa \Rightarrow aabbaa$

À esquerda: $S \Rightarrow aAS \Rightarrow aSbAS \Rightarrow aabAS \Rightarrow aabbaS \Rightarrow aabbaa$



Gramáticas Sensíveis ao Contexto

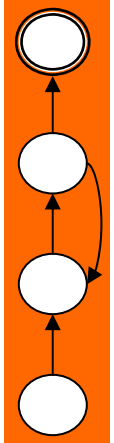
(c) Se toda produção estiver na forma $\alpha \rightarrow \beta$, com $\alpha, \beta \in (V \cup \Sigma)^+$ e $|\alpha| \leq |\beta|$, G é uma **gramática sensível ao contexto** (ou **tipo 1**).

- Gramáticas SC também são ditas monotônicas ou não-contráteis.
- Pode ser mostrado que GSCs podem ser escritas com regras na forma $\boldsymbol{u}A\boldsymbol{v} \rightarrow \boldsymbol{uwv}$, com $A \in V$, $\boldsymbol{w} \in (V \cup \Sigma)^+$ e $\boldsymbol{u}, \boldsymbol{v} \in (V \cup \Sigma)^*$
- Note que em GSCs não ocorre produção de cadeias nulas!

Exemplo: A gramática G = (V, Σ, P, S), com Σ ={a,b,c}, V={S,A,B,C,D,E}, P={S→aAB, S→aB, A→aAC, A→aC, B→Dc, D→b, CD→CE, CE→DE, DE→DC, Cc→Dcc} é **sensível ao contexto**

Derivações possíveis: abc, aabbcc, aaabbbccc,…

$L(G)=\{a^n b^n c^n \mid n>0\} \Rightarrow$ não existe uma gramática livre de contexto G com L=L(G); assim, L é sensível ao contexto e não é livre de contexto!



Gramáticas Irrestritas

(d) Se toda produção de G estiver na forma $\alpha \rightarrow \beta$, com $\alpha \in [(V \cup \Sigma)^* - \Sigma^*]$ e $\beta \in (V \cup \Sigma)^*$, G é uma **gramática irrestrita** (ou **tipo 0**).

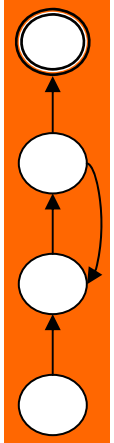
- Nesta gramática, nenhuma limitação é imposta.

Exemplo: A gramática G = (V, Σ, P, S), com Σ ={a,b,c}, V={S,A,C},

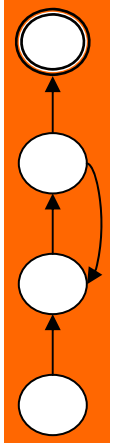
P={$S \rightarrow aAbc$, $A \rightarrow aAbC|\varepsilon$, $Cb \rightarrow bC$, $Cc \rightarrow cc$} é **irrestrita.**

$$L(G)=\{a^i b^i c^i \mid i>0\}$$

Como vc classifica a linguagem L(G)?

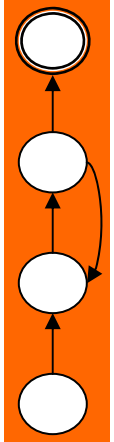


- Uma gramática regular é uma gramática livre de contexto.
- Uma gramática livre de contexto, sem produções do tipo $A \rightarrow \varepsilon$, é uma gramática sensível ao contexto.
- Uma gramática sensível ao contexto é uma gramática irrestrita.



Normalização de GLCs

- Para GLCs, é possível restringir a quantidade e forma das produções, sem reduzirmos seu poder gerador de linguagem. Isto garante propriedades interessantes:
  - Garantia de terminação do processo de *parsing* (análise sintática);
  - Facilitação da caracterização das linguagens geradas pela gramática.
- Passos preliminares para normalização:
  - Eliminação de Recursão sobre Símbolo Inicial S
  - Eliminação de Regras ε
  - Eliminação de Regras Encadeadas
  - Eliminação de Símbolos Inúteis



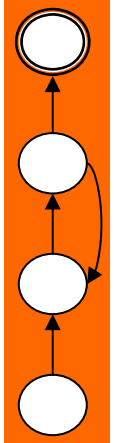
Eliminação de Recursão sobre S

- Força o símbolo inicial S a atuar apenas como um iniciador de derivações.

