Transformadores de Predicados y Semántica de Programas

A la memoria de mi padre,
Manuel Ruiz Hoyos,
sencillo matemático y gran Profesor.

A mi nieto recién nacido,
Manuel Ruiz Sánchez,
con el deseo de que complete la
cuarta generación de matemáticos.

Blas C. Ruiz Jiménez

Blas Carlos Ruiz Jiménez

Profesor Titular de Universidad
Departamento de Lenguajes y Ciencias de la Computación

E.T.S.I. Informática. Universidad de Málaga

TRANSFORMADORES DE PREDICADOS Y SEMÁNTICA DE PROGRAMAS

(CORRECCIÓN DE LA 2ª ED. – SETIEMBRE DE 2003)

Málaga, Octubre de 2010

IMPRIME: *IMAGRAF–Impresores.*
C/Nabuco, Nave 14-D. 29006–Málaga. Tel.: 2328597

I.S.B.N.: **84-607-5971-7**
Depósito Legal: MA-1203-2003

Composición realizada por el autor en LaTeX2ϵ.

Índice general

El estilo Semántico de Dijkstra 41

Índice de figuras

Capítulo 5

El cálculo de Hoare

DEFINICIÓN 5.0 (Cálculo o Sistema Axiomático)

1. *Un cálculo $\mathcal{C}$ es un conjunto de reglas de una de las formas siguientes*

$$\frac{}{\alpha} \text{ (axioma)} \qquad \frac{\pi_1\ \pi_2 \ldots\ \pi_n}{\gamma} \text{ (regla de inferencia propia)}$$

   La segunda regla se lee: a partir de π_1, π_2, ..., π_n es posible derivar γ.

2. *Diremos que ψ es deducible desde M en el cálculo $\mathcal{C}$, para abreviar $M \vdash_{\mathcal{C}} \psi$, si existe una derivación de la forma $\dfrac{\pi_1\ \ldots\ \pi_n}{\psi}$, donde, para cada i, o bien $\pi_i \in M$, o bien $M \vdash_{\mathcal{C}} \pi_i$.*

Un ejemplo de cálculo es el *Sistema de Hilbert* para la lógica de predicados, que puede verse en [Sperschneider y Antoniou, 1991]:69. Otro ejemplo de cálculo es el cálculo de Hoare, que veremos a continuación.

5.0. Las reglas del cálculo de Hoare

El cálculo de Hoare es una extensión del cálculo de predicados con unos objetos especiales (los tripletes) y unas reglas también especiales para derivar tripletes a partir de otros predicados y tripletes; así, los tripletes pueden ser utilizados conjuntamente con el resto de predicados.

Años después de la publicación por CAR Hoare del célebre *An Axiomatic Basis for Computer Programming* [Hoare, 1969], Nikolas Wirth y C.A.R. Hoare escribieron en 1973 *An axiomatic Definition of the Programming Language Pascal*, en el cual definen en forma axiomática el lenguaje PASCAL [Wirth y Hoare, 1973]; tal cálculo considera las reglas que aparecen en la Figura 5.0. La regla (ref) o de refinamiento es genérica para cualquier sentencia y mezcla tripletes con otros predicados[1]. El resto de reglas son particulares para cada construcción del lenguaje. Dos de ellas son para las sentencias básicas: $(nada)$ y $(:=)$; en éstas no aparece ningún antecedente y son axiomas. El resto de reglas son para la selectiva, la composición y el bucle.

[1] En la presentación original [Hoare, 1969], en el antecedente de la regla aparecen la implicaciones $P \Rightarrow P'$ y $Q' \Rightarrow Q$ sin el operador $[\,]$ (*ptle*), aunque obviamente hay que entenderlas tal como las hemos descrito.

$$\frac{[P \Rightarrow P'] \qquad \{P'\}S\{Q'\} \qquad [Q' \Rightarrow Q]}{\{P\}S\{Q\}}\,(ref)$$

$$\frac{}{\{Q\}nada\{Q\}}\,(nada) \qquad\qquad \frac{}{\{x := E.Q\}x := E\{Q\}}\,(:=)$$

$$\frac{\{P \wedge b\}S\{Q\} \;\; \{P \wedge \neg b\}T\{Q\}}{\{P\}if\ b\ then\ S\ else\ T\{Q\}}\,(si_1) \qquad \frac{\{P \wedge b\}S\{Q\} \;\; [P \wedge \neg b \Rightarrow Q]}{\{P\}if\ b\ then\ S\{Q\}}\,(si_2)$$

$$\frac{\{P\}S\{Y\} \;\; \{Y\}T\{Q\}}{\{P\}S;T\{Q\}}\,(;) \qquad\qquad \frac{\{P \wedge b\}S\{P\}}{\{P\}while\ b\ do\ S\{P \wedge \neg b\}}\,(rep)$$

Figura 5.0: Las reglas del cálculo de Hoare.

DEFINICIÓN 5.1 (Cálculo de Hoare) *Sea $\mathcal{A}$ una Σ-álgebra correspondiente a un lenguaje de programación*[2]*, y consideremos la teoría $T(\mathcal{A})$; es decir, el conjunto de fórmulas sin variables libres y válidas en $\mathcal{A}$.*

1. *El cálculo de Hoare es el conjunto de reglas*

$$\mathcal{H} = \{(ref), (nada), (;), (si_1), (si_2), (rep)\}.$$

2. *La teoría o lógica de Hoare (para simplificar $\mathcal{LH}$) viene dada por el conjunto*

$$H(\mathcal{A}) = \{\ \{P\}S\{Q\}\ \mid\ T(\mathcal{A}) \vdash_{\mathcal{H}} \{P\}S\{Q\}\ \}$$

Para simplificar, sobrentendiendo el álgebra y su teoría, también usaremos la forma $\vdash_{\mathcal{H}} \{P\}S\{Q\}$ en lugar de $T(\mathcal{A}) \vdash_{\mathcal{H}} \{P\}S\{Q\}$.

Una propiedad importante del conjunto $H(\mathcal{A})$ es que es inductivo. En el Teorema 5.14 veremos una aplicación.

Si consideramos la equivalencia $if\ b\ then\ S \equiv if\ b\ then\ S\ else\ nada$, el lector debe observar que la regla (si_2) es consecuencia de las reglas (si_1), $(nada)$ y (ref). En efecto; consideremos los antecedentes de la regla (si_2):

$$h_1 \doteq \{P \wedge b\}S\{Q\} \qquad\qquad h_2 \doteq [P \wedge \neg b \Rightarrow Q]$$

Entonces tenemos la siguiente derivación utilizando las reglas de la Figura 5.0

$$\dfrac{\overbrace{\{P \wedge b\}S\{Q\}}^{h_1} \qquad \dfrac{\overbrace{[P \wedge \neg b \Rightarrow Q]}^{h_2} \qquad \dfrac{}{\{Q\}nada\{Q\}}\,(nada)}{\{P \wedge \neg b\}nada\{Q\}}\,(ref)}{\{P\}if\ b\ then\ S\{Q\}}\,(si_1)$$

Según la Definición 5.0, tenemos $\{h_1, h_2\} \vdash_{\mathcal{H}} \{P\}if\ b\ then\ S\{Q\}$. Por tanto, podemos suprimir la regla (si_2) sin alterar la $\mathcal{LH}$. Esto significa que para estudiar las propiedades de $\mathcal{LH}$ podemos prescindir de la regla (si_2), aunque podemos usarla para inferir tripletes.

