

Linguagens Formais e Autômatos




Aula 27 - Problemas indecidíveis

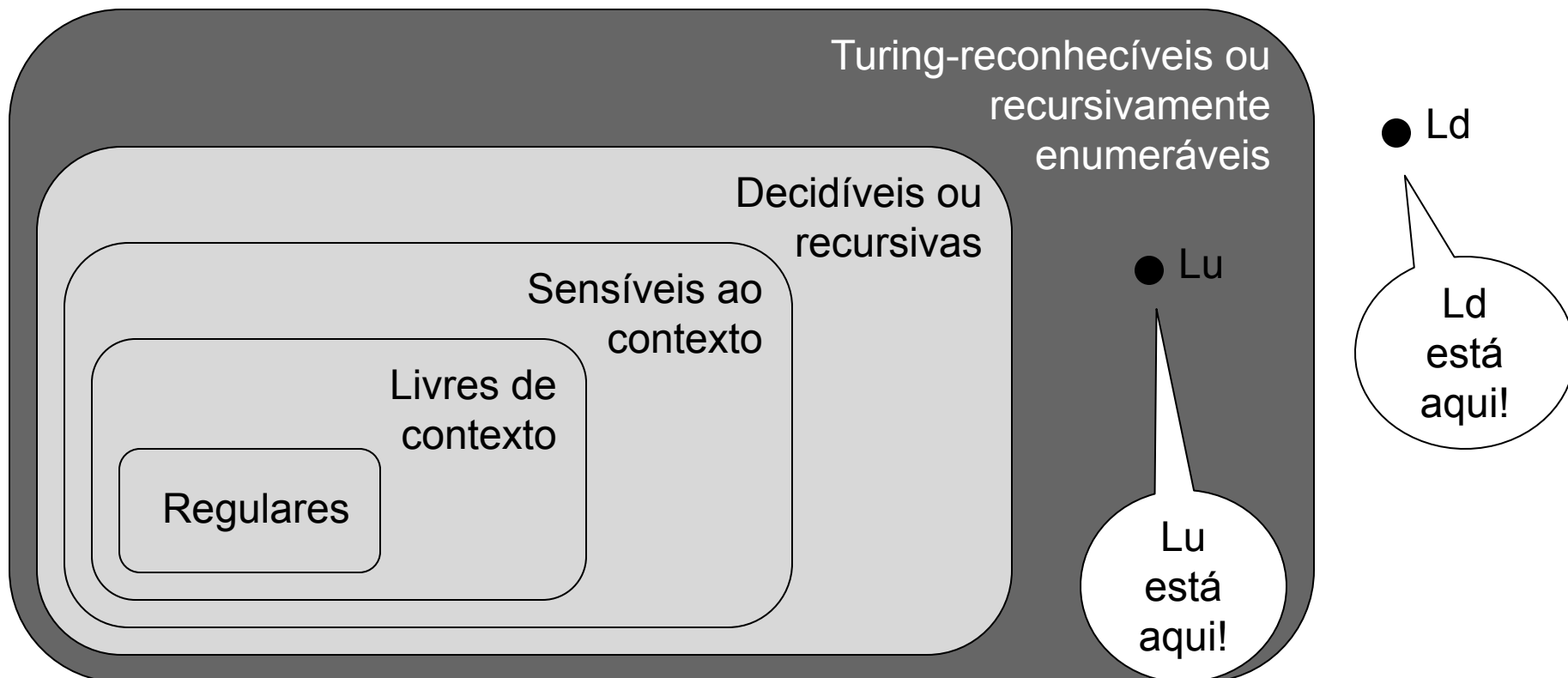
Prof. Dr. Daniel Lucrédio
Departamento de Computação / UFSCar
Última revisão: ago/2015

Referências bibliográficas

- **Introdução à teoria dos autômatos, linguagens e computação / John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman** ; tradução da 2.ed. original de Vandenberg D. de Souza. - Rio de Janeiro : Elsevier, 2002 (Tradução de: Introduction to automata theory, languages, and computation - ISBN 85-352-1072-5)
 - Capítulo 9 - Seção 9.3 e 9.5
- **Introdução à teoria da computação / Michael Sipser** ; tradução técnica Ruy José Guerra Barretto de Queiroz ; revisão técnica Newton José Vieira. -- São Paulo : Thomson Learning, 2007 (Título original : Introduction to the theory of computation. "Tradução da segunda edição norte-americana" - ISBN 978-85-221-0499-4)
 - Capítulo 5

