# ACH2043
# INTRODUÇÃO À TEORIA DA COMPUTAÇÃO

## Aula 23

## Cap 5 – REDUTIBILIDADE
## Cap 5.1 – Problemas indecidíveis (parte 2)

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 A for indecidível, B também será

Ou seja, se eu digo que A é redutível a B, então é porque eu consigo escrever uma MT $M_A$ que decida A usando uma MT $M_B$ que decide B (se tal máquina $M_B$ existir)

A escrita de tal máquina $M_A$ (usando $M_B$) é a redução que precisa ser mostrada.

# 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 A for indecidível, B também será

Chave para provar que certos problemas são indecidíveis: reduzindo um problema conhecidamente indecidível (A) a ele (B)

A: problema que eu JÁ SEI que é indecidível

B: problema que eu quero provar que é indecidível

Então basta mostrar que A é redutível a B (e a prova é por CONTRADIÇÃO!!!)

# Redutibilidade - resumo

- Como provar que um problema B é indecidível usando a técnica de redutibilidade:
  - Assumo por contradição que B é decidível, e que portanto tem uma MT decisora R
  - 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!

# Ex: quero provar que o problema da parada é indecidível utilizando redutibilidade

- $PARA_{MT}$ = {<M,w> : M é uma MT e M pára sobre a entrada w}
- $A_{MT}$ = {<M,w> : M é uma MT e M aceita w}
- $A_{MT}$ (que é indecidível) pode ser reduzido a $PARA_{MT}$ , pois se eu tiver a solução de $PARA_{MT}$ terei a solução de $A_{MT}$. Ou seja, se existir uma MT que decida $PARA_{MT}$, então essa máquina poderia ser usada para decidir $A_{MT.}$
- Mas $A_{MT}$ é indecidível! Logo, $PARA_{MT}$ é indecidível

# Ex: O Problema da Parada é indecidível

- Prova (tem que mostrar a redução!): assuma, por contradição, que uma MT $M_B$ decida $PARA_{MT}$. Então construímos $M_A$ (que usa $M_B$) para decidir $A_{MT}$ :

$M_A$ = "Sobre a entrada <M, w>, uma codificação de uma MT M e uma cadeia w:

1. Rode a MT $M_B$ sobre a entrada <M, w>.
2. Se $M_B$ rejeita, rejeite. # isto quer dizer que M não pára, logo w $\notin$ L(M)
3. Se $M_B$ aceita, simule M sobre w até que ela páre.
4. Se M aceitou, *aceite*; se M rejeitou, *rejeite*."

**Essa máquina $M_A$ DECIDIRIA $A_{MT}$ CASO $M_B$ EXISTISSE !!!**

- Logo o problema $A_{MT}$ pode ser reduzido ao $PARA_{MT}$
- Como $A_{MT}$ é indecidível, $PARA_{MT}$ é indecidível também
  - Pois se $M_A$ fosse decisora eu estaria decidindo $A_{MT}$ → CONTRADIÇÃO!!!

# EXERCÍCIOS – Prova de indecidibilidade por redutibilidade

Para os problemas do próximo slide:

- Escreva o problema sob a forma de linguagem
- Prove que o problema é indecidível utilizando redução a partir de um problema já provado ser indecidível (e caindo em uma contradição)

# EXERCÍCIOS – Prova de indecidibilidade por redutibilidade

1) Prove que o problema da vacuidade de uma MT (saber se uma MT não aceita nenhuma sequência) é indecidível

(aqui você conhece 2 problemas indecidíveis: $A_{MT}$ e $PARA_{MT}$)

2) Prove que o problema de saber se duas MT's são equivalentes é indecidível

(aqui você conhece 3 problemas indecidíveis: $A_{MT}$, $PARA_{MT}$ e $V_{MT}$)

3) Prove que o problema de saber se a linguagem reconhecida por uma MT é regular é indecidível

(aqui você conhece 4 problemas indecidíveis: $A_{MT}$, $PARA_{MT}$, $V_{MT}$ e $EQ_{MT}$)

# EXERCÍCIOS – Prova de indecidibilidade por redutibilidade

1) Prove que o problema da vacuidade de uma MT (saber se uma MT não aceita nenhuma sequência) é indecidível

