Teoria da Computação
Redutibilidade

Leonardo Takuno
{leonardo.takuno@gmail.com}

Centro Universitário Senac

Sumário

Sumário

PCP

- Dado um conjunto de "dominós":
$$\left\{ \left[\frac{b}{ca}\right], \left[\frac{a}{ab}\right], \left[\frac{ca}{a}\right], \left[\frac{abc}{c}\right] \right\}$$
- Emparelhamento:
$$\left[\frac{a}{ab}\right] \left[\frac{b}{ca}\right] \left[\frac{ca}{a}\right] \left[\frac{a}{ab}\right] \left[\frac{abc}{c}\right]$$
- Para algumas coleções de dominós, encontrar um emparelhamento pode não ser possível:
$$\left\{ \left[\frac{abc}{ab}\right], \left[\frac{ca}{a}\right], \left[\frac{acc}{ba}\right] \right\}$$

PCP e PCPM

- Uma instância do PCP é uma coleção P de dominós:
  $P = \left\{ \left[\frac{t_1}{b_1} \right], \left[\frac{t_2}{b_2} \right], ..., \left[\frac{t_k}{b_k} \right] \right\}$
  e um **emparelhamento** é uma seqüência $i_1, i_2, ..., i_l$
  onde $t_{i_1} t_{i_2} ... t_{i_l} = b_{i_1} b_{i_2} ... b_{i_l}$.
    - $PCP = \{\langle P \rangle |$ P é uma instância do PCP com um emparelhamento $\}$
    - $PCPM = \{\langle P \rangle |$ P é uma instância do PCP com um emparelhamento que começa com o primeiro dominó $\}$

PCP é indecidível

Teorema 5.15: PCP é indecidível

- A prova envolve muitos detalhes técnicos, mas a idéia é simples.
- Vamos reduzir A_{MT} para PCP via histórias de computação.
- Isto é, para uma MT M e w construiremos uma instância do PCP tal que P tem uma solução se, e somente se, M aceita w.

PCP é indecidível

Teorema 5.15: PCP é indecidível
Prova: Supomos que a MT R decide o PCP e construímos S que decide A_{MT}.
Seja $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$
Para isso, S primeiro constrói uma instância P' de PCPM.
Seja P' uma coleção de dominós.

(1) Primeiro dominó em P'
$$\left[\frac{\#}{\# q_0 w_1 w_2 \ldots w_n \#}\right]$$
(2) Para todo $a, b \in \Gamma$ e todo $q, r \in Q$
se $\delta(q, a) = (r, b, D)$, $\left[\frac{qa}{br}\right] \in P'$

PCP é indecidível

Teorema 5.15: PCP é indecidível
Prova: (Continuação)

(3) Para todo $a, b, c \in \Gamma$ e todo $q, r \in Q$
se $\delta(q, a) = (r, b, E)$, $\left[\frac{cqa}{rcb}\right] \in P'$

(4) Para todo $a \in \Gamma$
$\left[\frac{a}{a}\right] \in P'$

PCP é indecidível

Teorema 5.15: PCP é indecidível
Prova: (Continuação)

(5) Vamos permitir copiar o símbolo # que marca a separação das configurações e também possibilitar a inserção de um espaço em branco $\sqcup$ no final da configuração:
$\left[\frac{\#}{\#}\right]$ e $\left[\frac{\#}{\sqcup\#}\right] \in P'$

(6) Para todo $a \in \Gamma$
$\left[\frac{aq_{accept}}{q_{accept}}\right]$ e $\left[\frac{q_{accept}\,a}{q_{accept}}\right] \in P'$

(7) Finalmente, adicionamos o nó
$\left[\frac{q_{accept}\#\#}{\#}\right]$
e completamos o emparelhamento.