Hierarquia de linguagens

-  Existe uma MT que sempre para (decisor)
 -  Não existe MT
 -  Existe uma MT, mas ela pode entrar em loop (reconhecedor)
- Não-recursivamente enumeráveis



Hierarquia de linguagens

- Agora conhecemos L_d e L_u , duas linguagens diferentes, ambas indecidíveis
- Podemos utilizá-las para provar que outras linguagens são também indecidíveis
- Utilizaremos essas como ponto de partida nas demonstrações
- Para isso, introduziremos o conceito de redução

Redução

- Uma redução é uma maneira de converter um problema em outro de forma que uma solução para o segundo problema possa ser usada para resolver o primeiro
- Dois problemas: A e B
 - Se A se reduz a B, podemos usar uma solução para B para resolver A
- Exemplo:
 - A = problema de se orientar em uma cidade
 - B = problema de se obter um mapa da cidade

Redução

- Outro exemplo:
 - A = problema de se medir a área de um retângulo
 - B = problema de se medir seu comprimento e altura
- Outro exemplo:
 - A = problema de se resolver um sistema de equações lineares
 - B = problema de se inverter uma matriz

Redução

- Informalmente:
 - Se existe um algoritmo para converter instâncias de um problema A em instâncias de um problema B, dizemos que A se reduz a B
- Formalmente:
 - Uma redução de A a B é uma máquina de Turing que toma uma instância de A gravada em sua fita e para com uma instância de B em sua fita

Redução

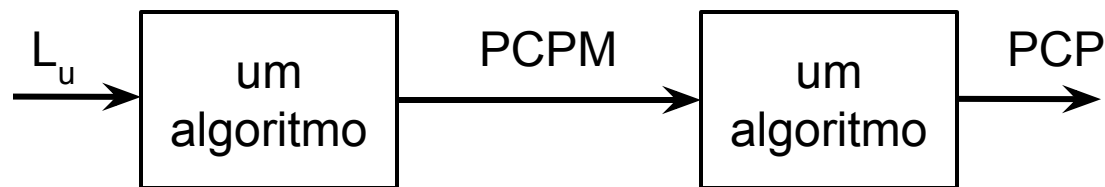
- Papel importante na classificação de problemas por decidibilidade (e complexidade também, veremos depois)
 - Quando A é redutível a B
 - resolver A não pode ser mais difícil do que resolver B
 - pois uma solução para B dá uma solução para A
 - ou seja, B é pelo menos tão difícil quanto A
 - Se A for redutível a B, então:
 - Se A é indecidível, então B também o é
 - Além disso, se A é não-RE, então B também o é
 - mas esse resultado é menos prático, já que saber que B é indecidível basta

Redução

- A prova de que L_u é indecidível utilizou uma redução
 - Ao aplicar $w111w$ sobre M_{nu} , reduzimos M_d a M_{nu} , ou seja:
 - Problema A = M_d = linguagem da diagonalização
 - Problema B = M_{nu} = complemento da linguagem universal
 - A é não-RE (já sabíamos disso), portanto B é não-RE

Problema da Correspondência de Post

- Iremos agora fechar o ciclo:
 - Provaremos que o PCP é indecidível
 - Estratégia: reduziremos L_u a PCP
 - Utilizaremos um passo intermediário para facilitar, produzindo um PCP “modificado” (PCPM)



- Como sabemos que L_u é indecidível, provaremos que o PCP é indecidível também

PCP “Modificado”

- O PCP “Modificado” é igual ao PCP original, mas com um requisito adicional:
 - O primeiro par nas listas A e B tem de ser o primeiro par na solução
 - Ou seja, a solução deve começar com 1
 - Na versão com dominós, a solução deve começar com a primeira peça
 - Formalmente, uma instância do PCPM é formada por duas listas $A = w_1, w_2, \dots, w_k$ e $B = x_1, x_2, \dots, x_k$, e uma solução é uma lista de 0 ou mais inteiros $S = (i_1, i_2, \dots, i_m)$ tais que
 - $w_{i_1} w_{i_2} \dots w_{i_m} = x_{i_1} x_{i_2} \dots x_{i_m}$

	Lista A	Lista B
i	w_i	x_i
1	10	101
2	011	11
3	101	011

Nesse exemplo, as cadeias devem começar com:
CA = 10....
CB = 101...

