Prova de Correção de Programas Imperativos

Diego Pinto da Jornada, Matthias Oliveira de Nunes

[1]Faculdade de Informática – Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)

{diego.jornada, matthias.nunes}@acad.pucrs.br

Resumo. *Este artigo apresenta duas provas de correção de programas imperativos, a partir de dois algoritmos descritos na forma de pseudo código.*

Abstract. *This paper presents two proofs of correctness of imperative programs, from two algorithms described in pseudocode form.*

1. Introdução

No escopo de disciplina de Métodos Formais o segundo trabalho pode ser desenvolvido da seguinte maneira: dado dois algoritmos, na forma de pseudo código, deve ser demonstrada a prova de correção utilizando os conceitos da lógica de Hoare.

A enunciado da tarefa apresenta dois algoritmos no seguinte formato:

Algorithm 1 Algoritmo 1

function ALGORITMO 01(x)
 $i \leftarrow 1$
 $j \leftarrow 4$
 while $i \neq x$ **do**
  $j \leftarrow j + 2 * i + 3$
  $i \leftarrow i + 1$
 end while
end function

Algorithm 2 Algoritmo 2

function ALGORITMO 02(x, y, n)
 $i \leftarrow 0$
 $j \leftarrow x$
 while $i \neq n$ **do**
  $j \leftarrow j * i$
  $i \leftarrow i + 1$
 end while
end function

Após a definiçãos dos algoritmos é necessário explicar com os devidos detalhes a função que cada um computa, bem como a tupla de Hoare correspondente. Também será definido, para cada um dos algoritmos, a invariante do laço e sua prova por indução correspondente. Na seção 3 será abordada a prova de correção parcial para cada tupla de Hoare que for apresentada, mas antes disso vamos falar um pouco sobre a lógica de hoare.

2. Lógica de Hoare

Lógica de Hoare é um sistema formal com um conjunto de regras lógicas para um raciocínio rigoroso sobre a corretude de um programa de computador.

2.1. Tripla de Hoare

Uma das mais importantes propriedades de um programa é se ele faz ou não faz a sua função. A intencional função de um programa, ou parte de um programa, pode ser especificada fazendo asserções gerais sobre os valores em que as variáveis relevantes vão ter *depois* da execução do programa. Essas asserções geralmente não vão atribuir valores particulares para cada variável, mas sim especificar certas propriedades gerais dos valores e das relações entre elas.

Em muitos casos, a validade dos resultados de um programa, ou parte de um programa, vai depender dos valores tomados pelas variáveis antes desse programa ser iniciado. Essas precondições iniciais de uso bem sucedido podem ser especificadas pelo mesmo tipo de asserção geral como é usado para descrever os resultados obtidos na terminação. Para estabelecer a conexão requirida entre a precondição (P), um programa (Q) e a descrição do resultado da sua execução (R), foi introduzido uma nova notação:

$$(\!|\ P\ |\!)\ Q\ (\!|\ R\ |\!)$$

Isso pode ser interpretado como "Se a asserção P for verdadeira antes do inícido do programa Q, então a asserção R será verdadeira após sua execução."Se não houver precondições impostas, escrevemos como $(\!|\ true\ |\!)\ Q\ (\!|\ R\ |\!)$[Hoare 1968].

3. Algoritmo 1

O Algortimo 1 tem o objetivo de computar a seguinte operação: dado uma variável *x* o algoritmo retorna $(x\ +\ 1)^2$ e pode ser representado com a seguinte tupla de hoare:

$$(\!|\ x > 0\ |\!)\ i := 1;\ j := 4 \text{ while } i \neq x\, \{j := j + 2 \times i + 3;\ i := i + 1\}\ \ (\!|\ j = (x+1)^2\ |\!)$$

3.0.1. Prova da Inavariante do Loop

Nesta seção demonstraremos a prova da invariante do loop. Após ser feita a análise do comportamento das variáveis duranda a execução do programa chegamos a conclusão que a invariate do loop era:

$$j\ =\ (i\ +\ 1)^2\ \wedge i\ \leq\ x$$

Porém é preciso provar que a invariante é válida para qualquer momento do código. A prova será feita utilizando indução, com isso apresentaremos a prova para um caso base, neste caso a base será repsentada pelo número zero que marca que o laço não ocorreu. Assumiremos a validade da invariante para *n* e desmontraremos que a também se mantém vardadeira para *n + 1*.

