ACH2043
INTRODUÇÃO À TEORIA DA COMPUTAÇÃO

Aula 17

Cap 5.3 – Redutilibidade por Mapeamento

Profa. Ariane Machado Lima
ariane.machado@usp.br

Na aula passada...

- Redução: conversão de um problema A em outro problema B de forma que a solução de B seja usada para solucionar A
- Se A é redutível a B
  - A não pode ser mais difícil do que B
  - Se B for decidível, A também será
  - Se A for indecidível, B também será
- Provamos que vários problemas são indecidíveis

Na aula de hoje

- Pularemos o cap. 5.2
- Formalização de uma redução
- Redutibilidade por mapeamento
- Mapeamento por uma função f computável
- Funções computáveis e funções não-computáveis

Funções computáveis

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

Funções computáveis

- 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
- Uma função é não-computável se não existe tal máquina (por mais que se possa calcular o valor de f para **alguns** pontos do Domínio)

Termos equivalentes ou relacionados

- Problema solúvel, problema ou linguagem decidível, linguagem recursiva
  - Função computável
- Problema insolúvel, problema ou linguagem indecidível ou semi-decidível (mas reconhecível), linguagem recursivamente enumerável não-recursiva
  - Função incomputável
- Problema completamente insolúvel, problema ou linguagem indecidível e irreconhecível, linguagem não recursivamente enumerável
  - Função incomputável

Exemplos de funções computáveis

- Operações aritméticas sobre inteiros
- Transformações em descrições de máquinas de Turing
  - Ex: f(<M>) = <M'>, onde M' reconhece a mesma linguagem que M, mas nunca tenta mover a cabeça de fita para além da extremidade esquerda (adicionando estados). Retorna ε se M não for uma descrição de uma MT legítima

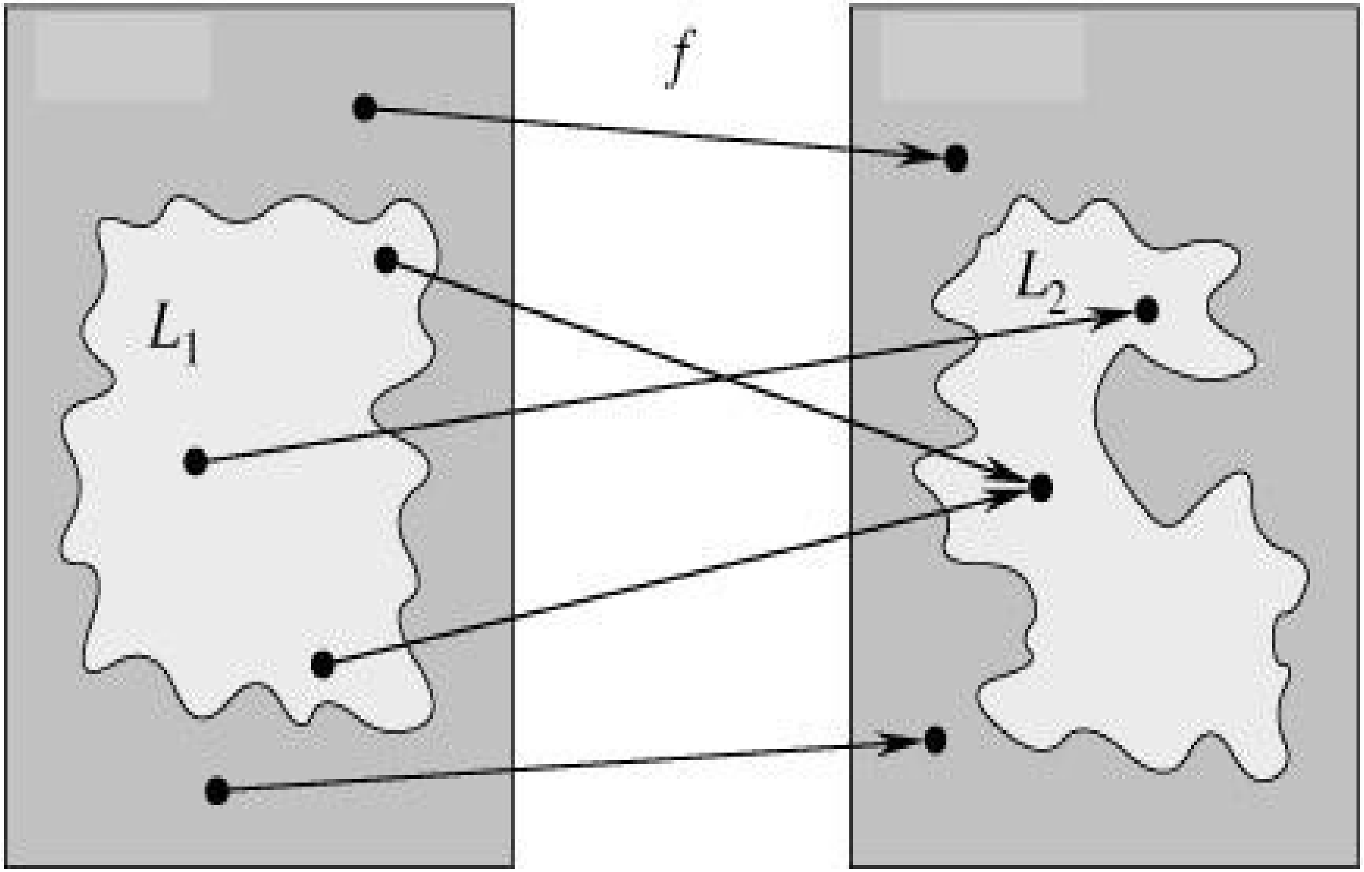
Definição formal de redutibilidade por mapeamento

