Teoria da Computação
Indecidibilidade

Leonardo Takuno
{leonardo.takuno@gmail.com}

Centro Universitário Senac

Sumário

Sumário

Apresentação

- Apresentar outros problemas indecidíveis
- Redutibilidade
  - Método para relacionar dois problemas indecidíveis.
  - Mapeamento de redutibilidade

Sumário

Redutibilidade

- Dizemos que um problema Q reduz ao problema P se podemos usar P para resolver Q.
- No contexto da Teoria da computabilidade dizemos:
  - Se A for redutível a B e B for decidível, A também será decidível; e
  - Se A for redutível a B e A for indecidível, B também será indecidível.
- Isto permite provar que vários problemas são indecidíveis.

Redutibilidade

- Em outras palavras:
  - Para mostrar que um novo problema A é decidível, mostre como transformá-lo em um problema decidível conhecido B para que a solução de B possa ser usada para resolver A.
  - Para mostrar que um novo problema A é indecidível, mostre como transformar um problema indecidível conhecido B em A para que A possa ser usado para resolver B.
- Redutibilidade é uma das técnicas mais importantes e mais usadas na teoria da computação.

Redutibilidade

Vamos utilizar a redutibilidade para provar o seguinte:

Teorema 5.1: A linguagem

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

é indecidível.

- Sabemos que $A_{MT} = \{\langle M, w\rangle | M \text{ é uma MT que aceita } w\}$ é indecidível.
- Idéia:
  - Suponha que $PARA_{MT}$ é decidível
  - Mostre que podemos usar $PARA_{MT}$ para decidir A_{MT}. (Redução)
  - Conclua que A_{MT} é decidível. Contradição.

Redutibilidade

Teorema 5.1: A linguagem

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

é indecidível.

Prova: Prova por contradição. Supomos que $PARA_{MT}$ seja decidível e usamos essa suposição para mostrar que A_{MT} é decidível, contradizendo o Teorema 4.11. Suponha que MT R decida $PARA_{MT}$.

Redutibilidade

Teorema 5.1: A linguagem

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

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. Rode MT R sobre a entrada $\langle M, w\rangle$
2. Se R rejeita, *rejeite*
3. Se R aceita, simule M sobre w até que ela pare
4. Se M aceita, *aceite*; se M rejeita, *rejeite* ”

Redutibilidade

Nós vimos que A_{MT} é indecidível, logo é uma contradição e nossa hipótese de que $PARA_{MT}$ é decidível deve ser incorreta. $\square$

Redutibilidade

Teorema 5.2: A linguagem

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

é indecidível.

- Utilizando a idéia anterior:
  - Seja R uma MT que decida V_{MT}
  - Usaremos R para construir uma máquina MT S que decide A_{MT}
  - Prove A_{MT} é decidível. Contradição.
  - Então a MT R não pode existir.

Redutibilidade

Teorema 5.2: A linguagem

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

é indecidível.

Como S funcionará quando ela receber a entrada $\langle M, w\rangle$

- **Idéia 1:**
  - Rodar R sobre a entrada $\langle M\rangle$ e ver se aceita.
  - Se aceita, sabemos que $L(M) = \oslash$, então M não aceita w.
  - Mas se R rejeita $\langle M\rangle$, M aceita alguma cadeia, mas não sabemos se M aceita w.
  - **Assim, usaremos uma idéia diferente**.

Redutibilidade

Teorema 5.2: A linguagem

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

é indecidível.

Como S funcionará quando ela receber a entrada $\langle M, w \rangle$

- **Idéia 2:**
  - Em vez de rodar R sobre $\langle M \rangle$, rodamos R sobre uma modificação de $\langle M \rangle$, a qual chamaremos de $M1$.
  - $M1$ rejeita todas as cadeias, exceto w.
  - Então usamos R para determinar se M1 reconhece uma linguagem vazia.
  - Se a linguagem é vazia, então $M1$ não aceita w, caso contrário, $M1$ aceita w.