[2] Véase la Sección 1.6, en página 24.

EJEMPLO 5.2 Un ejemplo simple de derivación en el Cálculo de Hoare es:

$$\dfrac{[x > 8 \;\Rightarrow\; x > 2] \qquad \dfrac{}{\{x > 2\}x := x + 1\{x > 3\}}\,(:=)}{\{x > 8\}x := x + 1\{x > 3\}}\,(ref)$$

5.3 *Probad que si eliminamos la regla* (ref) *del cálculo de Hoare, entonces es imposible inferir el triplete anterior* $\{x > 8\}x := x + 1\{x > 3\}$.

Otro ejemplo de derivación en el Cálculo de Hoare es:

$$\dfrac{\dfrac{}{\{a + x > 0\}x := x + 1\{a - 1 + x > 0\}}\,(:=) \qquad \dfrac{}{\{a - 1 + x\}a := a - 1\{a + x > 0 > 0\}}\,(:=)}{\{a + x > 0\}\; x := x + 1; a := a - 1\; \{a + x > 0\}}\,(;)$$

Obsérvese que la regla $(;)$ necesita un predicado intermedio y éste hay que buscarlo durante la prueba de la corrección del triplete objetivo; después se verán algunos ejemplos donde la búsqueda de tales predicados puede ser más complicada. EJEMPLO

5.4 *Si incluimos la sentencia* $aborta$*, ¿cual sería la regla de Hoare para tal sentencia?.*

EJEMPLO 5.5 Para probar la corrección del programa:

$$\begin{array}{l} m := a; \\ if\ b > m\ then\ m := b; \\ if\ c > m\ then\ m := c; \\ \{m = \text{máx}(a, b, c)\} \end{array}$$

observamos que es suficiente probar la corrección de

$$\{m = a\}\ if\ b > m\ then\ m := b\ \{m = \text{máx}(a, b)\} \qquad (*)$$

ya que aplicamos la derivación

$$\dfrac{\dfrac{\dfrac{}{\{C\}m := a\{m = a\}}\,(:=) \qquad (*)}{\{C\}m := a; if\ b > m\ then\ m := b\{m = \text{máx}(a, b)\}}\,(;) \qquad (*)}{\{C\}m := a; if\ b > m\ then\ m := b; if\ c > m\ then\ m := c\{m = \text{máx}(\text{máx}(a, b), c)\}}\,(;)$$

Ahora basta observar que $\text{máx}(\text{máx}(a, b), c) = \text{máx}(a, b, c)$. Queda probar $(*)$; lo hacemos con ayuda de la regla (si_2):

$$\dfrac{\{m = a \wedge b > m\}m := b\{m = \text{máx}(a, b)\} \quad [m = a \wedge b \leq m \;\Rightarrow\; m = \text{máx}(a, b)]}{\{m = a\}\ if\ b > m\ then\ m := b\{m = \text{máx}(a, b)\}}\,(si_2)$$

La implicación del antecedente sigue de la definición de máx. Probemos el primer antecedente:

$$\dfrac{[m = a \wedge b > m \;\Rightarrow\; b = \text{máx}(a, b)] \qquad \dfrac{}{\{b = \text{máx}(a, b)\}m := b\{m = \text{máx}(a, b)\}}\,(:=)}{\{m = a \wedge b > m\}m := b\{m = \text{máx}(a, b)\}}\,(ref)$$

Y la implicación primera sigue de la definición de máx. EJEMPLO

EJEMPLO 5.6 Probaremos, para cada sentencia S el triplete $\vdash_{\mathcal{H}} \{Falso\}S\{Q\}$. La primera observación es que bastará probar

$$\forall S : S \in \mathcal{P}rog : \vdash_{\mathcal{H}} \{F\}S\{F\} \tag{1}$$

y aplicar $[F \Rightarrow Q]$ junto a la regla de refinamiento. (1) se demuestra por inducción sobre la estructura de la sentencia. Los casos bases corresponden a la asignación y la sentencia $nada$. El primero es consecuencia de $[x := a.F \equiv F]$, y el segundo es trivial. Veamos ahora los pasos inductivos; para la composición:

$$\frac{\dfrac{HI}{\{F\}S\{F\}} \qquad \dfrac{HI}{\{F\}T\{F\}}}{\{F\}S;T\{F\}}\,(;)$$

Para la selectiva partimos de la hipótesis de inducción

$$\begin{array}{ll} & \{F\}S\{F\} \wedge \{F\}T\{F\} \\ = & \quad \because \text{véase el Lema 5.7 siguiente} \\ & \{b \wedge F\}S\{F\} \wedge \{\neg b \wedge F\}T\{F\} \\ \Rightarrow & \quad \because \text{regla } (si_1) \\ & \{F\}[\![\, b \rightarrow S \,\square\, T \,]\!]\{F\} \end{array}$$

Y para el bucle, a partir de la hipótesis de inducción:

$$\begin{array}{ll} & \{F\}S\{F\} \\ = & \quad \because \text{Lema 5.7} \\ & \{b \wedge F\}S\{F\} \\ \Rightarrow & \quad \because \text{regla } (rep) \text{ y cálculo} \\ & \{F\}while\ b\ do\ S\{F\} \end{array}$$

EJEMPLO

En la prueba anterior hacemos uso del siguiente resultado.

LEMA 5.7 *El cálculo de Hoare es sustitutivo. Es decir: Si $[Y \equiv Y']$ y $[X \equiv X']$, entonces el triplete $\vdash_{\mathcal{H}} \{Y\}S\{X\}$ es inferible si y solo si lo es el triplete $\vdash_{\mathcal{H}} \{Y'\}S\{X'\}$.*

Demostración.– Basta aplicar la regla de refinamiento. LEMA

5.8 *[259]* **(Enero, 96)** *Probad, $\forall S, P :: \vdash_{\mathcal{H}} \{P\}S\{Cierto\}$.*

5.9 *[260] ¿Es posible inferir los tripletes $\vdash_{\mathcal{H}} \{Falso\}S\{Q\}$ y $\vdash_{\mathcal{H}} \{P\}S\{Falso\}$?*

DEFINICIÓN 5.10 (Equivalencia de Programas en el Cálculo de Hoare) *En $\mathcal{LH}$, dos programas S y T son equivalentes (escribiremos $S =_{\mathcal{H}} T$) si*

$$\forall P, Q : P, Q \in \mathcal{P} : \vdash_{\mathcal{H}} \{P\}S\{Q\} \Longleftrightarrow \vdash_{\mathcal{H}} \{P\}T\{Q\}$$

5.11 *[260] Demostrad la equivalencia $S; nada =_{\mathcal{H}} S$.*

5.1. Corrección del Cálculo de Hoare (sin bucles)

TEOREMA 5.12 (Corrección de las reglas de Hoare) *Salvo la regla* (rep)*, las reglas del cálculo de Hoare (Figura 5.0) son válidas para la semántica de Dijkstra.*

NOTA *5.13 Obsérvese que por aplicación de la regla* (rep) *del bucle podemos inferir*

$$\frac{\{x>0\}x:=x+1\{x>0\}}{\{x>0\}while\ x>0\ do\ x:=x+1\{x>0\wedge x\leq 0\}}\ (rep)$$

y el triplete obtenido nunca será válido en la $\mathcal{LD}$ *ya que ésta considera corrección total.*