Prova: Devemos provar o seguinte: $P \triangleq \forall_k : \mathbb{N}.\ j_k = (i_k\ +\ 1)^2\ \wedge\ i_k \leq x$. Assumindo o teorema da distributividade do para todo temos o seguinte:

$$\forall_w : \mathbb{N}.\ j_w = (i_w\ + 1)^2\ \wedge \forall_z : \mathbb{N}.i_z \leq x$$

Com isso devemos demonstrar a prova para cada lado da conjunção. Agora Veja:

Caso Base P(0) É necessário provar o seguinte: $j_0 = (i_0 \ + \ 1)^2$ Agora veja que:

$$
\begin{array}{rll}
j_0 = & \textit{4} & \textit{(j := 4)} \\
= & 2^2 & \textit{(Artmimética)} \\
= & (1 \ + \ 1)^2 & \textit{(Aritmética)} \\
= & (i_0 \ + \ 1)^2 & \textit{(i := 1)}
\end{array}
$$

q.e.d

Também precisamos provar o seguinte:$i_0 \leq x$ Agora veja que:

$$
\begin{array}{rl}
x \ \geq \ 0 & \textit{PRE} \\
x \ \geq \ 1 & \textit{Aritimética} \\
x \ \geq \ i_0 & i_0 \textit{ := 1} \\
i_0 \ \leq \ x & \textit{Aritimética}
\end{array}
$$

q.e.d

Caso Indutivo P(n) Assumindo *n* como um valor arbitrário e como hipótese de indução $\forall k : \mathbb{N}.\ j_k = (i_k \ + \ 1)^2$ temos que mostrar que: $j_{k+1} = (i_{k+1} \ + \ 1)^2$ agora veja que:

$$
\begin{array}{rll}
(i_{k+1} \ + \ 1) = & ((i_w \ + \ 1) \ + \ 1)^2 & \textit{(i := i + 1)} \\
= & (i_w + 1)^2 + 2(i_w + 1) + 1 & \textit{(Artmimética)} \\
= & j_w + 2(i_w + 1) + 1 & \textit{(H.I)} \\
= & j_w + 2i_w + 3 & \textit{(Aritimética)} \\
= & j_{w+1} & \textit{(j := j + 2*i + 3)}
\end{array}
$$

q.e.d

Também precisamos provar assumindo *z* como um valor arbitrário e como hipótese de indução $\forall z : \mathbb{N}\ .\ i_z \leq x$ temos que mostrar que: $i_{z+1} \leq x$ agora veja que:

$$
\begin{array}{rl}
i_z \leq x \wedge \neg(i_z = x) & \wedge I\textit{, HI, LOOP} \\
\neg(i_z = x) \wedge (i_z = x) \vee \neg(i_z = z) \wedge (i_z < x) & \textit{Distributividade Conjunção} \\
\neg(i_z = x) \wedge (i_z < x) & \textit{Lógica} \\
i_z < x & \wedge\varepsilon \\
i_z + 1 \leq x & \textit{Aritimética} \\
i_{z+1} \leq x & \textit{Aritimética}
\end{array}
$$

q.e.d

3.1. Prova da Tripla de Hoare

Depois de termos provado a invariante do *laço*

$$INV = j = (i + 1)^2 \wedge i \leq x$$

estamos aptos para verificar se o programa correspondente ao Algoritmo 1 está correto.

1	$(\!\vert\ j = (i+2)^2 \wedge i+1 \leq x\ \vert\!)\ i := i+1\ (\!\vert\ INV\ \vert\!)$	ATR
2	$(\!\vert\ j + 2 \times i + 3 = (i+2)^2 \wedge i+1 \leq x\ \vert\!)$ $j := j + 2 \times i + 3$ $(\!\vert\ j = (i+2)^2 \wedge i+1 \leq x\ \vert\!)$	ATR
3	$(\!\vert\ j + 2 \times i + 3 = (i+2)^2 \wedge i+1 \leq x\ \vert\!)$ $j := j + 2 \times i + 3;\ i := i+1$ $(\!\vert\ INV\ \vert\!)$	1,2COMP
4	$(j = (i+1)^2 \wedge i \leq x) \wedge i \neq x$	H
5	$j = (i+1)^2 \wedge i \leq x$	4, $\wedge$EL
6	$j = (i+1)^2$	5, $\wedge$EL
7	$i \leq x$	5, $\wedge$ER
8	$j = i^2 + 2 \times i + 1$	6, ARIT
9	$j + 3 = i^2 + 2 \times i + 4$	8, ARIT
10	$j + 2 \times i + 3 = i^2 + 4i + 4$	9, ARIT
11	$j + 2 \times i + 3 = (i+2)^2$	10, ARIT
12	$i < x \vee i = x$	7, DEF$\leq$
13	$i \neq x$	4, $\wedge$ER
14	$(i < x \vee i = x) \wedge i \neq x$	12,13 $\wedge$I
15	$i < x \wedge i \neq x \vee i = x \wedge i \neq x$	14, DIST$\wedge$
16	$i < x \wedge i \neq x \vee \perp$	15, CONTRADIÇÃO
17	$i < x \wedge i \neq x$	16, LÓGICA
18	$i < x$	17, $\wedge$EL
19	$i + 1 \leq x$	18, ARIT
20	$j + 2 \times i + 3 = (i+2)^2 \wedge i+1 \leq x$	11,19 $\wedge$I
21	$(j = (i+2)^2 \wedge i \leq x) \wedge i \neq x$ $\rightarrow$ $j + 2 \times i + 3 = (i+2)^2 \wedge i+1 \leq x$	4-20, $\rightarrow$I
22	$(\!\vert\ (j = (i+1)^2 \wedge i \leq x) \wedge i \neq x\ \vert\!)$ $j := j + 2 \times i + 3;\ i := i+1$ $(\!\vert\ INV\ \vert\!)$	3,21,PreStren
23	$(\!\vert\ INV\ \vert\!)$ while $i \neq x\ \{j := j + 2 \times i + 3;\ i := i+1\}$ $(\!\vert\ INV \wedge \neg(i \neq x)\ \vert\!)$	22, PWhile