Reduzindo PCPM ao PCP

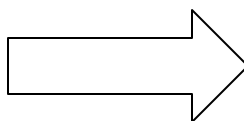
- Iremos reduzir o PCPM ao PCP
 - Ou seja, iremos providenciar um algoritmo que converte instâncias do PCPM em instâncias do PCP
- De forma que: se existir uma solução para o PCP, poderemos usá-la para obter uma solução para o PCPM

Reduzindo PCPM ao PCP

- O algoritmo de redução é o seguinte:
 - Dada uma instância do PCPM, com k elementos
 - Passaremos de w_i, x_i para y_i, z_i

	Lista A	Lista B
i	w_i	x_i
1	1	111
2	10111	10
3	10	0

Instância do PCPM



	Lista A	Lista B
i	y_i	z_i
0		
1		
2		
3		
4		

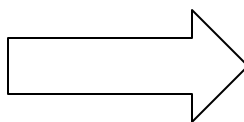
Instância do PCP

Reduzindo PCPM ao PCP

- Primeiro, copia-se os elementos das listas A e B para a nova instância, deixando um “espaço” antes e depois

	Lista A	Lista B
i	wi	xi
1	1	111
2	10111	10
3	10	0

Instância do PCPM



	Lista A	Lista B
i	yi	zi
0		
1	1	111
2	10111	10
3	10	0
4		

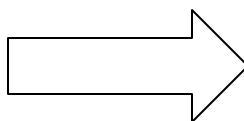
Instância do PCP

Reduzindo PCPM ao PCP

- Depois, modifica-se a Lista A, inserindo um * após cada símbolo

	Lista A	Lista B
i	wi	xi
1	1	111
2	10111	10
3	10	0

Instância do PCPM



	Lista A	Lista B
i	yi	zi
0		
1	1*	111
2	1*0*1*1*1*	10
3	1*0*	0
4		

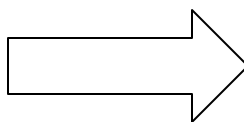
Instância do PCP

Reduzindo PCPM ao PCP

- Depois, modifica-se a Lista B, inserindo um * antes de cada símbolo

	Lista A	Lista B
i	wi	xi
1	1	111
2	10111	10
3	10	0

Instância do PCPM



	Lista A	Lista B
i	yi	zi
0		
1	1*	*1*1*1
2	1*0*1*1*1*	*1*0
3	1*0*	*0
4		

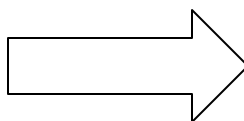
Instância do PCP

Reduzindo PCPM ao PCP

- Na primeira linha da Lista A, insere-se o elemento da segunda linha, com um * antes

	Lista A	Lista B
i	wi	xi
1	1	111
2	10111	10
3	10	0

Instância do PCPM



	Lista A	Lista B
i	yi	zi
0	*1*	
1	1*	*1*1*1
2	1*0*1*1*1*	*1*0
3	1*0*	*0
4		

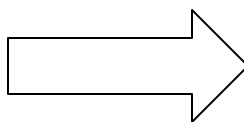
Instância do PCP

Reduzindo PCPM ao PCP

- Na primeira linha da Lista B, insere-se o elemento da segunda linha, sem modificá-lo

	Lista A	Lista B
i	wi	xi
1	1	111
2	10111	10
3	10	0

Instância do PCPM



	Lista A	Lista B
i	yi	zi
0	*1*	*1*1*1
1	1*	*1*1*1
2	1*0*1*1*1*	*1*0
3	1*0*	*0
4		

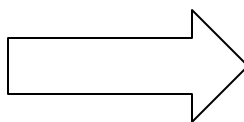
Instância do PCP

Reduzindo PCPM ao PCP

- Na última linha da Lista A, insere-se um \$

	Lista A	Lista B
i	wi	xi
1	1	111
2	10111	10
3	10	0

Instância do PCPM



	Lista A	Lista B
i	yi	zi
0	*1*	*1*1*1
1	1*	*1*1*1
2	1*0*1*1*1*	*1*0
3	1*0*	*0
4	\$	

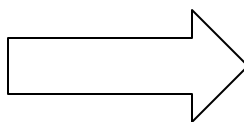
Instância do PCP

Reduzindo PCPM ao PCP

- Na última linha da Lista B, insere-se a cadeia *\$

	Lista A	Lista B
i	wi	xi
1	1	111
2	10111	10
3	10	0

Instância do PCPM



	Lista A	Lista B
i	yi	zi
0	*1*	*1*1*1
1	1*	*1*1*1
2	1*0*1*1*1*	*1*0
3	1*0*	*0
4	\$	*\$