(aqui você conhece 2 problemas indecidíveis: $A_{MT}$ e $PARA_{MT}$)

2) Prove que o problema de saber se duas MT's são equivalentes é indecidível

(aqui você conhece 3 problemas indecidíveis: $A_{MT}$, $PARA_{MT}$ e $V_{MT}$)

3) Prove que o problema de saber se a linguagem reconhecida por uma MT é regular é indecidível

(aqui você conhece 4 problemas indecidíveis: $A_{MT}$, $PARA_{MT}$, $V_{MT}$ e $EQ_{MT}$)

# Redutibilidade - resumo

- Como provar que um problema B ($V_{MT}$) é indecidível usando a técnica de redutibilidade:
  - Assumo por contradição que B é decidível, e que portanto tem uma MT decisora R ($M_B$)
  - Uso a MT decisora (R) de B para construir uma MT decisora (S = $M_A$) de um problema A ($A_{MT}$) que sabemos que é indecidível (redução de A a B)
  - Contradição! Portanto R não pode existir!

# Vacuidade de uma linguagem de uma MT

- $V_{MT}$ = { <M> | M é uma MT e L(M) = Ø}
- Como podemos usar $V_{MT}$ para resolver $A_{MT}$?

  $A_{MT}$ = {<M,w> | M é uma MT e M aceita w}

  - Se uma linguagem for vazia, ela não aceita w. Mas e se não for?
  - 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”

Ou seja, ou M1 aceita w ou não aceita nada

# Vacuidade de uma linguagem de uma MT

- Suponha que R decide $V_{MT}$, vamos construir S que decide $A_{MT}$
- 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 ← não entra em loop
  2. Rode R sobre M1 ← Pela hipótese inicial R é decisora, logo não entra em loop
  3. Se R aceita, *rejeite*; se R rejeita, *aceite*.” ← não entra em loop

- Contradição! Como $A_{MT}$ é indecidível, $V_{MT}$ é indecidível

# Aula de hoje

- Terminar os dois exercícios seguintes
- Redutibilidade por mapeamento

# EXERCÍCIOS – Prova de indecidibilidade por redutibilidade

1) Prove que o problema da vacuidade de uma MT (saber se uma MT não aceita nenhuma sequência) é indecidível

(aqui você conhece 2 problemas indecidíveis: $A_{MT}$ e $PARA_{MT}$)

2) Prove que o problema de saber se duas MT's são equivalentes é indecidível

(aqui você conhece 3 problemas indecidíveis: $A_{MT}$, $PARA_{MT}$ e $V_{MT}$)

3) Prove que o problema de saber se a linguagem reconhecida por uma MT é regular é indecidível

(aqui você conhece 4 problemas indecidíveis: $A_{MT}$, $PARA_{MT}$, $V_{MT}$ e $EQ_{MT}$)

# Equivalência entre MTs

Como escrevo esse problema em termos de linguagem?

# Equivalência entre MTs é indecidível

- $EQ_{MT}$ = {<M1, M2> | M1 e M2 são MTs e L(M1) = L(M2)}
- Podemos usar $EQ_{MT}$ para resolver qual desses problemas? $A_{MT}$, $PARA_{MT}$ OU $V_{MT}$?

# Equivalência entre MTs é indecidível

- $EQ_{MT}$ = {<M1, M2> | M1 e M2 são MTs e L(M1) = L(M2)}
- Podemos usar $EQ_{MT}$ para resolver $V_{MT}$!

TENTE FAZER!!!

# Equivalência entre MTs é indecidível

- $EQ_{MT}$ = {<M1, M2> | M1 e M2 são MTs e L(M1) = L(M2)}
- Podemos usar $EQ_{MT}$ para resolver $V_{MT}$!
- Ideia:

# Equivalência entre MTs é indecidível

- $EQ_{MT}$ = {<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 por contradição que R é uma MT que decide $EQ_{MT}$
  - Vamos construir S que decide $V_{MT}$ usando R

# Equivalência entre MTs é indecidível - PROVA

- A prova é por redução do problema $V_{MT}$
-
-
-

# Equivalência entre MTs é indecidível - PROVA

- A prova é por redução do problema $V_{MT}$
- Vamos assumir que o problema/linguagem $EQ_{MT}$ é decidível por uma MT R
-
-

# Equivalência entre MTs é indecidível - PROVA

- A prova é por redução do problema $V_{MT}$
- Vamos assumir que o problema/linguagem $EQ_{MT}$ é decidível por uma MT R
- A MT S que decide $V_{MT}$ utilizando R é:

S = “Sobre a entrada <M> onde M é uma MT:

1. Rode R sobre a entrada <M, M1>, em que M1 é uma MT que rejeita todas as entradas.
2. Se R aceita, ; se R rejeita, .”

M1 = “Sobre a entrada x,
1. rejeite.”

