INE5602 – Introdução à Informática

Modelos abstratos e computabilidade

Aula 1: Definições e exemplos

Prof. Laércio Lima Pilla

laercio.pilla@ufsc.br

Material modificado pelo Prof. Alexandre em 2016/1

Sumário

- Definições
- Histórico
- Computabilidade
- Considerações finais

DEFINIÇÕES

Definições

- Programa
- Máquina
- Computação
- Função computada
- Algoritmo

Definições

- ***Programa***
  - Conjunto estruturado de instruções
    - Composição de instruções
  - Capacitam uma máquina a aplicar sucessivamente:
    - ***Operações básicas***
    - ***Testes***
  - Possuem uma estrutura de controle

Definições

- Estruturação
  - ***Monolítica***
    1. Se T vá para 2 senão vá para 3
    2. Faça F e vá para 1
  - Iterativa
  - Recursiva

Definições

- Estruturação
  - Monolítica
  - ***Iterativa***
    - Enquanto T faça F
  - Recursiva

Definições

- Estruturação
  - Monolítica
  - Iterativa
  - ***Recursiva***
    - P é R onde
      - R def (se T então F;R senão v)
      - onde v representa a operação vazia

Definições

- ***Máquina***
  - Dá significado aos identificadores das operações e testes
    - ***Operações: levam a uma transformação na memória***
    - ***Testes: possuem uma função verdade***
  - Nem todo identificador precisa ser definido em uma máquina
  - Existe apenas uma função associada a cada identificador (não ambiguidade)

Definições

- ***Computação***
  - Histórico das instruções de um programa executadas em uma máquina
  - Exemplo
    - Máquina de dois registradores
      - testa se é zero; subtrai 1; adiciona 1

    1. Se ***a_zero*** então vá para 9, senão vá para 2
    2. Faça ***subtrai_a*** vá para 3
    3. Faça ***adiciona_b*** vá para 1

Definições

- Computação finita em máquina de dois registradores, considerando a=3 e b=0:

  (1, ***(3,0)***)

  (2, ***(3, 0)***)

  (3, ***(2, 0)***)

  (1, ***(2, 1)***)

  (2, ***(2, 1)***)

  (3, ***(1, 1)***)

  (1, ***(1, 2)***)

  (2, ***(1, 2)***)

  (3, ***(0, 2)***)

  (1, ***(0, 3)***)

  (9, ***(0, 3)***)

Definições

- ***Função computada***
  - Computação de um programa associada a uma entrada e uma saída
  - Adicionalmente, espera-se que a resposta (saída) seja gerada em um ***tempo finito***

Definições

- ***Algoritmo***
  - Noção intuitiva
    - Solução de um problema
    - Descrito de forma finita e não-ambígua
    - Consiste de passos discretos
    - Executável em tempo finito
  - Usa recursos “tão grandes quanto necessários”

HISTÓRICO

Histórico

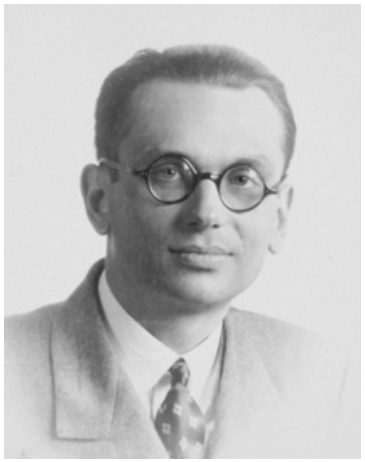
- Grandes nomes
  - Hilbert
  - Gödel
  - Church
  - Turing
  - Post
  - Markov

Histórico

- ***Hilbert***, David
  - ***Entscheidungsproblem (1928)***
    - Procedimento para demonstrar se uma dada fórmula no ***cálculo de predicados de primeira ordem*** era válida ou não, em tempo finito
  - Encontrar um conjunto completo e consistente de axiomas para toda a matemática

Histórico

- ***Gödel***, Kurt
  - Teorema da Não-Completude (1931)
    - Demonstrou que a mecanização do processo de provas não tem solução
    - A consistência dos axiomas não pode ser provada usando o próprio sistema formal
    - Uso de codificação de primos (***número de Gödel***), de forma que as formulações axiomáticas eram codificadas em números naturais

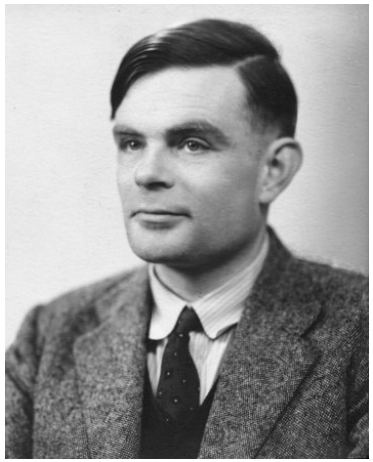


Histórico

- ***Church***, Alonzo
  - Cálculo Lambda
    - $f(x) = x^2+4 \rightarrow \lambda x.\ x^2+4$
      - função tal que, para um argumento arbitrário x resulta em x^2+4
  - Mostrou que o problema de Hilbert não tem solução (1936)
  - ***Hipótese de Church***