Seja G=(V,Σ,P,S) uma GLC. Existe uma GLC G' =(V',Σ,P',S') (e um algoritmo correspondente para produzí-la) que satisfaz:

i) L(G')=L(G).

ii) As regras de P' são da forma A$\rightarrow$w, onde A$\in$V' e w $\in((V-\{S'\})\cup \Sigma)^*$.



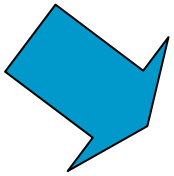
Eliminação de Recursão sobre S: Exemplo

G: $S \rightarrow aS \mid AB \mid AC$

$A \rightarrow aA \mid \varepsilon$

$B \rightarrow bB \mid bS$

$C \rightarrow cC \mid \varepsilon$



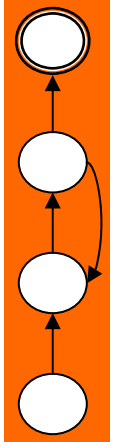
G': $S' \rightarrow S$

$S \rightarrow aS \mid AB \mid AC$

$A \rightarrow aA \mid \varepsilon$

$B \rightarrow bB \mid bS$

$C \rightarrow cC \mid \varepsilon$



Eliminação de Regras ε

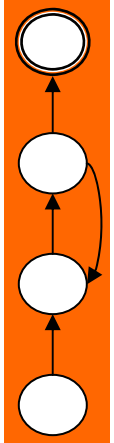
- Elimina regras que produzem cadeias nulas (a menos que seja parte da linguagem e a única regra ε seja $S \rightarrow \varepsilon$).

Seja $G=(V,\Sigma,P,S)$ uma GLC sem recursão sobre S. Existe uma GLC $G_\varepsilon=(V,\Sigma,P_\varepsilon,S)$ (e um algoritmo correspondente para produzí-la) que satisfaz:

i) $L(G_\varepsilon)=L(G)$.
ii) $A \rightarrow \varepsilon \in P_\varepsilon$ sss $\varepsilon \in L(G)$ e $A=S$.

Uma gramática G_ε satisfazendo estas condições é dita essencialmente não-contrátil.

Uma variável que pode produzir ε é dita anulável.



Eliminação de Regras ε: Por que?

G: $S \rightarrow SaB \mid aB$ $\qquad$ $L(G) = (a^+b^*)^+$
$\quad B \rightarrow bB \mid \varepsilon$

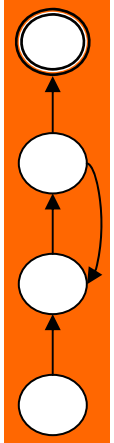
Derivação de aaaa: $S \Rightarrow SaB \Rightarrow SaBaB \Rightarrow SaBaBaB \Rightarrow aaBaBaB \Rightarrow$
$\quad aaaBaB \Rightarrow aaaaB \Rightarrow aaaa$

Ineficiente! Gera B's depois eliminados por aplicação de $B \rightarrow \varepsilon$ ...