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

w pertence a A <=> f(w) pertence a B.

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

Definição formal de redutibilidade por mapeamento



Definição formal de redutibilidade por mapeamento

- **Teorema**: Se $A \leq_m B$ e B é decidível, então A é decidível.
- **Prova**: Seja M o decisor de B e f a redução de A para B. Um decisor N para A é:

N = “Sobre a entrada w:

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

Se w pertence a A, f(w) pertence a B.

Portanto M aceita f(w) sempre que w pertencer a A e rejeita caso contrário.

Logo, N decide A.

Definição formal de redutibilidade por mapeamento

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

Exemplo - $PARA_{MT}$

- Redução de A_{MT} para $PARA_{MT}$
- Temos que mostrar uma função computável f onde:

$x \in A_{MT} \Leftrightarrow f(x) \in PARA_{MT}$

ou seja,

Exemplo - $PARA_{MT}$

- Redução de A_{MT} para $PARA_{MT}$
- Temos que mostrar uma função computável f onde:

$x \in A_{MT} \iff f(x) \in PARA_{MT}$

ou seja,

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

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

Exemplo - $PARA_{MT}$

- Redução de A_{MT} para $PARA_{MT}$
- Temos que mostrar uma função computável f onde:

$x \in A_{MT} \iff f(x) \in PARA_{MT}$

ou seja,

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

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

- Temos que mostrar uma MT F que compute f

Exemplo - $PARA_{MT}$

- Temos que mostrar uma MT F que compute f:

F = “Sobre a entrada <M,w>:

1. Construa a seguinte máquina M'

   M' = “Sobre a entrada x:

   1. Rode M sobre x
   2. Se M aceita, *?*
   3. Se M rejeita, *?*”

2. Dê como saída <M', w>”

-

Exemplo - $PARA_{MT}$

- Temos que mostrar uma MT F que compute f:

F = “Sobre a entrada <M,w>:

1. Construa a seguinte máquina M'

   M' = “Sobre a entrada x:

   1. Rode M sobre x
   2. Se M aceita, *aceite*
   3. Se M rejeita, entre em *loop*”

2. Dê como saída <M', w>”

-

Exemplo - $PARA_{MT}$

- Temos que mostrar uma MT F que compute f:

F = “Sobre a entrada <M,w>:

1. Construa a seguinte máquina M'

    M' = “Sobre a entrada x:

    1. Rode M sobre x
    2. Se M aceita, *aceite*
    3. Se M rejeita, entre em *loop*”

2. Dê como saída <M', w>”

- Obs.: Se uma entrada y não está na forma correta (e portanto não pertence a A), f(y) deve dar como saída uma cadeia que não pertence a B.