Agora vemos que:

1	$\llparenthesis\ 4 = (i+1)^2 \wedge i \leq x\ \rrparenthesis\ j := 4\ \llparenthesis\ INV\ \rrparenthesis$	ATR
2	$\llparenthesis\ 4 = (1+1)^2 \wedge 1 \leq x\ \rrparenthesis\ i := 1\ \llparenthesis\ 4 = (i+1)^2 \wedge i \leq x\ \rrparenthesis$	ATR
3	$\llparenthesis\ 4 = (1+1)^2 \wedge 1 \leq x\ \rrparenthesis\ i := 1;\ j := 4\ \llparenthesis\ INV\ \rrparenthesis$	1,2,COMP
4	$\quad x > 0$	H
5	$\quad 4 = 4$	=
6	$\quad 4 = 2^2$	5, ARIT
7	$\quad 4 = (1+1)^2$	6, ARIT
8	$\quad x \geq 1$	4, ARIT
9	$\quad 1 \leq x$	8, ARIT
10	$\quad 4 = (1+1)^2 \wedge 1 \leq x$	7,9,$\wedge$I
11	$x > 0 \rightarrow 4 = (1+1)^2 \wedge 1 \leq x$	4-10, $\rightarrow$I
12	$\llparenthesis\ x > 0\ \rrparenthesis\ i := 1;\ j := 4\ \llparenthesis\ INV\ \rrparenthesis$	3,11, PreStren

Com o *while*, que foi concluído na primeira parte, e a composição, que foi concluída na seguinda parte, podemos seguir a prova desta maneira:

	$\llparenthesis\ INV\ \rrparenthesis$	
1	while $i \neq x\ \{j := j + 2 \times i + 3;\ i := i + 1\}$	
	$\llparenthesis\ INV \wedge \neg(i \neq x)\ \rrparenthesis$	
2	$\llparenthesis\ x > 0\ \rrparenthesis\ i := 1;\ j := 4\ \llparenthesis\ INV\ \rrparenthesis$	
	$\llparenthesis\ x > 0\ \rrparenthesis$	
3	$i := 1;\ j := 4$ while $i \neq x\ \{j := j + 2 \times i + 3;\ i := i + 1\}$	2,1, COMP
	$\llparenthesis\ INV \wedge \neg(i \neq x)\ \rrparenthesis$	
4	$\quad (j = (i+1)^2 \wedge i \leq x) \wedge \neg(i \neq x)$	H
5	$\quad j = (i+1)^2 \wedge i \leq x$	4, $\wedge$EL
6	$\quad j = (i+1)^2$	5, $\wedge$EL
7	$\quad \neg(i \neq x)$	4, $\wedge$ER
8	$\quad i = x$	7, LÓGICA
9	$\quad j = (x+1)^2$	=E$\{6, 8\}$
10	$INV \wedge \neg(i \neq x) \rightarrow j = (x+1)^2$	4-9, $\rightarrow$I
	$\llparenthesis\ x > 0\ \rrparenthesis$	
11	$i := 1;\ j := 4$ while $i \neq x\ \{j := j + 2 \times i + 3;\ i := i + 1\}$	3,10,PosWeak
	$\llparenthesis\ j = (x+1)^2\ \rrparenthesis$	