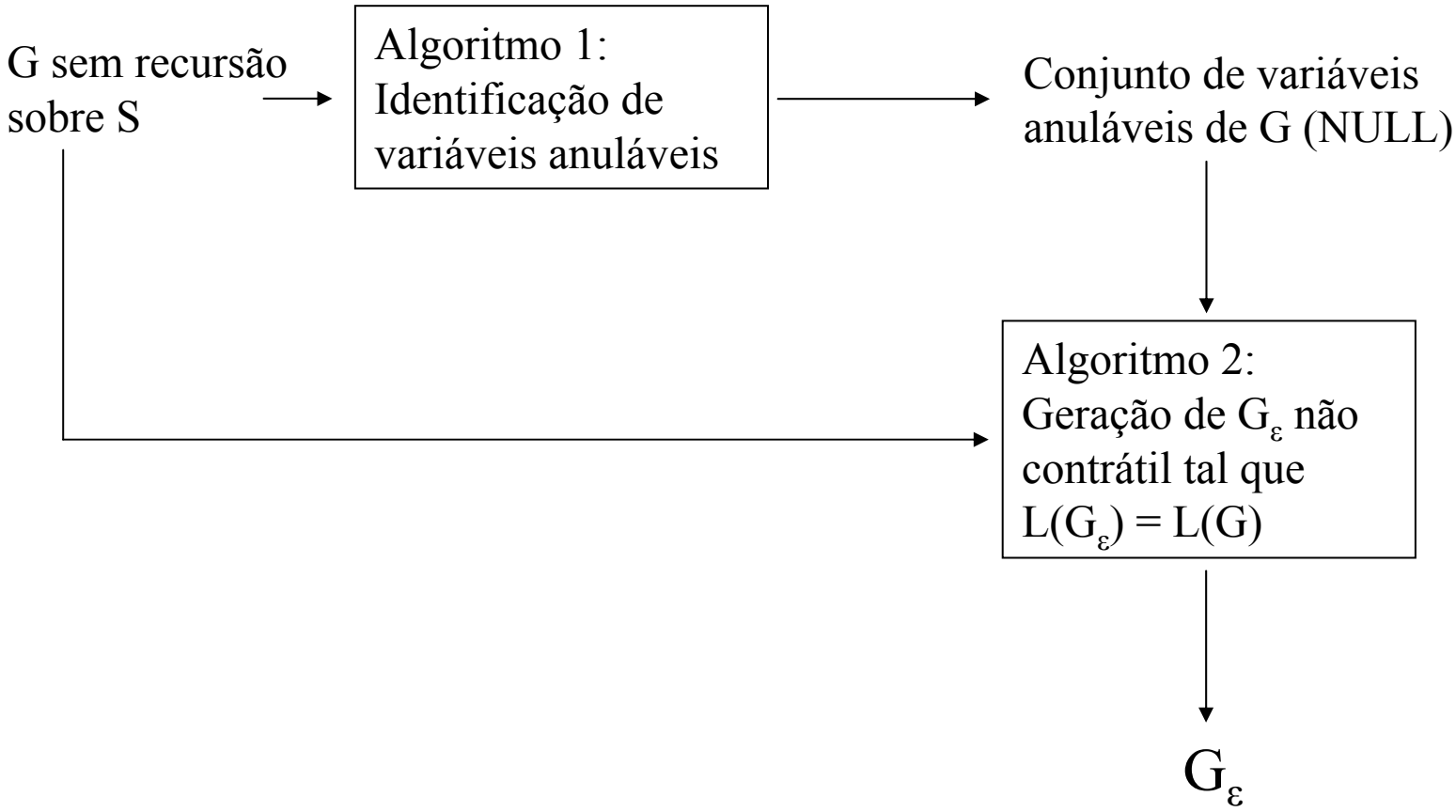
Compare com a seguinte gramática equivalente G':

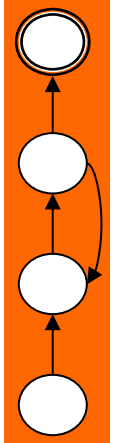
G': $S \rightarrow SaB \mid Sa \mid aB \mid a$ $\qquad$ $L(G) = (a^+b^*)^+$
$\quad B \rightarrow bB \mid b$

Derivação de aaaa: $S \Rightarrow \ldots$



Eliminação de Regras ε: Algoritmos





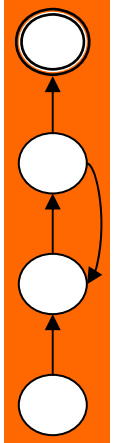
Algoritmo 1

Entrada: G = (V, Σ, P, S) GLC sem recursão sobre S.

