Problemas Indecidíveis e Reduções

Prof^a Jerusa Marchi

Departamento de Informática e Estatística

Universidade Federal de Santa Catarina

e-mail: jerusa@inf.ufsc.br

Decidibilidade

- Problemas indecidíveis
  - Problemas que não podem ser *resolvidos* por uma MT
- Como saber se um problema é solucionável ou não??
  - Escrevendo um algoritmo para ele
  - Provando que não existe tal algoritmo

Redução

- Uma **Redução** é um meio de converter um problema em outro de forma que uma solução para o segundo problema possa ser utilizada para resolver o primeiro
  - Sejam dois problemas A e B. Se A é reduzido a B, então, uma solução para B pode ser usada para resolver A
  - Isso significa que a solução para o problema A é tão custosa quanto a solução para o problema B, pois uma solução para B é uma solução para A
    - Se B é decidível, A também o é.
    - Se A é indecidível e redutível a B, então B também é indecidível.

Redução

- Sejam L_1 e $L_2 \subseteq \Sigma^*$ duas linguagens. Uma redução de L_1 para L_2 é uma função recursiva $\tau : \Sigma^* \mapsto \Sigma^*$, tal que $x \in L_1$ se e somente se $\tau(x) \in L_2$

Redução

- Para mostrar que uma linguagem L_2 é não recursiva, deve-se identificar uma linguagem L_1 sabidamente não recursiva e, então, reduzir L_1 a L_2
  - Observe que reduzir L_2 a L_1 seria inócuo, pois apenas mostra que L_2 só poderá ser decidida se pudermos decidir L_1
  - equivale a dizer "se L_1 é decidível, então L_2 é decidível", sendo portanto falsa a hipótese

Redução

- Formalmente, o uso correto de reduções em provas de indecidibilidade é o seguinte:
  - Se L_1 é uma linguagem não-recursiva, e se houver uma redução de L_1 para L_2 então L_2 também é não recursiva
    - **Prova:** Seja L_2 uma linguagem recursiva. Seja M_2 uma MT que decida L_2, e T uma MT que computa a redução τ. Nessas condições, a MT TM_2 deveria decidir L_1. Mas L_1 é indecidível. Contradição.
  - Em outras palavras
    - Se um problema A é indecidível, e se houver uma redução de A para B então B também é indecidível

Problemas Indecidíveis

- Da indecidibilidade do problema da parada, decorre a indecidibilidade de uma grande variedade de problemas
  - São indecidíveis os seguintes problemas acerca de Máquinas de Turing
    1. Dada uma máquina de Turing M, a linguagem de M é vazia
    2. Dada uma MT M, a linguagem que M semidecide é regular? Livre de contexto? Recursiva?
    3. Dadas duas máquinas de Turing M_1 e M_2, elas param em resposta às mesmas cadeias de entrada?

Problemas Indecidíveis

- Dada uma máquina de Turing M, a linguagem de M é vazia

$$V_{MT} = \{\langle M \rangle \mid M \text{ é uma MT e } L(M) = \emptyset\}$$

V_{MT} é indecidível.

- **Ideia da prova**:
  - Suponha que a linguagem V_{MT} é decidível. Seja R uma MT que decide V_{MT}. Use essa asserção para construir uma MT S que decide o problema da parada.
  - A ideia é S rodar R sobre a entrada $\langle M \rangle$ e ver se R aceita. Se aceita, então $L(M)$ é vazia, e por conseguinte, M não aceita w. Mas se R rejeita $\langle M \rangle$, então $L(M)$ é não vazia e consequentemente M aceita alguma cadeia, porém não sabemos se M aceita a cadeia específica w. Ou seja, precisamos tentar uma ideia diferente.

Problemas Indecidíveis

- Dada uma máquina de Turing M, a linguagem de M é vazia

$$V_{MT} = \{\langle M \rangle \mid M \text{ é uma MT e } L(M) = \emptyset\}$$

V_{MT} é indecidível.

- **Ideia da prova**:
  - Em vez de rodar R sobre $\langle M \rangle$, rodamos R sobre uma modificação de $\langle M \rangle$. $\langle M \rangle$ é modificada para garantir que M rejeite todas as cadeias, exceto w, mas que sobre a entrada w ela funcione normalmente. Então R é usada para determinar se a máquina modificada reconhece a linguagem vazia. A única cadeia que a máquina agora aceita é w e, portanto, sua linguagem será não vazia sse ela aceita w. Se R aceita quando é alimentada com uma descrição da máquina modificada, sabemos que a máquina modificada não aceita nada e que M não aceita w.

