ACH2043
INTRODUÇÃO À TEORIA DA COMPUTAÇÃO

Aula 22

Cap 5 – REDUTIBILIDADE
Cap 5.1 – Problemas indecidíveis

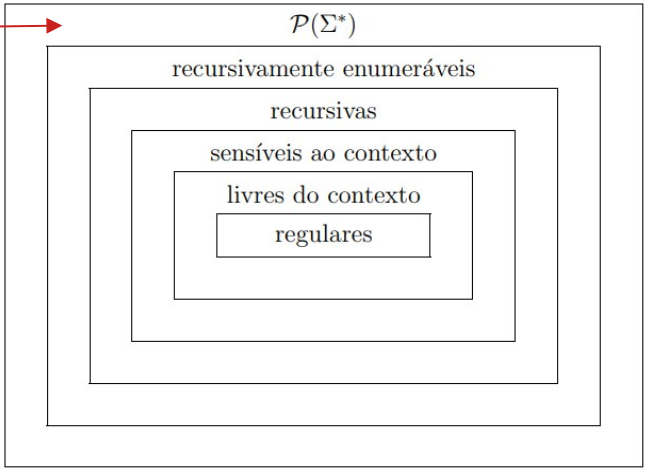
Profa. Ariane Machado Lima
ariane.machado@usp.br

Aula passada

Linguagens Turing-irreconhecíveis

COROLÁRIO 4.18

Algumas linguagens não são Turing-reconhecíveis.



- Ideia da Prova: vamos mostrar que:
  - Há um conjunto contável de Máquinas de Turing
  - Há um conjunto incontável de linguagens
  - Cada MT reconhece apenas uma linguagem
  - Logo, há linguagens que não são reconhecidas por nenhuma MT

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_{MT} é indecidível.

O problema da aceitação por uma MT é indecidível – Prova por contradição

- Supomos A_{MT} decidível e H um decisor:

$$H(\langle M, w\rangle) = \begin{cases} \textit{aceite} & \text{se } M \text{ aceita } w \\ \textit{rejeite} & \text{se } M \text{ não aceita } w \end{cases}$$

- D outra MT, que usa H para determinar o que M faz com <M>, e faz o oposto:

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

1. Rode H sobre a entrada $\langle M, \langle M \rangle \rangle$.
2. Dê como saída o oposto do que H dá como saída; ou seja, se H aceita, *rejeite* e se H rejeita, *aceite*."

O problema da aceitação por uma MT é indecidível – Prova por contradição

$$D(\langle M \rangle) = \begin{cases} \textit{aceite} & \text{se } M \text{ não aceita } \langle M \rangle \\ \textit{rejeite} & \text{se } M \text{ aceita } \langle M \rangle. \end{cases}$$

- E se D tiver <D> como entrada?

$$D(\langle D \rangle) = \begin{cases} \textit{aceite} & \text{se } D \text{ não aceita } \langle D \rangle \\ \textit{rejeite} & \text{se } D \text{ aceita } \langle D \rangle. \end{cases}$$

- Contradição! H e D não podem existir!

O problema da aceitação por uma MT é indecidível – Prova por contradição

- Prova similar que o problema da parada é indecidível (vídeo)

Aula de hoje: provas de indecidibilidade

Uma das formas de provar que uma linguagem L NÃO é Turing-reconhecível é provar que o complemento de L é Indecidível

- (se $\overline{L}$ fosse decidível L seria reconhecível)

- Para provar que uma linguagem é INdecidível pode-se usar a técnica de redutibilidade (cap 5)

E para quê?

- Saber que um problema é indecidível é importante para focarmos em soluções subótimas ou para um recorte do problema geral

Provas de indecidibilidade

E para provar que uma linguagem é INDECIDÍVEL?

- Pode-se usar a técnica de redutibilidade (cap 5 – AULA DE HOJE)

Redutibilidade

- Redução: conversão de um problema A em outro problema B de forma que a solução de B seja usada para solucionar A

- Exemplos matemáticos:
  - Medir a área de um retângulo pode ser reduzido a medir a altura e a largura do retângulo
  - Resolver um problema de equações lineares pode ser reduzido ao problema de inverter uma matriz

Exemplo de aplicação prática

Exemplo de problema indecidível

- Verificar se um trecho de código é um vírus

The 12th International Conference for Internet Technology and Secured Transactions (ICITST-2017)

Undecidable Problems in Malware Analysis

Ali Aydın Selçuk
Dept. of Computer Engineering
TOBB University of Economics and Technology
Ankara, Turkey
aliaydinselcuk@gmail.com

Fatih Orhan, Berker Batur
Comodo Security Solutions, Inc.
Clifton, NJ, USA
{fatih.orhan,berker.batur@comodo.com }

Bad news...

- O problema de, dados um programa e um trecho de código, saber se aquele trecho é um vírus para aquele programa é **indecidível** (prova por redução a partir do problema da parada)
- Consequências:
  - não há como desenvolver um anti-vírus universal
  - Há alguns vírus que são indetectáveis (polimórficos, alterando sua condição de disseminação ao longo do tempo)

- Há como identificar vírus por padrões semânticos (comportamento, ex: transposição de código, encriptar dados bancários, etc) então?

Bad news...

- Há como identificar vírus por padrões semânticos (comportamento, ex: transposição de código, encriptar dados bancários, etc) então?
- O problema de Template Matching para detecção de vírus é **indecidível** (prova por redução a partir do problema da parada)

Bad news...

- Há como identificar vírus por padrões semânticos (comportamento, ex: transposição de código, encriptar dados bancários, etc) então?
- O problema de Template Matching para detecção de vírus é **indecidível** (prova por redução a partir do problema da parada)
- Outros...

Nem tudo está perdido...

- Para *template matching*, toolkit que executa em duas fases:
  - Descompilação do programa binário → grafo de controle → representação intermediária (RI)
  - RI é comparada com um conjunto de templates de vírus (ex: envio de e-mail, descriptografia, ...)

Redutibilidade

Redutibilidade

- Redução: conversão de um problema A em outro problema B de forma que a solução de B seja usada para solucionar A

- Exemplos matemáticos:
  - Medir a área de um retângulo pode ser reduzido a medir a altura e a largura do retângulo
  - Resolver um problema de equações lineares pode ser reduzido ao problema de inverter uma matriz

Redutibilidade

- Utilidade:
  - Se A é redutível a B
    - A não pode ser mais (fácil/difícil?) do que B

Redutibilidade

- Utilidade:
  - Se A é redutível a B
    - A não pode ser mais fácil nem mais difícil do que B

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

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,

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

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,

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

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!!!)

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}
- Que problema indecidível pode ser reduzido a esse? Ou seja, que problema eu conseguiria resolver (decidir) se eu pudesse ter (e usar) uma MT que decide $PARA_{MT}$?

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}

Faz de conta que não assistimos aquele vídeo...

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} :

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, .
3. Se M_B aceita,

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,

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!!!

Em 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 M_B
  - Uso a MT decisora (M_B) de B para construir uma MT decisora (M_A) de um problema que sabemos que é indecidível (redução de A a B)
  - Contradição! Portanto M_B não pode existir!

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

- Como escrever isso em forma de linguagem?

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}

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?

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

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

Tente fazer antes de olhar os próximos slides!

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”

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

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
  - Uso a MT decisora (R) de B para construir uma MT decisora (S) 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

- Suponha que R decide V_{MT}, vamos construir S que decide A_{MT}
- S =

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:

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
  2. Rode R sobre M1
  3. Se R aceita, *?* ; se R rejeita, *?* ”

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

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})