1. NULL = $\{A \mid A \rightarrow \varepsilon \in P\}$
2. Repita
   PREV = NULL
   Para cada $A \in V$
   Se existe regra $A \rightarrow w$ com $w \in PREV^*$ então NULL = NULL $\cup$ {A}
   Até que NULL == PREV

G: $S \rightarrow ACA$
$A \rightarrow aAa \mid B \mid C$
$B \rightarrow bB \mid b$
$C \rightarrow cC \mid \varepsilon$

Iteração	NULL	PREV
0	{C}	
1	{A,C}	{C}
2	{S,A,C}	{A,C}
3	{S,A,C}	{S,A,C}



Algoritmo 2

Entrada: G = (V, Σ, P, S) GLC sem recursão sobre S, NULL

Saída: G_ε = (V, Σ, P_ε , S) GLC sem recursão sobre S e sem regras-ε.

1. Seja $A \rightarrow w \in P$. Se w puder ser escrito como $w_1A_1w_2A_2 \ldots w_kA_kw_{K+1}$ com $A_1, A_2, \ldots, A_k \subseteq$ NULL, então $A \rightarrow w_1w_2 \ldots w_kw_{k+1} \in P_\varepsilon$
2. Se $A \rightarrow \varepsilon \in P$ e $A \neq S$, então $A \rightarrow \varepsilon \notin P_\varepsilon$

Exemplo 1

G: $S \rightarrow ACA$
$A \rightarrow aAa \mid B \mid C$
$B \rightarrow bB \mid b$
$C \rightarrow cC \mid \varepsilon$

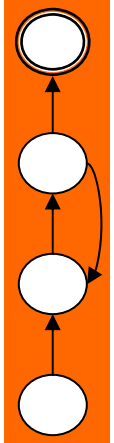
NULL = {S,A,C}

G_ε: $S \rightarrow ACA \mid AC \mid AA \mid CA \mid A \mid C \mid \varepsilon$
$A \rightarrow aAa \mid aa \mid B \mid C$
$B \rightarrow bB \mid b$
$C \rightarrow cC \mid c$

Exemplo 2

G: $S \rightarrow ABC$
$A \rightarrow aA \mid \varepsilon$
$B \rightarrow bB \mid \varepsilon$
$C \rightarrow cC \mid \varepsilon$

