ACH2043
INTRODUÇÃO À TEORIA DA COMPUTAÇÃO

Aula 24

Cap 5 – REDUTIBILIDADE
Cap 5.3 – Redutibilidade por mapeamento

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

Aula passada

Redutibilidade

Não pode ser mais fácil porque não faria sentido reduzir um problema mais fácil para um mais difícil.
Não pode ser mais difícil porque daí a solução de B não garantiria a solução de A.

- Utilidade:
  - Se A é redutível a B
    - A não pode ser mais fácil nem mais difícil do que B
    - Se B for decidível, A também será
    - Se B for reconhecível, A também será
    - Se A for indecidível, B também será
    - Se A for não-reconhecível, B também será

Se eu quero provar que um problema é **decidível/reconhecível**:
- **o problema que eu quero provar será o problema A**
- encontro um problema (B) que já sei que é decidível/reconhecível e mostro que A é redutível a B (ou seja, que a solução de B pode ser usada para solucionar A)

Redutibilidade

Não pode ser mais fácil porque não faria sentido reduzir um problema mais fácil para um mais difícil.
Não pode ser mais difícil porque daí a solução de B não garantiria a solução de A.

- Utilidade:
  - Se A é redutível a B
    - A não pode ser mais fácil nem mais difícil do que B
    - Se B for decidível, A também será
    - Se B for reconhecível, A também será
    - Se A for indecidível, B também será
    - Se A for não-reconhecível, B também será

Se eu quero provar que um problema é **indecidível/não-reconhecível**:
 - **o problema que eu quero provar será o problema B**
 - encontro um problema (A) que já sei que é indecidível e mostro que A é redutível a B (ou seja, que a solução de B poderia ser usada para solucionar A, o que é uma CONTRADIÇÃO)

Redutibilidade

- Como mostrar que um problema A é redutível a um problema B?
- Forma 1: redução informal (aulas 22 e 23 até agora)
  - escrever uma MT S que decida A usando uma MT R que decida B (se tal máquina R existir)
- Forma 2: redução formalizada por uma função de mapeamento entre os problemas A e B (redução por mapeamento) – veremos hoje
  - Daí basta aplicar a MT R (que soluciona o problema B) sobre o mapeamento de A

Redução informal (para provar indecidibilidade) - resumo

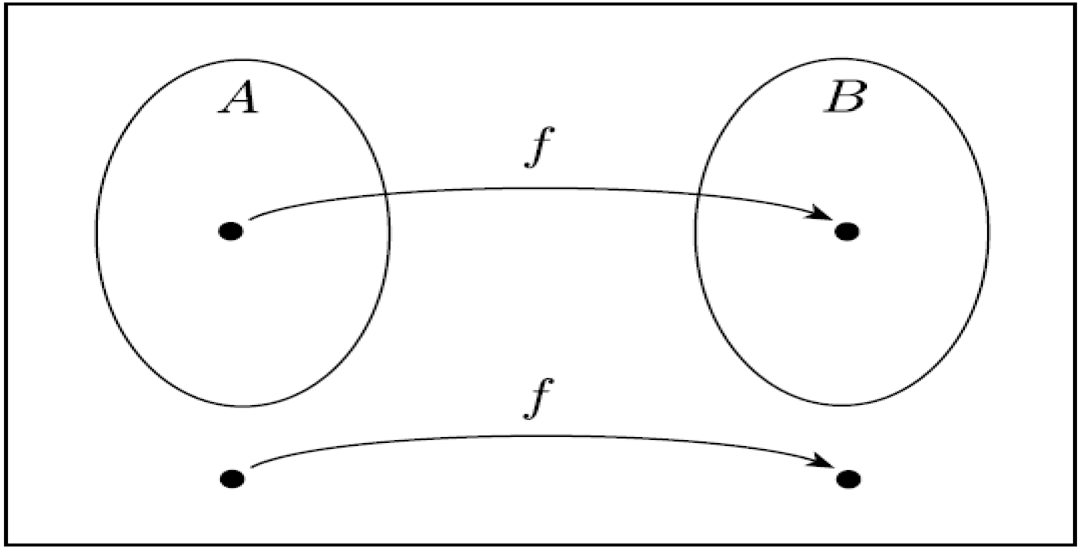
- Como provar que um problema B é indecidível usando a técnica de redutibilidade:
  - Assumo por contradição que B é decidível
  - Uso a MT decisora (R) de B para construir uma MT decisora (S) de um problema que sabemos que é indecidível (redução de A a B)
  - Contradição! Portanto R não pode existir!