# Equivalência entre MTs é indecidível - PROVA

- A prova é por redução do problema $V_{MT}$
- Vamos assumir que o problema/linguagem $EQ_{MT}$ é decidível por uma MT R
- A MT S que decide $V_{MT}$ utilizando R é:

S = “Sobre a entrada <M> onde M é uma MT:

1. Rode R sobre a entrada <M, M1>, em que M1 é uma MT que rejeita todas as entradas.
2. Se R aceita, *aceite*; se R rejeita, *rejeite*.”

M1 = “Sobre a entrada x,
1. rejeite.”

# Equivalência entre MTs é indecidível - PROVA

- A prova é por redução do problema $V_{MT}$
- Vamos assumir que o problema/linguagem $EQ_{MT}$ é decidível por uma MT R
- A MT S que decide $V_{MT}$ utilizando R é:

S = “Sobre a entrada <M> onde M é uma MT:

1. Rode R sobre a entrada <M, M1>, em que M1 é uma MT que rejeita todas as entradas.
2. Se R aceita, *aceite*; se R rejeita, *rejeite*.”

M1 = “Sobre a entrada x,
1. rejeite.”

- Contradição! Pois $V_{MT}$ é indecidível, então $EQ_{MT}$ também é

# EXERCÍCIOS – Prova de indecidibilidade por redutibilidade

1) Prove que o problema da vacuidade de uma MT (saber se uma MT não aceita nenhuma sequência) é indecidível

(aqui você conhece 2 problemas indecidíveis: $A_{MT}$ e $PARA_{MT}$)

2) Prove que o problema de saber se duas MT's são equivalentes é indecidível

(aqui você conhece 3 problemas indecidíveis: $A_{MT}$, $PARA_{MT}$ e $V_{MT}$)

3) Prove que o problema de saber se a linguagem reconhecida por uma MT é regular é indecidível

(aqui você conhece 4 problemas indecidíveis: $A_{MT}$, $PARA_{MT}$, $V_{MT}$ e $EQ_{MT}$)

# Classe da linguagem gerada por uma MT

- Dada uma MT M, a linguagem gerada por ela poderia ser reconhecida por um modelo mais simples?
- Por ex: saber se L(M) é regular
- Como escrever esse problema (identificar se a linguagem reconhecida por uma MT M é regular) em termos de linguagem?

# Determinação de se a linguagem gerada por uma MT é regular

- $REGULAR_{MT}$ = { <M> | M é uma MT e L(M) é regular}

# Determinação de se a linguagem gerada por uma MT é regular

- $REGULAR_{MT}$ = { <M> | M é uma MT e L(M) é regular}
- $REGULAR_{MT}$ é indecidível
- Ideia da Prova:

# Determinação de se a linguagem gerada por uma MT é regular

- $REGULAR_{MT}$ = { <M> | M é uma MT e L(M) é regular}
- $REGULAR_{MT}$ é indecidível
- Ideia da Prova:
  - Supomos que existe uma MT R que decide $REGULAR_{MT}$ e usamos R em uma MT S para decidir $A_{MT}$
  - S usará R para analisar se uma MT M2 é regular, sendo que M2 reconhece uma linguagem regular ($\Sigma^*$) sse M aceita w

# Determinação de se a linguagem gerada por uma MT é regular

- S que decide $A_{MT}$ usando R

Supondo que o alfabeto ∑ = {0,1}

- S = “Sobre a entrada <M,w>, onde M é uma MT e w é uma cadeia:

1. Construa (imprima) a MT M2:

Ou seja,
Se M não aceitar w, então o que M2 aceita? L(M2) ?

