ACH2043
INTRODUÇÃO À TEORIA DA COMPUTAÇÃO

Aula 21

Cap 5.1 – Problemas indecidíveis (parte 3)

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

Nas aulas passadas...

- 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!

Provamos que os seguintes problemas são indecidíveis:

- $PARA_{MT}$ = {<M,w> : M é uma MT e M pára sobre a entrada w}
- V_{MT} = { <M>: M é uma MT e L(M) = Ø} (Vacuidade de uma MT)
- $REGULAR_{MT}$ = { <M> | M é uma MT e L(M) é regular}

Na aula de hoje

- Outros exemplos de provas de indecidibilidade por redutibilidade
- Reduções via histórias de computação

Equivalência entre MTs

Como escrevo esse problema em termos de linguagem?

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:

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

Equivalência entre MTs

- $EQ_{MT} = \{\langle M1, M2 \rangle \mid$ 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 $L(M) = \emptyset$
  - Assuma por contradição que R é uma MT que decide EQ_{MT}
  - Vamos construir S que decide V_{MT} usando R

Essa é fácil. Tentem!!!!

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

Reduções via histórias de computação

- História de computação: sequência de configurações de uma MT, da inicial à de aceitação ou rejeição
- Uma história de computação deve ser finita
  - Ou seja, se uma MT não pára para uma dada cadeia w, não há uma história de computação dessa máquina para w
- Apenas uma para MTs determinísticas, e possivelmente várias para MTs não-determinísticas
- Aqui consideramos apenas MTs determinísticas

Reduções via histórias de computação

- Útil para provar a redutibilidade de A_{MT} a outras linguagens (principalmente aquelas que envolvem a existência de algo, como das raízes inteiras de um polinômio)

Ex: Vacuidade de autômatos linearmente limitados

- Autômatos linearmente limitados (ALL) são os modelos que reconhecem linguagens sensíveis ao contexto
- A_{ALL} = {<M,w> | M é um ALL que aceita a cadeia w}
- A_{ALL} é decidível

Autômatos linearmente limitados

- A_{ALL} é decidível:
  - Se ALL tem q estados, g símbolos de fita e uma fita de comprimento n, então só existem qng^n configurações distintas (estado atual, posição da cabeça, conteúdo da fita)
  - Se uma configuração se repetir, o ALL está em loop.
  - Como é uma MT que decide A_{ALL}?

Autômatos linearmente limitados

- A_{ALL} é decidível:

  L = “Sobre a entrada <M,w>, 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 M não parou, *rejeite*.”

Vacuidade de autômatos linearmente limitados

- Mas o problema da vacuidade de ALLs é indecidível
- $V_{ALL} = \{\langle M \rangle \mid M \text{ é um ALL e } L(M) = \emptyset\}$
- Para provar, vamos usar “histórias da computação” e redução a partir de A_{MT}

Vacuidade de autômatos linearmente limitados

- Podemos construir um ALL B que reconheça apenas histórias de computação de aceitação de uma cadeia w em uma MT M
- Se L(B) for vazia, então M não aceita w, caso contrário aceita
- Como seria esse B?
  - TEMOS QUE MOSTRAR COMO UMA MT S PARA A_{MT} CONSTRUIRIA O ALL B A PARTIR DE DE <M,w>

Vacuidade de autômatos linearmente limitados

- ALL B:
  - Fita contém inicialmente uma história de computação, ou seja, C1 até Cl, cada configuração separada por “#”

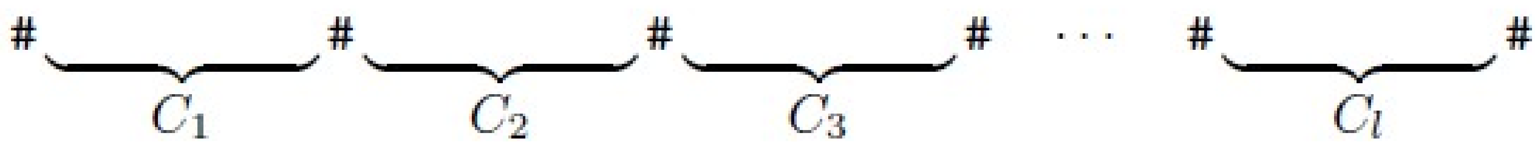

FIGURA 5.11
Uma possível entrada para B

Vacuidade de autômatos linearmente limitados

- ALL B:
  - Fita contém inicialmente uma história de computação, ou seja, C1 até Cl, cada configuração separada por “#”
  - Deve verificar 3 propriedades:
    1. C1 é a configuração inicial de M sobre w
    2. Cada Ci+1 é originada de Ci
    3. Cl é uma configuração de aceitação para M

