Fundamentos de Teoria da Computação Aula 09

Priscila Marques Kai

AGENDA DA AULA

- Demonstração de correção
  - Verificação e Validação de um Programa
  - Asserções
  - Axioma de Atribuição
  - Regra Condicional

Verificação e Validação de um Programa

Verificação do programa: tenta garantir que o programa de computador está correto.

- Um programa está **correto** se ele se comporta de acordo com suas especificações.
- Não significa que resolva o problema.

Validação do programa: garante que o programa atende às necessidades originais do cliente.

Verificação e Validação de um Programa

A verificação de um programa pode ser abordada por

- meio de testes;
- *demonstração de correção*.

Testes de programa: tentam mostrar que valores particulares de dados de entrada geram respostas aceitáveis.

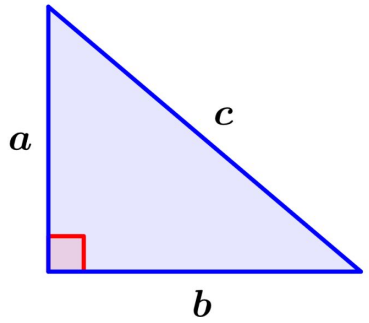
os testes podem provar a existência de erros, nunca sua ausência

Demonstração de correção: uso de técnicas de um sistema de lógica formal.

DEMONSTRAÇÃO DE CORREÇÃO

Distinção entre demonstração de correção e testes de programa:

> Exemplo: programa para calcular o comprimento *c* da hipotenusa de um triângulo retângulo, dados os valores positivos *a* e *b* para o comprimento dos lados.



$$c^2 = a^2 + b^2$$

Demonstração da correção do programa: estabelece que se *a* e *b* satisfizem as propriedades $a > 0$ e $b > 0$, então, após a execução do programa, o predicado $a^2 + b^2 = c^2$ seria satisfeito.

Testes de programa: escolher diversos valores particulares para *a* e *b*, calcular o resultado *c* e verificar a igualdade de $a^2 + b^2$ com c^2 em cada caso.

ASSERÇÕES

Para descrever a demonstração de correção mais formalmente, vamos denotar por:

X: coleção arbitrária de valores de entrada para algum programa, ou segmento de programa, P.

$Y = P(X)$: valores de Y dependem de X através das ações do programa P.

Q(X) : condições que os valores de entrada são supostos de satisfazer. (Predicado!)

Ex: programa para calcular a raiz quadrada de um número positivo. Então X consiste em um valor de entrada, x, e $Q(x)$ pode ser "$x > 0$".

R(X,Y) ou R[X,P(X)]: descreve as condições que os valores de saída devem satisfazer. (Predicado!)

Ex: se y é o único valor de saída, então y será a raiz quadrada de x, de modo que $R(x, y)$ será "$y^2 = x$".

ASSERÇÕES

Exemplo:

x ∈ X: valor numérico

Q(x): x > 0

y = P(x): valor numérico de saída

R(x,y): $y^2 = x$

$$(\forall X)\,(Q(X) \rightarrow R(x,y))$$

$$(\forall X)\,(x > 0 \rightarrow [P(X)]^2 = x)$$

ASSERÇÕES

O programa P está correto se o condicional abaixo for válido

$$(\forall X)\,(Q(X) \rightarrow R(x,y))$$

ou

$$(\forall X)\,(Q(X) \rightarrow R[X,P(X)])$$

sempre que *Q* for verdadeiro com os valores de entrada, *R* é verdadeiro com os valores de entrada e saída.

$$(\forall X)\,(x > 0 \rightarrow [P(X)]^2 = x)$$

A notação tradicional para a correção de programa é a **tripla de Hoare**.

$$\{Q\}P\{R\}$$

***Q*:** precondição

***P*:** programa

***R*:** pós-condição.

Na notação de Hoare, o quantificador universal não aparece explicitamente; fica subentendido.

