Consejos para la Derivación de Programas Imperativos

Franco M. Luque

Algoritmos y Estructuras de Datos I
2$^{\text{do}}$ cuatrimestre 2017

1. Rangos con Pares de Elementos de Arreglo

Ejemplo (p5, ej. 9): Dado un arreglo $A : array[0, N)\ of\ Num$ con $N \geq 0$, contar cuántas veces coinciden dos elementos:
Const $N : Int, A : array\ [0, N)\ of\ Int$;
Var $r : Int$;
$\{P : N \geq 0\}$
S
$\{Q :\ r = \langle N\, i, j\ : 0 \leq i < j < N :\ A.i = A.j \rangle\}$

Es evidente que **se necesita un ciclo** para recorrer el arreglo.

Usando la técnica de reemplazo de constante por variable, se crea una nueva variable $n : Int$ y se obtienen el invariante y la guarda:

$$I \doteq r = \langle N\, i, j\ : 0 \leq i < j < n :\ A.i = A.j \rangle \wedge 0 \leq n \leq N$$
$$B \doteq n < N$$

De esta forma queda garantizado el requisito $I \wedge \neg B \Rightarrow Q$.

El programa será de la forma:
Const $N : Int,\ A : array[0, N)\ of Num$;
Var $r, n : Int$;
$\{P : N \geq 0\}$
S_0 **;**
$\{I\}$
do $n < N \rightarrow$
  $\{I \wedge B\}\ S_1\ \{I\}$
od
$\{Q :\ r = \langle N\, i, j\ : 0 \leq i < j < N :\ A.i = A.j \rangle\}$
Falta derivar S_0 (inicialización) y S_1 (cuerpo del ciclo).

1.1. Inicialización (S_0)

Probamos con S_0 de la forma $n, r := 0, E$ con E siendo una incógnita a elegir tal que valga la terna:

$$\{P\}\ n, r := 0, E\ \{I\}$$

O, equivalentemente,

$$P \Rightarrow wp.(n, r := 0, E).I$$

Luego, tomamos como hipótesis P y vemos la wp:

$$\begin{array}{ll} & wp.(n, r := 0, E).I \\ \equiv & \{ \text{ definición } wp \text{ para } := \} \\ & E = \langle N\, i, j \ : 0 \leq i < j < 0 : \ A.i = A.j \rangle \wedge 0 \leq 0 \leq N \\ \equiv & \{ \text{ hipótesis } P : N \geq 0 \} \\ & E = \langle N\, i, j \ : 0 \leq i < j < 0 : \ A.i = A.j \rangle \\ \equiv & \{ \text{ rango vacío } \} \\ & E = 0 \\ \equiv & \{ \text{ elijo } \boxed{E = 0} \} \\ & True \end{array}$$

1.2. Cuerpo del Ciclo (S_1)

Probamos con S_1 de la forma $n, r := n + 1, E$ con E siendo una incógnita a elegir tal que valga la terna:

$$\{I \wedge B\}\ n, r := n + 1, E\ \{I\}$$

O, equivalentemente,

$$I \wedge B \Rightarrow wp.(n, r := n + 1, E).I$$

Luego, tomamos como hipótesis $I \wedge B$ y vemos la wp:

$$\begin{array}{ll} & wp.(n, r := n + 1, E).I \\ \equiv & \{ \text{ definición } wp \text{ para } := \} \\ & E = \langle N\, i, j \ : 0 \leq i < j < n + 1 : \ A.i = A.j \rangle \wedge 0 \leq n + 1 \leq N \\ \equiv & \{ \ 0 \leq n + 1 \leq N \text{ vale por hipótesis } 0 \leq n \text{ y } n < N \ \} \\ & E = \langle N\, i, j \ : 0 \leq i < j < n + 1 : \ A.i = A.j \rangle \\ \equiv & \{ \text{ partición de rango con } j < n \vee j = n. \text{ eliminación de variable con } j = n. \ \} \\ & E = \langle N\, i, j \ : 0 \leq i < j < n : \ A.i = A.j \rangle + \langle N\, i \ : 0 \leq i < n : \ A.i = A.n \rangle \\ \equiv & \{ \text{ hipótesis para } r \ \} \\ & E = r + \langle N\, i \ : 0 \leq i < n : \ A.i = A.n \rangle \end{array}$$

