ACH2043
INTRODUÇÃO À TEORIA DA COMPUTAÇÃO

Introdução à Teoria da Computação. Michael Sipser. Thomson Learning, 2007.

Aula 15

Cap 3.3 – Definição de algoritmo

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

O que é um algoritmo?

O que é um algoritmo?

Muito “usado” há tempos, mas formalmente definido apenas no século XX

Um pouco de história

- 1833 – Charles Babbage e a concepção da máquina analítica (programável)
- Ada Lovelace
  - criou estruturas de programas para a máquina analítica (loops, saltos condicionais, sub-rotinas,...)
  - Inventou a palavra algoritmo em homenagem ao matemático Al-Khawarizmi (820 D.C.)
- Mas algoritmos ainda eram uma noção intuitiva...

Um pouco de história

- 1900 – palestra do matemático David Hilbert
  - 23 desafios matemáticos para o próximo século
  - Décimo problema: “um processo pelo qual possa ser determinado, com um número finito de operações”, se um polinômio tem raízes inteiras.
- 1936 – artigos de Alonzo Church e Alan Turing definindo formalmente um algoritmo
  - Church com lambda-cálculo
  - Turing com Máquinas de Turing
  - As duas formulações são equivalentes

Tese de Church-Turing

Noção intuitiva de algoritmos	é igual a	*algoritmos de máquina de Turing*

Vamos olhar com mais detalhe essa tese —> slides Tese de Church

Algoritmo para o problema de Hilbert

- Problema de Hilbert:

um processo pelo qual possa ser determinado, com um número finito de operações", se um polinômio tem raízes inteiras

- 1970 – Yuri Matijasevic mostrou que não existe tal "processo" (ou algoritmo)

A tese de Church–Turing provê a definicao de algoritmo necessaria para resolver o decimo problema de Hilbert.

Algoritmo para o problema de Hilbert

- Problema de Hilbert:

D = { p | p é um polinômio com uma raiz inteira}

D é decidível?

Problema de Hilbert na forma de linguagem

Algoritmo para o problema de Hilbert

- Problema de Hilbert:

  D = { p | p é um polinômio com uma raiz inteira}

  D é decidível?

- 1970 – Yuri Matijasevic mostrou que não
- Próximo capítulo: como fazer esse tipo de prova.

Problema simplificado

$D_1 = \{p \mid p \text{ é um polinômio sobre } x \text{ com uma raiz inteira}\}.$

Problema simplificado

$D_1 = \{p \mid p \text{ é um polinômio sobre } x \text{ com uma raiz inteira}\}$.

Aqui está uma MT M_1 que reconhece D_1:

M_1 = "A entrada é um polinômio p sobre a variável x.

1. Calcule o valor de p com x substituída sucessivamente pelos valores $0, 1, -1, 2, -2, 3, -3, \ldots$ Se em algum ponto o valor do polinômio resulta em 0, *aceite*."

Decidível?

Problema simplificado

$D_1 = \{p \mid p \text{ é um polinômio sobre } x \text{ com uma raiz inteira}\}$.

Aqui está uma MT M_1 que reconhece D_1:

M_1 = "A entrada é um polinômio p sobre a variável x.

1. Calcule o valor de p com x substituída sucessivamente pelos valores $0, 1, -1, 2, -2, 3, -3, \ldots$ Se em algum ponto o valor do polinômio resulta em 0, *aceite*."

Decidível? Sim...
As raízes de um polinômio de uma só variável devem residir entre os dois valores:

$$\pm k \frac{c_{\text{máx}}}{c_1},$$

Exercício: Refinar a descrição de M1 para mostrar que D1 é decidível

onde k é o número de termos no polinômio, $c_{\text{máx}}$ é o coeficiente com o maior valor absoluto, e c_1 é o coeficiente do termo de mais alta ordem. Se uma raiz não for encontrada dentro desses limitantes, a máquina *rejeita*.

O problema original

Matijasevic mostra que, para polinômios com várias variáveis, não é possível calcular tais limitantes

Logo, D é

O problema original

Matijasevic mostra que, para polinômios com várias variáveis, não é possível calcular tais limitantes

Logo, D é Turing-reconhecível mas não Turing-decidível

Exercício: Mostrar que D é Turing-reconhecível

Terminologia para descrever Máquinas de Turing

- Mudança de foco no curso: algoritmos
  - Máquina de Turing como modelo
  - Precisamos estar convencidos de que podemos descrever qualquer algoritmo com uma máquina de Turing

A prática com a construção de mTs (em baixo nível) ajuda a entender mTs e ganhar confiança no uso delas…