Vacuidade de autômatos linearmente limitados

- ALL B:
  - Fita contém inicialmente uma história de computação, ou seja, C1 até Cl, cada configuração separada por “#”
  - Deve verificar 3 propriedades:

1. C1 é a configuração inicial de M sobre w

   $C1 = q_0w_1w_2...w_n$, onde q_0 é o estado inicial de M

2. Cada Ci+1 é originada de Ci

3. Cl é uma configuração de aceitação para M

Vacuidade de autômatos linearmente limitados

- ALL B:
  - Fita contém inicialmente uma história de computação, ou seja, C1 até Cl, cada configuração separada por “#”
  - Deve verificar 3 propriedades:
    1. C1 é a configuração inicial de M sobre w

       C1 = $q_0w_1w_2...w_n$, onde q_0 é o estado inicial de M
    2. Cada Ci+1 é originada de Ci

       Ci e ci+1 são idênticas exceto pelas posições sob e adjacentes à cabeça em Ci (estas, atualizadas conforme a função de transição)
    3. Cl é uma configuração de aceitação para M

Vacuidade de autômatos linearmente limitados

- ALL B:
  - Fita contém inicialmente uma história de computação, ou seja, C1 até Cl, cada configuração separada por "#"
  - Deve verificar 3 propriedades:
    1. C1 é a configuração inicial de M sobre w

       $C1 = q_0w_1w_2...w_n$, onde q_0 é o estado inicial de M
    2. Cada Ci+1 é originada de Ci

       Ci e ci+1 são idênticas exceto pelas posições sob e adjacentes à cabeça em Ci (estas, atualizadas conforme a função de transição)
    3. Cl é uma configuração de aceitação para M

       Cl contém q_{aceita}

Vacuidade de autômatos linearmente limitados

- ALL B:
  - Fita contém inicialmente uma história de computação, ou seja, C1 até Cl, cada configuração separada por “#”
  - Deve verificar 3 propriedades:
    1. C1 é a configuração inicial de M sobre w

       $C1 = q_0w_1w_2...w_n$, onde q_0 é o estado inicial de M
    2. Cada Ci+1 é originada de Ci **COMO?**

       Ci e ci+1 são idênticas exceto pelas posições sob e adjacentes à cabeça em Ci (estas, atualizadas conforme a função de transição)
    3. Cl é uma configuração de aceitação para M

       Cl contém q_{aceita}

Vacuidade de autômatos linearmente limitados

- ALL B:
  - Fita contém inicialmente uma história de computação, ou seja, C1 até Cl, cada configuração separada por “#”
  - Deve verificar 3 propriedades:
    1. C1 é a configuração inicial de M sobre w

       $C1 = q_0 w_1 w_2 ... w_n$, onde q_0 é o estado inicial de M
    2. Cada Ci+1 é originada de Ci **COMO? ZIGUE-ZAGUIANDO NA FITA**

       Ci e ci+1 são idênticas exceto pelas posições sob e adjacentes à cabeça em Ci (estas, atualizadas conforme a função de transição)
    3. Cl é uma configuração de aceitação para M

       Cl contém q_{aceita}

Vacuidade de autômatos linearmente limitados

- Redução de A_{MT} a V_{ALL}
- Suponha que existe uma MT R que decida V_{ALL}:
- S decide A_{MT}:

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

1. Construa o ALL B a partir de M e w
2. Rode R sobre <B>
3. Se R rejeita, *aceite*; se R aceita, *rejeite*.”

Mas A_{MT} é indecidível, então V_{ALL} também é

Vacuidade de autômatos linearmente limitados

- Redução de A_{MT} a V_{ALL}
- Suponha que existe uma MT R que decida V_{ALL}:
- S decide A_{MT}:

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

1. Construa o ALL B a partir de M e w
2. Rode R sobre <B>
3. Se R rejeita, *aceite*; se R aceita, *rejeite*."

Ou seja, não pretendemos executar B, apenas usar sua descrição para executar R !!!

