
CORREÇÃO DE PROGRAMAS (Cormen, §2.1)
---

1. INTRODUÇÃO: Programas de computador são objetos matematicamente definíveis, e portanto matematicamente tratáveis. Por exemplo, a sintaxe de uma linguagem de programação pode ser especificada por meio de uma gramática formal, que estabelece quais textos são programas sintaticamente válidos da linguagem em questão. A semântica de uma linguagem de programação também pode ser definida formalmente, e existem inclusive lógicas formais para se provar fatos a respeito de programas, como por exemplo o fato de que um programa realmente computa o que nós tínhamos em mente quando o escrevemos.

   A LÓGICA DE HOARE talvez seja a lógica formal mais famosa para se raciocinar formalmente sobre programas. Mais recentemente, a LÓGICA DE SEPARAÇÃO foi desenvolvida para tratar também programas que manipulam a memória de maneira mais explícita, por meio de ponteiros etc. Nós não abordaremos o assunto da correção de programas nesse nível de detalhe; para mais informações, consulte as referências ao fim desta nota de aula.

   Nesta parte do conteúdo, serão apresentados conceitos e técnicas que nos auxiliam a argumentar com precisão sobre a execução de algoritmos. Assim como na matemática em geral, o nível de detalhe dos argumentos pode variar. Via de regra, nós devemos buscar um nível de detalhe que seja preciso o suficiente para eliminar dúvidas sobre o funcionamento do algoritmo em análise, e que seja sucinto o suficiente para não ofuscar as ideias mais importantes.

2. ESTRUTURAS DE SELEÇÃO: a argumentação da correção de código composto por estruturas de seleção se dá, como esperado, por uma análise COMPLETA dos casos possíveis.

   Assim, por exemplo, considere este algoritmo:

   ```
   ========================================
   Algoritmo: máx
   Entrada: números reais "a" e "b".
   Saída: um número real.
   ----------------------------------------
   1. SE a < b
   2. | RETORNE b.
   3. SENÃO  // linha omitível
   4. | RETORNE a.
   ========================================
   ```

   Argumentar que ele faz o esperado é imediato:

   TEOREMA: o algoritmo "máx" retorna o máximo dentre os argumentos recebidos.
   ==========
   PROVA: Nós mostraremos que o enunciado é verdadeiro em cada um dos casos possíveis, que são os seguintes:

   a) b > a: nesse caso, a condição da linha 1 será verdadeira e o algoritmo retornará o valor de "b", corretamente (já que, nesse caso, o valor de "b" é o máximo dentre os valores de "a" e "b").

   b) a >= b: nesse caso, a condição da linha 1 será falsa e o algoritmo retornará o valor de "a", corretamente.
   ==========

3. EXERCÍCIO: Escreva um algoritmo que recebe como argumento um número real e retorna o valor absoluto (módulo) do número em questão. Em seguida, argumente que o algoritmo está correto.

   OBSERVAÇÃO: o seu algoritmo não deve pressupor um comando que já faz o

cálculo do módulo de um número; ao invés disso, deve fazer o cálculo usando testes e operadores aritméticos mais básicos.

4. ESTRUTURAS DE REPETIÇÃO:

* INVARIANTE: afirmação verdadeira no início e no fim de CADA iteração de um laço, isto é, logo antes de cada avaliação da condição do laço.

```
=======================================================
Algoritmo: busca_linear
Entrada: um vetor A[1..n] e um valor "v"
Saída: um número natural.
-------------------------------------------------------
1. i := 1
   // INVARIANTES: * 1 <= i <= n+1
   //              * Se i > 1, então A[i-1] != v
2. ENQUANTO i <= n
3. | SE A[i] = v
4. | | RETORNE i.
5. | ++i  // o mesmo que "i := i + 1"
6. RETORNE i.
=======================================================
```

5. EXERCÍCIOS:

a) Se incluirmos as linhas

```
4.1. | j := i
4.2. | i := -3
4.3. | i := j
```

entre as linhas 4 e 5 do algoritmo acima, a afirmação "1 <= i <= n+1" deixa de ser um invariante do laço?

b) O invariante "Se i > 1, então A[i-1] != v" é suficiente para provarmos que o algoritmo acima é correto? Se não, obtenha um invariante forte o suficiente (ou mais de um, se for o caso).

c) Tente argumentar, com base nos invariantes que você conseguir encontrar, que o algoritmo acima é correto (ou aponte um erro no programa...).

d) Como você acha que nós podemos PROVAR, de forma matematicamente precisa, que uma afirmação como "1 <= i <= n+1" ou "Se i > 1, então A[i-1] != v" é um invariante de um laço? Tente encontrar uma estratégia genérica, que, se realizada com sucesso, serviria, em princípio, para qualquer invariante.

6. REFERÊNCIAS.

a) Sobre invariantes de laço:

* Cormen, §2.1
* http://en.wikipedia.org/wiki/Loop_invariant

b) Um curso sobre lógica de Hoare, por MIKE GORDON:

* http://www.cl.cam.ac.uk/~mjcg/HoareLogic/

c) Um livro recente sobre lógica de separação:

* http://www.amazon.com/dp/110704801X