4. Algoritmo 2

O algotimo 2 retorna o n-ésimo termo de uma progressão geométrica de razão *y* e como termo inicial o valor *x*. Este algoritmo pode ser expresso, utilizando a notação de Hoare, da seguinte forma:

4.0.1. Prova da Inavariante do Loop

Nesta seção demonstraremos a prova da invariante do loop. Após ser feita a análise do comportamento das variáveis duranda a execução do programa chegamos a conclusão que a invariate do loop era:

$$j \;=\; xy^{i_0} \;\wedge n \;\geq\; i$$

Porém é preciso provar que a invariante é válida para qualquer momento do código. A prova será feita utilizando indução, com isso apresentaremos a prova para um caso base, neste caso a base será repsentada pelo número zero que marca que o laço não ocorreu. Assumiremos a validade da invariante para *n* e desmontraremos que a também se mantém vardadeira para *n + 1*.

Prova: Devemos provar o seguinte: $P \triangleq \forall_k : \mathbb{N}.\ j_k = xy^{i_n} \ \wedge \ n \geq i_n$. Assumindo o teorema da distributividade do para todo temos o seguinte:

$$\forall_w : \mathbb{N}.\ j_w = xy^{i_w} \ \wedge \forall_z : \mathbb{N}. n \geq i_z$$

Com isso devemos demonstrar a prova para cada lado da conjunção. Agora Veja:

Caso Base P(0) É necessário provar o seguinte: $j_0 = xy^{i_0}$ Agora veja que:

$$\begin{aligned} j_0 = &\quad x && \textit{(j := x)} \\ = &\quad x * 1 && \textit{(Artmimética)} \\ = &\quad x * y^0 && \textit{(Aritimética)} \\ = &\quad xy^{i_0} && \textit{(i := 0)} \end{aligned}$$

q.e.d

Também precisamos provar o seguinte:$n \geq i_0$ Agora veja que:

$$\begin{aligned} i_0 &= 0 && \textit{i:=0} \\ n &\geq 0 && \textit{PRE} \\ n &\geq i_1 && \textit{Aritimética} \end{aligned}$$

q.e.d

Caso Indutivo P(n) Assumindo *w* como um valor arbitrário e como hipótese de indução $\forall w : \mathbb{N}.\ j_w = xy^i_w$ temos que mostrar que: $j_{w+1} = xy^{i_{w+1}}$ agora veja que:

$$\begin{array}{rcll} xy^{i_w+1} & = & (xy^w)y & \textit{(Aritimética)} \\ & = & j_w * y & \textit{(Artmimética)} \\ & = & j_{w+1} & \textit{(j := j*y)} \end{array}$$

q.e.d

Também precisamos provar assumindo z como um valor arbitrário e como hipótese de indução $\forall z : \mathbb{N} \,.\, i_z \leq x$ temos que mostrar que: $i_{z+1} \leq x$ agora veja que:

$$\begin{array}{rl} i_k \leq n \wedge \neg(i_k = n) & \wedge I\textit{, HI, LOOP} \\ (i_k \leq n \vee i_k = n) \wedge \neg(i = n) & \textit{Definição} \leq \\ (\neg(i_k = n) \wedge i_k = n) \vee (\neg(i_k = n) \wedge i_k < n) & \textit{Distributividade} \wedge \\ \perp \vee (\neg(i_k = n) \wedge i_k < n) & \textit{Lógica} \\ \neg(i_k = n) \wedge i_k < n & \textit{Lógica} \\ i_k < n & \wedge\varepsilon \\ i_k + 1 \leq n & \textit{Aritimética} \\ i_{k+1} \leq n & \textit{i := i + 1} \end{array}$$

q.e.d

$$(\!|\, x \;\geq\; 0 \wedge x \;=\; x_0 \wedge y = y_0 |\!)\, i \;:=\; 0; j := \; x;\; while\, i \neq \; n\{j \;:=\; j * y;\; i \;:=\; i + 1\}(\!| j = xy^n |\!)$$

Depois de termos provado a invariante do *laço*

$$INV = j = jy^i \wedge n \geq i$$

estamos aptos para verificar se o programa correspondente ao Algoritmo 2 está correto.

