

# Indecidibilidade do problema da vacuidade da interseção de LLCs

Thiago Lages de Alencar

# Gramática

$$G = (V, T, P, S)$$

V são as variáveis ou não terminais

T são terminais

P são as regras de produção

S é o símbolo inicial ou variável inicial

# Gramática Livre do Contexto

$$G = (V, T, P, S)$$

$V$  = variáveis ou não terminais

$T$  = terminais

$P = (V \cup T)^* =$  qualquer combinação entre terminais e não terminais

$S$  = símbolo inicial ou variável inicial

# Linguagem Livre do Contexto

$$L(G) = \{\mathbf{w} \in T^* \mid S \Rightarrow^+ \mathbf{w}\}$$

$V$  = variáveis ou não terminais

$T$  = terminais

$P = (V \cup T)^*$  = qualquer combinação entre terminais e não terminais

$S$  = símbolo inicial ou variável inicial

# Linguagem Livre do Contexto

$$L(G) = \{\mathbf{w} \in T^* \mid S \Rightarrow^+ \mathbf{w}\}$$

$V$  = variáveis ou não terminais

$T$  = terminais

$P = (V \cup T)^*$  = qualquer combinação entre terminais e não terminais

$S$  = símbolo inicial ou variável inicial

$w$  é uma string

# Composição

$$w \in L(G)$$

$G$  é uma gramática livre de contexto

$w$  é uma string

Dada uma string  $w$  e uma gramática livre de contexto  $G$ ,  $w$  pertence a linguagem gerada por aquela gramática?

# Composição

$$\mathbf{w} \in L(G)$$

$G$  é uma gramática livre de contexto

$\mathbf{w}$  é uma string

Dada uma string  $\mathbf{w}$  e uma gramática livre de contexto  $G$ ,  $\mathbf{w}$  pertence a linguagem gerada por aquela gramática?

Decidível

# Composição - Algoritmo

Entradas:

- $G$  é uma gramática livre de contexto
- $w$  é uma string

Etapa 1: Forma Normal de Chomsky

Etapa 2: Gerar todas derivações com tamanho  $2N - 1$

Etapa 3: Procurar  $w$  entre todas as derivações



# Composição - Algoritmo - Forma Normal de Chomsky

Etapa 1:

- Transformar para forma normal de Chomsky

$$S \rightarrow \varepsilon$$

$$A \rightarrow BC$$

$$A \rightarrow a$$

# Composição - Algoritmo - Forma Normal de Chomsky

Exemplo:

$$S \rightarrow \varepsilon \mid A$$
$$A \rightarrow CB \mid a$$
$$B \rightarrow b$$
$$C \rightarrow AB$$
$$S \Rightarrow A \Rightarrow CB \Rightarrow ABB \Rightarrow AB BBB$$

4 passos

$$AB BBB \Rightarrow aB BBB \Rightarrow ab BBB \Rightarrow abb BB \Rightarrow abbb B \Rightarrow abbbb$$

5 passos

Uma string de 5 letras custou 9 passos

# Composição - Algoritmo - Forma Normal de Chomsky

Dada um string **w** com tamanho **N**, você precisará de  $2*N - 1$  passos para encontrar ela.

$S \Rightarrow A \Rightarrow CB \Rightarrow ABB \Rightarrow ABBBB$

4 passos

$ABBBB \Rightarrow aBBBB \Rightarrow abBBB \Rightarrow abbBB \Rightarrow abbbB \Rightarrow abbbb$

5 passos

Total: 9 passos

# Composição - Algoritmo

Etapa 2:

Gerar todas derivações com tamanho  $2 \cdot N - 1$ .

Etapa 3:

Se alguma delas for  $w$  então  $w \in L(G)$ , caso contrário não.

Sem saber o máximo de passos para encontrar  $w$  o algoritmo teria que gerar todas derivações possíveis e isso poderia entrar em loop.

# Finitude

$L(G)$  é finito

$G$  é uma gramática livre de contexto

Dada uma gramática livre de contexto  $G$ , a linguagem gerada por aquela gramática é finita?

