Práctico 6: Especificación y Verificación de Programas Imperativos

Algoritmos y Estructuras de Datos I
2^{do} cuatrimestre 2021

En esta guía usamos el transformador de predicados wpque calcula la pre-condición más débil. Recordemos que:

1. la wp, por su propia definición, garantiza la validez de $\{\ \text{wp}.S.Q\ \}\ S\ \{\ Q\ \}$
2. podemos usar wp para encontrar la pre-condición más débil si tenemos un programa y su post-condición;
3. y también para verificar la corrección de una terna de Hoare $\{\ P\ \}\ S\ \{\ Q\ \}$: para ello calculamos la pre-condición más débil $\text{wp}.S.Q$ y probamos $P \Rightarrow \text{wp}.S.Q$;
4. finalmente, si asumimos $\{\ P\ \}\ S\ \{\ Q\ \}$ y en S hay algunas incógnitas, entonces podemos descubrirlas usando $\text{wp}.S.Q$ y razonando con $P \Rightarrow \text{wp}.S.Q$.
5. Recuerde que I es un invariante de un ciclo **do** $B \to S$ **od** si $\{\ I \wedge B\ \}\ S\ \{\ I\ \}$.

1. Para cada uno de los siguientes programas, calcule la precondición más débil y las anotaciones intermedias. Para ello utilice el transformador de predicados wp.

   a) Var $x, y : Num$;
   $\{\qquad\qquad\}$
   $x := x + y$
   $\{x = 6 \wedge y = 5\}$

   b) Var $x : Num$;
   $\{\qquad\}$
   $x := 8$
   $\{x = 8\}$

   c) Var $x : Num$;
   $\{\qquad\}$
   $x := 8$
   $\{x = 7\}$

   d) Var $x, y : Num$;
   $\{\qquad\}$
   $x, y := y, x$
   $\{x = B \wedge y = A\}$

   e) Var $x, y, a : Num$;
   $\{\qquad\qquad\}$
   $a, x := x, y$;
   $\{\qquad\qquad\}$
   $y := a$
   $\{x = B \wedge y = A\}$

   f) Var $x, y : Num$;
   $\{\qquad\}$
   if $x \geq y \to$
   $\quad\{\qquad\}$
   $\quad x := 0$
   $\quad\{\qquad\qquad\}$
   $\square\ x \leq y \to$
   $\quad\{\qquad\}$
   $\quad x := 2$
   $\quad\{\qquad\qquad\}$
   fi
   $\{(x = 0 \vee x = 2) \wedge y = 1\}$

2. Demuestre que las siguientes ternas de Hoare son correctas. En todos los casos las variables x, y son de tipo Int, y a, b de tipo $Bool$.

   a) $\{True\}$
   if $x \geq 1 \to x := x + 1$
   $\square\ x \leq 1 \to x := x - 1$
   fi
   $\{x \neq 1\}$

   b) $\{x \neq y\}$
   if $x > y \to$ **skip**
   $\square\ x < y \to x, y := y, x$
   fi
   $\{x > y\}$

   c) $\{True\}$
   $x, y := y * y, x * x$;
   if $x \geq y \to x := x - y$
   $\square\ x \leq y \to y := y - x$
   fi
   $\{x \geq 0 \wedge y \geq 0\}$

   d) $\{True\}$
   if $\neg a \vee b \to a := \neg a$
   $\square\ a \vee \neg b \to b := \neg b$
   fi
   $\{a \vee b\}$

   e) $\{N \geq 0\}$
   $x := 0$;
   do $x \neq N \to x := x + 1$
   od
   $\{x = N\}$

   f) $\{True\}$
   $r := N$;
   do $r \neq 0 \to$
   $\quad$**if** $r < 0 \to r := r + 1$
   $\quad\square\ r > 0 \to r := r - 1$
   $\quad$**fi**
   od
   $\{r = 0\}$

3. Para cada uno de los siguientes programas, elija valores para las expresiones **E** y **F** de modo que las ternas de Hoare sean correctas.

   a) Var $x, y : Nat$;
   $\{True\}$
   $x, y := x + 1, \mathbf{E}$
   $\{y = x + 1\}$

   b) Var $a, q, c, w : Num$;
   $\{q = a * c \wedge w = c^2\}$
   $a, q := a + c, \mathbf{E}$
   $\{q = a * c\}$

   c) Const $A, B : Nat$;
   Var $q, r : Nat$;
   $\{A = q * B + r\}$
   $q := \mathbf{E};\ r := r - B$
   $\{A = q * B + r\}$

   d) Const $N : Num$;
   Var $x, y, p, q : Num$;
   $\{x * y + p * q = N\}$
   $x := x - p$;
   $q := \mathbf{F}$
   $\{x * y + p * q = N\}$

4. Especifique los siguientes problemas, enunciando pre y postcondición, y luego derive programas imperativos a partir de las especificaciones.

   a) Calcular el mínimo de dos valores.

   b) Calcular el valor absoluto de un número.

5. Demuestre que si la terna de Hoare (a) es correcta, entonces la terna (b) también lo es:[1]

   a) $\{P\}$
   if $B_0 \to S_0$
   $\Box\ B_1 \to S_1$
   fi
   $\{Q\}$

   b) $\{P\}$
   if $B_0 \to S_0$
   $\Box\ \neg B_0 \to S_1$
   fi
   $\{Q\}$

   ¿Qué utilidad tiene esta propiedad cuando se programa en lenguaje C?