Histórico

- ***Turing***, Alan
  - Máquina de Turing
    - Modelo elementar que imita o comportamento de um computador
    - Fita com alfabeto finito
    - Leitura, escrita, e movimentos laterais
    - Número de estados finitos



Histórico

- ***Post***, Emil
  - Máquina de Post
    - Uso de uma ***fila*** para armazenar dados
    - Alfabeto finito
    - Leitura destrutiva
    - Elementos
      - Partida
      - Parada: uma de aceitação (aceita) e outra de rejeição (rejeita)
      - Desvio ou teste - múltiplos caminhos
      - Atribuição

Histórico

- ***Markov***, Andrey Jr.
  - Algoritmo de Markov
    - Regras de substituição em ordem
    - Finaliza quando não houver regra
    - Exemplo:

Regras:

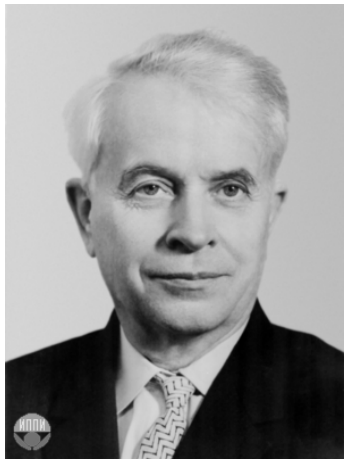
1. "|0" -> "0||"
2. "1" -> "0|"
3. "0" -> ""

Cadeia de entrada:

"101"

Execução:

1. "0|01"
2. "00||1"
3. "00||0|"
4. "00|0|||"
5. "000|||||"
6. "00|||||"
7. "0|||||"
8. "|||||"



COMPUTABILIDADE

Computabilidade

- Computabilidade e máquinas universais
  - Hipótese de Church
  - Máquinas universais
  - Como tudo se conecta
  - Exemplos do que não é computável
  - P e NP

Computabilidade

- ***Hipótese de Church***
  - "A capacidade de computação representada pela Máquina de Turing é o limite máximo que pode ser atingido por qualquer dispositivo de computação"
  - Qualquer função computável pode ser processada por uma Máquina de Turing
  - Ou seja, existe um algoritmo expresso na forma de Máquina de Turing capaz de processar a função

Computabilidade

- ***Máquina universal***
  - Possível representar qualquer algoritmo como um programa na máquina
  - Quaisquer recursos adicionais não a tornam mais poderosa
  - Computacionalmente equivalente a outras máquinas

Computabilidade

- Máquinas universais
  - Cálculo Lambda
  - Funções Recursivas
  - Máquina de Turing
  - Sistema Canônico de Post
  - Algoritmo de Markov

Computabilidade

- O que ***não é computável***
  - Equivalência de compiladores
    - Não existe algoritmo genérico que sempre pare capaz de comparar quaisquer dois compiladores (de linguagens livres do contexto como PASCAL), verificando se são equivalentes (se reconhecem a mesma linguagem);
  - Detector universal de loops
    - Dados um programa e uma entrada quaisquer, não existe algoritmo genérico capaz de verificar se o programa vai parar ou não para a entrada. Este problema é universalmente conhecido como o ***Problema da Parada***.

Computabilidade

- P versus NP
  - ***P = NP*** ou ***P != NP***?
  - Problema do Prêmio Millenium
    - Não solucionado
    - Vale $1,000,000

Computabilidade

- P versus NP
  - ***P = NP*** ou ***P != NP***?
    - Suponha que as soluções para um problema possam ser verificadas rapidamente. Então, suas soluções podem ser calculadas rapidamente também? A noção teórica do termo “rapidamente” usado aqui significa um algoritmo que executa em tempo polinomial. A classe geral dos problemas para os quais algum algoritmo pode fornecer uma resposta em tempo polinomial é chamada de “classe P” ou apenas “P”. Para alguns problemas, não há nenhuma maneira conhecida para encontrar uma resposta rapidamente, mas se houver informação que mostre qual é a resposta, esta pode ser verificada rapidamente. A classe de problemas em que uma reposta pode ser verificada em tempo polinomial é chamada NP.

CONSIDERAÇÕES FINAIS

Considerações finais

- Definições
- Grandes nomes
- O que é computável

Considerações finais

- Bibliografia
  - T. A. Diverio e P. B. Menezes. Teoria da Computação: Máquinas Universais e Computabilidade. 3ª ed. 2011.
    - http://books.google.com.br/books?id=459EInmoh2cC
- Próxima aula
  - Máquinas de estados
    - ***Tragam papel e caneta***

INE5602 – Introdução à Informática

Modelos abstratos e computabilidade

Aula 1: Definições e exemplos

Prof. Laércio Lima Pilla

laercio.pilla@ufsc.br