Demostración.– Hemos de probar que si se dan los antecedentes de cada regla de la Figura 5.0, podemos inferir, dentro del cálculo de predicados sobre un espacio de estados, el correspondiente consecuente. Es decir, hay que probar las siguientes implicaciones:

$$\begin{array}{c}[P\ \Rightarrow\ P']\wedge\vdash_{\mathcal{D}}\{P'\}S\{Q'\}\wedge[Q'\ \Rightarrow\ Q]\\ \overset{1}{\Rightarrow}\quad\text{- -}\\ \vdash_{\mathcal{D}}\{P\}S\{Q\}\end{array}$$

$$\begin{array}{c}C\\ \overset{2}{\Rightarrow}\quad\text{- - - - - - - - - - -}\\ \vdash_{\mathcal{D}}\{Q\}nada\{Q\}\end{array}\qquad\begin{array}{c}C\\ \overset{3}{\Rightarrow}\quad\text{- - - - - - - - - - - - -}\\ \vdash_{\mathcal{D}}\{x:=E.Q\}x:=E\{Q\}\end{array}$$

$$\begin{array}{c}\vdash_{\mathcal{D}}\{P\}S\{Y\}\wedge\vdash_{\mathcal{D}}\{Y\}T\{Q\}\\ \overset{4}{\Rightarrow}\quad\text{- - - - - - - - - - - - - - - - -}\\ \vdash_{\mathcal{D}}\{P\}S;T\{Q\}\end{array}$$

$$\begin{array}{c}\vdash_{\mathcal{D}}\{P\wedge b\}S\{Q\}\wedge\vdash_{\mathcal{D}}\{P\wedge\neg b\}T\{Q\}\\ \overset{5}{\Rightarrow}\quad\text{- - - - - - - - - - - - - - - - - - -}\\ \vdash_{\mathcal{D}}\{P\}if\ b\ then\ S\ else\ T\{Q\}\end{array}$$

$$\begin{array}{c}\vdash_{\mathcal{D}}\{P\wedge b\}S\{R\}\wedge[P\wedge\neg b\ \Rightarrow\ R]\\ \overset{6}{\Rightarrow}\quad\text{- -}\\ \vdash_{\mathcal{D}}\{P\}if\ b\ then\ S\{R\}\end{array}$$

La primera sigue en la forma siguiente:

$$\begin{array}{ll}& [P\ \Rightarrow\ P']\wedge\vdash_{\mathcal{D}}\{P'\}S\{Q'\}\wedge[Q'\ \Rightarrow\ Q]\\ =& \quad\because\text{definición de triplete de Dijkstra (Definición 3.4)}\\ & [P\ \Rightarrow\ P']\wedge[P'\ \Rightarrow\ S.Q']\wedge[Q'\ \Rightarrow\ Q]\\ \Rightarrow& \quad\because\text{transitividad de }\Rightarrow\text{ y monotonía de }S\\ & [P\ \Rightarrow\ S.Q']\wedge[S.Q'\ \Rightarrow\ S.Q]\\ \Rightarrow& \quad\because\text{transitividad de }\Rightarrow\\ & [P\ \Rightarrow\ S.Q]\\ =& \quad\because\text{definición de triplete}\\ & \vdash_{\mathcal{D}}\{P\}S\{Q\}\end{array}$$

Las dos siguientes son inmediatas:

$\vdash_{\mathcal{D}} \{Q\} nada \{Q\}$
$=$ $\quad \because$ definición de triplete
$[Q \ \Rightarrow \ nada.Q]$
$=$ $\quad \because$ semántica de $nada$
$[Q \ \Rightarrow \ Q]$
$=$ $\quad \because$ CP
$Cierto$

$\vdash_{\mathcal{D}} \{z := E.Q\} z := E \{Q\}$
$=$ $\quad \because$ definición de triplete
$[z := E.Q \ \Rightarrow \ z := E.Q]$
$=$ $\quad \because$ CP
$Cierto$

Para la *composición* tenemos:

$\vdash_{\mathcal{D}} \{P\} S \{Y\} \wedge \vdash_{\mathcal{D}} \{Y\} T \{Q\}$
$=$ $\quad \because$ definición de triplete
$[P \ \Rightarrow \ S.Y] \wedge [Y \ \Rightarrow \ T.Q]$
$\Rightarrow$ $\quad \because [\,]$ es conjuntivo, S es monótona
$[(P \ \Rightarrow \ S.Y) \wedge (S.Y \ \Rightarrow \ S.T.Q)]$
$\Rightarrow$ $\quad \because$ transitividad de $\Rightarrow$
$[P \ \Rightarrow \ S.T.Q]$
$=$ $\quad \because$ semántica de la composición y definición de triplete
$\vdash_{\mathcal{D}} \{P\} S; T \{Q\}$

Para la *selectiva* (si_1) (o para (si_2) tomando $T == nada$)

$\vdash_{\mathcal{D}} \{P \wedge b\} S \{Q\} \wedge \vdash_{\mathcal{D}} \{P \wedge \neg b\} T \{Q\}$
$=$ $\quad \because$ definición de triplete
$[P \wedge b \ \Rightarrow \ S.Q] \wedge [P \wedge \neg b \ \Rightarrow \ T.Q]$
$=$ $\quad \because$ conjuntividad de $[\,]$, cálculo de predicados
$[P \ \Rightarrow \ (b \ \Rightarrow \ S.Q) \wedge P \ \Rightarrow \ (\neg b \ \Rightarrow \ T.Q)]$
$=$ $\quad \because \Rightarrow$ es conjuntivo en el consecuente
$[P \ \Rightarrow \ (b \ \Rightarrow \ S.Q) \wedge (\neg b \ \Rightarrow \ T.Q)]$
$=$ $\quad \because$ tercio excluido
$[P \ \Rightarrow \ (b \vee \neg b) \wedge (b \ \Rightarrow \ S.Q) \wedge (\neg b \ \Rightarrow \ T.Q)]$
$=$ $\quad \because$ semántica selección
$[P \ \Rightarrow \ [\![b \to S \,\square\, \neg b \to T]\!] \,.\, Q]$
$=$ $\quad \because$ definición de triplete
$\vdash_{\mathcal{D}} \{P\} [\![b \to S \,\square\, \neg b \to T]\!] \{Q\}$
$=$
$\vdash_{\mathcal{D}} \{P\} if \ b \ then \ S \ else \ T \{Q\}$ TEOREMA

Como aplicación del teorema anterior probaremos el siguiente:

TEOREMA 5.14 *La $\mathcal{LH}$ para un lenguaje sin bucles es correcta cra $\mathcal{LD}$. Es decir, los tripletes obtenidos a través del cálculo de Hoare, sin el uso de la regla para los bucles (rep), son válidos en el cálculo de Dijkstra:*

$$\vdash_{\mathcal{H}} \{P\} S \{R\} \qquad \Rightarrow \qquad \vdash_{\mathcal{D}} \{P\} S \{R\} \qquad (correc)$$

Demostración.– La propiedad $correc$ la escribimos con algo más de rigor como:

$$\forall \{P\} S \{R\} \ : \ \{P\} S \{R\} \in H(\mathcal{A}) \ : \vdash_{\mathcal{D}} \{P\} S \{R\} \qquad (correc)$$

donde recordemos de la Definición 5.1, que el conjunto $H(\mathcal{A})$ es

$$H(\mathcal{A}) = \{ \ \{P\} S \{Q\} \mid T(\mathcal{A}) \vdash_{\mathcal{H}} \{P\} S \{Q\} \ \}$$

y resulta ser un conjunto inductivo para la siguiente relación entre tripletes

$$t' < t \ \doteq \text{ el triplete } t' \text{ aparece en el árbol de derivación del triplete } t$$

Inducción sobre las derivaciones La técnica que utilizaremos para probar la propiedad $(correc)$ es *inducción sobre la derivación del triplete* $\{P\}S\{R\}$ *en el cálculo de Hoare*, cuyo esquema es el siguiente:

(cb) CASOS BASE: $\vdash_{\mathcal{D}} \{P\}S\{R\}$ es cierta para aquellos tripletes $\{P\}S\{R\}$ derivados a través de reglas donde no aparecen otros tripletes en el antecedente; estos son los casos $\{P\}S\{R\}$, tales que no existe t' verificando $t' < \{P\}S\{R\}$.