Mas A_{MT} é indecidível, então V_{ALL} também é

Reduções via histórias de computação

- A mesma estratégia pode ser usada para provar a indecidibilidade de outros problemas
- Por exemplo...

Se uma GLC gera todas as cadeias

- $TODAS_{GLC}$ =
-
-
  -
  -
-
-

Se uma GLC gera todas as cadeias

- $TODAS_{GLC}$ = {<G>|G é uma GLC e L(G) = Σ^*} é indecidível
- Prova por "histórias da computação" e redução a partir de A_{MT}
- Para uma dada MT M e uma cadeia w, construimos uma GLC G que reconheça todas as cadeias que NÃO sejam histórias de computação de M sobre w
  - Se M aceita w, G gera todas as cadeias MENOS a história de computação de M sobre w
  - Se M não aceita w, G gera todas as cadeias
- Se G gerar todas as cadeias, então M NÃO aceita w, caso contrário M aceita w
- Se $TODAS_{GLC}$ fosse decidível, A_{MT} também seria. Logo, $TODAS_{GLC}$ é indecidível

Se uma GLC gera todas as cadeias

- $TODAS_{GLC}$ = {<G>|G é uma GLC e L(G) = Σ*} é indecidível
- Prova por "histórias da computação" e redução a partir de A_{MT}
- Para uma dada MT M e uma cadeia w, construimos uma GLC G que reconheça todas as cadeias que NÃO sejam histórias de computação de M sobre w
  - Se M aceita w, G gera todas as cadeias MENOS a história de computação de M sobre w
  - Se M não aceita w, G gera todas as cadeias
- Se G gerar todas as cadeias, então M NÃO aceita w, caso contrário M aceita w
- Se $TODAS_{GLC}$ fosse decidível, A_{MT} também seria. Logo, $TODAS_{GLC}$ é indecidível

Se uma GLC gera todas as cadeias

- Vamos construir um autômato a pilha equivalente a G (para aceitar cadeias que NÃO sejam histórias de computação de M sobre w), ou seja, cadeias que apresente AO MENOS UMA das seguintes propriedades (testadas não-deterministicamente):

1. **não** começa com C1

   C1 = $q_0w_1w_2...w_n$, onde q_0 é o estado inicial de M

2. alguma Ci+1 **não** é originada de Ci

3. **não** termina com uma configuração de aceitação

Se uma GLC gera todas as cadeias

- Vamos construir um autômato a pilha equivalente a G (para aceitar cadeias que NÃO sejam histórias de computação de M sobre w), ou seja, cadeias que apresente AO MENOS UMA das seguintes propriedades (testadas não-deterministicamente):

1. **não** começa com C1

   C1 = $q_0w_1w_2...w_n$, onde q_0 é o estado inicial de M

2. alguma Ci+1 **não** é originada de Ci

   como?

3. **não** termina com uma configuração de aceitação

Se uma GLC gera todas as cadeias

- Vamos construir um autômato a pilha equivalente a G (para aceitar cadeias que NÃO sejam histórias de computação de M sobre w), ou seja, cadeias que apresente AO MENOS UMA das seguintes propriedades (testadas não-deterministicamente):

1. **não** começa com C1

   C1 = $q_0w_1w_2...w_n$, onde q_0 é o estado inicial de M

2. alguma Ci+1 **não** é originada de Ci

   como? Fita: $\#C1\#C2^R\#C3\#C4^R\#...\#Cl\#$

   Vou empilhando símbolos até terminar de ler Ci, e começo a desempilhar para comparar com Ci+1 (não deterministicamente)

3. **não** termina com uma configuração de aceitação

Se uma GLC gera todas as cadeias

- Ex. 5.1 para provar que a equivalência entre duas GLCs é indecidível.
  - Para isso, usar $TODAS_{GLC}$