ASSERÇÕES

Um programa, ou segmento de programa, é dividido em declarações individuais s_i, com predicados (**asserções**) inseridos entre declarações e afirmam o que é suposto de ser verdadeiro sobre as variáveis do programa naquele ponto.

$$
\begin{array}{c}
\{Q\} \\
s_0 \\
\{R_1\} \\
s_1 \\
\{R_2\} \\
\vdots \\
s_{n-1} \\
\{R\}
\end{array}
\quad \Longrightarrow \quad
\begin{array}{c}
\{Q\}\ s_0\ \{R_1\} \\
\{R_1\}\ s_1\ \{R_2\} \\
\{R_2\}\ s_2\ \{R_3\} \\
\vdots \\
\{R_{n-1}\}\ s_{n-1}\ \{R\}
\end{array}
$$

P é correto se cada condicional é válido

AXIOMA DE ATRIBUIÇÃO

Seja uma proposição declarada na forma $x = e$, onde a variável x assume o valor da expressão e. A correção dessa proposição é demonstrada pela tripla de Hoare na forma:

$$\{R_i\}\ x = e\ \{R_{i+1}\}$$

Para que essa tripla seja válida, as asserções R_i e R_{i+1} têm que estar relacionadas de forma especial (em sequência).

Dado: $\{R_i\}s_i\{R_{i+1}\}$
s_i tem a forma x = e (Atribuição)
$\{R_i\}$é$\{R_{i+1}\}$

AXIOMA DE ATRIBUIÇÃO

EXEMPLO

Considere as seguintes declarações de atribuição, junto com a precondição e a pós-condição dadas:

$\{x - 1 > 0\}$
$x = x - 1$
$\{x > 0\}$

Dado: $\{R_i\}s_i\{R_{i+1}\}$
s_i tem a forma x = e (Atribuição)
$\{R_i\}$é$\{R_{i+1}\}$

AXIOMA DE ATRIBUIÇÃO

EXEMPLO

Considere as seguintes declarações de atribuição, junto com a precondição e a pós-condição dadas:

```
{x - 1 > 0}
   x = x - 1
{x > 0}
```

Pré-condição: {x - 1 > 0}
Atribuição: x = x - 1
Pós-condição: {x > 0}

Asserções

Dado: $\{R_i\}s_i\{R_{i+1}\}$
s_i tem a forma x = e (Atribuição)
$\{R_i\}$é$\{R_{i+1}\}$

AXIOMA DE ATRIBUIÇÃO

EXEMPLO

Considere as seguintes declarações de atribuição, junto com a precondição e a pós-condição dadas:

$\{x - 1 > 0\}$
$x = x - 1$
$\{x > 0\}$

Pré-condição: {x - 1 > 0}
Atribuição: x = x - 1
Pós-condição: {x > 0}

Asserções

Na **pós-condição,** temos x > 0
a **atribuição** gera x - 1 > 0
Logo, temos x - 1 > 0 → **x > 1**
que é a **pré-condição**

Dado: $\{R_i\}s_i\{R_{i+1}\}$
s_i tem a forma x = e (Atribuição)
$\{R_i\}$é$\{R_{i+1}\}$

AXIOMA DE ATRIBUIÇÃO

EXEMPLO

Considere as seguintes declarações de atribuição, junto com a precondição e a pós-condição dadas:

$\{x - 1 > 0\}$
$x = x - 1$
$\{x > 0\}$

Pré-condição: $\{x - 1 > 0\}$
Atribuição: $x = x - 1$
Pós-condição: $\{x > 0\}$

Para todo x, se $x - 1 > 0$ antes da execução da declaração (note que isso significa que $x > 1$), então, após a redução do valor de x por 1, teremos $x > 0$. Portanto,

$$\{x - 1 > 0\}\ x = x - 1\ \{x > 0\}$$

O ponto da lógica de predicados é permitir a determinação da validade de um modo mais mecânico, por meio da aplicação de regras de inferência.

AXIOMA DE ATRIBUIÇÃO

A regra de inferência apropriada para declarações de atribuição é o **axioma de atribuição**:

TABELA 1.18

De	Pode-se Deduzir	Nome da Regra	Restrições sobre o Uso
	$\{R_i\}s_i\{R_{i+1}\}$	atribuição	1. s_i tem a forma $x = e$. 2. R_i é R_{i+1} com e substituído em todos os lugares por x.