Instância do PCP

Reduzindo PCPM ao PCP

- Esses passos são claramente um algoritmo: um passo-a-passo simples de ser seguido
- Vamos agora mostrar que esse algoritmo resulta em uma redução válida, ou seja:
 - Uma solução para o PCP pode ser usada para se obter uma solução para o PCPM

	Lista A	Lista B
i	y_i	z_i
0	$*1*$	$*1*1*1$
1	$1*$	$*1*1*1$
2	$1*0*1*1*1*$	$*1*0$
3	$1*0*$	$*0$
4	$\$$	$*\$$

Reduzindo PCPM ao PCP

- Primeiro, observe que as instâncias reduzidas para o PCP só possuem soluções do tipo:
 - $0, i_1, i_2, \dots, i_m, k+1$
 - Ou seja, o primeiro elemento da solução (primeira peça do dominó) só pode ser o elemento da linha 0
- Cadeia A = $*1*$...
- Cadeia B = $*1*1*1*$...
 - Pois de outra maneira, as cadeias iriam começar de forma diferente

	Lista A	Lista B
i	y_i	z_i
0	$*1*$	$*1*1*1$
1	$1*$	$*1*1*1$
2	$1*0*1*1*1*$	$*1*0$
3	$1*0*$	$*0$
4	$\$$	$*\$$

Reduzindo PCPM ao PCP

- Da mesma maneira o último elemento da solução (última peça do dominó) só pode ser o elemento da linha $k+1$ (4)
- Cadeia A = $*1* \dots \dots \$$
- Cadeia B = $*1*1*1 \dots \dots *\$$
 - Pois de outra maneira, as cadeias iriam terminar de forma diferente

	Lista A	Lista B
i	y_i	z_i
0	$*1*$	$*1*1*1$
1	$1*$	$*1*1*1$
2	$1*0*1*1*1*$	$*1*0$
3	$1*0*$	$*0$
4	$\$$	$*\$$

Reduzindo PCPM ao PCP

- Os elementos intermediários correspondem ao PCP original
 - Os *s inseridos não modificam a lógica
 - A cada nova peça do dominó, a cadeia A irá terminar com *, e a B não
 - Mas cada nova peça do dominó, a cadeia B irá começar com *, e a A não
- Ou seja, se antes a solução sem os *s “servia”, com os *s a mesma solução vai “servir” também

	Lista A	Lista B
i	yi	zi
0	*1*	*1*1*1
1	1*	*1*1*1
2	1*0*1*1*1*	*1*0
3	1*0*	*0
4	\$	*\$

Reduzindo PCPM ao PCP

- Suponha agora que eu tenho uma solução para o PCP
 - $0, i_1, i_2, \dots, i_m, k+1$
 - Ou seja:
 - $y_0 y_{i_1} y_{i_2} \dots y_{i_m} y_{k+1} = z_0 z_{i_1} z_{i_2} \dots z_{i_m} z_{k+1}$
 - Lembrando que y_i e z_i são as cadeias no PCP, com *s e \$s inseridos conforme o algoritmo de redução
 - Enquanto que w_i e x_i são as cadeias do PCPM, sem *s e sem \$s
 - Agora, se removermos os *s e \$s de ambos os lados, teremos
 - $w_1 w_{i_1} w_{i_2} \dots w_{i_m} = x_1 x_{i_1} x_{i_2} \dots x_{i_m}$
 - Ou seja, $1, i_1, i_2, \dots, i_m$ é uma solução para o PCPM

Reduzindo PCPM ao PCP

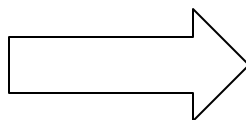
- Resumindo: se encontrei a solução para o PCP:
 - $0, i_1, i_2, \dots, i_m, k+1$
- Basta remover o primeiro e último elementos, começar com 1, e teremos uma solução para o PCPM
 - $1, i_1, i_2, \dots, i_m$

Reduzindo PCPM ao PCP

- Exemplo: aplicando o algoritmo

	Lista A	Lista B
i	wi	xi
1	a	ab
2	b	ca
3	ca	a
4	abc	c

Instância do PCPM



	Lista A	Lista B
i	yi	zi
0	*a*	*a*b
1	a*	*a*b
2	b*	*c*a
3	c*a*	*a
4	a*b*c*	*c
5	\$	*\$

Instância do PCP

Reduzindo PCPM ao PCP

- Encontrando uma solução para o PCP

	Lista A	Lista B
i	yi	zi
0	*a*	*a*b
1	a*	*a*b
2	b*	*c*a
3	c*a*	*a
4	a*b*c*	*c
5	\$	*\$