M2 = “Sobre a entrada x:

1. Se x tem a forma $0^n1^n$, *aceite*

2. senão, rode M sobre a entrada w e *aceite* (x) se M aceita w, *rejeite* se M rejeita”

# Determinação de se a linguagem gerada por uma MT é regular

- S que decide $A_{MT}$ usando R

Supondo que o alfabeto ∑ = {0,1}

- S = “Sobre a entrada <M,w>, onde M é uma MT e w é uma cadeia:

1. Construa (imprima) a MT M2:

Ou seja,
Se M não aceitar w, então M2 só aceita $0^n1^n$, (L(M2) = {$0^n1^n$})
que NÃO é regular
E se M aceitar w?

M2 = “Sobre a entrada x:

1. Se x tem a forma $0^n1^n$, *aceite*

2. senão, rode M sobre a entrada w e *aceite* (x) se M aceita w, *rejeite* se M rejeita”

# Determinação de se a linguagem gerada por uma MT é regular

- S que decide $A_{MT}$ usando R

Supondo que o alfabeto ∑ = {0,1}

- S = “Sobre a entrada <M,w>, onde M é uma MT e w é uma cadeia:

1. Construa (imprima) a MT M2:

M2 = “Sobre a entrada x:

1. Se x tem a forma $0^n1^n$, *aceite*

2. senão, rode M sobre a entrada w e *aceite* (x) se M aceita w, *rejeite* se M rejeita”

Ou seja,
Se M não aceitar w, então M2 só aceita $0^n1^n$, (L(M2) = {$0^n1^n$}) que NÃO é regular
Se M aceitar w, M2 aceita qualquer x (Σ*, que é regular).

M2 é como se fosse uma função como essa ao lado.
Dependendo de `aux`, M2 retorna true para qualquer x (ou seja, Σ*), pois faz isso tanto no if quanto no else que compara x , ou apenas para quando x tem a forma $0^n1^n$ (que é livre de contexto).
E *aux* é justamente a resposta de se **M** aceita **w**.

```
bool M2(x){
    if (x tem a forma 0^n1^n) return true;
    else {
        aux ← M(w)
        if (aux == true) return true;
        else return false;
}
```

# Determinação de se a linguagem gerada por uma MT é regular

- S que decide $A_{MT}$ usando R

Supondo que o alfabeto ∑ = {0,1}

- S = “Sobre a entrada <M,w>, onde M é uma MT e w é uma cadeia:

1. Construa (imprima) a MT M2:

   M2 = “Sobre a entrada x:

   1. Se x tem a forma $0^n1^n$, *aceite*

   2. senão, rode M sobre a entrada w e *aceite* (x) se M aceita w, *rejeite* se M rejeita”

Ou seja,
Se M não aceitar w, então M2 só aceita $0^n1^n$, ($L(M2) = \{0^n1^n\}$) que NÃO é regular
Se M aceitar w, M2 aceita qualquer x (Σ*, que é regular).

2. Rode R sobre a entrada <M2>

R é a suposta MT que decide de L(M) é regular

3. Se R aceita, ; se R rejeita, ”

-

# Determinação de se a linguagem gerada por uma MT é regular

- S que decide $A_{MT}$ usando R

Supondo que o alfabeto ∑ = {0,1}

- S = “Sobre a entrada <M,w>, onde M é uma MT e w é uma cadeia:

1. Construa (imprima) a MT M2:

   M2 = “Sobre a entrada x:

   1. Se x tem a forma $0^n1^n$, *aceite*

   2. senão, rode M sobre a entrada w e *aceite* (x) se M aceita w, *rejeite* se M rejeita”

Ou seja,
Se M não aceitar w, então M2 só aceita $0^n1^n$, (L(M2) = {$0^n1^n$}) que NÃO é regular
Se M aceitar w, M2 aceita qualquer x (Σ*, que é regular).