NULL = {S,A,B,C}



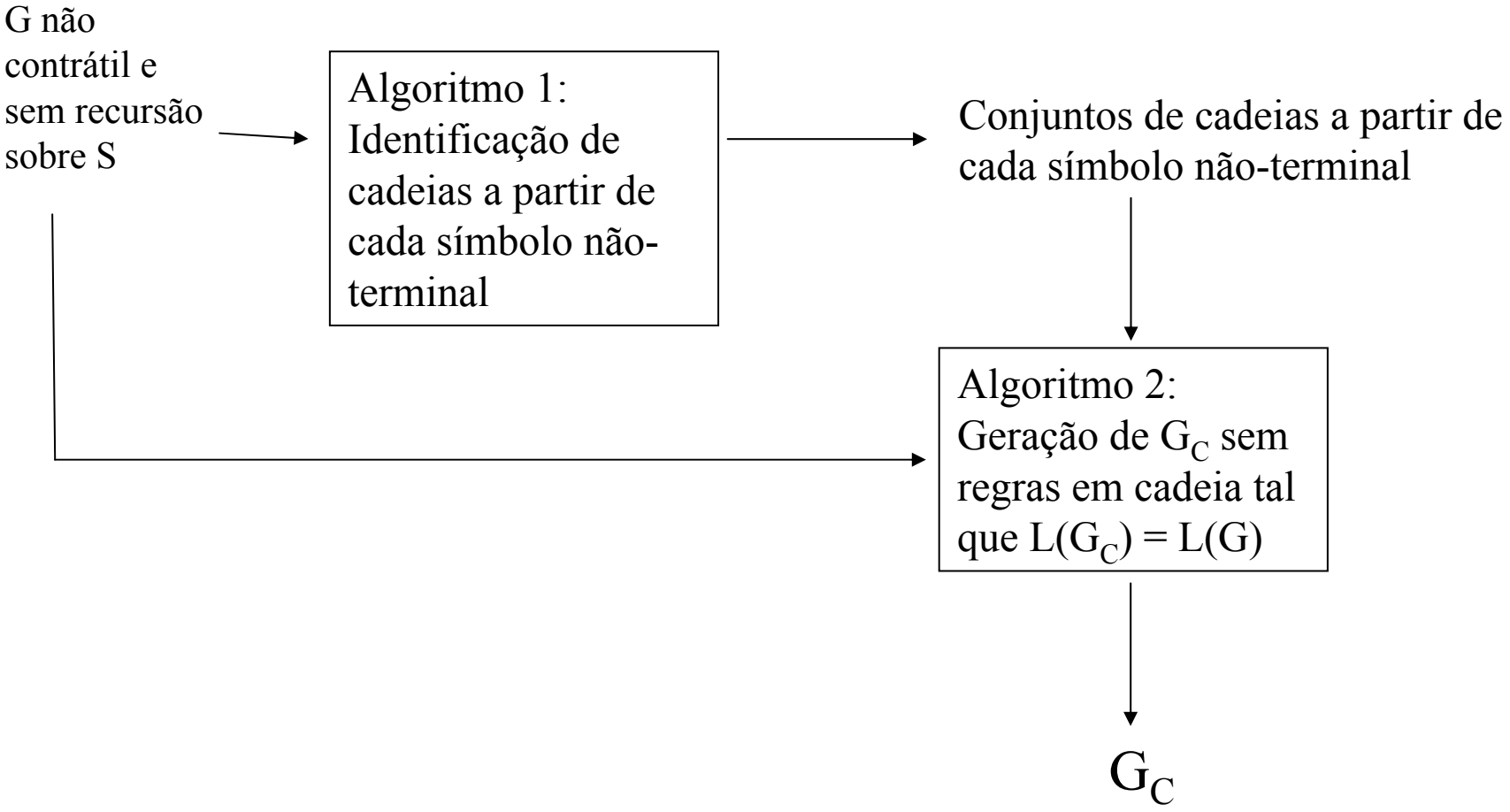
Eliminação de Regras em Cadeia

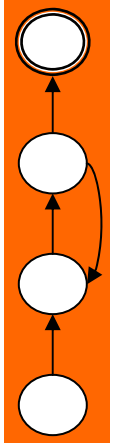
- Elimina regras do tipo $A \rightarrow B$, que não são nada mais do que uma renomeação de variáveis.

Seja $G=(V,\Sigma,P,S)$ uma GLC essencialmente não-contrátil e sem recursão sobre S. Existe uma GLC $G_C=(V,\Sigma,P_C,S)$ (e um algoritmo correspondente para produzí-la) que satisfaz:

i) $L(G_C)=L(G)$.
ii) G_C não tem regras em cadeia, é essencialmente não contrátil e não tem recursão sobre S.

Eliminação de Regras em Cadeia: Algoritmos





Algoritmo 1

Entradas: G = (V, Σ, P, S) GLC não-contrátil e sem recursão sobre S,
Símbolo não-terminal A.