Resposta:

Cadeia A = [*a*] [b*] [c*a*] [a*] [a*b*c*] [\$]

Cadeia B = [*a*b] [*c*a] [*a] [*a*b] [*c] [*\$]

Solução = 0, 2, 3, 1, 4, 5

Reduzindo PCPM ao PCP

	Lista A	Lista B
i	wi	xi
1	a	ab
2	b	ca
3	ca	a
4	abc	c

Instância do PCPM

- Usando a solução do PCP para resolver o PCPM
 - Solução PCP
 - 0,2,3,1,4,5
 - Solução PCPM (removendo primeiro e último da solução do PCP, e adicionando 1 no início)
 - 1,2,3,1,4

Resposta:

Cadeia A = [a] [b] [ca] [a] [abc] = abcaaabc

Cadeia B = [ab] [ca] [a] [ab] [c] = abcaaabc

Reduzindo Lu ao PCPM

- Concluimos que, se conseguirmos resolver o PCP, é simples resolver o PCPM
- Agora vamos fazer um passo a mais
 - Reduziremos Lu ao PCPM
 - De forma similar, se conseguirmos resolver o PCPM, conseguiremos resolver o Lu
 - Considerando a redução anterior: se conseguirmos resolver o PCP, conseguiremos resolver o Lu
- Mas sabemos que Lu é indecidível!!
 - Vimos a prova nessa aula, anteriormente
- Portanto, a conclusão é que o PCP não pode ser resolvido por um algoritmo!!
 - Pois se fosse, teríamos um algoritmo para resolver o Lu!!

Reduzindo Lu ao PCPM

- FAQ:
 - P: Mas no último exemplo, encontramos uma solução para o PCP! E a utilizamos para encontrar uma solução para o PCPM!! Então como podemos concluir que o PCP é indecidível?? Acabamos de decidi-lo!
 - R: Na verdade, não decidimos! Encontramos apenas UMA solução, usando nossa capacidade inventiva e criativa de seres humanos! Não usamos um algoritmo para resolvê-lo! A questão aqui é: existe um algoritmo para resolvê-lo? A resposta (como veremos) é não! Mas isso não impede que consigamos eventualmente encontrar UMA ou DUAS soluções, usando intuição, raciocínio, sorte, etc...

Reduzindo Lu ao PCPM

- Lembrando: o que é Lu?
 - É o conjunto de pares (M, w) tal que M aceita w , onde:
 - M é uma máquina de Turing
 - w é uma entrada
- A redução de Lu ao PCPM irá simular a execução de M
 - A solução para o PCPM será um “log” ou registro da execução de M sobre a entrada w
 - De forma parecida com o que fazemos com as configurações instantâneas
- Veremos que o PCPM pode ser usado para “implementar” a função de transição de M

Reduzindo Lu ao PCPM

- Utilizaremos uma abordagem baseada em exemplos:
- Considere a entrada $w=01$
- Considere a seguinte MT:
 - $M = (\{q1, q2, q3\}, \{0, 1\}, \{0, 1, B\}, \delta, q1, B, \{q3\})$
- Onde δ é dado por:

	0	1	B
q1	(q2,1,D)	(q2,0,E)	(q2,1,E)
q2	(q3,0,E)	(q1,0,D)	(q2,0,D)
q3	-	-	-

Reduzindo Lu ao PCPM

- Regra 1:
 - O primeiro par é
 - $\#, \#q_0 w \#$
 - Onde q_0 é o estado inicial
 - E w é a entrada
 $w=01$

Regra	Lista A	Lista B
(1)	#	$\#q_1 01 \#$

	0	1	B
q1	(q2,1,D)	(q2,0,E)	(q2,1,E)
q2	(q3,0,E)	(q1,0,D)	(q2,0,D)
q3	-	-	-

Reduzindo Lu ao PCPM

- Regra 2:
 - Insira um par para cada símbolo de fita e o separador #
 - Sem considerar B

w=01

	0	1	B
q1	(q2,1,D)	(q2,0,E)	(q2,1,E)
q2	(q3,0,E)	(q1,0,D)	(q2,0,D)
q3	-	-	-

Regra	Lista A	Lista B
(1)	#	#q ₁ 01#
(2)	0	0
	1	1
	#	#

Reduzindo Lu ao PCPM

- Regra 3:
 - Insira pares que simulam o movimento de M
 - qX,Yp se $\delta(q,X) = (p,Y,D)$
 - ZqX,pZY se $\delta(q,X) = (p,Y,E)$
 - (Repetir para todo símbolo de fita Z)
 - $q\#,Yp\#$ se $\delta(q,B) = (p,Y,D)$
 - $Zq\#,pZY\#$ se $\delta(q,B) = (p,Y,E)$
 - (Repetir para todo símbolo de fita Z)