Redutibilidade

- Como mostrar que um problema A é redutível a um problema B?
- Forma 1: redução informal (aula 22)
  - escrever uma MT S que decida A usando uma MT R que decida B (se tal máquina R existir)

- Forma 2: redução formalizada por uma função de mapeamento entre os problemas A e B (redução por mapeamento) – veremos hoje
  - Daí basta aplicar a MT R (que soluciona o problema B) sobre o mapeamento de A

Redutibilidade por mapeamento



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

Aula de hoje

Cap 5.3 – Redutilibidade por Mapeamento

Cap 5.3 – Redutilibidade por Mapeamento

- Mapeamento por uma função f computável
- Funções computáveis e funções não-computáveis

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

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
- 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

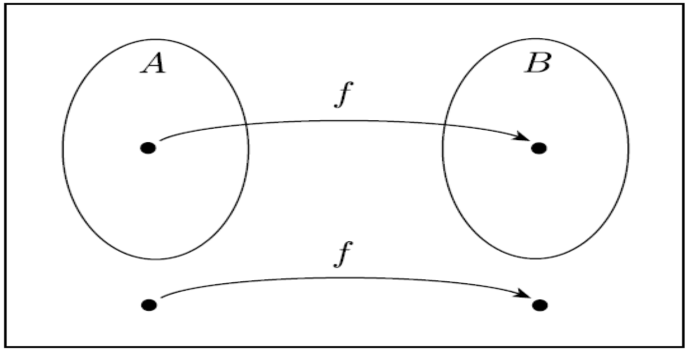
- Operações aritméticas sobre inteiros
- Transformações de descrições de máquinas de Turing
  - Ex 1: f(<M>) = <M'>, sendo que M' reconhece a mesma linguagem que M, mas nunca tenta mover a cabeça de fita para além da extremidade esquerda (faz isso adicionando estados). Retorna ε se M não for uma descrição de uma MT legítima
  - Ex 2: f(<M>) = <M’>, sendo que M’ diz o contrário de M

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 $f:\Sigma^* \rightarrow \Sigma^*$ em que para toda cadeia w,

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

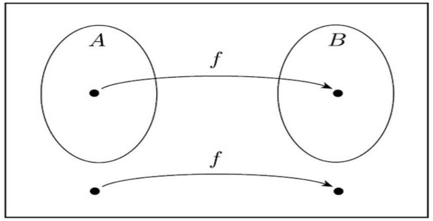
A função f é denominada a redução de A para B.