—> tese de Church-Turing

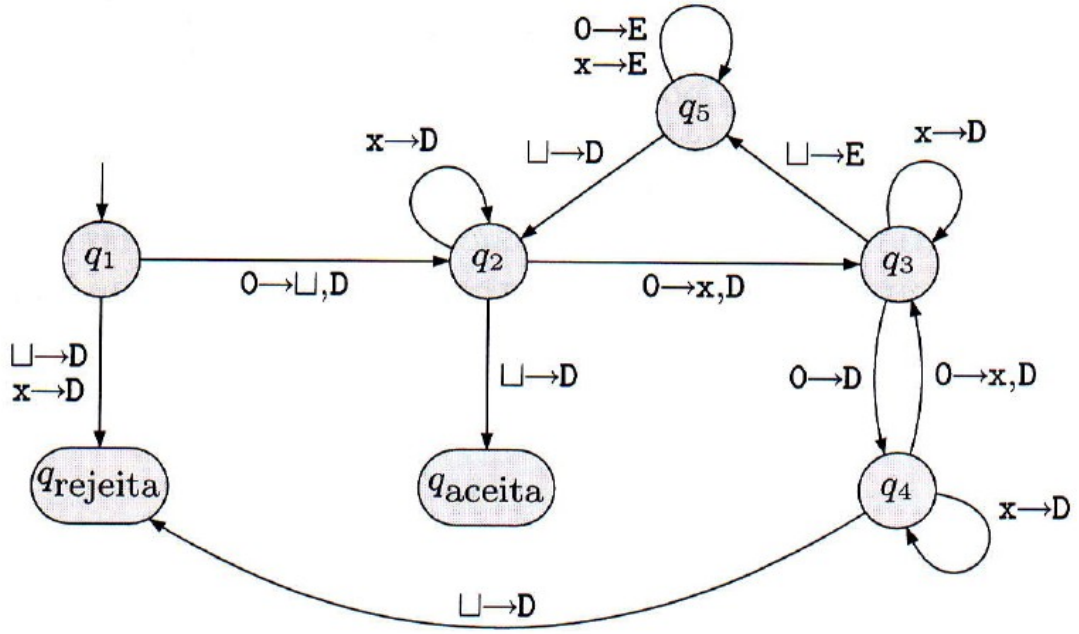
Terminologia para descrever Máquinas de Turing

- 3 níveis de descrição de algoritmos:
  - **Descrição formal**: detalhes da máquina: estados, função de transição, etc.
  - **Descrição de implementação**: escrito em língua natural para descrever como a máquina move a cabeça da fita, lê e escreve dados, etc (sem descrever estados ou função de transição)
  - **Descrição de alto nível**: escrito em língua natural para descrever um algoritmo, omitindo detalhes de implementação

Exemplo – descrição formal (se o nr de zeros de uma cadeia é uma potência de 2)

Agora, damos a descrição formal de $M_2 = (Q, \Sigma, \Gamma, \delta, q_1, q_{\text{aceita}}, q_{\text{rejeita}})$:

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_{\text{aceita}}, q_{\text{rejeita}}\}$,
- $\Sigma = \{0\}$ e
- $\Gamma = \{0, x, \sqcup\}$.
- Descrevemos δ com um diagrama de estados (veja a Figura 3.8).
- Os estados inicial, de aceitação e de rejeição são q_1, q_{aceita} e q_{rejeita}.



Exemplo – descrição de implementação (se o nr de zeros de uma cadeia é uma potência de 2)

EXEMPLO 3.7

Aqui descrevemos uma máquina de Turing (MT) M_2 que decide $A = \{0^{2^n} \mid n \geq 0\}$, a linguagem consistindo em todas as cadeias de 0s cujo comprimento é uma potência de 2.

M_2 = "Sobre a cadeia de entrada w:

1. Faça uma varredura da esquerda para a direita na fita, marcando um 0 não, e outro, sim.
2. Se no estágio 1, a fita continha um único 0, *aceite*.
3. Se no estágio 1, a fita continha mais que um único 0 e o número de 0s era ímpar, *rejeite*.
4. Retorne a cabeça para a extremidade esquerda da fita.
5. Vá para o estágio 1."

Exemplo – descrição de alto nível (se um polinômio sobre x tem raiz inteira)

M_1 = "A entrada é um polinômio p sobre a variável x.

1. Calcule o valor de p com x substituída sucessivamente pelos valores $0, 1, -1, 2, -2, 3, -3, \ldots$ Se em algum ponto o valor do polinômio resulta em 0, *aceite*."

Terminologia para descrever Máquinas de Turing

- Até agora usamos as descrições formais e de implementação
- Passaremos a usar mais a descrição de alto nível
  - Objetos (O) convertidos em cadeias (<O>)
  - Vários objetos em uma única cadeia (<O_1, O_2, ..., O_k>)
  - Assumimos que as MTs são capazes de decodificar essas cadeias

Exemplos de objetos: textos, gramáticas, autômatos ….

Descrição de alto nível de Máquinas de Turing

- M = “ …

“

- Primeira linha: entrada da máquina
  - w é cadeia
  - <w> é objeto codificado em cadeia – implicitamente MT testa se a codificação está ok, se não estiver rejeita

Como converter uma mT qualquer
em uma cadeia sobre o alfabeto {0,1} ?