$w=01$

	0	1	B
q1	(q2,1,D)	(q2,0,E)	(q2,1,E)
q2	(q3,0,E)	(q1,0,D)	(q2,0,D)
q3	-	-	-

Regra	Lista A	Lista B
...
(3)	$q_1 0$	$1 q_2$
	$0 q_1 1$	$q_2 00$
	$1 q_1 1$	$q_2 10$
	$0 q_1 \#$	$q_2 01 \#$
	$1 q_1 \#$	$q_2 11 \#$
	$0 q_2 0$	$q_3 00$
	$1 q_2 0$	$q_3 10$
	$q_2 1$	$0 q_1$
	$q_2 \#$	$0 q_2 \#$

Reduzindo Lu ao PCPM

- Regra 4:
 - Insira pares, para todo símbolo de fita X e Y e para todo estado de aceitação q:
 - XqY, q
 - Xq, q
 - qY, q

$w=01$

	0	1	B
q1	(q2,1,D)	(q2,0,E)	(q2,1,E)
q2	(q3,0,E)	(q1,0,D)	(q2,0,D)
q3	-	-	-

Regra	Lista A	Lista B
...
(4)	$0q_30$	q_3
	$0q_31$	q_3
	$1q_30$	q_3
	$1q_31$	q_3
	$0q_3$	q_3
	$1q_3$	q_3
	q_30	q_3
	q_31	q_3

Reduzindo Lu ao PCPM

- Regra 5:
 - Para cada estado de aceitação q , insira um par:
 - $q\#\#, \#$

$w=01$

	0	1	B
q1	(q2,1,D)	(q2,0,E)	(q2,1,E)
q2	(q3,0,E)	(q1,0,D)	(q2,0,D)
q3	-	-	-

Regra	Lista A	Lista B
...
(5)	$q_3\#\#$	$\#$

Reduzindo Lu ao PCPM

- Executando a MT original

w=01

	0	1	B
q1	(q2,1,D)	(q2,0,E)	(q2,1,E)
q2	(q3,0,E)	(q1,0,D)	(q2,0,D)
q3	-	-	-

[q1]01
1[q2]1
10[q1]B
1[q2]01
[q3]101

Regra	Lista A	Lista B
(1)	#	#q ₁ 01#
(2)	0	0
	1	1
	#	#
(3)	q ₁ 0	1q ₂
	0q ₁ 1	q ₂ 00
	1q ₁ 1	q ₂ 10
	0q ₁ #	q ₂ 01#
	1q ₁ #	q ₂ 11#
	0q ₂ 0	q ₃ 00
	1q ₂ 0	q ₃ 10
	q ₂ 1	0q ₁
	q ₂ #	0q ₂ #
(4)	0q ₃ 0	q ₃
	0q ₃ 1	q ₃
	1q ₃ 0	q ₃
	1q ₃ 1	q ₃
	0q ₃	q ₃
	1q ₃	q ₃
	q ₃ 0	q ₃
	q ₃ 1	q ₃
(5)	q ₃ ##	#

Resolvendo o PCPM

- Deve começar pelo primeiro elemento:

A : #

B : #q₁01#

Regra	Lista A	Lista B
(1)	#	#q ₁ 01#
(2)	0	0
	1	1
	#	#
(3)	q ₁ 0	1q ₂
	0q ₁ 1	q ₂ 00
	1q ₁ 1	q ₂ 10
	0q ₁ #	q ₂ 01#
	1q ₁ #	q ₂ 11#
	0q ₂ 0	q ₃ 00
	1q ₂ 0	q ₃ 10
	q ₂ 1	0q ₁
	q ₂ #	0q ₂ #
(4)	0q ₃ 0	q ₃
	0q ₃ 1	q ₃
	1q ₃ 0	q ₃
	1q ₃ 1	q ₃
	0q ₃	q ₃
	1q ₃	q ₃
	q ₃ 0	q ₃
	q ₃ 1	q ₃
(5)	q ₃ ##	#