(pi) PASOS INDUCTIVOS: para cada regla $\boxed{r}$, supuesta la propiedad para los tripletes $\{P'\}S'\{R'\}$ que aparecen en el antecedente de la regla $\boxed{r}$ (Hipótesis de Inducción), hay que demostrar la propiedad para el triplete que aparece en el consecuente de la regla $\boxed{r}$.

A la vista del esquema anterior, el caso base puede dividirse en dos subcasos, ya que existen dos reglas sin tripletes en el antecedente:

$(cb1)$ $\vdash_{\mathcal{D}} \{Q\}nada\{Q\}$.

$(cb2)$ $\vdash_{\mathcal{D}} \{z := E.Q\}z := E\{Q\}$.

Ambas han sido probadas en la prueba del Teorema 5.12.

De la misma forma, el paso inductivo (pi) se divide en cuatro subcasos, ya que hay cuatro reglas en las cuales aparecen tripletes en el antecedente. Aplicaremos reiteradamente el Teorema 5.12 sobre la corrección de las diferentes reglas para la semántica de Dijkstra.

$(pi1)$ $\{P\}S\{Q\}$ ha sido inferido por la regla de refinamiento; entonces, partimos del antecedente

$$
\begin{array}{ll}
 & [P \Rightarrow P'] \wedge \vdash_{\mathcal{H}} \{P'\}S\{Q'\} \wedge [Q' \Rightarrow Q] \\
\Rightarrow & \because \text{HI, ya que el triplete aparece en el antecedente} \\
 & [P \Rightarrow P'] \wedge \vdash_{\mathcal{D}} \{P'\}S\{Q'\} \wedge [Q' \Rightarrow Q] \\
\Rightarrow & \because \text{corrección de la regla } (ref) \text{ en la semántica de Dijkstra} \\
 & \vdash_{\mathcal{D}} \{P\}S\{Q\}
\end{array}
$$

$(pi2)$ $\{P\}S\{Q\}$ ha sido inferido por la regla de la composición; entonces, partimos del antecedente:

$$
\begin{array}{ll}
 & \vdash_{\mathcal{H}} \{P\}S\{Y\} \wedge \vdash_{\mathcal{H}} \{Y\}T\{Q\} \\
\Rightarrow & \because \text{HI} \\
 & \vdash_{\mathcal{D}} \{P\}S\{Y\} \wedge \vdash_{\mathcal{D}} \{Y\}T\{Q\} \\
\Rightarrow & \because \text{corrección de la regla } (;) \text{ en la semántica de Dijkstra} \\
 & \vdash_{\mathcal{D}} \{P\}S;T\{Q\}
\end{array}
$$

$(pi3)$ El triplete $\{P\}S\{Q\}$ ha sido inferido por aplicación de la regla para la selectiva (si_1); entonces, el antecedente es:

$$\vdash_{\mathcal{H}} \{P \wedge b\}S\{Q\} \wedge \vdash_{\mathcal{H}} \{P \wedge \neg b\}T\{Q\}$$

$\Rightarrow$ $\because$ HI

$\vdash_{\mathcal{D}} \{P \wedge b\} S \{Q\} \wedge \vdash_{\mathcal{D}} \{P \wedge \neg b\} T \{Q\}$

$\Rightarrow$ $\because$ corrección de la regla (si_1) en la semántica de Dijkstra

$\vdash_{\mathcal{D}} \{P\} if\ b\ then\ S\ else\ T \{Q\}$

$(pi4)$ El triplete $\{P\}S\{Q\}$ ha sido inferido por aplicación de la regla para la selectiva (si_2); entonces

$\vdash_{\mathcal{H}} \{P \wedge b\} S \{Q\} \wedge [P \wedge \neg b \ \Rightarrow\ Q]$

$\Rightarrow$ $\because$ HI

$\vdash_{\mathcal{D}} \{P \wedge b\} S \{Q\} \wedge [P \wedge \neg b \ \Rightarrow\ Q]$

$\Rightarrow$ $\because$ corrección de la regla (si_2) en la semántica de Dijkstra

$\{P\} if\ b\ then\ S \{Q\}$ TEOREMA

5.2. Completitud de $\mathcal{LH}$ para un lenguaje sin bucles

Veamos ahora el problema recíproco al de la corrección. Que el cálculo de Hoare es *completo* significa que cualquier triplete de Dijkstra puede ser inferido utilizando el Cálculo de Hoare.

TEOREMA 5.15 *$\mathcal{LH}$ es completa cra $\mathcal{LD}$. Es decir,* $\forall S : S \in \mathcal{P}rog$:

$$\forall P, R :: \vdash_{\mathcal{D}} \{P\} S \{R\} \quad \Rightarrow \quad \vdash_{\mathcal{H}} \{P\} S \{R\} \qquad (compl)$$

Demostración.– Teniendo en cuenta la regla (ref), bastará demostrar

$$\forall S : S \in \mathcal{P}rog : \forall R :: \vdash_{\mathcal{H}} \{S.R\} S \{R\} \qquad (*)$$

En efecto:

$$\dfrac{\dfrac{\vdash_{\mathcal{D}} \{P\} S \{R\}}{[P \ \Rightarrow\ S.R]}\ \text{definición} \qquad \dfrac{}{\{S.R\} S \{R\}}\ (*)}{\{P\} S \{R\}}\ (ref)$$

La técnica que usaremos para probar $(*)$ es muy diferente a la utilizada en la demostración del Teorema 5.14; utilizaremos el esquema de *inducción sobre la estructura* de la sentencia S descrito en la Sección 4.3; recordemos que para demostrar que las sentencias del lenguaje simple propuesto (sin bucles y con selecciones binarias) verifican cierta propiedad $\boxed{p}$ hay que demostrar:

(cb) Casos base: p es cierta para la sentencias elementales: $p.nada$, $p.(x := E)$.