PCP é indecidível

- Seja $\Gamma = \{0, 1, 2, \sqcup\}$ e $w = 0100$ e que o estado inicial de M seja q_0
- A configuração inicial é $q_0 0100$
- Agora suponha
$$\delta(q_0, 0) = (q_7, 2, D)$$
- **Parte 1:** Colocar o primeiro dominó.

$$\left[\frac{\#}{\# q_0 0100 \#}\right]$$

PCP é indecidível

$$\left[\frac{\#}{\#q_0 0100\#}\right]$$

- Para ocorrer o emparelhamento nós queremos um dominó que tenha $q_0 0$ na parte superior do dominó. a **parte 2** coloca o dominó:

$$\left[\frac{q_0 0}{2 q_7}\right]$$

pois $\delta(q_0, 0) = (q_7, 2, D)$

PCP é indecidível

- O emparelhamento se apresenta da seguinte maneira:

$$\left[\frac{\#}{\#q_0 0100\#}\right]\left[\frac{q_0 0}{2q_7}\right]$$

- A **parte 4** coloca os dominós

$$\left[\frac{0}{0}\right], \left[\frac{1}{1}\right], \left[\frac{2}{2}\right], \left[\frac{\sqcup}{\sqcup}\right]$$

PCP é indecidível

- O emparelhamento junto com a parte 5 apresenta-se:

$$\left[\frac{\#}{\#q_0 0100\#}\right]\left[\frac{q_0 0}{2q_7}\right]\left[\frac{1}{1}\right]\left[\frac{0}{0}\right]\left[\frac{0}{0}\right]\left[\frac{\#}{\#}\right]$$

PCP é indecidível

- Suponha que:

$$\delta(q_7, 1) = (q_5, 0, D)$$

Então temos o dominó

$$\left[\frac{q_7 1}{0 q_5}\right]$$

- O último emparelhamento se estende para

$$\left[\frac{\#}{\# q_0 0100 \#}\right]\left[\frac{q_0 0}{2 q_7}\right]\left[\frac{1}{1}\right]\left[\frac{0}{0}\right]\left[\frac{0}{0}\right]\left[\frac{\#}{\#}\right]\left[\frac{2}{2}\right]\left[\frac{q_7 1}{0 q_5}\right]\left[\frac{0}{0}\right]\left[\frac{0}{0}\right]\left[\frac{\#}{\#}\right]$$

PCP é indecidível

- Suponha que:

$$\delta(q_5, 0) = (q_9, 2, E)$$

Então temos os dominós

$$\left[\frac{0q_50}{q_902}\right], \left[\frac{1q_50}{q_912}\right], \left[\frac{2q_50}{q_922}\right], \left[\frac{\sqcup q_50}{q_9 \sqcup 2}\right]$$

- O último emparelhamento se estende para

$$\left[\frac{\#}{\#q_00100\#}\right]\left[\frac{q_00}{2q_7}\right]\left[\frac{1}{1}\right]\left[\frac{0}{0}\right]\left[\frac{0}{0}\right]\left[\frac{\#}{\#}\right]\left[\frac{2}{2}\right]\left[\frac{q_71}{0q_5}\right]\left[\frac{0}{0}\right]\left[\frac{0}{0}\right]\left[\frac{\#}{\#}\right]$$

$$\left[\frac{\#}{\#}\right]\left[\frac{2}{2}\right]\left[\frac{0q_50}{q_902}\right]\left[\frac{0}{0}\right]\left[\frac{\#}{\#}\right]$$

PCP é indecidível

- Para construir um emparelhamento nós temos que simular M sobre a entrada w.
- Entretanto, há um problema
- A string na parte superior está sempre “atrás” da parte inferior!

PCP é indecidível

- O emparelhamento pode ser completado utilizando os três dominós dos itens 6 e 7. Assim, inserimos “pseudopassos” da máquina de Turing depois que ela parou, onde a cabeça “consome” os símbolos adjacentes até que neste ponto não reste mais nenhum.

PCP é indecidível

(6) Para todo $a \in \Gamma$

$$\left[\frac{aq_{accept}}{q_{accept}}\right] \text{ e } \left[\frac{q_{accept}a}{q_{accept}}\right] \in P'$$

(7) Adicionamos o nó

$$\left[\frac{q_{accept}\#\#}{\#}\right]$$

- Quando a M alcança o estado de aceitação as partes 6 e 7 permitem completar o emparelhamento.
- Entretanto, se M rejeita w ou entrar em loop. Não ocorrerá emparelhamento.

PCP é indecidível

Teorema

PCPM é indecidível

- Assuma que PCPM seja decidível. Então temos um decisor R para PCPM.