Problemas Indecidíveis

seja M_1 a máquina modificada construída a partir de M:

$M_1 =$ sobre a entrada x :

1. Se $x \neq w,$ rejeite
2. Se $x = w,$ roda M com a entrada w e aceite se M aceita

Essa máquina tem a cadeia w como parte de sua descrição. Ela conduz o teste $x = w$, fazendo a varredura na entrada e comparando-a com w.

Problemas Indecidíveis

- Supomos que R decide V_{MT} e construímos a MT S que decide o problema da parada como segue:
  $S =$ Sobre a entrada $\langle M, w \rangle$:
  1. Use a descrição de M e w para construir a MT M_1 como descrito
  2. Rode R com a entrada $\langle M_1 \rangle$
  3. Se R aceita, *rejeite*; se R rejeita, *aceite*

Se R fosse um decisor para V_{MT}, S deve ser um decisor para o problema da parada. Um decididor para o problema da parada não existe, logo, V_{MT} é indecidível.

Problemas Indecidíveis

- saber quando a linguagem de uma máquina de Turing é regular

$$L_{MT_{reg}} = \{\langle M \rangle \mid M \text{ é uma MT e } L(M) \text{ é uma linguagem regular}\}$$

- Ideia de prova: Redução a partir do problema da parada. Assuma que $L_{MT_{reg}}$ é decidível por uma MT R e use esta asserção para construir uma MT S que decide o problema da parada.
- S recebe como entrada $\langle M, w \rangle$, então modifica-se M de forma que a MT reconheça uma linguagem regular sse M aceita w. Chamemos esta máquina modificada de M_2. M_2 é projetada para reconhecer a linguagem não regular $\{0^n 1^n | n \geq 0\}$ se M não aceita w e para reconhecer a linguagem regular Σ^* se M aceita w.
- Devemos especificar como S constrói M_2 a partir de M e w.

Problemas Indecidíveis

- Seja R uma MT que decide $L_{MT_{reg}}$ e seja S uma MT para decidir o problema da parada. Então S trabalha da seguinte forma:
  S = Sobre a entrada $\langle M, w \rangle$

  1. Constrói a seguinte MT M_2
     M_2 = Sobre a entrada x:
     (a) Se x tem a forma $0^n 1^n$, aceite
     (b) Se x não tem esta forma, então rode M sobre a entrada w e aceite se M aceita w
  2. Roda R com a entrada $\langle M_2 \rangle$
  3. Se R aceita, aceite; Se R rejeita, rejeite.

- Porém o problema da parada é indecidível, logo R não existe.

Problemas Indecidíveis

- Similarmente, também são problemas indecidíveis:
  1. saber quando a linguagem de uma máquina de Turing é livre de contexto
  2. saber quando a linguagem de uma máquina de Turing é sensível ao contexto
  3. saber quando a linguagem de uma máquina de Turing é decidível
  4. saber quando a linguagem de uma máquina de Turing é finita
- Ou seja é indecidível testar qualquer propriedade das linguagens reconhecidas por máquinas de Turing
- Tais propriedades são ditas não triviais

Teorema de Rice

- Dada uma propriedade P de uma linguagem recursivamente enumerável, ela é trivial se e somente se ela não é satisfeita por nenhuma linguagem Recursivamente Enumerável, ou é satisfeita por todas as linguagens Recursivamente Enumeráveis.
- Teorema de Rice: Seja $\mathcal{C}$ um subconjunto próprio não-vazio da classe de linguagens recursivamente enumeráveis. Então o seguinte problema é indecidível: dada uma máquina de Turing M, $L(M) \in \mathcal{C}$?
  - O teorema de Rice prova a indecidibilidade de todas as propriedades não triviais de linguagens Recursivamente Enumeráveis ("ser livre de contexto", "ser finita", "ser decidível", etc)
  - (sugestão: leitura do problema 5.28 do Sipser)

Problemas Indecidíveis

- A redução é transitiva, ou seja, qualquer outro problema demonstrado indecidível pode ser usado em provas por redutibilidade
- Dadas duas máquinas de Turing M_1 e M_2, elas param em resposta às mesmas cadeias de entrada.

$$EQ_{MT} = \{\langle M_1, M_2 \rangle \mid M_1 \text{ e } M_2 \text{ são MTs e } L(M_1) = L(M_2)\}$$

EQ_{MT} é indecidível

- ideia da prova: Mostrar que, se EQ_{MT} fosse decidível, V_{MT} também seria, dando uma redução de V_{MT} para EQ_{MT}. Se umas das linguagens de M_1 ou M_2 for vazia, precisamos apenas determinar se a linguagem da outra máquina é vazia, ou seja V_{MT}.