(pi) Paso inductivo, que consta de las siguientes implicaciones

$$\begin{array}{lcll} p.S \wedge p.T & \Rightarrow & p.(S;T) & \text{— composición} \\ p.S \wedge p.T & \Rightarrow & p.[\![\, b \rightarrow S \,\square\, \neg b \rightarrow T \,]\!] & \text{— selección binaria} \end{array}$$

Probemos pues $(*)$ por inducción sobre la sentencia, utilizando el esquema antes descrito, y tomando como propiedad

$$p.S \ \doteq\ \forall R :: \vdash_{\mathcal{H}} \{S.R\} S \{R\}$$

– CASOS BASE. Hay que demostrar, $p.nada$ y $p.x := E$; es decir

$$\forall R :: \vdash_{\mathcal{H}} \{nada.R\} S \{R\}$$
$$\forall R :: \vdash_{\mathcal{H}} \{x := E.R\} S \{R\}$$

lo cual es evidente teniendo en cuenta la semántica de $nada$ y las reglas $(nada)$ y $(:=)$ del cálculo de Hoare.

– PASO INDUCTIVO 1.– COMPOSICIÓN. Tenemos

$$p.(S;T) \equiv \forall R \ :: \vdash_{\mathcal{H}} \{S;T.R\} S;T \{R\}$$

Sea un predicado arbitrario R; entonces

$\vdash_{\mathcal{H}} \{S;T.R\} S;T \{R\}$
$=$ $\quad \because$ semántica composición
$\vdash_{\mathcal{H}} \{S.(T.R)\} S;T \{R\}$
$\Leftarrow$ $\quad \because$ regla de la composición
$\vdash_{\mathcal{H}} \{S.(T.R)\} S \{T.R\} \ \wedge \ \vdash_{\mathcal{H}} \{T.R\} T \{R\}$
$\Leftarrow$ $\quad \because$ HI: por $p.S$, tomando $R == T.R$
$\vdash_{\mathcal{H}} \{T.R\} T \{R\}$
$\Leftarrow$ $\quad \because$ HI: por $p.T$
$Cierto$

– PASO INDUCTIVO 2.– SELECCIÓN BINARIA: $SI \doteq [\![b \to S \,\square\, \neg b \to T]\!]$ (para la sentencia $if\ b\ then\ S$ el razonamiento es similar),

$\vdash_{\mathcal{H}} \{SI.R\} SI \{R\}$
$\Leftarrow$ $\quad \because$ regla (si_1)
$\vdash_{\mathcal{H}} \{b \wedge SI.R\} S \{R\} \ \wedge \ \vdash_{\mathcal{H}} \{\neg b \wedge SI.R\} T \{R\}$
$\Leftarrow$ $\quad \because$ regla refinamiento, junto a la HI dos veces
$[b \wedge SI.R \ \Rightarrow \ S.R] \wedge [\neg b \wedge SI.R \ \Rightarrow \ T.R]$
$=$ $\quad \because$ CP
$[SI.R \ \Rightarrow \ (b \ \Rightarrow \ S.R)] \wedge [SI.R \ \Rightarrow \ (\neg b \ \Rightarrow \ T.R)]$
$=$ $\quad \because [\,]$ conjuntivo, cp
$[SI.R \ \Rightarrow \ (b \ \Rightarrow \ S.R) \wedge (\neg b \ \Rightarrow \ T.R)]$
$\Leftarrow$ $\quad \because$ semántica de la selectiva
$[SI.R \ \Rightarrow \ SI.R]$ TEOREMA

5.3. Un teorema fundamental para la selectiva

En ocasiones, ni siquiera los teoremas de corrección y completitud son suficientemente prácticos. Sin embargo, el siguiente teorema proporcionará reglas prácticas para inferir tripletes en presencia de sentencias selectivas.

TEOREMA 5.16 (Teorema fundamental para la sentencia selectiva) *Se verifica la siguiente equivalencia para cualquier sentencia selectiva*

$$\begin{array}{rl} & [P \;\Rightarrow\; OB] \wedge \forall j : 1 \leq j \leq n : [P \wedge b_j \;\Rightarrow\; S_j.R] \\ \equiv & \\ & [P \;\Rightarrow\; SI.R] \end{array}$$

donde SI es la selectiva $[\![\; \square : 1 \leq j \leq n : b_j \rightarrow S_j \;]\!]$.

Demostración.– Tenemos

$$\begin{array}{ll} & [P \;\Rightarrow\; SI.R] \\ = & \quad \because \text{semántica} \\ & [P \;\Rightarrow\; OB \wedge (\forall j : 1 \leq j \leq n : b_j \;\Rightarrow\; S_j.R)] \\ = & \quad \because \text{cálculo de predicados, } [\,] \text{ es conjuntivo} \\ & [P \;\Rightarrow\; OB] \wedge [P \;\Rightarrow\; (\forall j : 1 \leq j \leq n : b_j \;\Rightarrow\; S_j.R)] \\ = & \quad \because \text{cálculo} \\ & [P \;\Rightarrow\; OB] \wedge \forall j : 1 \leq j \leq n : [P \;\Rightarrow\; (b_j \;\Rightarrow\; S_j.R)] \\ = & \quad \because \text{cálculo} \\ & [P \;\Rightarrow\; OB] \wedge \forall j : 1 \leq j \leq n : [P \wedge b_j \;\Rightarrow\; S_j.R] \end{array}$$

TEOREMA

El teorema tiene aplicaciones prácticas en las que encontrar la precondición más débil es difícil y nos bastaría con encontrar precondiciones P tales que

$$[P \;\Rightarrow\; SI.R]$$

Según el teorema, no es necesario en estos casos calcular el transformador SI y nos basta con asegurar las implicaciones para cada sentencia guardada. Veamos dos aplicaciones del teorema anterior.

Demostraciones comentadas

Si tomamos $[OB \equiv Cierto]$, entonces, de la corrección del esquema:

$$\begin{array}{lll} [\![& b_1 \rightarrow \{P \wedge b_1\} S_1 \{Q_1\} \\ & \cdots \\ \square & b_n \rightarrow \{P \wedge b_n\} S_n \{Q_n\} &]\!] \end{array}$$

junto a $\forall i :: [Q_i \;\Rightarrow\; R]$, aplicando refinamiento y el Teorema 5.14, se tiene $[P \;\Rightarrow\; SI.R]$. En particular, para la selectiva binaria, necesitamos los tripletes del antecedente de la regla (si_1); obsérvese que esto último podría haberse aplicado en el paso $(pi3)$ de la demostración del Teorema 5.14.

Esta idea proporciona una técnica: añadir comentarios o predicados intermedios alrededor de las sentencias que envuelven las sentencias selectivas (y los bucles, como más tarde veremos). Comencemos con un ejemplo simple.