PCP é indecidível

Construímos a MT S para decidir A_{MT} da seguinte forma:
S = "Sobre a entrada $\langle M, w \rangle$, uma codificação de uma MT M e uma cadeia w:

1. Construa P' como descrito em 7 partes.
2. Execute R sobre P'
3. Se R aceita, *aceite*;
4. Se R rejeita, *rejeite* "

Então S é um decisor para A_{MT}. Que é uma contradição para o fato de que A_{MT} é indecidível.

PCP é indecidível

- Concluimos a construção de P'. Lembre-se que P' é uma instância de PCPM.
- Reduzimos A_{MT} para PCPM por encontrar um emparelhamento que simula a computação de M sobre w.
- Como PCPM é uma instância de PCP, é possível encontrar uma maneira de converter P' em P, uma instância de PCP que simula M sobre w.

PCPM para PCP

- Notação:

$$\begin{aligned} \star u &= *u_1 * u_2 * u_3 * \cdots * u_n \\ u\star &= u_1 * u_2 * u_3 * \cdots * u_n* \\ \star u\star &= *u_1 * u_2 * u_3 * \cdots * u_n* \end{aligned}$$

PCPM com

$$\left\{ \left[\frac{t_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \left[\frac{t_3}{b_3}\right], \ldots, \left[\frac{t_k}{b_k}\right] \right\}$$

é equivalente ao PCP com
PCPM com

$$\left\{ \left[\frac{\star t_1}{\star b_1 \star}\right], \left[\frac{\star t_2}{b_2 \star}\right], \left[\frac{\star t_3}{b_3 \star}\right], \ldots, \left[\frac{\star t_k}{b_k \star}\right], \left[\frac{\star\Diamond}{\Diamond}\right] \right\}$$

PCP é indecidível

Teorema

PCP é indecidível

- Assuma que PCP seja decidível. Então temos um decisor R para PCP.

PCP é indecidível

Construímos a MT S da seguinte forma:
S = “Sobre a entrada $\langle P' \rangle$:

1. Construa P como descrito.
2. Execute R sobre P
3. Se R aceita, *aceite*;
4. Se R rejeita, *rejeite* ”

Então S é um decisor para PCP. Que é uma contradição para o fato de que PCPM é indecidível.

Sumário

Funções computáveis

Definição 5.17: Uma função $f : \Sigma^* \rightarrow \Sigma^*$ é uma **função computável** se alguma máquina de Turing M, sobre toda entrada w, pára com exatamente $f(w)$ sobre sua fita.

Funções computáveis

- Exemplos de funções computáveis
  - $f(\langle m, n \rangle) = m + n$
  - Seja A uma matriz $n \times n$ então a função

    $$f(\langle A \rangle) = \langle A^2 \rangle$$

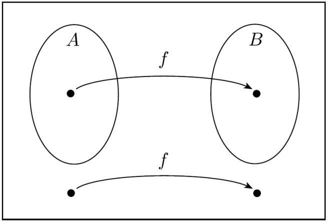
    é computável.

Redutibilidade por Mapeamento

Definição 5.20: A linguagem A é **redutível por mapeamento** à linguagem B, escrito $A \leq_m B$, se existe uma função computável $f : \Sigma^* \rightarrow \Sigma^*$, onde para toda w,

$$w \in A \Longleftrightarrow f(w) \in B$$

A função f é denominada a **redução** de A para B.



Redutibilidade por Mapeamento

Teorema 5.22: Se $A \leq_m B$ e B é decidível, então A é decidível.

Prova: Fazemos M ser o decisor para B e f a redução de A para B. Descrevemos um decisor N para A da seguinte forma.
$N =$ "Sobre a entrada w:

1. Compute $f(w)$.
2. Rode M sobre a entrada $f(w)$ e dê como o que M der como saída.

Claramente, se $w \in A$, então $f(w) \in B$, porque f é uma redução de A para B. Portanto, M aceita $f(w)$ sempre que $w \in A$.

Redutibilidade por Mapeamento

Corolário 5.23: Se $A \leq_m B$ e A é indecidível, então B é indecidível.

Prova: Assuma que B é decidível, seja M um decisor para B e seja f uma redução de A para B, então podemos construir um decisor N para A da seguinte forma.