Para o caso do Exemplo 41,

$$\{x-1>0\}$$
$$x=x-1$$
$$\{x>0\}$$

a tripla

$$\{x-1>0\}\,x=x-1\,\{x>0\}$$

é válida pelo axioma de atribuição. A pós-condição é

$$x>0$$

Substituindo x por $x-1$ na pós-condição, obtemos

$$x-1>0 \text{ ou } x>1$$

que é a precondição. Aqui, não tivemos que pensar; apenas verificamos que o axioma de atribuição foi seguido.

AXIOMA DE ATRIBUIÇÃO

EXEMPLO 2

De acordo com o axioma de atribuição, qual deveria ser a pré-condição no segmento de programa a seguir?

```
{pré-condição}
    x = x - 2
{x = y}
```

AXIOMA DE ATRIBUIÇÃO

EXEMPLO 2

De acordo com o axioma de atribuição, qual deveria ser a pré-condição no segmento de programa a seguir?

```
{pré-condição}
   x = x - 2
{x = y}
```

Pré-condição: ?
Atribuição: x = x - 2
Pós-condição: {x = y}

Solução:

```
                Pós-condição: {x = y}
Substituindo na atribuição: y = x - 2
                Pré-condição: y = x - 2 ou x - y - 2 ou x = y + 2
```

AXIOMA DE ATRIBUIÇÃO

EXEMPLO 3

Verifique a correção do segmento de programa a seguir, que troca os valores de x e y

```
temp = x

x = y

y = temp
```

AXIOMA DE ATRIBUIÇÃO

EXEMPLO 3

Verifique a correção do segmento de programa a seguir, que troca os valores de x e y

(a,b)

```
temp = x

x = y

y = temp
```

(b,a)

AXIOMA DE ATRIBUIÇÃO

EXEMPLO 3

Verifique a correção do segmento de programa a seguir, que troca os valores de x e y

(a,b)

```
temp = x

x = y

y = temp
```

{x=b,y=a}

(b,a)

AXIOMA DE ATRIBUIÇÃO

EXEMPLO 3

Verifique a correção do segmento de programa a seguir, que troca os valores de x e y

(a,b)

```
temp = x

x = y
{x=b,temp=a}
y = temp
{x=b,y=a}
```

(b,a)

AXIOMA DE ATRIBUIÇÃO

EXEMPLO 3

Verifique a correção do segmento de programa a seguir, que troca os valores de x e y

(a,b)

```
temp = x
{y=b,temp=a}
x = y
{x=b,temp=a}
y = temp
{x=b,y=a}
```

(b,a)

AXIOMA DE ATRIBUIÇÃO

EXEMPLO 3

Verifique a correção do segmento de programa a seguir, que troca os valores de x e y

(a,b)

```
        {x=a,y=b}
        temp = x
        {y=b,temp=a}
        x = y
        {x=b,temp=a}
        y = temp
        {x=b,y=a}
```

(b,a)

EXEMPLO 43

Verifique a correção do segmento de programa a seguir, que troca os valores de x e y:

$\text{temp} = x$

$x = y$

$y = \text{temp}$

No início desse segmento de programa, x e y têm certos valores. Assim, podemos expressar essa precondição de fato como $x = a$ e $y = b$. A pós-condição desejada é, então, $x = b$ e $y = a$. Usando o axioma de atribuição, podemos começar nossa análise pela pós-condição para encontrar as asserções anteriores (leia o segmento a seguir de baixo para cima).

$\{y = b, x = a\}$

$\text{temp} = x$

$\{y = b, \text{temp} = a\}$

$x = y$

$\{x = b, \text{temp} = a\}$

$y = \text{temp}$

$\{x = b, y = a\}$

A primeira asserção está de acordo com a precondição; o axioma de atribuição, aplicado repetidamente, nos garante que o segmento de programa está correto.