Redutibilidade

Teorema 5.2: A linguagem

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

é indecidível.

Prova: Por contradição. Assuma que V_{MT} é decidível e R é o decisor, e aí mostramos que A_{MT} é decidível - uma contradição. Supomos que a MT R decide V_{MT} e construímos a MT S que decide A_{MT}.

Redutibilidade

Teorema 5.2: A linguagem

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

é indecidível.

S = "Sobre a entrada $\langle M, w \rangle$, uma codificação de uma MT M e uma cadeia w:

1. Construa a máquina M1 a seguir :
   M1 = " Sobre a entrada x
   1. Se $x \neq w$, rejeite
   2. Se $x = w$, rode M sobre a entrada w e aceite se M aceita.
2. Execute R sobre $\langle M1 \rangle$
3. Se R aceita, rejeite, se R rejeita, aceite." $\square$

Redutibilidade

Teorema 5.3: A linguagem

$$REGULAR_{MT} = \{\langle M \rangle | \text{M é MT e L(M) é uma linguagem regular}\}$$

é indecidível.

- Suponha que $REGULAR_{MT}$ é decidível por uma MT R.
- Usaremos essa suposição para construir uma MT S que decide A_{MT} - uma contradição.

Redutibilidade

Teorema 5.3: A linguagem

$$REGULAR_{MT} = \{\langle M \rangle | \text{M é MT e L(M) é uma linguagem regular}\}$$

é indecidível.

- **idéia:** Sobre a entrada $\langle M, w \rangle$, modificar M de modo que a máquina resultante M2 reconhece uma linguagem regular se, e somente se, M reconhece w. Assim, temos
    - Se $w \in L(M)$ então $L(M2) = \Sigma^*$ (Uma linguagem regular qualquer)
    - Se $w \notin L(M)$ então $L(M2) = \{0^n 1^n | n \geq 0\}$ (Uma linguagem qualquer que não seja regular)

Redutibilidade

Teorema 5.3: A linguagem

$$REGULAR_{MT} = \{\langle M\rangle | \text{M é MT e L(M) é uma linguagem regular}\}$$

é indecidível.

A máquina S decide A_{MT}, construída usando R é:

S = "Entrada $\langle M, w\rangle$, onde M é uma MT e w é uma string

1. Construa uma MT M2:
   M2 = Entrada string x
   1. Se x tem a forma 0^n1^n, aceite
   2. Se x não tem essa forma, então executa M sobre a entrada w e aceita, se M aceita w.
2. Execute R sobre a entrada $\langle M2\rangle$
3. Se R aceita, *aceite*; se R rejeita, *rejeite* ." □

Redutibilidade

Teorema 5.4: A linguagem

$$EQ_{MT} = \{\langle M1, M2\rangle | M_1 \text{ e } M_2 \text{ são MTs e } L(M_1) = L(M_2)\}$$

é indecidível.

- $V_{MT} = \{\langle M\rangle|$ M é uma MT L(M) $= \emptyset\}$
- V_{MT} é redutível a EQ_{MT}
- Mostre que se EQ_{MT} fosse decidível, V_{MT} também seria.

Redutibilidade

Teorema 5.4: A linguagem

$$EQ_{MT} = \{\langle M1, M2\rangle | M_1 \text{ e } M_2 \text{ são MTs e } L(M_1) = L(M_2)\}$$

é indecidível.

- Mostre que se EQ_{MT} fosse decidível, V_{MT} também seria.
  - Assuma que uma MT R decide EQ_{MT} e construa uma MT S para decidir V_{MT}
  - Utilize uma MT M1 que rejeite todas as entradas, ou seja, tenha linguagem vazia
  - Utilize R para comparar M com M1 e verificar se $L(M) = \emptyset$, aceitando ou rejeitando.
  - Conclua que V_{MT} é decidível. Contradição.