Definição formal de redutibilidade por mapeamento

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

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



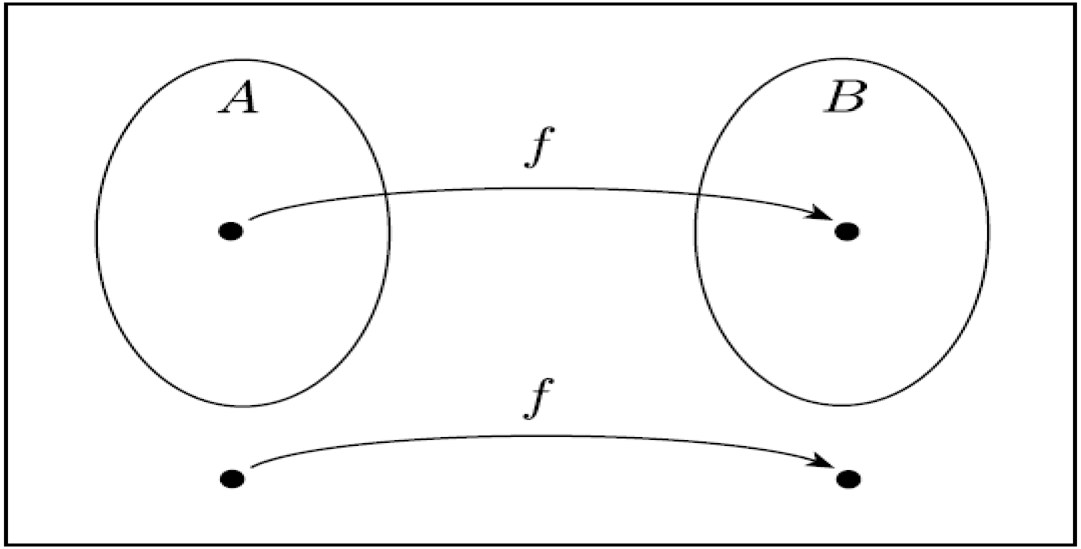
A função f é denominada a redução de A para B.

Quero solucionar A = { w | w está em determinado formato e possui determinadas características}.
Seja B = { z | z está em (potencialmente outro) determinado formato e possui (potencialmente outras) características}
Se:

- existe um solucionador para o problema B (ie, uma MT que decide B) e
- existe um **mapeamento** f que transforma uma cadeia w entrada do problema A em f(w) = z entrada do problema B

Então posso utilizar o solucionador de B sobre a entrada f(w), e assim estou solucionando A sobre w