$N =$ "Sobre a entrada w:

1. Compute $f(w)$.
2. Rode M sobre a entrada $f(w)$ e dê como o que M der como saída.

Claramente, como A é indecidível esta máquina não pode existir, portanto, nossa suposição de que B é decidível deve estar incorreta.

Exemplo $A_{MT} \leq_m PARA_{MT}$

A linguagem

$$PARA_{MT} = \{\langle M, w\rangle | M \text{ é uma MT para sobre } w\ \}$$

é indecidível

Ideia da prova: Podemos mostrar que $PARA_{MT}$ é indecidível apresentando uma redução por mapeamento de

$$A_{MT} \leq_m PARA_{MT}$$

Exemplo $A_{MT} \leq_m PARA_{MT}$

A linguagem

$$PARA_{MT} = \{\langle M, w\rangle | M \text{ é uma MT para sobre } w\ \}$$

é indecidível

Para mostrar que

$$A_{MT} \leq_m PARA_{MT}$$

temos que encontrar uma função computável

$$f(\langle M, w\rangle) = \langle M', w'\rangle$$

tal que:

$$\langle M, w\rangle \in A_{MT} \iff \langle M', w'\rangle \in PARA_{MT}$$

Exemplo $A_{MT} \leq_m PARA_{MT}$

A linguagem

$$PARA_{MT} = \{\langle M, w\rangle | M \text{ é uma MT para sobre } w \}$$

é indecidível

Para construir a função computável f, construa uma MT F que computa a redução:

F = "Sobre a entrada $\langle M, w\rangle$:

1. Construa a seguinte máquina M'
   M' = "Sobre a entrada x:
   1. Rode M sobre x
   2. Se M aceitar, aceite.
   3. Se M rejeitar, entrar em loop."
2. Dê como saída $\langle M', w\rangle$

Observe que $\langle M', w\rangle \in PARA_{MT} \Leftrightarrow \langle M, w\rangle \in A_{MT}$ como requerido.

Exemplo $A_{MT} \leq_m PARA_{MT}$

A linguagem

$$PARA_{MT} = \{\langle M, w\rangle | M \text{ é uma MT para sobre } w\ \}$$

é indecidível

Vamos analisar por que isto funciona:

- $\langle M, w\rangle \in A_{MT} \rightarrow M$ aceita $w \rightarrow M'$ aceita $w \rightarrow \langle M', w\rangle \in PARA_{MT}$
- $\langle M, w\rangle \notin A_{MT}$
  - M rejeita $w \rightarrow M'$ entra em loop $\rightarrow \langle M', w\rangle \notin PARA_{MT}$
  - M entra em loop sobre $w \rightarrow M'$ entra em loop $\rightarrow \langle M', w\rangle \notin PARA_{MT}$

Exemplo: $A_{MT} \leq_m PCPM \leq_m PCP$

Na prova da indecidibilidade do problema da correspondência de Post mostramos duas reduções por mapeamento:

$$A_{MT} \leq_m PCPM$$

$$PCPM \leq_m PCP$$

Note que as reduções por mapeamento são transitivas.

Exemplo: $A_{MT} \leq_m \overline{V_{MT}}$

A linguagem

$$V_{MT} = \{\langle M \rangle | \text{M é uma MT L(M)} = \oslash\}$$

é indecidível.

Ideia da prova: Podemos mostrar que V_{MT} é indecidível, apresentando uma redução por mapeamento de

$$A_{MT} \leq_m \overline{V_{MT}}$$

Exemplo: $A_{MT} \leq_m \overline{V_{MT}}$

A linguagem

$$V_{MT} = \{\langle M\rangle | \text{M é uma MT L(M)} = \oslash\}$$

é indecidível.

Vamos mostrar que

$$A_{MT} \leq_m \overline{V_{MT}}$$

Defina:

$$f(\langle M, w\rangle) = \langle M_1\rangle$$

tal que

$$M \text{ aceita } w \text{ se e somente se } L(M_1) \neq \oslash$$

$A_{MT} \leq_m \overline{V_{MT}}$

A linguagem

$$V_{MT} = \{\langle M \rangle | \text{M é uma MT L(M)} = \oslash\}$$

é indecidível.