Diferença da prova da aula passada para a da aula de hoje

- Prova de que $PARA_{MT}$ é indecidível utilizando A_{MT}
- Em ambos os casos, supomos que existe uma MT R que decide $PARA_{MT}$
- Aula passada (redução informal)
  - Utilizamos R sobre a entrada <M,w> para decidir A_{MT} sobre o mesmo <M,w>
- Aula de hoje (redução formal por mapeamento)
  - Criamos uma MT F que mapeia cada cadeia de A_{MT} em uma cadeia de $PARA_{MT}$ (de <M,w> no problema A_{MT} computamos <M',w>=F(<M,w>) para o problema $PARA_{MT}$)
  - A resposta de R sobre <M', w> é a resposta para <M,w> no problema A_{MT}
  - Ou seja, um decisor N para A_{MT} é R(F(<M,w>)

Para ver se entenderam....

- Vamos ver uma prova da aula passada...
- Diga como deveria ser a redução por mapeamento

Equivalência entre MTs

- EQ_{MT} = {<M1, M2> | M1 e M2 são MTs e L(M1) = L(M2)}
- Podemos usar EQ_{MT} para resolver V_{MT}!
- Ideia: se uma MT M for equivalente a outra que rejeita qualquer cadeia, então L(M) = Ø
- Assuma que R é uma MT que decide EQ_{MT}
- Vamos construir S que decide V_{MT} usando R

Equivalência entre MTs

- S = “Sobre a entrada <M> onde M é uma MT:
  1. Rode R sobre a entrada <M, M1>, onde M1 é uma MT que rejeita todas as entradas.
  2. Se R aceita, *aceite*; se R rejeita, *rejeite*.”
- Mas V_{MT} é indecidível, então EQ_{MT} também é

Equivalência entre MTs

- S = "Sobre a entrada <M> onde M é uma MT:
  1. Rode R sobre a entrada <M, M1>, onde M1 é uma MT que rejeita todas as entradas.
  2. Se R aceita, *aceite*; se R rejeita, *rejeite*."
- Mas V_{MT} é indecidível, então EQ_{MT} também é
- Como seria a redução por mapeamento?

Equivalência entre MTs

- Assuma que R é uma MT que decide o problema EQ_{MT}
- Preciso escrever uma MT F que faça o mapeamento de V_{MT} em EQ_{MT}, de forma que um decisor de V_{MT} para uma dada entrada <M> seja R(F(<M>))

Equivalência entre MTs

- F = “Sobre a entrada <M>, onde M é uma MT:

  1. Construa a seguinte máquina M1

     M1 = “Sobre a entrada x:

     rejeite.”

  2. Dê como saída <M, M1>”

Outro exercício considerando uma prova da aula passada...

Vacuidade de uma linguagem de uma MT

- V_{MT} = { <M>: M é uma MT e L(M) = Ø}
- Usar um decisor R de V_{MT} para decidir A_{MT}
- Ideia: construir uma versão de M que apenas teste w

M1 = "Sobre a entrada x:

1. Se x ≠ w *rejeite*
2. Se x = w, rode M sobre a entrada w e *aceite* se M aceita, e *rejeite* se M rejeita"

Vacuidade de uma linguagem de uma MT

- S = “Sobre a entrada <M,w>, uma codificação de uma MT M e uma cadeia w:
  1. Use a descrição de M e w para construir M1
  2. Rode R sobre M1
  3. Se R aceita, *rejeite*; se R rejeita, *aceite*.”

Mas como A_{MT} é indecidível, V_{MT} é indecidível

Vacuidade de uma linguagem de uma MT

- S = “Sobre a entrada <M,w>, uma codificação de uma MT M e uma cadeia w:
  1. Use a descrição de M e w para construir M1
  2. Rode R sobre M1
  3. Se R aceita, *rejeite*; se R rejeita, *aceite*.”

