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

w é uma string

Dada uma string **w** e uma gramática livre de contexto G, **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 → ε | A
A → CB | a
B → b
C → AB

S ⇒ A ⇒ CB ⇒ ABB ⇒ ABBBB
4 passos

ABBBB ⇒ aBBBB ⇒ abBBB ⇒ abbBB ⇒ abbbB ⇒ 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 ⇒ A ⇒ CB ⇒ ABB ⇒ ABBBB

4 passos

ABBBB ⇒ aBBBB ⇒ abBBB ⇒ abbBB ⇒ abbbB ⇒ abbbb

5 passos

Total: 9 passos

Composição - Algoritmo

Etapa 2:

Gerar todas derivações com tamanho 2***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**

S

A

C B

A B b

C B b b

…

Tamanho >= **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 → ε | A

A → CB | a

B → b

C → DB

D → d

Número de não terminais: **5**

S

A

C B

D B b

d b b

Tamanho >= **5** + 1

Vacuidade

$$L(G) = \varnothing$$

G é uma gramática livre de contexto

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

Vacuidade

$$L(G) = \varnothing$$

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:
$S \rightarrow \varepsilon \mid A$
$A \rightarrow A \mid aA$

$S \Rightarrow \varepsilon$
$S \Rightarrow A \Rightarrow aA \Rightarrow aaA \Rightarrow aaaA \Rightarrow ...$

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 → ε

S → ABCD

A → BB

B → AC

C → CD

C → zzz

D → dC

Vacuidade - Algoritmo

Etapa 1 - Marcar todos os terminais no lado direito:

S → ε

S → ABCD

A → BB

B → AC

C → CD

C → zzz

D → dC

Vacuidade - Algoritmo

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

S → ε

S → ABCD

A → BB

B → AC

C → CD

C → zzz

D → dC

Vacuidade - Algoritmo

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

S → ε

S → ABCD

A → BB

B → AC

C → CD

C → zzz

D → dC

Vacuidade - Algoritmo

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

S → ε

S → ABCD

A → BB

B → AC

C → CD

C → zzz

D → 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 → ε

S → ABCD

A → BB

B → AC

C → CD

C → zzz

D → dC

(repetir etapa 2 enquanto conseguir marcar algum não terminal)

Vacuidade - Algoritmo

Etapa 3 - Verificar se S foi marcado:

S → ε

S → ABCD

A → BB

B → AC

C → CD

C → zzz

D → 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 → DABC

A → BB

B → AC

B → bbC

C → CD

C → xy

D → Cd

Vacuidade - Algoritmo

Etapa 1 - Marcar todos os terminais no lado direito:

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 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 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 ⇒ DABxy ⇒ DAbbCxy ⇒ DAbbxyxy ⇒ DBBbbxyxy ⇒ DBbbCbbxyxy ⇒ DBbbxybbxyxy ⇒ DbbCbbxybbxyxy ⇒ Dbbxybbxybbxyxy ⇒ Cdbbxybbxybbxyxy ⇒ xydbbxybbxybbxyxy

Post Correspondence Problem

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

α_1	α_2	α_3

β_1	β_2	β_3

Post Correspondence Problem

α_1	α_2	α_3
a	ab	bba

β_1	β_2	β_3
baa	aa	bb

Post Correspondence Problem

α_1	α_2	α_3
a	ab	bba

β_1	β_2	β_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

α_1	α_2	α_3
a	ab	bba

β_1	β_2	β_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

bbaabbbaa = bbaabbbaa

Sequência: 3,2,3,1

Post Correspondence Problem

α_1	α_2	α_3
a	ab	bba

β_1	β_2	β_3
baa	aa	bb

O problema de decisão é:
Existe tal solução ou não

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

bba ab bba a = bb aa bb baa

bbaabbbaa = bbaabbbaa

Sequência: 3,2,3,1

Post Correspondence Problem

α_1	α_2	α_3
a	ab	bba

β_1	β_2	β_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

bbaabbbaa = bbaabbbaa

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:

$S \rightarrow A \mid B$

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

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

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

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

Post Correspondence Problem para LLC

$S \rightarrow A \mid B$

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

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

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

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

x é o par solução de α em A ou o par solução em β em B

Post Correspondence Problem para LLC

$S \rightarrow A \mid B$

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

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

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

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

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.

Indecidibilidade

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

Cada uma das provas seguintes envolve reduzir o problema ao de PCP

$L(G_1) \cap L(G_2) = \varnothing$

$L(G_1) = L(G_2)$

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

$L(G_1) = \sum^*$

Vacuidade da interseção de LLCs

$L(G_1) \cap L(G_2) = \varnothing$

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.

FIM