EJEMPLO 5.17 Tratemos de probar la corrección del programa

$$\begin{array}{llllll} [\![& x > y & \rightarrow & [\![& z > x & \rightarrow m := z \\ & & & \square & z \leq x & \rightarrow m := x \;]\!] \\ \square & x \leq y & \rightarrow & [\![& z > y & \rightarrow m := z \\ & & & \square & z \leq y & \rightarrow m := y \;]\!] \;]\!] \end{array}$$

$$\{m = \text{máx}(x, y, z)\}$$

que es consecuencia de la validez de

$$\begin{array}{llllll} [\![& x > y & \rightarrow & \{x > y\} & S_1 & \{m = \text{máx}(x,y,z)\} \\ \square & x \leq y & \rightarrow & \{x \leq y\} & S_2 & \{m = \text{máx}(x,y,z)\} \end{array}$$

Para probar los tripletes que envuelven a S_1 y S_2, podemos de nuevo aplicar el mismo razonamiento, y bastará probar los tripletes:

$$\begin{array}{lll} \{x > y \wedge z > x\} & m := z & \{m = \text{máx}(x,y,z)\} \\ \{x > y \wedge z \leq x\} & m := x & \{m = \text{máx}(x,y,z)\} \\ \{x \leq y \wedge z > y\} & m := z & \{m = \text{máx}(x,y,z)\} \\ \{x \leq y \wedge z \leq y\} & m := y & \{m = \text{máx}(x,y,z)\} \end{array}$$

Los cuatro tripletes se prueban utilizando la regla de refinamiento y la regla de la asignación; veamos el primero:

$$\dfrac{\dfrac{}{[x > y \wedge z > x \;\Rightarrow\; z = \text{máx}(x,y,z)]}\,(\text{def. de máx}) \qquad \dfrac{}{\{z = \text{máx}(x,y,z)\}\; m := z\; \{m = \text{máx}(x,y,z)\}}\,(:=)}{\{x > y \wedge z > x\}\; m := z\; \{m = \text{máx}(x,y,z)\}}\,(ref)$$

Otra forma consiste en probar directamente las equivalencias

$$[x > y \;\Rightarrow\; S_1.(m = \text{máx}(x,y,z))] \qquad\qquad [x \leq y \;\Rightarrow\; S_2.(m = \text{máx}(x,y,z))]$$

Veamos por ejemplo la primera; tenemos, *ptle*:

$$\begin{array}{ll} & S_1.(m = \text{máx}(x,y,z)) \\ = & \quad \because \text{semántica selectiva binaria} \\ & z > x \wedge m := z.(m = \text{máx}(x,y,z)) \vee z \leq x \wedge m := x.(m = \text{máx}(x,y,z)) \\ = & \quad \because \text{semántica asignación} \\ & z > x \wedge z = \text{máx}(x,y,z) \vee z \leq x \wedge x = \text{máx}(x,y,z) \\ = & \quad \because\; z = \text{máx}(x,y,z) \equiv z \geq x, y, \ldots \\ & z > x \wedge z \geq x, y \vee z \leq x \wedge x \geq y, z \\ \Leftarrow & \quad \because \text{distributiva, transitividad de } \Rightarrow \\ & x > y \wedge (z > x \vee z \leq x) \\ = & \quad \because \text{tercio excluido} \\ & x > y \end{array}$$

EJEMPLO

NOTA 5.18 *La técnica utilizada en el ejemplo anterior consiste en la construcción de una* demostración comentada *(proof outline); esta es la técnica seguida en la teoría de Susan Owicki y David Gries formulada en 1976 [Apt, 1988].*

EJEMPLO 5.19 Vamos a probar la corrección del siguiente programa.

$$\begin{array}{l} x, y, z := a, b, c; \\ if\ x \leq y\ then\ x, y := y, x; \\ if\ x \leq z\ then\ x, z := z, x; \\ if\ y \leq z\ then\ y, z := z, y \\ \{x = \text{máx}(a,b,c) \wedge z = \text{mín}(a,b,c)\} \end{array}$$

Dada la simetría de las selecciones, rescribimos el programa en la forma

$$
\begin{array}{l}
x, y, z := a, b, c; \\
P(x, y); \\
P(x, z); \\
P(y, z); \\
\{x = \text{máx}(a, b, c) \wedge z = \text{mín}(a, b, c)\}
\end{array}
$$

donde $P(x, y) \doteq \mathit{if}\ x \leq y\ \mathit{then}\ x, y := y, x$. Veamos que es suficiente probar

$$\{x = a \wedge y = b \wedge z = c\} P(x, y) \{x = \text{máx}(a, b) \wedge y = \text{mín}(a, b) \wedge z = c\} \quad (*)$$

En efecto; basta aplicar la regla $(;)$ sucesivamente a los siguientes tripletes:

$\{Cierto\}$
$x, y, z := a, b, c$ — regla $(:=)$
$\{x = a \wedge y = b \wedge z = c\}$,

$\{x = a \wedge y = b \wedge z = c\}$
$P(x, y)$ — por $(*)$
$\{x = \text{máx}(a, b) \wedge y = \text{mín}(a, b) \wedge z = c\}$,

$\{x = \text{máx}(a, b) \wedge y = \text{mín}(a, b) \wedge z = c\}$
$P(x, z)$ — por $(*)$, con $a == \text{máx}(a, b), y == z, \ldots$
$\{x = \text{máx}(\text{máx}(a, b), c) \wedge y = \text{mín}(a, b) \wedge z = \text{mín}(\text{máx}(a, b), c)\}$,

$\{x = \text{máx}(a, b, c) \wedge y = \text{mín}(a, b) \wedge z = \text{mín}(\text{máx}(a, b), c)\}$
$P(y, z)$ — por $(*)$, idem
$\{x = \text{máx}(a, b, c) \wedge y = \ldots \wedge z = \text{mín}(\text{máx}(a, b), c, \text{mín}(a, b))\}$

La última expresión del último predicado puede simplificarse en la forma

$$
\begin{array}{ll}
 & \text{mín}(\text{máx}(a, b), c, \text{mín}(a, b)) \\
= & \quad \because \text{por ser } \text{máx}(a, b) \geq \text{mín}(a, b), \text{ eliminamos el primer término de la lista} \\
 & \text{mín}(c, \text{mín}(a, b)) \\
= & \\
 & \text{mín}(c, a, b)
\end{array}
$$

Finalmente, queda probar $(*)$; pero

$$
\begin{array}{ll}
 & \{x = a \wedge y = b \wedge z = c\} \\
 & P(x, y) \\
 & \{x = \text{máx}(a, b) \wedge y = \text{mín}(a, b) \wedge z = c\} \\
\Leftarrow & \quad \because \text{regla } (si_2) \\
 & \{x = a \wedge y = b \wedge z = c \wedge x \leq y\} \\
 & x, y := y, x \\
 \wedge & \{x = \text{máx}(a, b) \wedge y = \text{mín}(a, b) \wedge z = c\} \\
 & [x = a \wedge y = b \wedge z = c \wedge x > y \ \Rightarrow\ x = \text{máx}(a, b) \wedge y = \text{mín}(a, b) \wedge z = c] \\
= & \quad \because \text{semántica asignación, refinamiento y cálculo} \\
 & [x = a \wedge y = b \wedge z = c \wedge x \leq y \ \Rightarrow \\
 \wedge & y = \text{máx}(a, b) \wedge x = \text{mín}(a, b) \wedge z = c] \\
 & Cierto \\
= & \quad \because \text{cálculo} \\
 & Cierto
\end{array}
$$

EJEMPLO

Veremos otra forma de tratar el ejemplo utilizando el siguiente interesante resultado.