Resolvendo o PCPM

- Escolhendo o único par possível

A : #q₁0

B : #q₁01#1q₂

Regra	Lista A	Lista B
(1)	#	#q ₁ 01#
(2)	0	0
	1	1
	#	#
(3)	q ₁ 0	1q ₂
	0q ₁ 1	q ₂ 00
	1q ₁ 1	q ₂ 10
	0q ₁ #	q ₂ 01#
	1q ₁ #	q ₂ 11#
	0q ₂ 0	q ₃ 00
	1q ₂ 0	q ₃ 10
	q ₂ 1	0q ₁
	q ₂ #	0q ₂ #
(4)	0q ₃ 0	q ₃
	0q ₃ 1	q ₃
	1q ₃ 0	q ₃
	1q ₃ 1	q ₃
	0q ₃	q ₃
	1q ₃	q ₃
	q ₃ 0	q ₃
	q ₃ 1	q ₃
(5)	q ₃ ##	#

Resolvendo o PCPM

- Escolhendo o único par possível

A : #q₁01#1q₂1

B : #q₁01#1q₂1#10q₁

Regra	Lista A	Lista B
(1)	#	#q ₁ 01#
(2)	0	0
	1	1
	#	#
(3)	q ₁ 0	1q ₂
	0q ₁ 1	q ₂ 00
	1q ₁ 1	q ₂ 10
	0q ₁ #	q ₂ 01#
	1q ₁ #	q ₂ 11#
	0q ₂ 0	q ₃ 00
	1q ₂ 0	q ₃ 10
	q ₂ 1	0q ₁
	q ₂ #	0q ₂ #
(4)	0q ₃ 0	q ₃
	0q ₃ 1	q ₃
	1q ₃ 0	q ₃
	1q ₃ 1	q ₃
	0q ₃	q ₃
	1q ₃	q ₃
	q ₃ 0	q ₃
	q ₃ 1	q ₃
(5)	q ₃ ##	#

Resolvendo o PCPM

- Nesse ponto, daria para “copiar” os três símbolos seguintes 1,# e 0
 - Mas aí não iríamos muito longe
 - A solução é “copiar” apenas o 1 e #

A : #q₁01#1q₂1#1

B : #q₁01#1q₂1#10q₁#1

Regra	Lista A	Lista B
(1)	#	#q ₁ 01#
(2)	0	0
	1	1
	#	#
(3)	q ₁ 0	1q ₂
	0q ₁ 1	q ₂ 00
	1q ₁ 1	q ₂ 10
	0q ₁ #	q ₂ 01#
	1q ₁ #	q ₂ 11#
	0q ₂ 0	q ₃ 00
	1q ₂ 0	q ₃ 10
	q ₂ 1	0q ₁
	q ₂ #	0q ₂ #
(4)	0q ₃ 0	q ₃
	0q ₃ 1	q ₃
	1q ₃ 0	q ₃
	1q ₃ 1	q ₃
	0q ₃	q ₃
	1q ₃	q ₃
	q ₃ 0	q ₃
	q ₃ 1	q ₃
(5)	q ₃ ##	#

Resolvendo o PCPM

- Só tem uma opção agora

A: #q₁01#1q₂1#10q₁#

B: #q₁01#1q₂1#10q₁#1q₂01#

Regra	Lista A	Lista B
(1)	#	#q ₁ 01#
(2)	0	0
	1	1
	#	#
(3)	q ₁ 0	1q ₂
	0q ₁ 1	q ₂ 00
	1q ₁ 1	q ₂ 10
	0q ₁ #	q ₂ 01#
	1q ₁ #	q ₂ 11#
	0q ₂ 0	q ₃ 00
	1q ₂ 0	q ₃ 10
	q ₂ 1	0q ₁
	q ₂ #	0q ₂ #
(4)	0q ₃ 0	q ₃
	0q ₃ 1	q ₃
	1q ₃ 0	q ₃
	1q ₃ 1	q ₃
	0q ₃	q ₃
	1q ₃	q ₃
	q ₃ 0	q ₃
	q ₃ 1	q ₃
(5)	q ₃ ##	#

Resolvendo o PCPM

- Novamente, uma opção

A : #q₁01#1q₂1#10q₁#1q₂0

B : #q₁01#1q₂1#10q₁#1q₂01#q₃10

Regra	Lista A	Lista B
(1)	#	#q ₁ 01#
(2)	0	0
	1	1
	#	#
(3)	q ₁ 0	1q ₂
	0q ₁ 1	q ₂ 00
	1q ₁ 1	q ₂ 10
	0q ₁ #	q ₂ 01#
	1q ₁ #	q ₂ 11#
	0q ₂ 0	q ₃ 00
	1q ₂ 0	q ₃ 10
	q ₂ 1	0q ₁
	q ₂ #	0q ₂ #
(4)	0q ₃ 0	q ₃
	0q ₃ 1	q ₃
	1q ₃ 0	q ₃
	1q ₃ 1	q ₃
	0q ₃	q ₃
	1q ₃	q ₃
	q ₃ 0	q ₃
	q ₃ 1	q ₃
(5)	q ₃ ##	#