2. Rode R sobre a entrada <M2>

3. Se R aceita, *aceite*; se R rejeita, *rejeite*”

M2 é regular → M aceita w

M2 não é regular → M não aceita w

# Determinação de se a linguagem gerada por uma MT é regular

- S que decide $A_{MT}$ usando R

Supondo que o alfabeto ∑ = {0,1}

- S = “Sobre a entrada <M,w>, onde M é uma MT e w é uma cadeia:

1. Construa (imprima) a MT M2:

   M2 = “Sobre a entrada x:

   1. Se x tem a forma $0^n1^n$, *aceite*

   2. senão, rode M sobre a entrada w e *aceite* se M aceita w, *rejeite* se M rejeita”

Ou seja,
Se M não aceitar w, então M2 só aceita $0^n1^n$, (L(M2) = {$0^n1^n$}) que NÃO é regular
Se M aceitar w, M2 aceita qualquer x (Σ*, que é regular).

2. Rode R sobre a entrada <M2>

3. Se R aceita, *aceite*; se R rejeita, *rejeite*”

M2 é regular → M aceita w

M2 não é regular → M não aceita w

- Contradição, com R eu decidiria $A_{MT}$!!! Mas $A_{MT}$ é indecidível, então $REGULAR_{MT}$ também é

# Determinação de propriedades da linguagem gerada por uma MT

- Da mesma forma, os seguintes problemas são indecidíveis (para uma dada MT M)
  - Determinar se L(M) é livre-de-contexto
  - Determinar se L(M) é sensível ao contexto
  - Determinar se L(M) é decidível (recursiva)
  - ...
  - Na verdade, determinar qualquer propriedade de L(M) (Teorema de Rice)

# Um outro exercício – co-Turing reconhecibilidade

Prove que o problema da equivalência de duas gramáticas livres de contexto é co-Turing reconhecível

(primeiro escreva o problema na forma de linguagem)

# Um outro exercício – co-Turing reconhecibilidade

Prove que o problema da equivalência de duas gramáticas livres de contexto é co-Turing reconhecível

(primeiro escreva o problema na forma de linguagem)

$EQ_{GLC} = \{ \langle G_1, G_2 \rangle | \; G_1 \text{ e } G_2 \text{ são duas GLCs e } L(G_1) = L(G_2) \}$

# Um outro exercício – co-Turing reconhecibilidade

Prove que o problema da equivalência de duas gramáticas livres de contexto é co-Turing reconhecível

(primeiro escreva o problema na forma de linguagem)

$EQ_{GLC} = \{ \langle G_1, G_2 \rangle | \; G_1 \text{ e } G_2 \text{ são duas GLCs e } L(G_1) = L(G_2)\}$

O que seria o complemento disso?

# Um outro exercício – co-Turing reconhecibilidade

Prove que o problema da equivalência de duas gramáticas livres de contexto é co-Turing reconhecível

(primeiro escreva o problema na forma de linguagem)

$EQ_{GLC} = \{ \langle G_1, G_2 \rangle | \; G_1 \text{ e } G_2 \text{ são duas GLCs e } L(G_1) = L(G_2)\}$

O que seria o complemento disso? Duas GLCs que não são equivalentes

# Um outro exercício – co-Turing reconhecibilidade

Prove que o problema da equivalência de duas gramáticas livres de contexto é co-Turing reconhecível

(primeiro escreva o problema na forma de linguagem)

$EQ_{GLC} = \{ \langle G_1, G_2 \rangle | \; G_1 \text{ e } G_2 \text{ são duas GLCs e } L(G_1) = L(G_2) \}$

Preciso mostrar uma MT reconhecedora para essa parte

O que seria o complemento disso? Duas GLCs que não são equivalentes

# $EQ_{GLC}$ é co-Turing reconhecível

A MT M abaixo reconhece o complemento de $EQ_{GLC}$:

M = “Sobre a entrada <G1, G2>, sendo G1 e G2 GLCs,

1. Para cada cadeia s de $\sum^*$, seguindo a ordem lexicográfica, use o algoritmo de CYK para verificar se G1 e G2 aceitam s. Se derem o mesmo resultado (ambas aceitando ou rejeitando), vá para a próxima cadeia s. Se derem resultado diferentes, *aceite*.”

# $EQ_{GLC}$ é co-Turing reconhecível

A MT M abaixo reconhece o complemento de $EQ_{GLC}$:

M = “Sobre a entrada <G1, G2>, sendo G1 e G2 GLCs,

1. Para cada cadeia s de $\sum^*$, seguindo a ordem lexicográfica, use o algoritmo de CYK para verificar se G1 e G2 aceitam s. Se derem o mesmo resultado (ambas aceitando ou rejeitando), vá para a próxima cadeia s. Se derem resultado diferentes, *aceite*.”

