A Tese de Church-Turing

Teoria da Computação – Ciência da Computação

INSTITUTO
FEDERAL
Brasília

Prof. Daniel Saad Nogueira Nunes

IFB – Instituto Federal de Brasília, Campus Taguatinga

Sumário

Sumário

Sumário

A Definição de algoritmo

- O que é um algoritmo?

A Definição de algoritmo

- Informalmente: uma sequência finita de instruções simples para realizar uma tarefa.
- Mas qual a definição formal de algoritmo?
- Precisamos de uma noção formal deste conceito para podermos saber os limites da computação.

Sumário

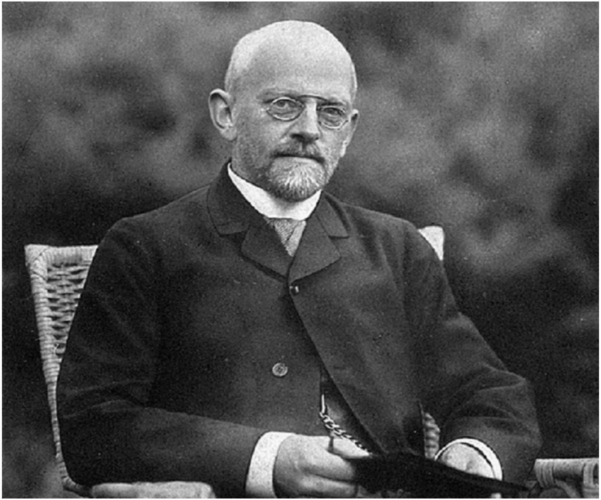
O 10° problema de Hilbert

- Em 1900, Hilbert uma palestra na conferência "International Congress of Mathematicians" em Paris.
- Nesta palestra ele propôs 23 problemas matemáticos para o século XX.
- O décimo problema na lista era relacionado com o conceito de algoritmo.

O 10° problema de Hilbert

- Em 1900, Hilbert uma palestra na conferência "International Congress of Mathematicians" em Paris.
- Nesta palestra ele propôs 23 problemas matemáticos para o século XX.
- O décimo problema na lista era relacionado com o conceito de algoritmo.
- E também era relacionado com polinômios.

O 10° problema de Hilbert



O 10° problema de Hilbert

- Um polinômio é uma soma de **termos**.
- Cada termo é um produto de variáveis e uma constante, chamada de **coeficiente**.
- Exemplo de termo:

$$6 \cdot x \cdot x \cdot x \cdot y \cdot z \cdot z = 6x^3yz^2$$

- Exemplo de polinômio:

$$6x^3yz^2 + 3xy^2 - x^3 - 10$$

O 10° problema de Hilbert

- Para esta discussão vamos considerar apenas polinômios com coeficientes inteiros.

O 10° problema de Hilbert

- Para esta discussão vamos considerar apenas polinômios com coeficientes inteiros.

O 10° problema de Hilbert

- Uma raiz de um polinômio é uma valoração das variáveis do mesmo de modo que o valor do polinômio seja 0.
- Para o polinômio:

$$6x^3yz^2 + 3xy^2 - x^3 - 10$$

temos as raízes $x = 5$, $y = 3$ e $z = 0$.

O 10 $^\circ$ problema de Hilbert

- Alguns polinômios possuem raízes inteiras.
- Outros não.
- O décimo problema de Hilbert consistia em

  Determinar um processo que pode ser especificado de acordo com um número finito de operações e que determina se um polinômio tem, ou não, raízes inteiras.
- Note que Hilbert sequer utilizou a palavra algoritmo, uma vez que este conceito ainda não estava definido precisamente.

O 10 ° problema de Hilbert

- Interessantemente, da maneira com que Hilbert definiu o problema, deu a entender que o algoritmo existia, alguém só precisaria descobri-lo.
- Hoje sabemos que um algoritmo para este problema não existe.
- Mas para os matemáticos da época, era difícil responder esta questão sem uma noção precisa do que é um algoritmo.
- A noção imprecisa de algoritmos era útil para descrever e automatizar algumas tarefas, mas era inútil para mostrar que não poderia existir algoritmos para tarefas específicas.

Sumário

A noção precisa de algoritmo

- A definição precisa de algoritmo veio com os trabalhos de Alonzo Church e Alan Turing, ambos em 1936.
- Church propôs um mecanismo conhecido como cálculo-λ para definir algoritmos.
- Turing, por sua vez, formalizou o conceito através de suas máquinas.