AXIOMA DE ATRIBUIÇÃO

EXEMPLO 4

Verifique a correção do segmento de programa a seguir, com a precondição e a pós-condição dadas:

```
{x = 3}

y = 4

z = x + y

{z = 7}
```

AXIOMA DE ATRIBUIÇÃO

EXEMPLO 4

Verifique a correção do segmento de programa a seguir, com a precondição e a pós-condição dadas:

```
Q:{x = 3}

        s1: y = 4

        s2: z = x + y

R: {z = 7}
```

AXIOMA DE ATRIBUIÇÃO

EXEMPLO 4

Verifique a correção do segmento de programa a seguir, com a precondição e a pós-condição dadas:

```
s1: y = 4

s2: z = x + y
{z = 7}
```

AXIOMA DE ATRIBUIÇÃO

EXEMPLO 4

Verifique a correção do segmento de programa a seguir, com a precondição e a pós-condição dadas:

```
s1: y = 4
{x + y = 7}
s2: z = x + y
{z = 7}
```

AXIOMA DE ATRIBUIÇÃO

EXEMPLO 4

Verifique a correção do segmento de programa a seguir, com a precondição e a pós-condição dadas:

```
{x = 3}     Pré-Condição!
{x + 4 = 7}
s1: y = 4
{x + y = 7}
s2: z = x + y
{z = 7}
```

AXIOMA DE ATRIBUIÇÃO

EXEMPLO 5

Verifique a correção do segmento de programa a seguir, com a precondição e a pós-condição dadas:

```
{x = 3}     Pré-Condição!
{x + 4 = 7}
s1: y = 4
{x + y = 7}
s2: z = x + y
{z = 7}
```

REGRA CONDICIONAL

Uma **proposição** ou **declaração condicional** é uma declaração, em um programa, da forma