Se as gramáticas forem diferentes, M irá parar quando analisar a primeira cadeia s que pertence à linguagem de uma mas não à de outra.

Se as gramáticas forem equivalentes, M entra em loop.

# E a $EQ_{GLC}$ ?

- Se $EQ_{GLC}$ é indecidível  e o complemento de $EQ_{GLC}$ é Turing-reconhecível, o que podemos falar sobre $EQ_{GLC}$?

  Não é nem Turing-reconhecível

  (provaremos isso mais à frente)

# Computabilidade

- Dado um problema, como provar que ele é
  - Turing decidível
  - Turing indecidível
  - Turing reconhecível
  - Turing não-reconhecível

- Há várias formas para isso, parte delas se baseiam em redutibilidade

# Redutibilidade

- 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 A for indecidível, B também será

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.

# 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

# 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!

# Determinar se uma MT aceita uma cadeia w é INdecidível

$$A_{\text{MT}} = \{\langle M, w\rangle \mid M \text{ é uma MT e } M \text{ aceita } w\}.$$

**TEOREMA 4.11**

$A_{\text{MT}}$ é indecidível.

# Ex: O Problema da Parada é indecidível

Prova (tem que mostrar a redução!): vamos reduzir $A_{MT}$ a $PARA_{MT}$.

- 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, logo w ϵ L(M)
3. Se R aceita, simule M sobre w até que ela pare.
4. Se M aceitou, *aceite*; se M rejeitou, *rejeite*.”

- Logo o problema $A_{MT}$ pode ser reduzido ao $PARA_{MT}$
- Como $A_{MT}$ é indecidível, $PARA_{MT}$ é indecidível também
  - Pois se S fosse decisora eu estaria decidindo $A_{MT}$ → CONTRADIÇÃO!!!

# Problemas indecidíveis (provas nas aulas passada e de hoje usando redução informal)

1) $V_{MT}$ : vacuidade de uma MT (saber se uma MT não aceita nenhuma sequência)

2) $REGULAR_{MT}$ : saber se a linguagem reconhecida por uma MT é regular

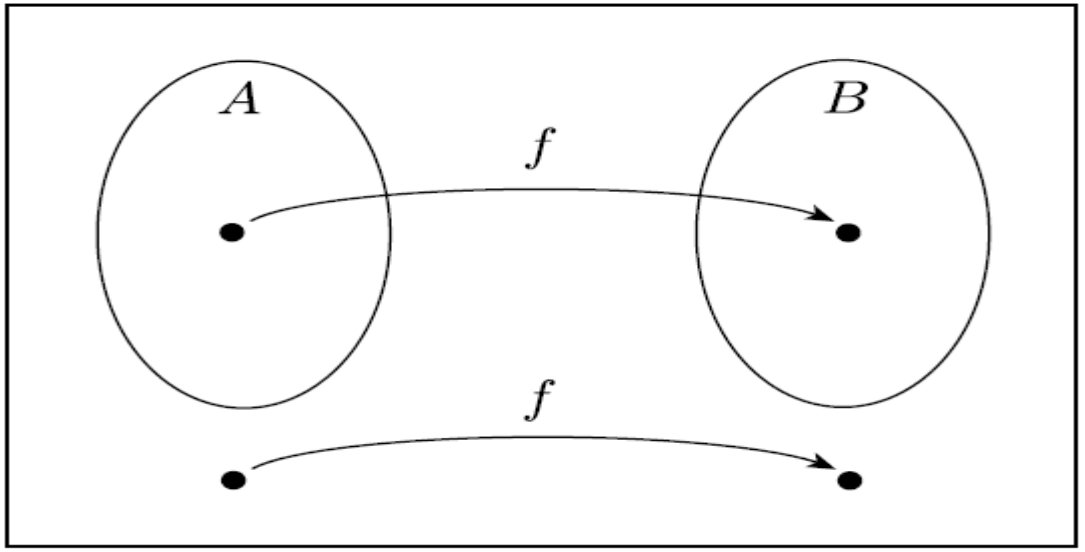
3) $EQ_{MT}$ : saber se duas MT's são equivalentes

# 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.