TEOREMA 5.20 *Sea P un predicado invariante para una colección inicial de sentencias $\mathcal{C}$. Es decir, $\forall S : S \in \mathcal{C} : \{P\}S\{P\}$. Entonces, P es invariante para cualquier sentencia compuesta escrita a partir de sentencias de la colección $\mathcal{C}$ con los operadores composición secuencial y selectiva binaria.*

Demostración.– Sea $\mathcal{C}'$ la colección de sentencias compuestas a partir de sentencias de la colección $\mathcal{C}$. $\mathcal{C}'$ es un conjunto inductivo. Hemos de probar:

$$\forall S : S \in \mathcal{C}' : \{P\}S\{P\}$$

y lo hacemos por inducción sobre la estructura de $\mathcal{C}'$. Los casos base corresponden a las sentencias S de la colección $\mathcal{C}$, que obviamente satisfacen $\{P\}S\{P\}$ por hipótesis. Los pasos inductivos se corresponden con las dos construcciones permitidas: composición y selección. Luego hay que probar:

$$\begin{array}{lcl} \{P\}S\{P\} \wedge \{P\}T\{P\} & \Rightarrow & \{P\}S;T\{P\} \\ \{P\}S\{P\} \wedge \{P\}T\{P\} & \Rightarrow & \{P\}if\ b\ then\ S\ else\ T\{P\} \end{array}$$

La primera implicación sigue trivialmente de la regla de la composición $(:=)$. La segunda sigue de la regla de refinamiento, y del Teorema 5.16. En efecto,

$\{P\}S\{P\} \wedge \{P\}T\{P\}$
$\Rightarrow$ $\quad \because$ refinamiento
$\{P \wedge b\}S\{P\} \wedge \{P \wedge \neg b\}T\{P\}$
$\Rightarrow$ $\quad \because$ Teorema 5.16
$\{P\}if\ b\ then\ S\ else\ T\{P\}$ TEOREMA

EJEMPLO 5.21 Volvamos a la corrección del esquema del Ejemplo 5.19. Es fácil demostrar que el predicado

$$P \doteq (x, y, z) \text{ es una permutación de (a,b,c)}$$

es invariante bajo todas las asignaciones del programa en cuestión. Entonces, por el Teorema 5.20, P es invariante para la composición de las selectivas, de donde tenemos la corrección del siguiente esquema:

$$\begin{array}{l} \{C\} \\ x, y, z := a, b, c; \{P\} \\ if\ x \leq y\ then\ x, y := y, x; \\ if\ x \leq z\ then\ x, z := z, x \\ \{P\} \end{array}$$

Pero las dos primeras sentencias calculan el máximo sobre la variable x (véase Ejemplo 5.5); entonces, por conjuntividad de los transformadores, tenemos

$$\begin{array}{l} \{C\}x, y, z := a, b, c; \{P\} \\ if\ x \leq y\ then\ x, y := y, x; \\ if\ x \leq z\ then\ x, z := z, x \\ \{P \wedge x = \text{máx}(a, b, c)\} \end{array}$$

y solamente habrá que demostrar el triplete

$$\{P \wedge x = \text{máx}(a,b,c)\} if\ y \leq z\ then\ y,z := z,y \{z = \text{mín}(a,b,c)\} \qquad (1)$$

ya que el predicado $x = \text{máx}(a,b,c)$ es invariante bajo la selectiva anterior. Pero la selectiva termina satisfaciendo el predicado $y \geq z$, de donde, por conjuntividad, también termina satisfaciendo la conjunción:

$$\{P \wedge x = \text{máx}(a,b,c)\} if\ y \leq z\ then\ y,z := z,y \{Q\}$$

donde $Q \doteq P \wedge x = \text{máx}(a,b,c) \wedge y \geq z$. Para obtener (1) solamente hemos de aplicar refinamiento a la implicación $[Q \Rightarrow z = \text{mín}(a,b,c)]$, que por otro lado es trivial. EJEMPLO

EJEMPLO 5.22 El siguiente es otro ejemplo de demostración comentada. Queremos probar el triplete

$$\{i \leq j\}\ \ if\ j < k\ then\ m := k\ else\ m := j\ \ \{i,j,k \leq m\}$$

que rescribimos en la forma habitual, dejando una serie de huecos que iremos rellenando:

$$\begin{array}{lll} \{i \leq j\} & [\![\quad j < k \rightarrow & m := k \\ & \square \quad j \geq k \rightarrow & m := j \\ \{i,j,k \leq m\} &]\!] & \end{array}$$

Si nos guiamos por el Teorema 5.16 tratamos de introducir comentarios alrededor de las sentencias guardadas:

$$\begin{array}{lll} \{i \leq j\} & [\![\quad j < k \rightarrow \{i \leq j \wedge j < k\} & m := k\ \{i,j,k \leq m\} \\ & \square \quad j \geq k \rightarrow \{i \leq j \wedge j \geq k\} & m := j\ \{i,j,k \leq m\} \\ \{i,j,k \leq m\} &]\!] & \end{array}$$

y solamente hemos de probar los dos tripletes de la derecha. Para ello insertamos los comentarios delante de la sentencia de asignación de forma que hagan cierto los últimos tripletes usando la regla de la sentencia de asignación:

$$\begin{array}{llll} \{i \leq j\} & [\![\quad j < k \rightarrow \{i \leq j \wedge j < k\} & \{i,j,k \leq k\} & m := k\ \{i,j,k \leq m\} \\ & \square \quad j \geq k \rightarrow \{i \leq j \wedge j \geq k\} & \{i,j,k \leq j\} & m := j\ \{i,j,k \leq m\} \\ \{i,j,k \leq m\} &]\!] & & \end{array}$$

Todos los tripletes serían correctos si logramos probar las implicaciones intermedias:

$$\begin{array}{ll} i \leq j \wedge j < k & \Rightarrow i,j,k \leq k \\ i \leq j \wedge j \geq k & \Rightarrow i,j,k \leq j \end{array}$$

que por otro lado son triviales. EJEMPLO

Ejercicios

5.23 *[260] Sea la siguiente semántica informal para el operador $\odot$:*

$$S \odot T \equiv \textit{ejecutar } S \textit{ o } T \textit{ (elegida al azar)}$$

(A) Dad la semántica del operador $\odot$ con una regla al estilo de Hoare.

(B) Dad la semántica de $\odot$ con un transformador de predicados.

(C) Dad un ejemplo donde $S \odot T$ sea determinista y otro donde sea indeterminista.

(D) Implementad el operador $\odot$ en el lenguaje de Dijkstra.

5.24 *[260] Sea el lenguaje* $S ::= x := E \mid S; S' \mid [\![b \to S \,\square\, b' \to S']\!]$.

(A) Dad las reglas de un cálculo de Hoare si la selectiva debe tener un comportamiento indeterminista.

(B) En la $\mathcal{LH}$ del apartado anterior, estudiad si, para cualquier predicado, siempre es derivable $\vdash_{\mathcal{H}} \{P\}S\{Cierto\}$.

(C) En el sistema de Dijkstra, sea la definición $\vdash_{\mathcal{D}} \{X\}S\{Y\} \doteq [X \;\Rightarrow\; S.Y]$. ¿Qué significa que la $\mathcal{LH}$ anterior sea completa para la semántica de Dijkstra?

(D) Probad que es completa.

(E) ¿Qué significa que el sistema $\vdash_{\mathcal{H}}$ sea correcto para la semántica de Dijkstra?

(F) Probad que es correcto.

5.25 *[262]* **(Febrero, 96)**

(A) Demostrad la corrección de la siguiente regla con tripletes à la Dijkstra:

$$\frac{[P \;\Rightarrow\; b] \qquad \{P\}S\{Q\} \qquad \{P\}T\{Q\}}{\{P\}[\![b \to S \,\square\, b \to T]\!]\{Q\}}$$

(B) Probad, siendo $W = [\![x > 0 \to x := 2 \,\square\, x > 0 \to x := 4]\!]$,

$$\begin{array}{lcl} \{x > 1\}W\{x = 2\} & \equiv & Falso \\ \{x > 1\}W\{x = 2 \vee x = 4\} & \equiv & Cierto \end{array}$$

y deducid que W es indeterminista.