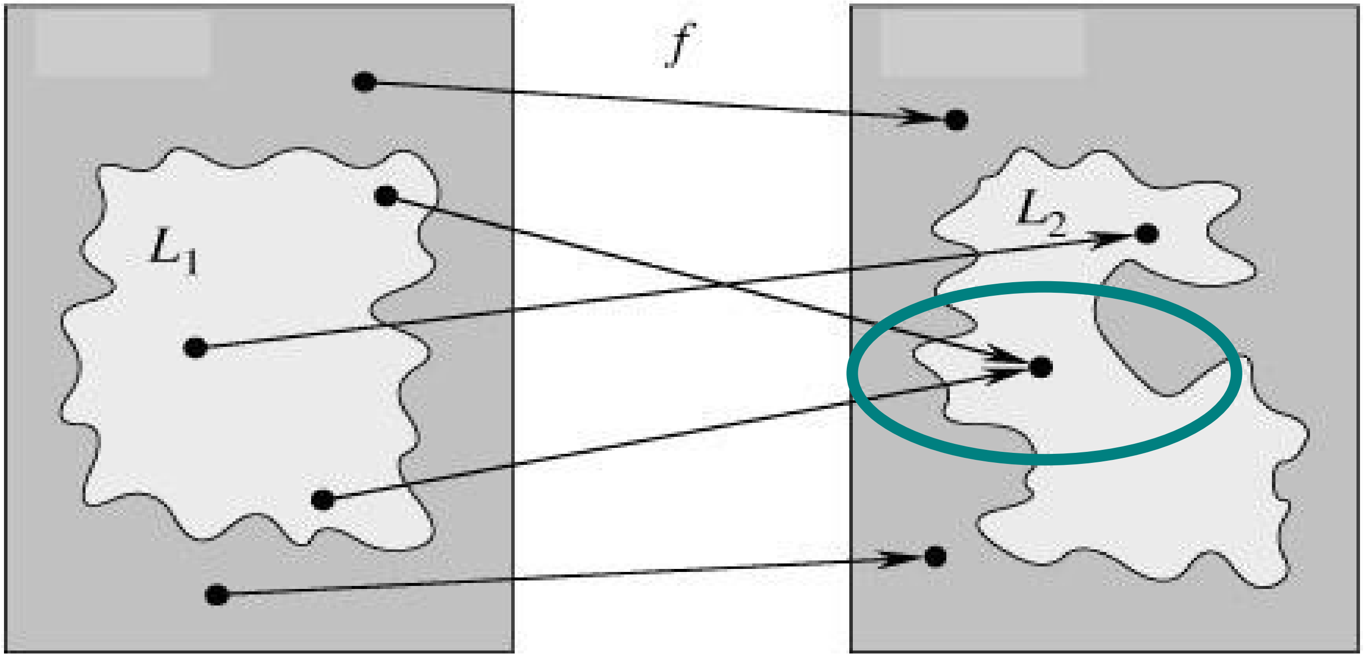
Redutibilidade por mapeamento



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

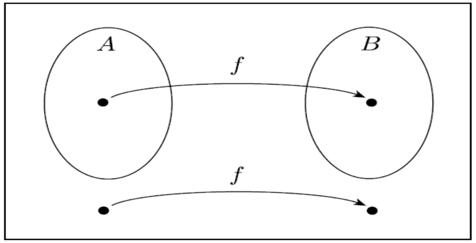
Redutibilidade por mapeamento

f não precisa ser uma função bijetora
Por isso também chamada de redutibilidade muitos-para-um



Definição formal de redutibilidade por mapeamento

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



S = “Sobre a entrada w:

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

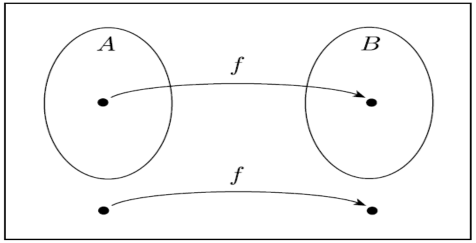
Se w pertence a A, f(w) pertence a B.

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

Logo, S decide A.

Definição formal de redutibilidade por mapeamento

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



S = “Sobre a entrada w:

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

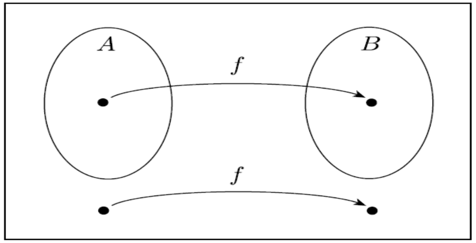
Se w pertence a A, f(w) pertence a B.

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

Logo, S decide A.

Na redutibilidade por mapeamento S será sempre essa; resta escrever F que compute f(w)

Definição formal de redutibilidade por mapeamento



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

Prova: Assuma por contradição que B seja decidível. Se A $\leq_m$ B e B é decidível (decisor R), então R(F(x)) decide A. CONTRADIÇÃO, pois A é indecidível!

Exemplo – $PARA_{MT}$
(prova por redução **por mapeamento**)

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

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

ou seja,

Exemplo – $PARA_{MT}$
(prova por redução **por mapeamento**)

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

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

ou seja,

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

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

Exemplo – $PARA_{MT}$ (prova por redução **por mapeamento**)

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

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

ou seja,

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

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

- Temos que mostrar uma MT F que compute f

Exemplo – $PARA_{MT}$
(prova por redução **por mapeamento**)

Temos que mostrar uma MT F que compute f: $A_{MT} \rightarrow PARA_{MT}$

w = <M, w> $\in$ A_{MT} , f(w) = <M´, w'> $\in$ $PARA_{MT}$

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 – $PARA_{MT}$
(prova por redução **por mapeamento**)

Temos que mostrar uma MT F que compute f: $A_{MT} \rightarrow PARA_{MT}$

w = <M, w> $\in$ A_{MT} , f(w) = <M´, w'> $\in$ $PARA_{MT}$

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, ???”