6. Analice los siguientes programas anotados. En cada caso, describa en lenguaje natural la postcondición, y decida si el programa efectivamente valida las anotaciones.

   a) Const $N : Int,\ A : array[0, N)\ of Num$;
   Var $s : Num,\ i : Int$;
   $\{N \geq 0\}$
   $i, s := 0, 0$ **;**
   do $i \neq N \to$
   $\quad s := s + A.i$
   od
   $\{s = \langle \sum k\ : 0 \leq k < N :\ A.k \rangle\}$

   b) Const $N : Int,\ A : array[0, N)\ of Num$;
   Var $s : Num,\ i : Int$;
   $\{N \geq 0\}$
   $i, s := 0, 0$ **;**
   do $i \neq N \to$
   $\quad i := i + 1$ **;**
   $\quad s := s + A.i$
   od
   $\{s = \langle \sum k\ : 0 \leq k < N :\ A.k \rangle\}$

   c) Const $N : Int,\ A : array[0, N)\ of Num$;
   Var $s : Num,\ i : Int$;
   $\{N \geq 0\}$
   $i, s := -1, 0$ **;**
   do $i \neq N \to$
   $\quad i := i + 1$ **;**
   $\quad s := s + A.i$
   od
   $\{s = \langle \sum k\ : 0 \leq k < N :\ A.k \rangle\}$

   d) Const $E : Num,\ N : Int,\ A : array[0, N)\ of Num$;
   Var $i : Int,\ r : Bool$;
   $\{N \geq 0\}$
   $i, r := 0, False$ **;**
   do $i \neq N \wedge \neg r \to$
   $\quad$ **if** $A.i = E \to r := True$
   $\quad \Box\ A.i \neq E \to$ **skip**
   $\quad$ **fi ;**
   $\quad i := i + 1$
   od
   $\{\langle \exists k\ : 0 \leq k < N :\ A.k = E \rangle \Rightarrow A.i = E\}$

[1] Atención: los programas no son equivalentes. Proponga guardas y sentencias concretas y una ejecución posible para el primer programa que no sea posible en el segundo (ayuda: piense en no-determinismo).

7. Decida si los siguientes predicados son invariantes válidos del ciclo del programa 6(b). Justifique.

   a) $\{1 \leq i\}$
   b) $\{0 \leq i\}$
   c) $\{0 \leq i \leq N\}$
   d) $\{s = \langle \sum k\ : 0 \leq k < N :\ A.i \rangle\}$
   e) $\{0 \leq s \leq \langle \sum k\ : 0 \leq k < N :\ A.i \rangle\}$

8. *Swap*(intercambio): Considere los siguientes programas que intercambian los valores de dos variables x e y de tipo Int:

$$x, y := y, x \qquad\qquad \begin{array}{l} z := x; \\ x := y; \\ y := z \end{array} \qquad\qquad \begin{array}{l} x := x - y; \\ y := x + y; \\ x := y - x \end{array}$$

   Especifique el *swap* (con pre y postcondición), y verifique que los programas satisfacen la especificación.

9. Derive un programa para calcular el máximo común divisor entre dos enteros positivos. Utilice la siguiente especificación:

   Const $X, Y : Int$;
   Var $x, y : Int$;
   $\{X > 0 \wedge Y > 0 \wedge x = X \wedge y = Y\}$
   S
   $\{x = mcd.X.Y\}$

   Utilice como invariante $\{I : x > 0 \wedge y > 0 \wedge mcd.x.y = mcd.X.Y\}$.

   Para la derivación serán de utilidad las siguientes propiedades del mcd:

   a) $mcd.x.x = x$
   b) $mcd.x.y = mcd.y.x$
   c) $x > y \Rightarrow mcd.x.y = mcd.(x - y).y$
   d) $y > x \Rightarrow mcd.x.y = mcd.x.(y - x)$

10. Considere las siguientes definiciones recursivas de la función de exponenciación $exp.x.y$, especificada como $exp.x.y = x^y$:

   a) Definición de complejidad lineal:

$$\begin{array}{lll} exp.x.y & = & (\ \ y = 0 \to 1 \\ & & \ \square\ y \neq 0 \to x * exp.x.(y-1) \\ & &) \end{array}$$

   b) Definición de complejidad logarítmica:

$$\begin{array}{lll} exp.x.y & = & (\ \ y = 0 \to 1 \\ & & \ \square\ y \neq 0 \to (\ \ y \bmod 2 = 0 \to exp.(x * x).(y \div 2) \\ & & \qquad\qquad\quad \square\ y \bmod 2 = 1 \to x * exp.x.(y-1) \\ & & \qquad\qquad\quad) \\ & &) \end{array}$$

   Derive **dos** programas imperativos que calculen la exponenciación, cada uno utilizando una de las definiciones recursivas. Utilice la siguiente especificación:

   Const $X, Y : Int$;
   Var $x, y, r : Int$;
   $\{x = X \wedge y = Y \wedge x \geq 0 \wedge y \geq 0\}$
   S
   $\{r = X^Y\}$

   Utilice como invariante $\{I : y \geq 0 \wedge r * x^y = X^Y\}$.