Linguagens Formais e Autômatos

Aula 01 - Introdução

Prof. Dr. Daniel Lucrédio
Departamento de Computação / UFSCar
Última revisão: ago/2015

Referências bibliográficas

- **Introdução à teoria dos autômatos, linguagens e computação / John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman** ; tradução da 2.ed. original de Vandenberg D. de Souza. - Rio de Janeiro : Elsevier, 2002 (Tradução de: Introduction to automata theory, languages, and computation - ISBN 85-352-1072-5)
  - Capítulo 1 - Seções 1.1 e 1.5
- **Introdução à teoria da computação / Michael Sipser** ; tradução técnica Ruy José Guerra Barretto de Queiroz ; revisão técnica Newton José Vieira. -- São Paulo : Thomson Learning, 2007 (Título original : Introduction to the theory of computation. "Tradução da segunda edição norte-americana" - ISBN 978-85-221-0499-4)
  - Capítulo 0 - Seção 0.1

Introdução

Autômatos

Quais são as capacidades e limitações fundamentais dos computadores?

Computabilidade

Complexidade

Introdução

- Década de 1930
- Matemáticos
- Antes da existência dos computadores

Teoria da complexidade

- Alguns problemas são simples
  - Ordenação
- Outros são complexos
  - Escalonamento de aulas em uma universidade
  - 1000 aulas: o tempo para achar o melhor escalonamento pode levar séculos
  - Número com 500 dígitos: fatorar leva todo o tempo de vida do universo

O que torna alguns problemas computacionalmente difíceis e outros fáceis?

Teoria da complexidade

- Não sabemos a resposta!
- Mas temos um esquema para classificar problemas conforme sua dificuldade
- Várias opções...
  - Alterar o problema (ou aquele aspecto)
  - Contentar-se com uma solução menos do que perfeita
  - Alguns problemas só são difíceis no pior caso
- Ex: criptografia

Teoria da computabilidade

- Alguns problemas não podem ser resolvidos por computadores
  - Ex: um enunciado matemático é verdadeiro ou falso?
- Problemas solúveis vs insolúveis

Teoria dos autômatos

- Definições e propriedades de modelos matemáticos de computação
- Autômatos são abstrações matemáticas que ajudam a entender este modelo
  - Tem implicações e aplicações práticas

Autômatos e Linguagens

Autômatos e Linguagens

- Conceitos centrais
- Alfabeto: conjunto finito, não vazio, de símbolos
  - Símbolos = membros de um alfabeto
- Σ (sigma) ou Γ (gamma)
  - Ex: Σ = {0,1}
  - Σ = {a,b,c,...,z}
  - Σ = conjunto de caracteres ASCII

Autômatos e Linguagens

- Cadeia/string sobre um alfabeto
  - Sequência finita de símbolos daquele alfabeto
  - Geralmente escritos um seguido do outro e não separados por vírgulas
- Ex: 01001, if, then, while
- Se w é uma cadeia sobre Σ, o comprimento de uma cadeia |w| é o número de símbolos que ela contém
  - Estritamente, é o número de posições que conta
- Cadeia de comprimento zero
  - ε (epsilon minúsculo)

Autômatos e Linguagens

- Se w tem comprimento n
  - $w = w_1 w_2 ... w_n$, onde cada $w_i \in \Sigma$
- Reverso de w = w^R
  - Cadeia obtida escrevendo-se w na ordem inversa
  - $w_n w_{n-1} ... w_1$
- z é uma subcadeia/substring de w se z aparece consecutivamente dentro de w
  - cad é uma subcadeia de abracadabra

Autômatos e Linguagens

- Cadeia x de comprimento m
- Cadeia y de comprimento n
- Concatenação de x e y, escrito xy é:
  - $x_1x_2...x_my_1y_2...y_n$
  - $|xy|=m+n$
  - $xx...x = x^k$
  - $w\varepsilon=\varepsilon w=w$