Problemas Indecidíveis

$$EQ_{MT} = \{\langle M_1, M_2\rangle \mid M_1 \text{ e } M_2 \text{ são MTs e } L(M_1) = L(M_2)\}$$

EQ_{MT} é indecidível

- Suponha que a MT R decide EQ_{MT} e construa a MT S para decidir $V_M T$ da seguinte forma: S = Sobre a entrada $\langle M\rangle$, onde M é uma MT:
  1. Rode R sobre a entrada $\langle M, M_1\rangle$ onde M_1 é uma MT que rejeita todas as entradas.
  2. Se R aceita, aceite; se R rejeita, rejeite.
- Se R decide EQ_{MT}, S decide V_{MT}. Mas V_{MT} é idecidível, portanto EQ_{MT} também o é.

Reduções via Histórias de Computação

- Uma história de computação para uma MT sobre uma entrada é simplesmente a sequência de configurações pelas quais a máquina passa à medida que processa a entrada.
- Seja M uma MT e w uma cadeia de entrada. Uma **história de computação de aceitação** para M sobre w é uma sequência de configurações, C_1, C_2,...,C_l, onde C_1 é a configuração inicial de M sobre w, C_l é uma configuração de aceitação de M e cada C_i segue legitimamente de C_{i-1} conforme as regras de M. Uma **história de computação de rejeição** é similar, exceto que C_l é uma configuração de rejeição

Reduções via Histórias de Computação

- Histórias de Computação são finitas
- Se M não pára sobre w então nenhuma história de computação de aceitação ou rejeição existe para M sobre w
- Máquinas determinísticas têm no máximo uma história de computação sobre qualquer entrada
- Máquina não determinísticas podem ter muitas histórias de computação sobre uma única entrada

Autômatos Linearmente Limitados

- Um Autômato Linearmente Limitado (ALL) é um tipo restrito de máquina de Turing no qual à cabeça de leitura-escrita não é permitido mover-se para fora da parte da fita contendo a entrada.
  - Se a máquina tentar mover sua cabeça para além de qualquer uma das extremidades da entrada, a cabeça permanecerá onde está
  - Ou seja, a memória da máquina é limitada
  - A utilização de um alfabeto de fita maior que o alfabeto de entrada permite que a memória disponível seja incrementada por, no máximo, um fator constante. Ou seja, para uma entrada de tamanho n, a quantidade de memória disponível é **linear** em n

Autômatos Linearmente Limitados

- ALL são poderosos:
  - Decisores para Linguagens Regulares, para vacuidade de AF, Linguagens Livres de Contexto e Vacuidade de AP.
  - Seja A_{ALL} o problema de se determinar se um ALL aceita sua entrada

$$A_{ALL} = \{\langle M, w\rangle \mid M \text{é um ALL que aceita a cadeia} w\}$$

  - A_{ALL} é decidível

Autômatos Linearmente Limitados

- **Lema:** Seja M um ALL com q estados e g símbolos no alfabeto de fita. Existem exatamente qng^n configurações distintas de M para uma fita de comprimento n.
  Prova: Uma configuração é constituída do estado do controle, posição da cabeça e conteúdo da fita. Aqui, M tem q estados. O comprimento da fita é n, portanto, a cabeça pode estar em uma das n posições e g^n cadeiras possíveis de símbolos de fita aparecem sobre a fita. O produto dessas três quantidades é o número total de configurações diferentes de M.

Autômatos Linearmente Limitados

- A_{ALL} é decidível
- **Ideia de Prova:** Para decidir se ALL M aceita a entrada w, simulamos M sobre w. Durante o curso da simulação, se M pára e aceita ou rejeita, aceitamos ou rejeitamos em conformidade com M. Para detectar quando M entrou em loop, à medida que M computa sobre w, ela vai de configuração em configuração. Pelo fato de M ser um ALL, a quantidade de fita disponível é limitada. Pelo lema anterior, M pode estar em apenas um número limitado de configurações sobre o tamanho da fita. É possível detectar que M está em loop simulando M pelo número de passos dados pelo lema anterior. Se M não parou, é porque ela está em loop.

Autômatos Linearmente Limitados

- A_{ALL} é decidível
- **Prova:** $L =$ Sobre a entrada $\langle M, w \rangle$ onde M é um ALL e w é uma cadeia:
  1. Simule M sobre w por qng^n passos ou até que M páre
  2. Se M parou, *aceite* se M aceitou ou *rejeite* se M rejeitou. Se M não parou, *rejeite*