Acá, no hay forma de elegir una expresión E válida en el lenguaje de programación. La asignación necesita una hipótesis adicional en la precondición, que diga que **una nueva variable auxiliar** s contiene el valor de la cuantificación que sobra:

$$s = \langle N\, i \ : 0 \leq i < n : \ A.i = A.n \rangle\,.$$

Para garantizar esta hipótesis, **debe introducirse una nueva sentencia que calcule el valor de s antes de la asignación**. Luego, se replantea el cuerpo del ciclo S_1 de la siguiente forma:

$\{I \wedge B\}$
S_2 **;**
$\{I \wedge B \wedge s = \langle N\, i \ : 0 \leq i < n : \ A.i = A.n \rangle\}$
$n, r := n + 1, E$
$\{I\}$

Con la nueva hipótesis, ya se puede derivar la asignación. Tomamos como hipótesis $I \wedge B \wedge s = \langle N\, i \ : 0 \leq i < n : \ A.i = A.n \rangle$ y vemos la wp:

$wp.(n, r := n + 1, E).I$
$\equiv$ { mismos pasos que antes }
$E = r + \langle N\, i \ : 0 \leq i < n : \ A.i = A.n \,\rangle$
$\equiv$ { nueva hipótesis para s }
$E = r + s$

1.2.1. Ciclo Anidado (S_2)

Sólo queda derivar S_2. Llamamos P' a la precondición y Q' a la postcondición de S_2:

$\{P' : I \wedge B\}$
S_2 **;**
$\{Q' : I \wedge B \wedge s = \langle N\, i \ : 0 \leq i < n : \ A.i = A.n \,\rangle\}$

Acá, S_2 debe preservar la validez de $I \wedge B$, y por lo tanto no debe modificar las variables r y n. Es decir, r y n **son constantes** para S_2.

Para calcular el valor de s, es evidente que **se necesita un ciclo que recorra los primeros** n **elementos del arreglo**. Luego se puede usar la técnica de reemplazo de constante por variable, creando una nueva variable $m : Int$ y obteniendo invariante y guarda:

$$I' \doteq I \wedge B \wedge s = \langle N\, i \ : 0 \leq i < m : \ A.i = A.n \,\rangle \wedge 0 \leq m \leq n$$
$$B' \doteq m < n$$

Observación importante: Sólo se reemplaza la ocurrencia de n en el rango de la nueva cuantificación. No se reemplaza en el término ni en ningún otra parte.

Luego, se derivan inicialización y cuerpo de este ciclo para obtener (**queda como ejercicio**):

$m, s := 0, 0$ **;**
do $m < n \rightarrow$
  if $A.m = A.n \rightarrow$
    $m, s := m + 1, s + 1$
  $\square$ $A.m \neq A.n \rightarrow$
    $m := m + 1$
  fi
od ;

1.3. Resultado Final

Const $N : Int,\ A : array[0, N)\ of Num;$
Var $r, n, m, s : Int;$
$\{P : N \geq 0\}$
$n, r := 0, 0$ **;**
do $n < N \rightarrow$
  $m, s := 0, 0$ **;**
  do $m < n \rightarrow$
    if $A.m = A.n \rightarrow$
      $m, s := m + 1, s + 1$
    $\square$ $A.m \neq A.n \rightarrow$
      $m := m + 1$
    fi
  od **;**
  $n, r := n + 1, r + s$ **;**
od
$\{Q :\ r = \langle N\, i, j\ : 0 \leq i < j < N :\ A.i = A.j\ \rangle\}$