1. CHAIN(A) = {A}; PREV = ∅
2. Repita
   NEW = CHAIN – PREV; PREV = CHAIN(A)
   Para cada B ∈ NEW
   Para cada regra B → C CHAIN(A) = CHAIN(A) ∪ {C}
   Até que CHAIN(A) == PREV

G: S → ACA|CA|AA|AC|A|C|ε
A → aAa|aa|B|C
B → bB|b
C → cC|c

Iteração	CHAIN(S)	NEW	PREV
0	{S}		∅
1	{S,A,C}	{S]	{S]
2	{S,A,B,C}	{A,C}	{S,A,C}
3	{S,A,B,C}	{B}	{S,A,B,C}

CHAIN(S)={S,A,B,C}. Similarmente: CHAIN(A)={A,B,C}, CHAIN(B)={B}, CHAIN(C)={C}

Algoritmo 2

Entrada: G = (V, Σ, P, S) GLC não-contrátil sem recursão sobre S, CHAIN(V)

Saída: G_C = (V, Σ, P_C, S) GLC sem recursão sobre S, sem regras-ε e sem regras em cadeia

Para cada A ∈ V

Para cada B ∈ CHAIN(A) tal que B → w ∈ P e w ∉ V

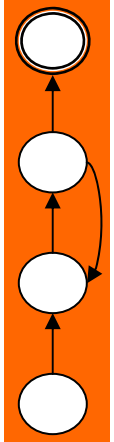
Faça A → w ∈ P_C e A → B ∉ P_C

Exemplo

G_ε: S → ACA | AC | AA | CA | A | C | ε
A → aAa | aa | B | C
B → bB | b
C → cC | c

CHAIN(S) = {S,A,B,C}
CHAIN(A) = {A,B,C}
CHAIN(B) = {B}
CHAIN(C) = {C}

G_C: S → ACA | AC | AA|CA|
aAa | aa| bB | b | cC|c|ε
A → aAa | aa | bB|b|cC|c
B → bB | b
C → cC | c

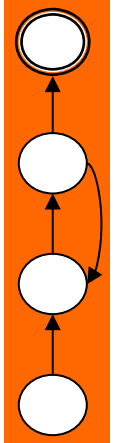


Eliminação de Símbolos Inúteis

- Elimina variáveis que não contribuem para a geração de cadeias da linguagem gerada pela gramática.
- Def.: Seja G uma GLC. Um símbolo $x \in (V \cup \Sigma)$ é útil se existir derivação $S \overset{*}{\Rightarrow} uxv \overset{*}{\Rightarrow} w$, *onde* $u, v \in (V \cup \Sigma)^*$ e $w \in \Sigma^*$. Caso contrário, o símbolo é dito inútil.

Seja $G=(V,\Sigma,P,S)$ uma GLC. Existe uma GLC $G_U=(V_U,\Sigma,P_U,S)$ (e um algoritmo correspondente para produzí-la) que satisfaz:

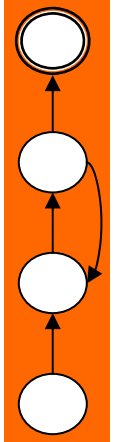
i) $L(G_U)=L(G)$.
ii) G_U não tem símbolos inúteis



Eliminação de Símbolos Inúteis: Algoritmos

G não contrátil e sem regras em cadeia → Algoritmo 1: Determinação de variáveis que produzem sentenças e eliminação de demais variáveis e regras correspondentes → Algoritmo 2: Determinação de variáveis que podem ser alcançadas a partir de S e eliminação de demais variáveis e regras correspondentes → G' não contrátil, sem regras em cadeia e sem símbolos inúteis

Algoritmos e exemplos.



Formas Normais de Chomsky e Greibach

- *Def.:* Uma GLC G está na forma normal de Chomsky se cada regra estiver em uma das seguintes formas: $A \rightarrow BC$, $A \rightarrow a$, $A \rightarrow \varepsilon$
- *Def.:* Uma GLC G está na forma normal de Greibach se cada regra estiver em uma das seguintes formas: $A \rightarrow aA_1A_2 \ldots A_n$, , $A \rightarrow a$, $A \rightarrow \varepsilon$

Seja G=(V,Σ,P,S) uma GLC. Existem GLCs G' e G'' (e algoritmos correspondente para produzí-las) que satisfazem:

i) L(G'')=L(G')=L(G).
ii) G' está na forma normal de Chomsky.
iii) G" está na forma normal de Greibach

Exemplos.