Mas como A_{MT} é indecidível, V_{MT} é indecidível

Como fazer uma redução por mapeamento?

Vacuidade de uma linguagem de uma MT

- Uma MT F que receba <M,w> e dê como saída M1 faz um mapeamento entre A_{MT} e o complemento de V_{MT}!
- Logo, formalmente, provou-se que o complemento de V_{MT} é indecidível
- Na verdade, não existe uma redução por mapeamento de A_{MT} para V_{MT}
- A prova de que V_{MT} é indecidível ainda funciona porque a decidibilidade **não** é afetada por complementação

Resumindo...

- Sei que o problema A é indecidível. Quero provar que o problema B é indecidível. Como?
- Prova por contradição: assumo que B é decidível por uma MT R. Se esse R puder ser usado para decidir o problema o A, CONTRADIÇÃO! Logo B é indecidível.
- O que falta na prova é mostrar como R pode ser usado para decidir A.
- Usando informalmente “redução”, essa solução era criada caso a caso.
- Em redução por mapeamento, a solução é sempre a mesma:

Resumindo...

- Um decisor D de A seria:

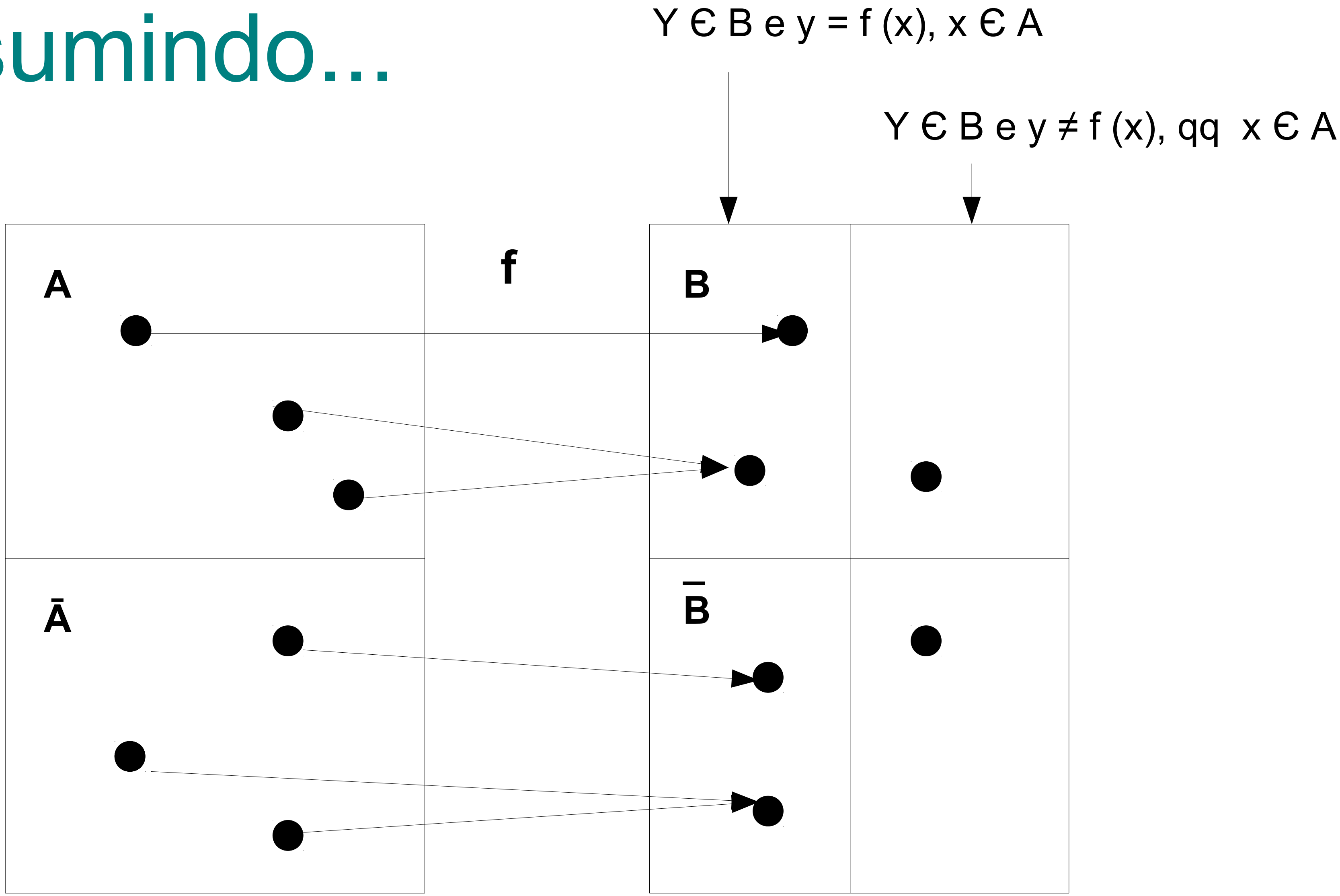
  D = “Sobre uma entrada x,

  1. Dê a resposta dada pela MT R sobre a entrada F(x).”

- Onde F é a função de mapeamento de A para B que funciona de tal forma que:

  x pertence a A <=> f(x) pertence a B

Resumindo...



Resumindo

A tarefa fica então em construir a F para um dado A e um dado B

Redutibilidade por mapeamento e reconhecibilidade

- **Teorema**: Se $A \leq_m B$ e B é Turing-reconhecível, então A é Turing-reconhecível.
- **Prova**: Seja M o reconhecedor de B e f a redução de A para B. Um reconhecedor N para A é:

N = “Sobre a entrada w:

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

Se w pertence a A, f(w) pertence a B.