1	$\llparenthesis\ j = jy^{i+1} \wedge n \geq i+1\ \rrparenthesis\ i := i+1\ \llparenthesis\ INV\ \rrparenthesis$	ATR
2	$\llparenthesis\ jy = jy^{i+1} \wedge n \geq i+1\ \rrparenthesis\ j := jy\ \llparenthesis\ j = jy^{i+1} \wedge n \geq i+1\ \rrparenthesis$	ATR
3	$\llparenthesis\ jy = jy^{i+1} \wedge n \geq i+1\ \rrparenthesis\ j := jy; i := i+1\ \llparenthesis\ INV\ \rrparenthesis$	1,2,COMP
4	$j = jy^i \wedge n \geq i \wedge i \neq n$	H
5	$n \geq i \wedge i \neq n$	4,∧ER
6	$(n > i \vee n = i) \wedge i \neq n$	5,DEF≥
7	$(n > i \wedge i \neq n) \vee (n = i \wedge i \neq n)$	6,DIST∧
8	$n > i \wedge i \neq n \vee \bot$	7,CONTRADIÇÃO
9	$n > i \wedge i \neq n$	8,LÓGICA
10	$n > i$	9,∧EL
11	$n \geq i+1$	10,ARIT
12	$j = jy^i$	4,∧EL
13	$jy = jyy^i$	12,ARIT
14	$jy = jy^{i+1} \wedge n \geq i+1$	13,ARIT 11,13,∧I
15	$j = jy^i \wedge n \geq i \wedge i \neq n \rightarrow jy = jy^{i+1} \wedge n \geq i+1$	4-14,→I
16	$\llparenthesis\ j = jy^i \wedge n \geq i \wedge i \neq n\ \rrparenthesis\ j := jy; i := i+1\ \llparenthesis\ INV\ \rrparenthesis$	3,15,PreStren
17	$\llparenthesis\ INV\ \rrparenthesis\ whilei \neq n\, \{j := jy;\ i := i+1\}\ \llparenthesis\ INV \wedge \neg(i \neq n)\ \rrparenthesis$	16,PWhile

Agora vemos que:

1	$\llparenthesis\ x = xy^i \wedge n \geq i\ \rrparenthesis\ j := x\ \llparenthesis\ INV\ \rrparenthesis$	ATR
2	$\llparenthesis\ x = xy^0 \wedge n \geq 0\ \rrparenthesis\ i := 0\ \llparenthesis\ x = xy^i \wedge n \geq i\ \rrparenthesis$	ATR
3	$\llparenthesis\ x = xy^0 \wedge n \geq 0\ \rrparenthesis\ i := 0;\ j := x\ \llparenthesis\ INV\ \rrparenthesis$	1,2,COMP
4	$n \geq 0$	H
5	$x = x$	=
6	$x = x \times 1$	5,ARIT
7	$x = x \times y^0$	6,ARIT
8	$x = xy^0 \wedge n \geq 0$	4,7,∧I
9	$n \geq 0 \rightarrow x = xy^0 \wedge n \geq 0$	4-8,→I
10	$\llparenthesis\ n \geq 0\ \rrparenthesis\ i := 0;\ j := x\ \llparenthesis\ INV\ \rrparenthesis$	3,9,PreStren

Com o *while*, que foi concluído na primeira parte, e a composição, que foi concluída na seguinda parte, podemos seguir a prova desta maneira:

1	$\llparenthesis\ INV\ \rrparenthesis\ while i \neq n\, \{j := jy;\ i := i+1\}\ \llparenthesis\ INV \wedge \neg(i \neq n)\ \rrparenthesis$	
2	$\llparenthesis\ n \geq 0\ \rrparenthesis\ i := 0;\ j := x\ \llparenthesis\ INV\ \rrparenthesis$	
3	$\llparenthesis\ n \geq 0\ \rrparenthesis$ $i := 0;\ j := x;\ while i \neq n\, \{j := jy;\ i := i+1\}$ $\llparenthesis\ INV \wedge \neg(i \neq n)\ \rrparenthesis$	1,2,COMP
4	$j = jy^i \wedge n \geq i \wedge \neg(i \neq n)$	H
5	$j = jy^i$	4,$\wedge$EL
6	$\neg(i \neq n)$	4,$\wedge$ER
7	$i = n$	6,LÓGICA
8	$j = jy^n$	=E$\{5, 7\}$
9	$INV \wedge \neg(i \neq n) \rightarrow j = jy^n$	4-8,$\rightarrow$I
10	$\llparenthesis\ n \geq 0\ \rrparenthesis$ $i := 0;\ j := x;\ while i \neq n\, \{j := jy;\ i := i+1\}$ $\llparenthesis\ j = jy^n\ \rrparenthesis$	3,9,PosWeak

Referências

Hoare, C. A. R. (1968). An axiomatic basis for computer programming.

Martini, A. (2014). Métodos formais para ciência da computação.