2. Dê como saída <M', w>”

•

Exemplo – $PARA_{MT}$
(prova por redução **por mapeamento**)

Temos que mostrar uma MT F que compute f: $A_{MT} \rightarrow PARA_{MT}$

$$w = \langle M, w\rangle \in A_{MT}, \quad f(w) = \langle M', w'\rangle \in PARA_{MT}$$

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>”

Poderia até ser “rejeite”! (desde que páre)
Note que se M entrar em loop M’ também entra

Exemplo – $PARA_{MT}$
(prova por redução **por mapeamento**)

Temos que mostrar uma MT F que compute f: $A_{MT} \rightarrow PARA_{MT}$

w = <M, w> Є A_{MT} , f(w) = <M´, w'> Є $PARA_{MT}$

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>”

Poderia até ser “rejeite”! (desde que páre)

Note que se M entrar em loop M’ também entra

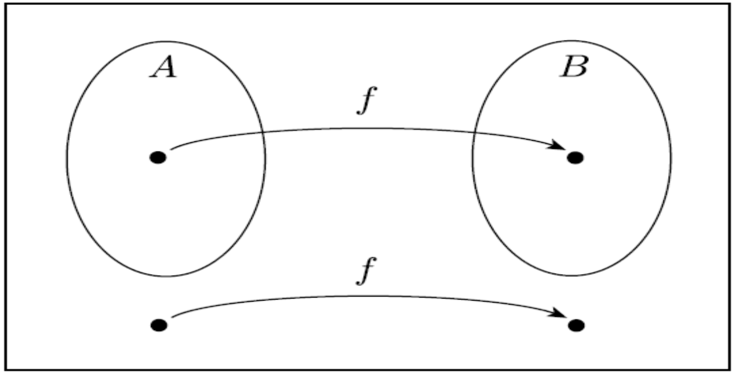
Se $PARA_{MT}$ fosse decidível por uma MT R, a MT S = R(F(M,w)) decidiria A_{MT}

CONTRADIÇÃO!!

Observação

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

(isso ficará subentendido)



Redução **por mapeamento** (para provar indecidibilidade) - resumo

- Como provar que um problema é indecidível usando a técnica de redutibilidade:
  - Assumo por contradição que ele seja decidível (será o problema B)
  - Escolho um problema A sabidamente indecidível para fazer a redução ao problema B
  - Construo uma MT F que calcula a função de redução (mapeamento) $f : A \rightarrow B$
  - Se R é a MT decisora (R) de B sobre uma entrada y, F é tal que y = F(x), a MT S decisora de A sobre uma entrada x, é S(x) = R(F(x))
  - Contradição! Portanto R não pode existir!

Diferença entre as duas reduções

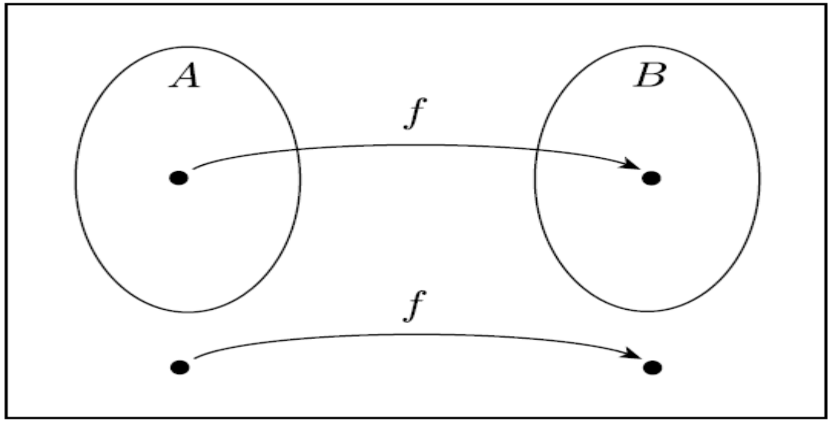
- Prova de que $PARA_{MT}$ é indecidível utilizando A_{MT}
- Em ambos os casos, supomos que existe uma MT R que decide $PARA_{MT}$
- Redução informal:

Parte que exige nossa “criatividade”

  – Construimos a MT S (utilizando R) sobre a entrada <M,w> para decidir A_{MT} sobre o mesmo <M,w>

Assuma, por contradição, que uma MT R decida $PARA_{MT}$.
Então construímos S que usa R para decidir A_{MT} :
S = “Sobre a entrada <M, w>, uma codificação de uma MT M e uma cadeia w:
1. Rode a MT R sobre a entrada <M, w>.
2. Se R rejeita, rejeite. # isto quer dizer que M não pára
3. Se R aceita, simule M sobre w até que ela pare.
4. Se M aceitou, aceite; se M rejeitou, rejeite.”

Logo A_{MT} pode ser reduzido a $PARA_{MT}$
Como A_{MT} é indecidível, $PARA_{MT}$ é indecidível



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>”

- 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 S para A_{MT} é R(F(<M,w>)

Parte que exige nossa “criatividade”

Diferença entre as duas reduções

Para um dado par de problemas, pode ser mais fácil usar um tipo ou outro de redução
(as MTs implementadas são diferentes)

Exercício

- Prove que a linguagem EQ_{MT} (equivalência de duas MT's) é indecidível utilizando redução por mapeamento

Para ver se entenderam....

- Vamos ver uma prova da seção 5.1...
- Digam como deveria ser a redução por mapeamento

Equivalência entre MTs

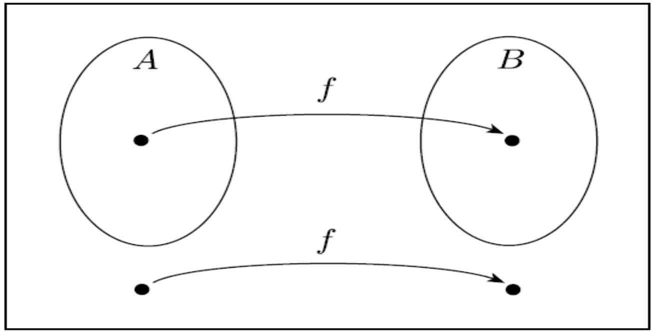
- EQ_{MT} = {<M1, M2> | M1 e M2 são MTs e L(M1) = L(M2)}
- Poderíamos usar EQ_{MT} para resolver V_{MT}!
  - Ou seja, quem é redutível a quem?
- Ideia: se uma MT M for equivalente a outra que rejeita qualquer cadeia, então L(M) = Ø
- Assuma que R é uma MT que decide EQ_{MT}

Equivalência entre MTs

- EQ_{MT} = {<M1, M2> | M1 e M2 são MTs e L(M1) = L(M2)}
- Poderíamos usar EQ_{MT} para resolver V_{MT}!
  – Ou seja, V_{MT} é redutível a EQ_{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}

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>))
- $F: V_{MT} \rightarrow EQ_{MT}$

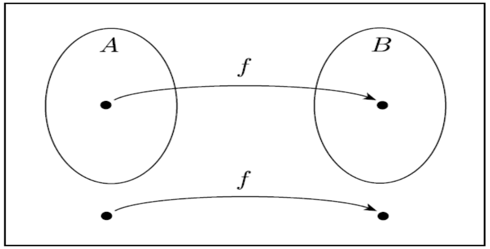


- $w \in V_{MT}$ = {<M> | M é uma MT e L(M) = Ø}
- $F(w) \in EQ_{MT}$ = {<M1, M2> | M1 e M2 são MTs e L(M1) = L(M2)}

Equivalência entre MTs é indecidível (prova)