Redutibilidade

Teorema 5.4: A linguagem

$$EQ_{MT} = \{\langle M1, M2\rangle | M_1 \text{ e } M_2 \text{ são MTs e } L(M_1) = L(M_2)\}$$

é indecidível.

Prova: Por contradição. Assuma que EQ_{MT} é decidível e R é um decisor. Mostramos que V_{MT} reduz EQ_{MT} por construir um decisor S para decidir V_{MT}.

Redutibilidade

Teorema 5.4: A linguagem

$$EQ_{MT} = \{\langle M1, M2\rangle | M_1 \text{ e } M_2 \text{ são MTs e } L(M_1) = L(M_2)\}$$

é indecidível.

S = "Sobre a entrada $\langle M \rangle$, onde M é uma MT:

1. Rode R sobre a entrada $\langle M, M_1 \rangle$, onde M_1 é uma MT que rejeita todas as entradas.
2. Se R aceita, *aceite*; se R rejeita, *rejeite*."

Se R decide EQ_{MT}, S decide V_{MT}. Mas V_{MT} é indecidível pelo Teorema 5.2 portanto EQ_{MT} também tem de ser indecidível. $\square$

Sumário

Teorema de Rice

Teorema: Todo teste de qualquer propriedade (não trivial) de linguagens reconhecidas por máquinas de Turing é indecidível.

Uma classe de problemas: Seja P uma linguagem constituída de descrições de máquinas de Turing, em que P satisfaz duas condições.

1. P é não-trivial: ela contém alguma descrição, mas não todas as descrições de MTs.
   - existe uma MT M1 para o qual $\langle M1 \rangle \in P$, e
   - existe uma MT M2 para o qual $\langle M2 \rangle \notin P$
2. P é uma propriedade da linguagem da linguagem da MT

$$\text{se } L(M1) = L(M2) \text{ então } \langle M1 \rangle \in P \text{ sse } \langle M2 \rangle \in P$$

Teorema de Rice

Teorema: Todo teste de qualquer propriedade (não trivial) de linguagens reconhecidas por máquinas de Turing é indecidível.

exemplo:

- Testar se a linguagem reconhecida por uma MT é livre do contexto.
- Testar se linguagem reconhecida por uma MT é decidível.
- Testar se a linguagem reconhecida por uma máquina de Turing é finito.

Teorema de Rice

Teorema: Todo teste de qualquer propriedade (não trivial) de linguagens reconhecidas por máquinas de Turing é indecidível.

Prova: Por contradição. Seja P uma propriedade não-trivial.
Queremos mostrar que

$$L_P = \{\langle M \rangle | L(M) \text{ satisfaz } P\},$$

é indecidível.
Assuma que L_P seja decidível e M_P é um decisor. Mostramos que podemos construir um decisor S para A_{MT}.

Teorema de Rice

Teorema: Todo teste de qualquer propriedade (não trivial) de linguagens reconhecidas por máquinas de Turing é indecidível.

S = "Sobre a entrada $\langle M, w \rangle$, onde M é uma MT e w uma string:

1. Use MT M e w para construir a seguinte MT M':
   M' = Sobre a entrada $\langle T, x \rangle$, onde T é uma MT e x uma string
   1. Simule M sobre w. Se pára e rejeita, *rejeite*. se aceita, proceda os estágio 2
   2. Simule T sobre x. Se aceita, *aceite* (note: L_P não é trivial, então $\langle T \rangle \in L_P$ tem que existir.)
2. Use M_P para determinar se $\langle M' \rangle \in L_P$. Se aceita, *aceite*. Caso contrário, rejeite."

Teorema de Rice

A MT M' simula T se M aceita w. Logo, $L(M')$ é igual a $L(T)$ se M aceita w e é igual a 0, em caso contrário. Portanto, $\langle M, w \rangle \in P$ sse M aceita w.
Visto que A_{MT} não é decidível, esta máquina não pode existir e nossa hipótese que L_P é decidível deve ser incorreta. $\square$

Sumário

Reduções via Histórias de Computação

Definição: História de computação é uma seqüência de configurações, C_1, C_2 ,... , C_l, onde C_1 é a configuração inicial de M sobre w, C_l é uma configuração de aceitação ou de rejeição de M, e cada C_i segue C_{i-1} conforme as regras de M.