# Finitude

$L(G)$  é finito

$G$  é uma gramática livre de contexto

Dada uma gramática livre de contexto  $G$ , a linguagem gerada por aquela gramática é finita?

Decidível

# Finitude - Algoritmo

Entradas:

- $G$  é uma gramática livre de contexto

Etapa 1: Forma Normal de Chomsky

Etapa 2: Contar o número de não terminais ( $N$ )

Etapa 3: Verificar se existe alguma árvore derivação com o tamanho maior ou igual a  $N + 1$

# Finitude - Algoritmo

Exemplo:

$S \rightarrow \varepsilon \mid A$

$A \rightarrow CB \mid a$

$B \rightarrow b$

$C \rightarrow AB$



# Finitude - Algoritmo

Etapa 2:

$S \rightarrow \varepsilon \mid A$

$A \rightarrow CB \mid a$

$B \rightarrow b$

$C \rightarrow AB$

Número de não terminais: 4

# Finitude - Algoritmo

Etapa 3:

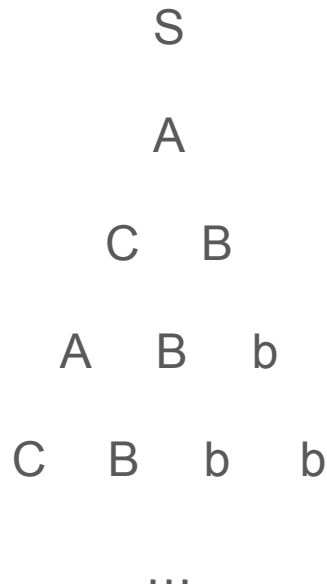
$S \rightarrow \varepsilon \mid A$

$A \rightarrow CB \mid a$

$B \rightarrow b$

$C \rightarrow AB$

Número de não terminais: **4**



Tamanho  $\geq 4 + 1$

# Finitude - Algoritmo

Exemplo 2:

$$S \rightarrow \varepsilon \mid A$$
$$A \rightarrow CB \mid a$$
$$B \rightarrow b$$
$$C \rightarrow DB$$
$$D \rightarrow d$$

# Finitude - Algoritmo

Etapa 2:

$S \rightarrow \varepsilon \mid A$

$A \rightarrow CB \mid a$

$B \rightarrow b$

$C \rightarrow DB$

$D \rightarrow d$

Número de não terminais: 5

# Finitude - Algoritmo

Etapa 3:

$S \rightarrow \varepsilon \mid A$

$A \rightarrow CB \mid a$

$B \rightarrow b$

$C \rightarrow DB$

$D \rightarrow d$

Número de não terminais: **5**

S  
A  
C B  
D B b  
d b b

Tamanho  $\geq$  **5** + 1

# Vacuidade

$$L(G) = \emptyset$$

G é uma gramática livre de contexto

Dada uma gramática livre de contexto G, a linguagem gerada por aquela gramática é vazia?

# Vacuidade

$$L(G) = \emptyset$$

G é uma gramática livre de contexto

Dada uma gramática livre de contexto G, a linguagem gerada por aquela gramática é vazia?

Decidível

# Vacuidade

Exemplo 1:

$S \rightarrow \varepsilon$

$S \Rightarrow \varepsilon$

Exemplo 2:

$S \rightarrow \varepsilon \mid A$

$A \rightarrow A \mid aA$

$S \Rightarrow \varepsilon$

$S \Rightarrow A \Rightarrow aA \Rightarrow aaA \Rightarrow aaaA \Rightarrow \dots$

É necessário um algoritmo que dê a resposta certa e sempre pare.

Dada uma gramática livre de contexto  $G$ , a linguagem gerada por aquela gramática é vazia?

Em outras palavras: Gramática  $G$  não gera nenhuma palavra?