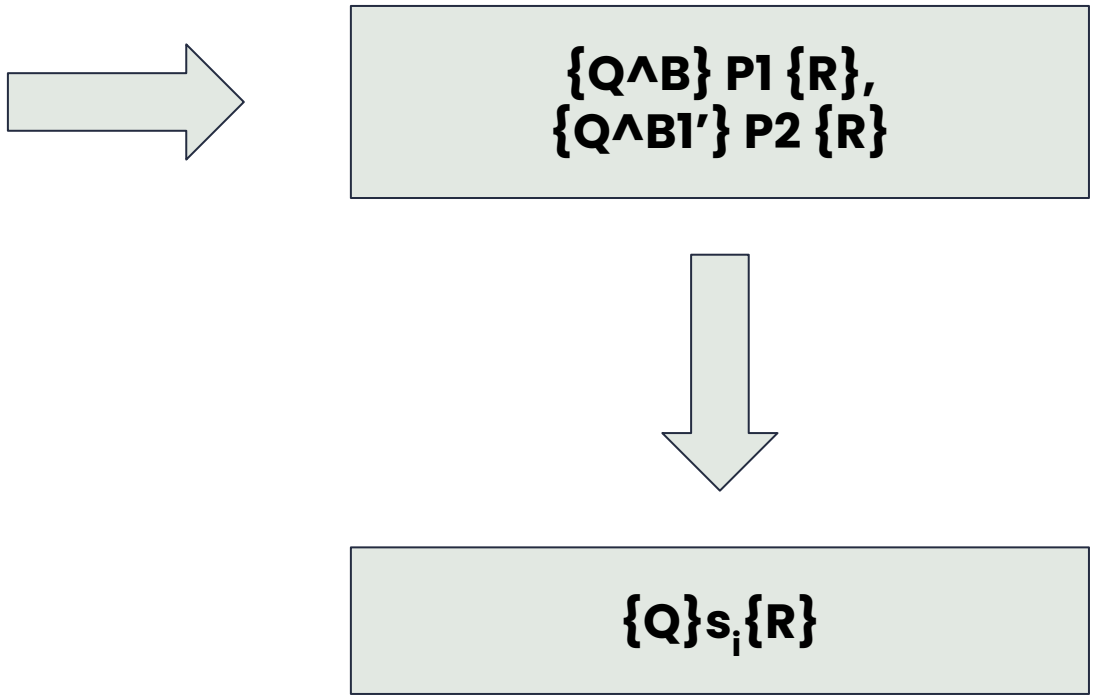
SE (condição B) **ENTÃO**

P1

SENÃO

P2

FIMSE



REGRA CONDICIONAL

EXEMPLO

{n = 5}

SE n>= 10 ENTÃO

y = 100

SENÃO

y = n + 1

{y = 6}

REGRA CONDICIONAL

EXEMPLO

```
{n = 5}
SE n>= 10 ENTÃO
        y = 100
SENÃO
        y = n + 1
{y = 6}
```

REGRA CONDICIONAL

EXEMPLO

```
{n = 5}
SE n>= 10 ENTÃO
        y = 100
SENÃO
        y = n + 1
{y = 6}
```


{Q^B} P1 {R},
{n = 5 e n >= 10} y = 100 {y = 6}

REGRA CONDICIONAL

EXEMPLO

```
{n = 5}
SE n>= 10 ENTÃO
        y = 100
SENÃO
        y = n + 1
{y = 6}
```


{Q^B} P1 {R},
{n = 5 e n >= 10} y = 100 {y = 6}

{Q^B'} P2 {R},
{n = 5 e n < 10} y = 5 + 1 {y = 6}

REGRA CONDICIONAL

EXEMPLO

{n = 5}
SE n>= 10 ENTÃO
y = 100
SENÃO
y = n + 1
{y = 6}

Pela primeira condição,

{Q^B} P1 {R},
{n = 5 e n >= 10} y = 100 {y = 6}

Temos:
n = 5 e n >= 10
Falso

REGRA CONDICIONAL

EXEMPLO

{n = 5}

SE n>= 10 ENTÃO

y = 100

SENÃO

y = n + 1

{y = 6}

{Q^B'} P2 {R},
{n = 5 e n < 10} y = 5 + 1 {y = 6}

Aplicando o axioma de atribuição:

y = n + 1

REGRA CONDICIONAL

EXEMPLO

{n = 5}
SE n>= 10 ENTÃO
y = 100
SENÃO
y = n + 1
{y = 6}

{Q^B'} P2 {R},
{n = 5 e n < 10} y = 5 + 1 {y = 6}

Aplicando o axioma de atribuição:

{n = 5}
{n + 1 = 6}
y = n + 1
{y = 6}

REGRA CONDICIONAL

EXEMPLO 2

{x = 4}

SE x < 5 ENTÃO

y = x - 1

SENÃO

y = 7

{y = 3}

REGRA CONDICIONAL

EXEMPLO 2

{x = 4}

SE x < 5 ENTÃO

y = x - 1

SENÃO

y = 7

{y = 3}

{Q^B} P1 {R},
{x = 4 e x < 5} y = x - 1 {y = 3}

Aplicando o axioma de atribuição:

{x = 4}

y = x - 1

{y = 3}

REGRA CONDICIONAL

EXEMPLO 2

{x = 4}
SE x < 5 ENTÃO
y = x - 1
SENÃO
y = 7
{y = 3}

VERDADEIRO

{Q^B} P1 {R},
{x = 4 e x < 5} y = x - 1 {y = 3}

FALSO

{Q^B'} P2 {R},
{x = 4 e x >= 5} y = 7 {y = 3}

REGRA CONDICIONAL

EXEMPLO 2

{x = 4}

SE x < 5 ENTÃO

y = x - 1

SENÃO

y = 7

{y = 3}

{Q^B} P1 {R},
{x = 4 e x < 5} y = x - 1 {y = 3}

Aplicando o axioma de atribuição:

{x = 4}

{x - 1 = 3} ⇒ {x = 3 + 1}

y = x - 1

{y = 3}

Exercícios

1) De acordo com o axioma de atribuição, qual é a precondição para o segmento de programa a seguir?
{precondição}
x = 2 * x
{x > y}

2) De acordo com o axioma de atribuição, qual é a precondição para o segmento de programa a seguir?
{precondição}
x = 3 * x – 1
{x = 2 * y – 1}