Autômato linearmente limitado

Definição: Um **autômato linearmente limitado** é um tipo restrito de máquina de Turing na qual à cabeça de leitura-escrita não é permitido mover-se para fora da parte da fita contendo a entrada.

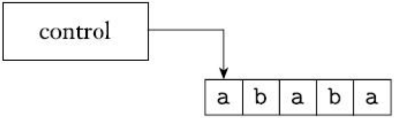


FIGURA 5.7
Esquemática de um autômato linearmente limitado

Autômato linearmente limitado

Lema 5.8: Seja M um ALL com q estados e g símbolos no alfabeto de fita. Existem exatamente qng^n configurações distintas distintas de M para uma fita de comprimente n.

Prova: M tem q estados. O comprimento da sua fita é n, portanto, a cabeça pode estar em uma das n posições e g^n cadeias possíveis de símbolos de fita aparecem sobre a fita. O produto dessas três quantidades é o número total de configurações diferentes de M com uma fita de comprimento n. $\square$

Autômato linearmente limitado

Teorema 5.9:

$$A_{ALL} = \{\langle M, w\rangle | \text{M é um ALL que aceita a cadeia } w\}$$

é decidível.

Prova: O algoritmo que decide A_{ALL} é como segue.
L = "Sobre a entrada $\langle M, w\rangle$, onde M é um ALL e w é uma cadeia:

1. Simule M sobre w por qng^n passos ou até que ela pare.
2. Se M parou, *aceite* se ela aceitou e *rejeite* se ela rejeitou. Se ela não parou, *rejeite*."

Autômato linearmente limitado

Teorema 5.9:

$$A_{ALL} = \{\langle M, w\rangle | \text{M é um ALL que aceita a cadeia } w\}$$

é decidível.

Se M sobre w não parou dentro de qng^n passos, ela tem que estar repetindo uma configuração conforme o Lema 5.8 e, consequentemente, estar em loop. É por isso que nosso algoritmo rejeita nessa instância. □

Autômato linearmente limitado

Teorema 5.10:

$$V_{ALL} = \{\langle M, w\rangle | \text{M é um ALL onde } L(M) = \oslash\}$$

é indecidível.

Prova: Construímos um ALL B para aceitar uma entrada x se x é uma história de computação de aceitação para M sobre w. Assumimos que a história de computação de aceitação é apresentada como uma única cadeia, com as configurações separadas umas das outras pelo símbolo $\#$ como a figura 5.11.

$$\#\underbrace{\qquad\qquad}_{C_1}\#\underbrace{\qquad\qquad}_{C_2}\#\underbrace{\qquad\qquad}_{C_3}\#\quad\cdots\quad\#\underbrace{\qquad\qquad}_{C_l}\#$$

FIGURA **5.11**
Uma possível entrada para B

Autômato linearmente limitado

Teorema 5.10:

$$V_{ALL} = \{\langle M, w\rangle | \text{M é um ALL onde } L(M) = \oslash\}$$

é indecidível.

Prova: O ALL B, então obtem a entrada x e verifica se é uma história de computação de aceitação,o qual deve satisfazer as três condições:

1. C_1 é a configuração inicial para M sobre w.
2. Cada C_{i+1} segue legitimamente de C_i.
3. C_l é uma configuração de aceitação para M.

Obs: Montamos o ALL B para alimentar o decisor de V_{ALL} que pressupomos existir

Autômato linearmente limitado

Teorema 5.10:

$$V_{ALL} = \{\langle M, w\rangle | \text{M é um ALL onde } L(M) = \oslash\}$$

é indecidível.