Autômatos e Linguagens

- Potências de um alfabeto
- Σ^k
  - Conjunto de cadeias de comprimento k, e o símbolo de cada um deles está em Σ
- $\Sigma^0=\{\varepsilon\}$ (independente do alfabeto)
- Se $\Sigma=\{0,1\}$
  - $\Sigma^1=\{0,1\}$
  - $\Sigma^2=\{00,01,10,11\}$
  - $\Sigma^3=\{000,001,010,011,100,101,110,111\}$
  - Σ^*=conjunto de todas as cadeias sobre um alfabeto
  - $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots$
  - $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \cdots$
  - $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$

Linguagens

- Um conjunto de cadeias escolhidas de um alfabeto
  - Se Σ é um alfabeto
  - $L \subseteq \Sigma^*$
  - L é uma linguagem sobre Σ
    - E sobre qualquer alfabeto que contenha Σ
- Outras conclusões
  - Σ^* é uma linguagem para qualquer alfabeto Σ
  - $\varnothing$ (linguagem vazia) é uma linguagem sobre qualquer alfabeto
  - $\{\varepsilon\}$ é uma linguagem sobre qualquer alfabeto

Linguagens e Problemas

- O termo “linguagem” pode parecer estranho
  - Mas linguagens comuns podem ser vistas como um conjunto de cadeias
  - Ex: português/alfabeto latino, grego/alfabeto grego
  - Ex: C/ASCII
- Exemplos mais abstratos
  - Ex: Linguagens de todas as cadeias que consistem de n 0’s seguidos por n 1’s
  - Ex: Conjunto de cadeias com número igual de 0’s e 1’s

Linguagens

- Linguagens podem possuir infinitas palavras, mas alfabetos são finitos
- Problemas: na teoria dos autômatos, um problema é a decisão sobre se uma dada cadeia/palavra é parte de uma linguagem particular
  - Se Σ é um alfabeto, e L é uma linguagem sobre Σ, então um problema (formal) é:
    - Dada uma string w em Σ^*, decidir se w pertence ou não a L
- Também enquadra a noção coloquial de problema

Linguagens e Problemas

- Linguagens podem descrever situações reais
  - Ex: uma linguagem que descreve os caminhos entre minha casa e o trabalho:
    - DDEEFFDED (a cada esquina)
    - Há vários caminhos (cadeias)
  - Problema: encontrar o caminho mais curto
    - Ou: dado um caminho, é o mais curto?
    - Ou: encontre todos os caminhos até a minha casa
- Em termos da teoria da computação, são a mesma coisa
  - A dificuldade – em essência – é a mesma

Linguagens e Problemas

- Porém em alguns casos problemas são mais do que linguagens
- Ex: Dada uma string ASCII
  - Decidir se é ou não um elemento de Lc (conjunto de programas válidos em C)
- Mas por trás:
  - Compilador precisa fazer uma tarefa complexa
  - Produz uma árvore de análise sintática
  - Entradas em uma tabela de símbolos
  - Transformar um programa C em código-objeto
  - Outras tarefas de compiladores
  - Vai além de responder “sim” ou “não” sobre a validade de um programa

Linguagens e problemas

- Problemas são úteis à teoria da complexidade
  - Técnicas podem provar que alguns problemas não podem ser resolvidos facilmente
- O termo preferido depende do ponto de vista
  - Se estamos interessados nas palavras em si: linguagem
  - Se estamos mais interessados nas coisas que representam – semântica: problema
- Para a teoria computacional, estamos interessados em saber os limites de complexidade
  - Encontrar uma solução para a versão linguagem (sim/não) é tão difícil quanto a versão “resolva isso”
- É muito útil pensar em linguagens
  - Ao estudar teoria da computação

Linguagens e problemas

- Definindo linguagens
- Formador de conjuntos
  - {w | algo sobre w}
- Exs:
  - {w | w consiste em um número igual de 0’s e 1’s}
  - {w | w é um número inteiro binário primo}
  - {w | w é um programa em C sintaticamente correto}
  - $\{0^i1^j \mid 0 \leq i \leq j\}$

Fim

Aula 01 - Introdução