3) Verifique a correção do segmento de programa a seguir com a precondição e a pós-condição indicadas.
{x = 1}
y = x + 3
y = 2 * y
{y = 8}

4) Verifique a correção do segmento de programa a seguir com a precondição e a pós-condição indicadas.
{x > 0}
y = x + 2
z = y + 1
{z > 3}

5) Verifique a correção do segmento de programa a seguir com a precondição e a pós-condição indicadas.
{x = 0}
z = 2 * x + 1
y = z – 1
{y = 0}

6) Verifique a correção do segmento de programa a seguir com a precondição e a pós-condição indicadas.
{x < 8}
z = x – 1
y = z – 5
{y < 2}

Exercícios

6) Verifique a correção do segmento de programa a seguir com a precondição e a pós-condição indicadas.

```
{y = 0}
        se y < 5 então
                y = y + 1
        senão
                y = 5
        fimse
{y = 1}
```

7) Verifique a correção do segmento de programa a seguir com a precondição e a pós-condição indicadas.

```
{x = 7}
        se x <= 0 então
                y = x
        senão
                y = 2 * x
        fimse
{y = 14}
```

8) Verifique a correção do segmento de programa a seguir com a precondição e a pós-condição indicadas.

```
{x ≠ 0}
        se x > 0 então
                 y = 2 * x
        senão
                y = (–2) * x
        fimse
{y > 0}
```

9) Verifique a correção do segmento de programa a seguir com a precondição e a pós-condição indicadas.

```
{x = 0}
        z = 2 * x + 1
        y = z – 1
{y = 0}
```

Exercícios

6) Verifique a correção do segmento de programa a seguir com as asserções dadas.

```
{z = 3}
        x = z + 1
        y = x + 2
{y = 6}
        se y > 0 então
                z = y + 1
        senão
                z = 2 * y
        fimse
{z = 7}
```

REGRA DO LAÇO

Suponha que s_i é uma proposição com laço da forma

```
enquanto condição B faça
    P
fimenquanto
```

em que *B* é uma condição que pode ser falsa ou verdadeira e P é um segmento de programa.

Caso B seja verdadeira, o segmento de programa P é executado e B é reavaliada.
Se a condição B se torna falsa em algum momento, o laço termina.

REGRA DO LAÇO

A precondição Q é válida antes de se entrar no laço;

- uma das condições é que Q continue válida depois que o laço termina (deve haver uma precondição Q que seja verdadeira quando o laço terminar).
- B' — a condição para o laço parar — também tem que ser verdadeira.

$$\{Q\}s_i\{Q \wedge B'\}$$

REGRA DO LAÇO

EXEMPLO

Considere a função em pseudocódigo a seguir, que realiza o produto entre dois inteiros não negativos x e y. P

```
Variáveis locais:
inteiros i, j
    i= 0
    j= 0
    enquanto i≠x faça
        j= j + y
        i= i + 1
    fimenquanto
    //j agora tem o valor x * y
    escreva j
fim da função Produto
```

REGRA DO LAÇO

EXEMPLO

Considere a função em pseudocódigo a seguir, que realiza o produto entre dois inteiros não negativos x e y. P

```
Variáveis locais:
inteiros i, j
    i= 0
    j= 0
    enquanto i≠x faça
        j= j + y
        i= i + 1
    fimenquanto
    //j agora tem o valor x * y
    escreva j
fim da função Produto
```

LAÇO

Condição **B** = i≠x

B' = i = x

Dado que **i = x** quando o laço termina, a asserção **j = i * y** também teria que ser verdadeira.

no término do laço, temos

$$Q \wedge B' = Q \wedge (i = x)$$

e

$$j = x * y$$

Para ter ambas as condições

$$Q \wedge (i = x) \; e \; j = x * y$$

Q tem que ser

$$j = i * y$$

o que ocorre de fato, já que, logo antes do laço, i = j = 0.

REGRA DO LAÇO

Parece que temos, para este exemplo, um candidato Q para o condicional, mas ainda não temos a regra de inferência que nos permite dizer quando é um condicional verdadeiro.