# Vacuidade - Algoritmo

Entrada:

- $G$  é uma gramática livre de contexto

Etapa 1: Marcar todos os terminais no lado direito

Etapa 2: Marcar todos os não terminais que tem o lado direito todo marcado  
(repetir etapa 2 enquanto conseguir marcar algum não terminal)

Etapa 3: Verificar se  $S$  foi marcado

# Vacuidade - Algoritmo

Exemplo 1:

$S \rightarrow \varepsilon$

$S \rightarrow ABCD$

$A \rightarrow BB$

$B \rightarrow AC$

$C \rightarrow CD$

$C \rightarrow zzz$

$D \rightarrow dC$

# Vacuidade - Algoritmo

Etapa 1 - Marcar todos os terminais no lado direito:

$S \rightarrow \varepsilon$

$S \rightarrow ABCD$

$A \rightarrow BB$

$B \rightarrow AC$

$C \rightarrow CD$

$C \rightarrow zzz$

$D \rightarrow dC$

# Vacuidade - Algoritmo

Etapa 1 - Marcar todos os terminais no lado direito:

$S \rightarrow \epsilon$

$S \rightarrow ABCD$

$A \rightarrow BB$

$B \rightarrow AC$

$C \rightarrow CD$

$C \rightarrow \text{zzz}$

$D \rightarrow \text{dC}$

# Vacuidade - Algoritmo

Etapa 2 - Marcar todos os não terminais que tem o lado direito todo marcado:

$S \rightarrow \epsilon$

$S \rightarrow ABCD$

$A \rightarrow BB$

$B \rightarrow AC$

$C \rightarrow CD$

$C \rightarrow \text{zzz}$

$D \rightarrow \text{dC}$

# Vacuidade - Algoritmo

Etapa 2 - Marcar todos os não terminais que tem o lado direito todo marcado:

$S \rightarrow \varepsilon$

$S \rightarrow ABCD$

$A \rightarrow BB$

$B \rightarrow AC$

$C \rightarrow CD$

$C \rightarrow zzz$

$D \rightarrow dC$

# Vacuidade - Algoritmo

Etapa 2 - Marcar todos os não terminais que tem o lado direito todo marcado:

$S \rightarrow \epsilon$

$S \rightarrow ABCD$

$A \rightarrow BB$

$B \rightarrow AC$

$C \rightarrow CD$

$C \rightarrow zzz$

$D \rightarrow dC$

(repetir etapa 2 enquanto conseguir marcar algum não terminal)

# Vacuidade - Algoritmo

Etapa 2 - Marcar todos os não terminais que tem o lado direito todo marcado:

$S \rightarrow \epsilon$

$S \rightarrow ABCD$

$A \rightarrow BB$

$B \rightarrow AC$

$C \rightarrow CD$

$C \rightarrow zzz$

$D \rightarrow dC$

(repetir etapa 2 enquanto conseguir marcar algum não terminal)



# Vacuidade - Algoritmo

Etapa 3 - Verificar se S foi marcado:

$S \rightarrow \epsilon$

$S \rightarrow ABCD$

$A \rightarrow BB$

$B \rightarrow AC$

$C \rightarrow CD$

$C \rightarrow zzz$

$D \rightarrow dC$

Como S não foi marcado, não é possível gerar uma palavra nessa gramática.  
Em outras palavras, essa linguagem é vazia.

# Vacuidade - Algoritmo

Exemplo 2:

$S \rightarrow DABC$

$A \rightarrow BB$

$B \rightarrow AC$

$B \rightarrow bbC$

$C \rightarrow CD$

$C \rightarrow xy$

$D \rightarrow Cd$

# Vacuidade - Algoritmo

Etapa 1 - Marcar todos os terminais no lado direito:

$S \rightarrow DABC$

$A \rightarrow BB$

$B \rightarrow AC$

$B \rightarrow bbC$

$C \rightarrow CD$

$C \rightarrow xy$

$D \rightarrow Cd$

# Vacuidade - Algoritmo

Etapa 2 - Marcar todos os não terminais que tem o lado direito todo marcado:

$S \rightarrow DABC$

$A \rightarrow BB$

$B \rightarrow AC$

$B \rightarrow bbC$

$C \rightarrow CD$

$C \rightarrow xy$

$D \rightarrow Cd$

# Vacuidade - Algoritmo

Etapa 2 - Marcar todos os não terminais que tem o lado direito todo marcado:

$S \rightarrow DABC$

$A \rightarrow BB$

$B \rightarrow AC$

$B \rightarrow bbC$

$C \rightarrow CD$

$C \rightarrow xy$

$D \rightarrow Cd$

# Vacuidade - Algoritmo

Etapa 2 - Marcar todos os não terminais que tem o lado direito todo marcado:

$S \rightarrow DABC$

$A \rightarrow BB$

$B \rightarrow AC$

$B \rightarrow bbC$

$C \rightarrow CD$

$C \rightarrow xy$

$D \rightarrow Cd$

# Vacuidade - Algoritmo

Etapa 2 - Marcar todos os não terminais que tem o lado direito todo marcado:

S → DABC

A → BB

B → AC

B → bbC

C → CD

C → xy

D → Cd

# Vacuidade - Algoritmo

Etapa 2 - Marcar todos os não terminais que tem o lado direito todo marcado:

S → DABC

A → BB

B → AC

B → bbC

C → CD

C → xy

D → Cd



# Vacuidade - Algoritmo

Etapa 3 - Verificar se S foi marcado:

S → DABC

A → BB

B → AC

B → bbC

C → CD

C → xy

D → Cd

Como S foi marcado, essa gramática consegue gerar uma palavra.

Exemplo:  $DABC \Rightarrow DABxy \Rightarrow DAbbCxy \Rightarrow DAbbxyxy \Rightarrow DBBbbxyxy \Rightarrow DBbbCbbxyxy \Rightarrow DBbbxybbxyxy \Rightarrow DbbCbbxybbxyxy \Rightarrow Dbbxybbxybbxyxy \Rightarrow Cdbbxybbxybbxyxy \Rightarrow xydbbxybbxybbxyxy$

# Post Correspondence Problem

- $A$  é um alfabeto com pelo menos 2 símbolos
- Existe duas listas: lista  $\alpha$  e lista  $\beta$
- Ambas as listas são listas de strings
- Ambas as listas tem tamanho  $N$

$\alpha_1$	$\alpha_2$	$\alpha_3$
$\beta_1$	$\beta_2$	$\beta_3$

# Post Correspondence Problem

$\alpha_1$	$\alpha_2$	$\alpha_3$
a	ab	bba

$\beta_1$	$\beta_2$	$\beta_3$
baa	aa	bb

# Post Correspondence Problem

$\alpha_1$	$\alpha_2$	$\alpha_3$
a	ab	bba

$\beta_1$	$\beta_2$	$\beta_3$
baa	aa	bb

$$\alpha_3 \alpha_2 \alpha_3 \alpha_1 = \beta_3 \beta_2 \beta_3 \beta_1$$

Sequência: 3,2,3,1

# Post Correspondence Problem

$\alpha_1$	$\alpha_2$	$\alpha_3$
a	ab	bba

$\beta_1$	$\beta_2$	$\beta_3$
baa	aa	bb

$$\alpha_3 \alpha_2 \alpha_3 \alpha_1 = \beta_3 \beta_2 \beta_3 \beta_1$$

bba ab bba a = bb aa bb baa

bbaabbbbaa = bbaabbbbaa

Sequência: 3,2,3,1

# Post Correspondence Problem

$\alpha_1$	$\alpha_2$	$\alpha_3$
a	ab	bba

$\beta_1$	$\beta_2$	$\beta_3$
baa	aa	bb

$$\alpha_3 \alpha_2 \alpha_3 \alpha_1 = \beta_3 \beta_2 \beta_3 \beta_1$$

$$\text{bba ab bba a} = \text{bb aa bb baa}$$

$$\text{bbaabbbbaa} = \text{bbaabbbbaa}$$

O problema de decisão é:  
Existe tal solução ou não

Sequência: 3,2,3,1

# Post Correspondence Problem

$\alpha_1$	$\alpha_2$	$\alpha_3$
a	ab	bba

$\beta_1$	$\beta_2$	$\beta_3$
baa	aa	bb

$$\alpha_3 \alpha_2 \alpha_3 \alpha_1 = \beta_3 \beta_2 \beta_3 \beta_1$$

$$\text{bba ab bba a} = \text{bb aa bb baa}$$

$$\text{bbaabbbbaa} = \text{bbaabbbbaa}$$

O problema de decisão é:  
Existe tal solução ou não

Indecidível

Sequência: 3,2,3,1

# Post Correspondence Problem para LLC

Podemos construir uma gramática livre de contexto para o Post Correspondence Problem:

$$S \rightarrow A \mid B$$

$$A \rightarrow \alpha_1 A x_1 \mid \alpha_2 A x_2 \mid \alpha_3 A x_3 \mid \dots \mid \alpha_N A x_N \mid$$

$$\alpha_1 x_1 \mid \alpha_2 x_2 \mid \alpha_3 x_3 \mid \dots \mid \alpha_N x_N \mid$$

$$B \rightarrow \beta_1 B x_1 \mid \beta_2 B x_2 \mid \beta_3 B x_3 \mid \dots \mid \beta_N B x_N \mid$$

$$\beta_1 x_1 \mid \beta_2 x_2 \mid \beta_3 x_3 \mid \dots \mid \beta_N x_N \mid$$



# Post Correspondence Problem para LLC

$$S \rightarrow A \mid B$$

$$A \rightarrow \alpha_1 A x_1 \mid \alpha_2 A x_2 \mid \alpha_3 A x_3 \mid \dots \mid \alpha_N A x_N \mid$$

$$\alpha_1 x_1 \mid \alpha_2 x_2 \mid \alpha_3 x_3 \mid \dots \mid \alpha_N x_N \mid$$

$$B \rightarrow \beta_1 B x_1 \mid \beta_2 B x_2 \mid \beta_3 B x_3 \mid \dots \mid \beta_N B x_N \mid$$

$$\beta_1 x_1 \mid \beta_2 x_2 \mid \beta_3 x_3 \mid \dots \mid \beta_N x_N \mid$$

x representa a escolha do par correspondente  $\alpha_i$  ou  $\beta_i$ .

# Post Correspondence Problem para LLC

$$S \rightarrow A \mid B$$

$$A \rightarrow \alpha_1 A x_1 \mid \alpha_2 A x_2 \mid \alpha_3 A x_3 \mid \dots \mid \alpha_N A x_N \mid$$

$$\alpha_1 x_1 \mid \alpha_2 x_2 \mid \alpha_3 x_3 \mid \dots \mid \alpha_N x_N \mid$$

$$B \rightarrow \beta_1 B x_1 \mid \beta_2 B x_2 \mid \beta_3 B x_3 \mid \dots \mid \beta_N B x_N \mid$$

$$\beta_1 x_1 \mid \beta_2 x_2 \mid \beta_3 x_3 \mid \dots \mid \beta_N x_N \mid$$

$L_A$  é a linguagem gerada pela lista A

$L_B$  é a linguagem gerada pela lista B

# Ambiguidade

Sabe se uma gramática livre de contexto é ambígua ou não é indecidível.

Suponha que 1, 2, 3 é uma solução do PCP na forma LLC

$$S \Rightarrow A \Rightarrow \alpha_1 A x_1 \Rightarrow \alpha_1 \alpha_2 A x_2 x_1 \Rightarrow \alpha_1 \alpha_2 \alpha_3 x_3 x_2 x_1$$

$$S \Rightarrow B \Rightarrow \beta_1 B x_1 \Rightarrow \beta_1 \beta_2 B x_2 x_1 \Rightarrow \beta_1 \beta_2 \beta_3 x_3 x_2 x_1$$

Já que 1, 2, 3 é uma solução, nós podemos dizer que  $\alpha_1 \alpha_2 \alpha_3 = \beta_1 \beta_2 \beta_3$

Como só é possível ter uma derivação em A e uma em B, a única maneira de acabarem sendo iguais é ambas estarem fazendo derivação da mesma string.

# Complemento

Dada a  $L_A$  como linguagem da lista A,  $L_A^C$  é uma linguagem livre de contexto.