Prova: Agora estamos prontos para enunciar a redução de A_{MT} para V_{ALL}. Suponha que MT R decide V_{ALL}. Construa MT S que decide A_{MT}.

Autômato linearmente limitado

Teorema 5.10:

$$V_{ALL} = \{\langle M, w \rangle | \text{M é um ALL onde } L(M) = \emptyset\}$$

é indecidível.

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

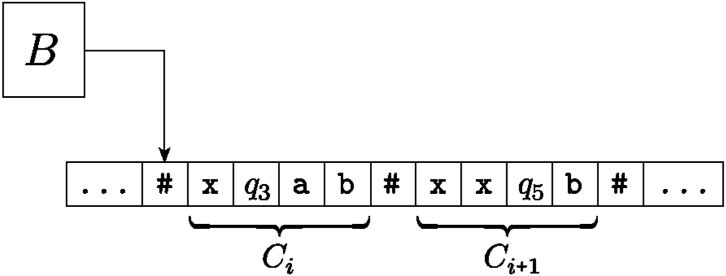
1. Construa o ALL B a partir de M e w conforme descrito.
2. Rode R sobre a entrada $\langle B \rangle$.
3. Se R rejeita, aceite; se R aceita, rejeite."

Autômato linearmente limitado

Se R aceita $\langle B \rangle$, então $L(B) = \emptyset$. Então, M não tem nenhuma história de computação de aceitação sobre w e M não aceita w. Conseqüentemente, S rejeita $\langle M, w \rangle$. Similarmente, se R rejeita $\langle B \rangle$, a linguagem de B é não vazia. A MT B aceita uma história de computação de aceitação para M sobre w. Portanto, M deve aceitar w. Como conseqüência, S aceita $\langle M, w \rangle$.

$\square$

Autômato linearmente limitado



Sumário

Problemas indecidíveis sobre GLCs

Teorema 5.10:

$$TODAS_{GLC} = \{\langle G \rangle | \text{G é uma GLC e } L(G) = \Sigma^*\}$$

é indecidível.

Prova: Para uma MT M e uma entrada w, construimos uma GLC G para gerar todas as strings que não são histórias de computação de aceitação para M sobre w.

Isto é, G gera todas as strings se e somente se M não aceita w.

Se $TODAS_{GLC}$ fosse decidível então A_{MT} também seria.

Problemas indecidíveis sobre GLCs

Teorema 5.10:

$$TODAS_{GLC} = \{\langle G\rangle | \text{G é uma GLC e } L(G) = \Sigma^*\}$$

é indecidível.

Prova: Assuma que $TODAS_{GLC}$ é decidível. Construímos uma Autômato de Pilha D que aceita a string $\#C_1\#C_2^{\mathcal{R}}\#C_3\#C_4^{\mathcal{R}}\#...\#C_l\#$, tal que $\#C_1\#C_2\#C_3\#C_4\#...\#C_l\#$ não represente uma história de computação de aceitação de M sobre w.

Problemas indecidíveis sobre GLCs

Teorema 5.10:

$$TODAS_{GLC} = \{\langle G \rangle | \text{G é uma GLC e } L(G) = \Sigma^*\}$$

é indecidível.

D = "Sobre a entrada $\#C_1\#C_2^{\mathcal{R}}\#C_3\#C_4^{\mathcal{R}}\#\ldots\#C_l\#$:

1. Se C_1 não é o estado inicial de M, então *aceite*
2. Se C_l não é o estado de aceitação de M, então *aceite*
3. Se C_i não produz C_{i+1}, então *aceite*"

No terceiro passo, D usa a pilha efetivamente.

Problemas indecidíveis sobre GLCs

Teorema 5.10:

$$TODAS_{GLC} = \{\langle G \rangle | \text{G é uma GLC e } L(G) = \Sigma^*\}$$

é indecidível.

Note que :

- $L(D) = \Sigma^* \Leftrightarrow M$ não aceita w, e
- $L(D) \neq \Sigma^* \Leftrightarrow M$ aceita w

$\square$