- F = ‘

$F: V_{MT} \rightarrow EQ_{MT}$

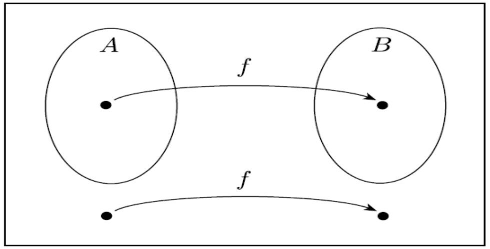


- w Є V_{MT} = {<M> | M é uma MT e L(M) = Ø}
- F(w) Є EQ_{MT} = {<M1, M2> | M1 e M2 são MTs e L(M1) = L(M2)}

Equivalência entre MTs é indecidível (prova)

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

$F: V_{MT} \rightarrow EQ_{MT}$



- w Є V_{MT} = {<M> | M é uma MT e L(M) = Ø}
- F(w) Є EQ_{MT} = {<M1, M2> | M1 e M2 são MTs e L(M1) = L(M2)}

Equivalência entre MTs é indecidível (prova)

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

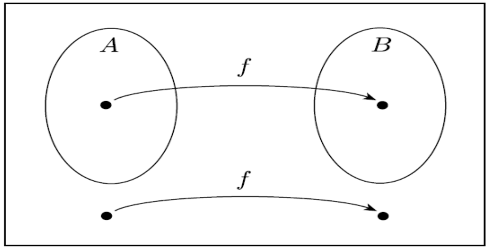
  1. Construa a seguinte máquina M1

     M1 = “Sobre a entrada x:

     rejeite.”

  2. Dê como saída <M, M1>”

$$F: V_{MT} \rightarrow EQ_{MT}$$



- $w \in V_{MT} = \{\langle M\rangle \mid M \text{ é uma MT e } L(M) = \emptyset\}$
- $F(w) \in EQ_{MT} = \{\langle M1, M2\rangle \mid M1 \text{ e } M2 \text{ são MTs e } L(M1) = L(M2)\}$

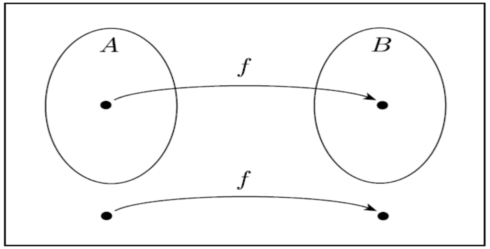
Equivalência entre MTs é indecidível (prova)

- Vamos assumir que EQ_{MT} seja decidível
- 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>”

$F: V_{MT} \rightarrow EQ_{MT}$



- Se a MT R decide EQ_{MT}, então a MT S = R(F(<M>)) decide V_{MT}
  Mas V_{MT} é indecidível! CONTRADIÇÃO!
  Logo EQ_{MT} é indecidível!

Vacuidade de uma linguagem de uma MT

- Prove que V_{MT} = { <M>: M é uma MT e L(M) = Ø} é indecidível
- Dica: utilizar A_{MT}:
  - Reduzir quem a quem?
  -
-

Vacuidade de uma linguagem de uma MT

- Prove que V_{MT} = { <M>: M é uma MT e L(M) = Ø} é indecidível
- Dica: utilizar A_{MT}:
  - Reduzir quem a quem?
  - Reduzir A_{MT} a V_{MT} – isto é, utilizar um decisor R de V_{MT} para decidir A_{MT}
- Ideia: construir uma versão de M que apenas teste w

Vacuidade de uma linguagem de uma MT – redução informal

- Assuma que V_{MT} é decidível e R é sua MT decisora
- A MT S que decide A_{MT} é:

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:

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”

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 – redução por mapeamento

A_{MT} = { <M,w>: M é uma MT, w é uma cadeia Є Σ* e M aceita w}

V_{MT} = { <M>: M é uma MT e L(M) = Ø}

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

Qual a F : $A_{MT} \rightarrow V_{MT}$?