Diferente de  $L_A$ , não é fácil criar uma gramática para  $L_A^C$ ,  
mas podemos fazer o design de um autômato com pilha para  $L_A^C$ .

# Complemento

Dada a  $L_A$  como linguagem da lista A,  $L_A^C$  é uma linguagem livre de contexto.

Diferente de  $L_A$ , não é fácil criar uma gramática para  $L_A^C$ ,  
mas podemos fazer o design de um autômato com pilha para  $L_A^C$ .

...

# Indecidibilidade

Nós podemos usar  $L_A$ ,  $L_B$  e seus complementos de diversas maneiras para demonstrar indecidibilidade

Muitos problema indecidíveis envolvem reduzir o problema ao de PCP

$$L(G_1) \cap L(G_2) = \emptyset ?$$

$$L(G_1) = L(G_2) ?$$

$$L(G_1) \subseteq L(G_2) ?$$

$$L(G_1) = \Sigma^* ?$$

# Vacuidade da interseção de LLCs

$$L(G_1) \cap L(G_2) = \emptyset ?$$

Considerando

$$L(G_1) = L_A$$

$$L(G_2) = L_B$$

Então  $L(G_1) \cap L(G_2)$  é o conjunto de soluções do problema PCP

A interseção é vazia apenas se não existe solução para o problema PCP

Mostrar que o complemento de um problema é indecidível equivale a mostrar que o problema é indecidível.

# Vacuidade da interseção de LLCs

$$L_A \cap L_B = (L_A^c \cup L_B^c)^c$$



# Vacuidade da interseção de LLCs

$$L_A \cap L_B = (L_A^c \cup L_B^c)^c$$

$$L_A = \emptyset ?$$

# Vacuidade da interseção de LLCs

$$L_A \cap L_B = (L_A^c \cup L_B^c)^c$$

$$L_A = \emptyset ?$$

$$L_A \cup L_B \text{ é LLC}$$

# Vacuidade da interseção de LLCs

$$L_A \cap L_B = (L_A^C \cup L_B^C)^C$$

$$L_A = \emptyset ?$$

$$L_A \cup L_B \text{ é LLC}$$

Como existe prova que  $L_A^C$  e  $L_B^C$  são LLC

# Vacuidade da interseção de LLCs

$$L_A \cap L_B = (L_A^C \cup L_B^C)^C$$

$$L_A = \emptyset ?$$

$$L_A \cup L_B \text{ é LLC}$$

Como existe prova que  $L_A^C$  e  $L_B^C$  são LLC

$$L_A^C \cup L_B^C \text{ é LLC}$$

# Vacuidade da interseção de LLCs

$$L_A \cap L_B = (L_A^C \cup L_B^C)^C$$

$$L_A = \emptyset ?$$

$$L_A \cup L_B \text{ é LLC}$$

Como existe prova que  $L_A^C$  e  $L_B^C$  são LLC

$$L_A^C \cup L_B^C \text{ é LLC}$$

$$L_A^C \cup L_B^C = \emptyset ?$$

# Vacuidade da interseção de LLCs

$$L_A \cap L_B = (L_A^C \cup L_B^C)^C$$

$$L_A = \emptyset ?$$

$$L_A \cup L_B \text{ é LLC}$$

Como existe prova que  $L_A^C$  e  $L_B^C$  são LLC

$$L_A^C \cup L_B^C \text{ é LLC}$$

$$L_A^C \cup L_B^C = \emptyset ? \text{ Decidível}$$

# Vacuidade da interseção de LLCs

$$L_A \cap L_B = (L_A^C \cup L_B^C)^C$$

$$L_A = \emptyset ?$$

$$L_A \cup L_B \text{ é LLC}$$

Como existe prova que  $L_A^C$  e  $L_B^C$  são LLC

$$L_A^C \cup L_B^C \text{ é LLC}$$

$$L_A^C \cup L_B^C = \emptyset ? \text{ Decidível}$$

$$(L_A^C \cup L_B^C)^C \text{ Indecidível}$$

**FIM**