Q (Predicado): deve ser verdadeiro antes do início do laço e precisa permanecer verdadeira após cada iteração, incluindo a final.

- Se Q for válida antes e depois de cada iteração do laço, essa relação não é afetada pela iteração, mesmo que os valores das variáveis mudem. Essa relação constante é chamada de **invariante do laço**.

REGRA DO LAÇO

Regra de inferência para laços: permite que a veracidade seja inferida de um condicional dizendo que Q é um invariante do laço.

> **Q invariante do laço:** Q deve permanecer verdadeira após cada iteração, o que pode ser expresso pela tripla de Hoare

$$\{Q \wedge B\}\ P\ \{Q\}$$

REGRA DO LAÇO

TABELA 2.4

De	Pode-se Deduzir	Nome da Regra	Restrições sobre o Uso
$\{Q \wedge B\}\ P\ \{Q\}$	$\{Q\}$ *si* $\{Q \wedge B'\}$	laço	*si* tem a forma **enquanto** condição B **faça** P **fim do enquanto**

REGRA DO LAÇO

Para usar essa regra de inferência, precisamos encontrar um invariante do laço Q que seja útil — um que afirme o que queremos e o que esperamos que aconteça — e depois provar o condicional

$$\{Q \wedge B\}\ P\ \{Q\}$$

E é aqui que a indução entra em cena. Vamos denotar por Q(n) a proposição de que um invariante do laço proposto Q seja verdadeiro após n iterações do laço. Como não sabemos quantas iterações serão executadas no laço (ou seja, por quanto tempo a condição B irá permanecer verdadeira), queremos mostrar que Q(n) é verdadeira para todo $n \geq 0$. (O valor $n = 0$ corresponde à asserção antes do início do laço, antes de qualquer iteração.)

Considere, novamente, a função em pseudocódigo do Exemplo 25. Naquele exemplo, conjecturamos que Q é a relação

$$j = i * y$$

Para usar a regra de inferência do laço, precisamos provar que Q é um invariante do laço.

As quantidades x e y permanecem constantes durante todo o processo, mas os valores de i e j variam dentro do laço. Vamos denotar por i_n e j_n, respectivamente, os valores de i e j após n iterações do laço. Então, Q_n é a proposição $j_n = i_n * y$.

Vamos provar por indução que $Q(n)$ é válida para todo $n \geq 0$. $Q(0)$ é a proposição

$$j_0 = i_0 * y$$

que, como observamos no Exemplo 25, é verdadeira, pois antes de qualquer iteração, ao chegar pela primeira vez no laço, é atribuído tanto a i quanto a j o valor 0. (Formalmente, o axioma de atribuição poderia ser usado para provar que essas condições sobre i e j são válidas nesse instante.)

Suponha $Q(k)$: $j_k = i_k * y$

Mostre $Q(k+1)$: $j_{k+1} = i_{k+1} * y$

Entre o instante em que j e i têm os valores j_k e i_k e o instante em que j e i têm os valores j_{kk+1} e i_{kk+1}, ocorre uma iteração do laço. Nessa iteração, j muda adicionando-se y a seu valor anterior e i muda adicionando-se 1. Ou seja,

$$j_{k+1} = j_k + y \tag{3}$$
$$i_{k+1} = i_k + 1 \tag{4}$$

Então

$$
\begin{aligned}
j_{k+1} &= j_k + y && \text{(de (3))} \\
&= i_k * y + y && \text{(pela hipótese de indução)} \\
&= (i_k + 1)y \\
&= i_{k+1} * y && \text{(de (4))}
\end{aligned}
$$

Acabamos de provar que Q é um invariante do laço.

A regra de inferência do laço nos permite inferir que, ao sair do laço, a condição $Q \wedge B'$ é válida, o que, nesse caso, se torna

$$j = i * y \wedge i = x$$

Portanto, nesse instante, a proposição

$$j = x * y$$

é verdadeira, o que é exatamente o que a função é suposta de calcular. ●