Portanto M aceita f(w) sempre que w pertencer a A

Logo, N reconhece A.

Redutibilidade por mapeamento e reconhecibilidade

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

Aplicações do corolário

- Já sabemos que o complemento de A_{MT} não é Turing-reconhecível (ponto de partida para mostrar que outras linguagens também não são)
- $A \leq_m B$ implica que

complemento(A) $\leq_m$ complemento(B)

- Assim, para provar que B não é Turing-reconhecível podemos usar

complemento(A_{MT}) $\leq_m$ B

ou

$A_{MT} \leq_m$ complemento(B)

Exemplo

- **Teorema**: EQ_{MT} não é Turing-reconhecível nem co-Turing-reconhecível
- **Prova**:

Provamos que EQ_{MT} não é Turing-reconhecível e depois que complemento(EQ_{MT}) também não é

Exemplo - EQ_{MT} não é Turing-reconhecível

- $A_{MT} \leq_m$ complemento(EQ_{MT})
- F = “Sobre a entrada <M,w>, onde M é uma MT e w é uma cadeia:

  1. Construa as seguintes MTs M1 e M2:

     M1 = “Sobre qualquer cadeia de entrada:

       1. *rejeite*.”

     M2 = “Sobre qualquer cadeia de entrada:

       1. Rode M sobre w. Se M aceita, *aceite*; se M rejeita, *rejeite*.”

  2. Dê como saída <M1, M2>.”

Exemplo - complemento(EQ_{MT}) não é Turing-reconhecível

- complemento(A_{MT}) $\leq_m$ complemento(EQ_{MT}), ou seja
- $A_{MT} \leq_m EQ_{MT}$
- F = “Sobre a entrada <M,w>, onde M é uma MT e w é uma cadeia:

  1. Construa as seguintes MTs M1 e M2:

     M1 = “Sobre qualquer cadeia de entrada:

     1. *aceite*.”

     M2 = “Sobre qualquer cadeia de entrada:

     1. Rode M sobre w. Se M aceita, *aceite*; se M rejeita, *rejeite*.”

  2. Dê como saída <M1, M2>.”

Lista 5

- Exercícios 5.1 e 5.2
- Data de entrega: 29/10