Introdução a Teoria da Computação

Juliana Kaizer Vizzotto

Universidade Federal de Santa Maria

Disciplina de Teoria da Computação

Quais são as capacidades e limitações fundamentais dos computadores?

Funções Computáveis

- Algoritmo:
  - descrição finitade uma computação em termos de operações (ou instruções) elementares bem-definidas;
  - um procedimento determinístico: o próximo passo é unicamente definido, se ele existe;
  - um método que sempre produz um resultado, não importando qual seja a entrada (i.e., a computação descrita por um algoritmo sempre termina).

Teoria da Computabilidade

- Formalizar o conceitos de *algoritmo* sem se referir a uma linguagem de programação ou dispositivo físico específicos.
- Um modelo de computação abstrai dos detalhes do dispositivo físico que estamos utilizando para fazer cálculos:
  - Abacus: caneta e papel
  - Linguagem de programação
  - Processador

Turing e Church (1930)

- Significado de computação como um processo mental abstrato
- Dispositivos teóricos para modelar o processo de computação.
- Utilizados para expressar algoritmos e computações não-terminantes

Funções Parciais

- A noção de função parcial generaliza a noção de algoritmo considerando processos que nem sempre levam a um resultado.

Funções Parciais

- A noção de função parcial generaliza a noção de algoritmo considerando processos que nem sempre levam a um resultado.
- Exemplos:
  - *True* $+ 4$ não é definida
  - $10/0$ não é definida
  - `fatorial(-1)` não tem um valor se `fatorial` é uma função recursiva definida como:

    ```
    fatorial(0)=1
    fatorial(n)= n * fatorial(n-1)
    ```

Funções Parciais

- O primeiro é um erro de tipo, pois a adição é uma função de números para números. Para qualquer número natural, a adição é uma função bem definida.
- A adição é uma função total sobre os números naturais.
- O segundo é um tipo diferente de problema: 10 e 0 são números! Mas a divisão por 0 não é definida!
- Dizemos que a divisão é uma função parcial sobre os números naturais.
- Existe ainda um outro caso no qual uma expressão pode não ter um valor: a computação pode ficar em um *loop* infinito.
- Dizemos que `fatorial` é uma função parcial sobre os números inteiros.

Funções Parciais

Definição: Sejam A e B dois conjuntos. Denotamos o produto cartesiano como $A \times B$, i.e., $A \times B$ denota o conjunto de todos os pares tal que o primeiro elemento é A e o segundo elemento é B. Utilizamos o símbolo $\in$ para denotar a relação de membro de conjuntos, i.e., escreve-se $a \in A$ para indicar que o elemento a está no conjunto A.
Uma função parcial f de A para B $(f : A \to B)$ é um subconjunto de $A \times B$, tal que se $(x, y) \in f$ e $(x, z) \in f$ então $y = z$. Em outras palavras, uma função parcial de A para B associa a cada elemento de A no máximo um elemento de B.
Se $(x, y) \in f$, escrevemos $f(x) = y$ e dizemos que y é a imagem de x. Os elementos de A, que tem uma imagem em B estão no domínio de f.

Funções Parciais

- Considerando um ponto de vista abstrato, pode-se dizer que cada programa define uma função parcial
- Na prática, estamos interessados não somente na função que um programa computa!
- Também gostaríamos de saber como a função é computada, quanto eficiente a computação é, quando espaço de memória vamos precisar, etc.
- Entretanto, nessa disciplina vamos nos concentrar em entender quando um problema tem uma solução computável ou não e como o mecanismo de computação é expressado

Modelos de Computação

- Algumas funções matemáticas são computáveis e algumas não são!
- Existem problemas para os quais nenhum programa de computador pode fornecer uma solução mesmo assumindo que o tempo e a memória computacional são infinitas.
- A *Teoria da complexidade* estuda os aspectos práticos da computabilidade: para uma função computável, ela responde as questões:
  - Quanto tempo e memória precisaremos para a computação
- Durante esse semestre vamos nos concentrar em computabilidade

Função Computável: Todas as funções sobre os números naturais que podem ser efetivamente computadas, tal que tempo e memória são ilimitados, são chamadas *funções recursivas parciais* ou *funções computáveis*

Modelos de Computação

- Precisamos de modelos de computação!
- Abstrair sobre detalhes materiais do processador utilizado, da linguagem de programação utilizada.
  - Máquina de Turing
  - Cálculo Lambda (Alonzo Church)
  - Teoria de funções recursivas (Kurt Gödel e Stephen Kleene)
- Os três modelos são equivalentes! Expressam a mesma classe de funções!
- These de Church diz que os três modelos expressam as assim chamadas funções computáveis

Modelos de Computação

- Dizemos que uma linguagem de programação é Turing Completa se qualquer função computável pode ser escrita nessa linguagem!
- Todas as linguagens de programação de propósito geral atuais são completas nesse sentido.
- A completude de uma linguagem é geralmente provada se a linguagem codifica um modelo universal de computação!

Funções Não-computáveis

- Desde 1930 sabe-se que certos problemas básicos não podem ser resolvidos através da computação.

- Um exemplo típico é o Problema da Parada, o qual foi provado não ser computável por Church e Turing.

- Estude alguns problemas não computáveis.

Intuitivamente, para resolver o problema da parada, precisamos de um algoritmo que verifique quando um dado programa irá parar ou não, dada uma certa entrada.
Formalmente, escreva um algoritmo H tal que:

- dada a descrição de um algoritmo A (o qual requer alguma entrada) e
- uma entrada I.

H retornará 1 se A parar com a entrada I e 0 se A não parar em I.

- O algoritmo H pode ser visto como uma função $H(A, I) = 1$ se o programa A para quando a entrada I é fornecida, e $H(A, I) = 0$ caso contrário.
- Church e Turing provaram que não existe um algoritmo H tal que, para qualquer par $A(I)$, H produz a saída requerida.

Prova: se existisse um algoritmo H, poderíamos usar ele para definir o seguinte programa C:
C recebe como entrada um algoritmo A e computada $H(A, A)$. Se o resultado é 0, então ele responde 1 e para; caso contrário ele entra em loop infinito.
Considere que $A(I) \uparrow$ representa o fato que A não para na entrada I. Utilizando C, para qualquer programa A, as seguintes propriedades valem:

- Se $H(A, A) = 1$, então $C(A) \uparrow$ e $A(A)$ para.
- Se $H(A, A) = 0$, então $C(A)$ para e $A(A) \uparrow$

Em outras palavras, $C(A)$ para se, e somente se, $A(A)$ não para. Como A é arbitrário, ele pode ser o próprio C e assim obtem-se a seguinte contradição:

$C(C)$ **para se, e somente se,** $C(C)$ **não para. Assim,** H **não existe.**