(“Cola” da prova por redutibilidade informal):

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 – redução por mapeamento

- 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}
- Mas o uso dessa F na prova de que V_{MT} é indecidível ainda funciona porque a decidibilidade **não** é afetada por complementação (ou seja, se o complemento de V_{MT} é indecidível, então V_{MT} também é!)

Complemento da vacuidade de uma linguagem de uma MT – redução por mapeamento

- Assuma que $\overline{V_{MT}}$ é decidível e R é sua MT decisora
- A MT F que calcula o mapeamento de A_{MT} para $\overline{V_{MT}}$ é

  F = “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:

     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, e *rejeite* se M rejeita”

  2. Dê M1 como saída

- A MT S que decide A_{MT} é:

S = “Sobre a entrada <M,w>, uma codificação de uma MT M e uma cadeia w:

1. Rode R sobre F(<M,w>) e dê como saída o que R der”

- Absurdo! Pois A_{MT} é indecidível! Portanto $\overline{V_{MT}}$ não pode ser decidível, logo V_{MT} também não

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 S de A seria:

S = “Sobre uma entrada x,

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

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

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

MAS ESTA F SIM É QUE PRECISA SER DEFINIDA CASO A CASO

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 R o reconhecedor de B e f a redução de A para B. Um reconhecedor S para A é:

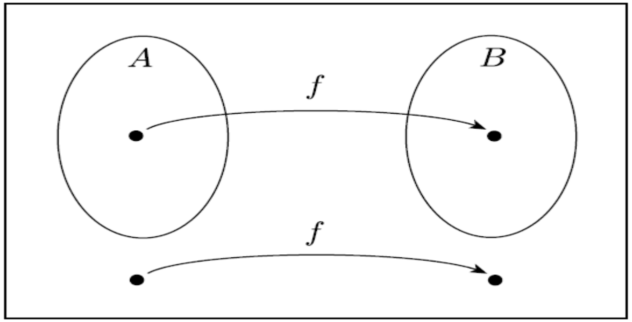
S = "Sobre a entrada w:

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

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

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

Logo, S reconhece A.



Redutibilidade por mapeamento e reconhecibilidade

- **Corolário**: Se A $\leq_m$ B e A 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

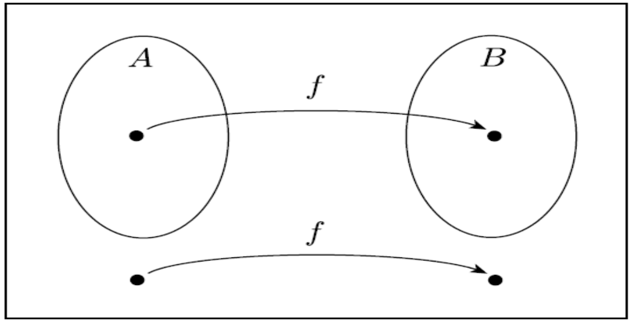
complemento(A) $\leq_m$ complemento(B)

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

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

ou

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



Exercício 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

Exercício exemplo - EQ_{MT} não é Turing-reconhecível

Exercício exemplo - EQ_{MT} não é Turing-reconhecível

- complemento(A_{MT}) $\leq_m$ EQ_{MT}
- => $A_{MT} \leq_m$ complemento(EQ_{MT})

F(<M, w>) = <M1, M2>

- 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. ? .”

     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>.”

Exercício exemplo - EQ_{MT} não é Turing-reconhecível

- complemento(A_{MT}) $\leq_m$ EQ_{MT}
- => $A_{MT} \leq_m$ complemento(EQ_{MT})

F(<M, w>) = <M1, M2>

- 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(<M, w>) = <M1, M2>

- 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>."

ACH2043
INTRODUÇÃO À TEORIA DA COMPUTAÇÃO

Cap 5.3 – Redutibilidade por mapeamento

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