Resolvendo o PCPM

- Continuando

A: #q₁01#1q₂1#10q₁#1q₂01#q₃101#q₃01#q₃1#

B: #q₁01#1q₂1#10q₁#1q₂01#q₃101#q₃01#q₃1#q₃#

Regra	Lista A	Lista B
(1)	#	#q ₁ 01#
(2)	0	0
	1	1
	#	#
(3)	q ₁ 0	1q ₂
	0q ₁ 1	q ₂ 00
	1q ₁ 1	q ₂ 10
	0q ₁ #	q ₂ 01#
	1q ₁ #	q ₂ 11#
	0q ₂ 0	q ₃ 00
	1q ₂ 0	q ₃ 10
	q ₂ 1	0q ₁
	q ₂ #	0q ₂ #
(4)	0q ₃ 0	q ₃
	0q ₃ 1	q ₃
	1q ₃ 0	q ₃
	1q ₃ 1	q ₃
	0q ₃	q ₃
	1q ₃	q ₃
	q ₃ 0	q ₃
	q ₃ 1	q ₃
(5)	q ₃ ##	#

Resolvendo o PCPM

- Pra encerrar, a regra 5

A: #q₁01#1q₂1#10q₁#1q₂01#q₃101#q₃01#q₃1#q₃##

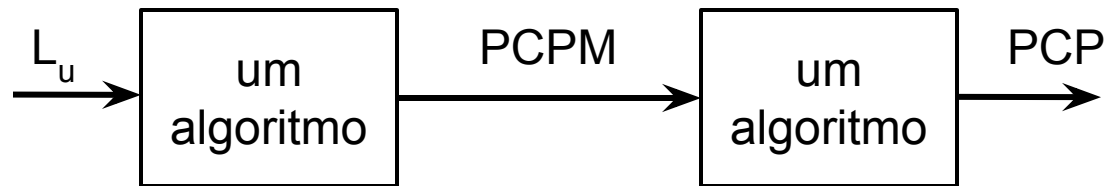
B: #q₁01#1q₂1#10q₁#1q₂01#q₃101#q₃01#q₃1#q₃##

Reduzindo Lu ao PCPM

- Vimos que a solução anterior “simula” a execução da máquina de turing sobre w
 - É possível demonstrar que não existe solução diferente para o PCPM, senão aquela que é uma simulação correta de M !
 - Ou seja: M aceita w se e somente se a instância do PCPM construída tem uma solução!!
- Dessa forma, confirmamos que o algoritmo anterior, com as 5 regras, é uma redução válida

Reduzindo Lu ao PCP

- Conseguimos completar a redução de Lu ao PCP



- Verificamos que as reduções são válidas
- Ou seja, se pudéssemos decidir o PCP, poderíamos usar o mesmo algoritmo como base para decidir um PCPM, e em seguida, decidir Lu
- Mas Lu é indecidível
 - Ou seja, não existe um algoritmo que decide o PCP
- Portanto PCP é um problema indecidível!

Outros problemas indecidíveis

- Agora temos 3 problemas indecidíveis “nas mãos”
 - Ld
 - Lu
 - PCP
- Podemos usá-los como ferramenta para provar que outros problemas são indecidíveis
 - Um exemplo é a ambiguidade de gramáticas livres de contexto
 - Problema AMB = decidir se uma gramática é ambígua
 - É possível reduzir PCP a esse problema AMB

Resumo

- Mostramos que existe uma forma de demonstrar indecidibilidade de problemas
 - Descrevendo-os como linguagens
 - Reduzindo esses problemas a outros conhecida e indecidíveis
 - Pode ser útil na prática
- Vimos também que problemas indecidíveis têm soluções, apenas não existe um algoritmo para encontrá-las

Resumo

- Veremos a seguir outra classe de problemas
 - São decidíveis, porém não é viável decidi-los
 - São problemas ditos intratáveis
 - Ou seja, existem algoritmos, mas eles são muito ineficientes
 - A ineficiência é tanta, que mesmo supercomputadores modernos levariam anos para decidir uma instância de tamanho razoável
 - Ou necessitariam de uma quantidade inimaginável de memória e/ou espaço em disco

Fim

Aula 27 - Problemas indecidíveis