Para construir a função computável f, construa uma MT F que computa a redução:

F = "Sobre a entrada $\langle M, w \rangle$

1. Seja M_1 a máquina:
   M_1 = "Sobre a entrada x:
   1. se $x \neq w$ rejeite
   2. se $x = w$, rode M sobre w e aceite se M aceita."
2. Dê como saida $\langle M_1 \rangle$"

$A_{MT} \leq_m \overline{V_{MT}}$

Vamos analisar por que isto funciona:

- $\langle M, w\rangle \in A_{MT} \rightarrow L(M_1) = \{w\} \rightarrow \langle M_1\rangle \notin V_{MT} \rightarrow \langle M_1\rangle \in \overline{V_{MT}}$
- $\langle M, w\rangle \notin A_{MT} \rightarrow L(M_1) = \emptyset \rightarrow \langle M_1\rangle \in V_{MT} \rightarrow \langle M_1\rangle \notin \overline{V_{MT}}$

$A_{MT} \leq_m \overline{V_{MT}}$

A redutibilidade anterior mostra que

- $\overline{V_{MT}}$ é indecidível

isto implica que

- V_{MT} é indecidível

$A_{MT} \leq_m \overline{V_{MT}}$

Conclusão:
Para mostrar que uma linguagem B é indecidível.

- Encontre uma linguagem A que foi provado ser indecidível.
- Encontre uma função de redução de A para B ou para $\overline{B}$

Exemplo: $V_{MT} \leq_m EQ_{MT}$

A linguagem

$$EQ_{MT} = \{\langle M_1, M_2\rangle \mid M_1 \text{ e } M_2 \text{ são uma MT e } L(M_1) = L(M_2)\}$$

é indecidível.

Ideia da prova: Para mostrar que EQ_{MT} é indecidível, podemos apresentar uma redução por mapeamento de

$$V_{MT} \leq_m EQ_{MT}$$

Exemplo: $V_{MT} \leq_m EQ_{MT}$

A linguagem

$$EQ_{MT} = \{\langle M_1, M_2\rangle \mid M_1 \text{ e } M_2 \text{ são uma MT e } L(M_1) = L(M_2)\}$$

é indecidível.

Vamos mostrar que

$$V_{MT} \leq_m EQ_{MT}$$

Seja M_1 uma MT que rejeita todas as entradas.
Defina:

$$f(\langle M\rangle) = \langle M, M_1\rangle$$

então teremos

$$L(M) = \emptyset \;\Leftrightarrow\; L(M) = L(M_1)$$

Exemplo: $V_{MT} \leq_m EQ_{MT}$

Para construir uma função computável f, construa uma MT F que computa a redução $V_{MT} \leq_m EQ_{MT}$, como segue:

F = "Sobre a entrada $\langle M \rangle$:

1. Construa uma máquina M_1 que rejeita todas as entradas;
2. Dê como saída $\langle M, M_1 \rangle$

Exemplo: $V_{MT} \leq_m EQ_{MT}$

Perceba que $L(M) = \oslash \Leftrightarrow L(M) = L(M_1)$, onde M_1 rejeita todas as entradas, então

$$F(\langle M \rangle) = \langle M, M_1 \rangle$$

é uma função computável para a redução $V_{MT} \leq_m EQ_{MT}$. Assim,

$$\langle M \rangle \in V_{MT} \Leftrightarrow \langle M, M_1 \rangle \in EQ_{MT}.$$

Redutibilidade por Mapeamento

Observação: Reduções de linguagens nem sempre existem. Isto é, nem sempre é possível especificar uma função computação que reduz uma linguagem na outra.

Ex: $A_{MT} \leq_m V_{MT}$

Não é possível construir uma redução, como pede o exercício 5.5 do livro.

Redutibilidade por Mapeamento

Teorema 5.28: Se $A \leq_m B$ e B é Turing-reconhecível, então A é Turing-reconhecível.

Prova: Seja M um reconhecedor para B e f é uma redução de A para B. A seguinte MT N reconhece A:
N = "Sobre a entrada w

1. Compute $f(w)$
2. Execute M sobre $f(w)$ e dê como saída o que M apresentar.

Claramente, se $w \in A$ então $f(w) \in B$ pois f é uma redução. Assim M aceita $f(w)$ quando $w \in A$.