A noção precisa de algoritmo

- Estas duas noções mostraram-se equivalentes.
- Tudo que um formalismo fazia, o outro também era capaz de fazer.
- Foi estabelecida uma conexão entre a noção informal de algoritmo e a definição precisa de algoritmo.

A noção precisa de algoritmo

- Com a definição de algoritmo, agora era possível formalizar o décimo problema de Hilbert.
- Basicamente queremos saber se a linguagem:

$$D = \{p | p \text{ é um polinômio com raiz inteira } \}$$

  é Turing-decidível.
- Ou seja, queremos saber se existe uma máquina de Turing que sempre para, e diz **aceita**, caso $w \in D$ e diz rejeita, caso $w \notin D$.

A noção precisa de algoritmo

- Obviamente a linguagem é turing reconhecível.
- Conseguimos construir uma máquina que valora as variáveis com $\{0, -1, 1, -2, 2, -3, 3, \ldots\}$.
- Caso alguma valoração faz com que o polinômio seja avaliado em 0, a máquina para e aceita o polinômio.

A noção precisa de algoritmo

- Está claro que a máquinas M reconhece D , mas será que existe alguma máquina M' que decide D?

A noção precisa de algoritmo

- Está claro que a máquinas M reconhece D , mas será que existe alguma máquina M' que decide D?
- **Não**. Provado em 1970 por Matijasevič.

A tese de Church-Turing

- **Tese de Church-Turing**: define que tudo que é computável, deve ser computável por Máquinas de Turing, ou λ-cálculo ou outro mecanismo Turing-completo.
- Não pode ser provada, pois ela tenta dar precisão para um conceito informal.
- No entanto é altamente aceita, pois qualquer outro modelo capaz de computar mais coisas que uma máquina de Turing ou modelo equivalente é demasiadamente "exagerado".

A tese de Church-Turing

Função Computável		Assembly x86
$\Updownarrow$		$\Updownarrow$
Máquina de Turing	$\Leftrightarrow$	Programa em `Python`
$\Updownarrow$		$\Updownarrow$
Cálculo-λ		Programa em `C`

Sumário

Algoritmos

- Chegamos em um ponto crítico do curso.
- Vamos continuar falando de máquinas de Turing.
- No entanto o nosso foco agora será sobre algoritmos.
- Máquinas de Turing serviram como um modelo preciso para capturar a noção de algoritmo.
- Mas agora só precisamos acreditar nisso e podemos discutir as coisas em um nível um pouco mais alto.
- Não perderemos nada com isso devido à tese de Church-Turing.

Algoritmos

- Existem várias formas de descrever algoritmos.
- Descrição formal: dando os estados e as transições de uma máquina de Turing.
- Descrição da implementação: descrevemos como a máquina opera em termos de movimento de fita.
- **Descrição alto nível**: usamos inglês ou pseudocódigo para descrever o algoritmo. Não nos preocupamos em mencionar como a máquina opera a fita ou a cabeça de leitura/escrita.

Algoritmos

- Praticamos com Máquinas de Turing para ganharmos intuição de como ela opera.
- Uma vez que acreditamos que esse formalismo captura a noção de algoritmo e já ganhamos confiança com ele, podemos trabalhar em um nível de descrição mais abstrato.
- Máquinas de turing conseguem codificar um objeto O com $\langle O \rangle$.

Algoritmos

Exemplo

Seja A a linguagem de todas as *strings* representando grafos conexos.

$$A = \{\langle G \rangle \,|G \text{ é um grafo conexo } \}$$

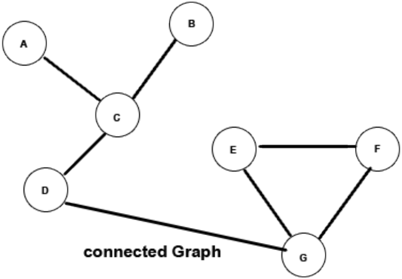
Como seria a descrição de um alto nível de uma MT que decide A? Ou seja, de um algoritmo.

Algoritmos

Algorithm 1: Testando a conectividade de G

Selecione qualquer nó $v \in G$ e o marque
Enquanto não houver novos nós marcados:
    Para cada nó em G, marque-o se ele possui aresta para um nó que já está marcado.
Inspecione todos os nó de G, se algum ainda não está marcado, **rejeite**, caso contrário, **aceite**

Algoritmos



Algoritmos