Redutibilidade por Mapeamento

Corolário 5.29: Se $A \leq_m B$ e A não é Turing-reconhecível, então B não é Turing-reconhecível.

Questão: Qual linguagem estudada não é Turing-Reconhecível ?

Redutibilidade por Mapeamento

Corolário 5.29: Se $A \leq_m B$ e A não é Turing-reconhecível, então B não é Turing-reconhecível.

Questão: Qual linguagem estudada não é Turing-Reconhecível ?
Resposta: $\overline{A_{MT}}$

Redutibilidade por Mapeamento

Corolário 5.29: Se $A \leq_m B$ e A não é Turing-reconhecível, então B não é Turing-reconhecível.

- A definição de redutibilidade por mapeamento implica que :

$$A \leq_m B \text{ significa o mesmo que } \overline{A} \leq_m \overline{B}$$

Para provar que B não é reconhecível, podemos mostrar que $A_{MT} \leq_m \overline{B}$

Teorema 5.30: EQ_{MT} não é nem Turing-reconhecível nem co-Turing-reconhecível.

Prova: Primeiro mostramos que EQ_{MT} não é Turing-reconhecível. Fazemos isso mostrando que $A_{MT} \leq_m \overline{EQ_{MT}}$. A função redutora f funciona da seguinte forma.

F = "sobre a entrada $\langle M, w \rangle$ onde M é uma MT e w, uma cadeia:

1. Construa as seguintes máquinas M_1 e M_2

   M_1 = "Sobre qualquer entrada:

   1. Rejeite"

   M_2 = "Sobre qualquer entrada:

   1. Rode M sobre w.
   2. Se ela aceita, *aceite*."

2. Dê como saída $\langle M_1, M_2 \rangle$"

EQ_{MT} não é Turing-reconhecível

$A_{MT} \leq_m \overline{EQ_{MT}}$.

Vamos analisar por que isto funciona:

- $\langle M, w \rangle \in A_{MT} \rightarrow M$ aceita $w \rightarrow L(M_2) = \Sigma^* \rightarrow L(M_1) \neq L(M_2) \rightarrow \langle M_1, M_2 \rangle \in \overline{EQ_{MT}}$
- $\langle M, w \rangle \notin A_{MT} \rightarrow M$ não aceita $w \rightarrow L(M_2) = \oslash \rightarrow L(M_1) = L(M_2) \rightarrow \langle M_1, M_2 \rangle \notin \overline{EQ_{MT}}$

Teorema 5.30: EQ_{MT} não é nem Turing-reconhecível nem co-Turing-reconhecível.

Prova: (continuação) Agora mostramos que $\overline{EQ_{MT}}$ não é Turing-reconhecível. Fazemos isso mostrando que $A_{MT} \leq_m EQ_{MT}$.
A MT G computa a função redutora g.
G = "sobre a entrada $\langle M, w \rangle$ onde M é uma MT e w, uma cadeia:

1. Construa as seguintes máquinas M_1 e M_2
   M_1 = "Sobre qualquer entrada:
   1. Aceite"

   M_2 = "Sobre qualquer entrada:
   1. Rode M sobre w.
   2. Se ela aceita, *aceite*."
2. Dê como saída $\langle M_1, M_2 \rangle$"

$\overline{EQ_{MT}}$ não é Turing-reconhecível

$A_{MT} \leq_m EQ_{MT}$.
Vamos analisar por que isto funciona:

- $\langle M, w\rangle \in A_{MT} \rightarrow M$ aceita $w \rightarrow L(M_2) = \Sigma^* \rightarrow L(M_1) = L(M_2) \rightarrow \langle M_1, M_2\rangle \in EQ_{MT}$
- $\langle M, w\rangle \notin A_{MT} \rightarrow M$ não aceita $w \rightarrow L(M_2) = \oslash \rightarrow L(M_1) \neq L(M_2) \rightarrow \langle M_1, M_2\rangle \notin EQ_{MT}$

A única diferença entre f e g está na máquina M_1. Em f, a máquina M_1 sempre rejeita enquento em g ela sempre aceita. Em ambas f e g, M aceita w sse M_2 sempre aceita.