(C) ¿Se puede utilizar la regla del apartado (A) para probar los tripletes del (B)?

5.26 *[263]* **(Febrero, 96)** *Demostrad la siguiente regla de derivación con tripletes de Dijkstra*

$$\frac{\{P \wedge b\}S_1\{Q\} \qquad \{P \wedge c\}S_2\{Q\} \qquad [P \wedge \neg c \;\Rightarrow\; Q]}{\{P\}if\ b\ then\ S_1\ else\ if\ c\ then\ S_2\{Q\}}$$

5.27 *[263]* **(Enero, 91)** *Se considera el cálculo de Hoare estándar; dad una definición de equivalencia $=_{\mathcal{H}}$ entre programas y probad $\forall \mathcal{S} : \mathcal{S} \in \mathcal{P}rog : \mathcal{S}; nada =_{\mathcal{H}} \mathcal{S}$.*

5.28 **(Febrero, 02)** *Sea el lenguaje $S ::= aborta | x := E | [\![b \to S \,\square\, b' \to S']\!]$. Consideremos el modelo de Hoare formado las reglas estándar (ref) y $(:=)$, a las que añadimos las dos reglas siguientes:*

$$\frac{}{\{Falso\}aborta\{Q\}}\,(aborta) \qquad \frac{[b \vee b'] \qquad \{b\}S\{Q\} \qquad \{b'\}S'\{Q\}}{\{b \vee b'\}[\![b \to S \,\square\, b' \to S']\!]\{Q\}}\,(si)$$

(A) ¿Cuándo se dice que una sentencia S es indeterminista según cierto modelo de tripletes de Hoare?.

(B) ¿$aborta$ es indeterminista?

(C) Prueba, describiendo la técnica que utilizas, alguna de las siguientes propiedades:

$(C1)$ *El modelo de tripletes anterior es correcto* cra *la semántica de Dijkstra.*

$(C2)$ *Si $\{X\}$aborta $\{Q\}$, entonces $[X \equiv Falso]$.*

5.29 *[264]* **(Febrero, 01)** *Consideremos el cálculo de Hoare estándar. Demuestra que si* $\vdash_{\mathcal{H}} \{P\}nada\{Q\}$*, entonces* $[P \Rightarrow Q]$*, indicando la técnica que utilizas para ello.*

5.30 *[264]* **(Diciembre, 94)** *Para cada sentencia sana* $\mathcal{S}$*, y predicados* X *e* Y*, se definen los tripletes de Dijkstra en la forma* $\{X\}S\{Y\} \doteq [X \Rightarrow S.Y]$*.*

(A) Calculad (e interpretad) con tal definición los tripletes

$$\begin{array}{ll} \{Cierto\}S\{Falso\} & \{Falso\}S\{Cierto\} \\ \{Cierto\}x := x+1\{Cierto\} & \{Falso\}S\{Falso\} \end{array}$$

(B) Dad un ejemplo con $\mathcal{S}$ *indeterminista verificando* $\{Cierto\}\mathcal{S}\{x=1\}$*.*

(C) Deducid la siguiente regla de derivación para triples de Dijkstra:

$$\frac{[P \Rightarrow R] \quad [Q \Rightarrow R] \quad [P \Rightarrow b] \quad \{P\}S\{Q\} \quad \{P\}T\{Q\}}{\{P\}[\![\, b \to S \,\square\, b \to T \,\square\, b \to nada \,]\!]\{R\}}$$

5.31 *[264]* **(Julio, 94)** *Demostrad la corrección de la siguiente regla con tripletes à la Dijkstra:*

$$\frac{[P \Rightarrow b \wedge c] \quad \{P\}S\{Q\} \quad \{P\}T\{Q\}}{\{P\}[\![\, b \to S \,\square\, c \to T \,]\!]\{Q\}}$$

5.32 *[265]* **(Diciembre, 94)** *Probad la siguiente regla de derivación con triples de Dijkstra:*

$$\frac{\{X\}S;A\{Y\} \quad \{X\}S;B\{Y\}}{\{X\}S;[\![\, b \to A \,\square\, \neg b \to B \,]\!]\{Y\}}$$

5.33 **(Febrero, 91)**

(A) ¿Qué diferencias importantes existen entre el estilo axiomático de Hoare y el de Dijkstra?

(B) ¿Qué ventajas tiene la definición semántica de Dijkstra frente a la de Hoare?

(C) Demostrad, en las dos semánticas, la corrección del siguiente programa (todas las variables son enteras):

$$\begin{array}{l} [\![\, z > x \to m := z \,\square\, z \leq x \to m := x \,]\!] \\ \{m = \text{máx}(x,z)\} \end{array}$$

5.34 *[265]* **(Febrero, 91)** *Probad que las sentencias* SI *y* SI' *siguientes son equivalentes:*

$$\begin{array}{lcl} SI & \doteq & [\![\, b_1 \to S_1 \,\square\, b_2 \to S_2 \,\square\, b_3 \to S_3 \,]\!] \\ SI' & \doteq & [\![\, b_1 \vee b_2 \to [\![\, b_1 \to S_1 \,\square\, b_2 \to S_2 \,]\!] \,\square\, b_3 \to S_3 \,]\!] \end{array}$$

5.35 *[265]* **(Febrero, 99)** *Enunciad alguna propiedad para el cálculo de Hoare que tenga la forma* $\forall S, ?, ? ::\vdash_{\mathcal{H}} \{?\}S\{?\}$*, y demostradla por inducción sobre la sentencia.*

5.36 *[265] Consideremos el cálculo de Hoare estándar, salvo que permitimos selectivas indeterministas y la siguiente regla:*

$$\frac{[X \Rightarrow b \vee b'] \quad \{b \wedge X\}S\{Y\} \quad \{b' \wedge X\}S'\{Y\}}{\{X\}[\![\, b \to S \,\square\, b' \to S' \,]\!]\{Y\}}\,(si)$$

Probad que entonces se verifica:

$$\vdash_{\mathcal{H}} \{P\}[\![\, b \to S \,\square\, b' \to S' \,]\!]\{Q\} \quad \Rightarrow \quad [P \Rightarrow b \vee b']$$

Capítulo 6

La sentencia de iteración o bucle

6.0. Transformador asociado a un bucle

Veamos ahora otra construcción a partir de un conjunto de sentencias con guardas; se trata de la sentencia bucle:

$$\begin{array}{lll} repite & & b_1 \to S_1 \\ & & \dots \\ & \square & b_n \to S_n \\ fin\_repite & & \end{array}$$

El significado operacional es el siguiente:

(a) Se evalúan todas las guardas.

(b) Entre las verdaderas se selecciona una en forma indeterminista, se ejecuta la correspondiente secuencia de sentencias, y volvemos al paso (a).

(b') Si todas las guardas son falsas termina la sentencia.

Denotaremos esta última sentencia con $\mathcal{R}$. Dos observaciones:

(1) Si todas las guardas son falsas entonces $\mathcal{R}$ equivale a $nada$.

(2) Al final del bucle todas las guardas deben ser falsas.

Estas dos afirmaciones deberán deducirse del transformador de $\mathcal{R}$.

Al igual que la sentencia selectiva, la sentencia $\mathcal{R}$ introduce una abstracción adicional (además del mecanismo de repetición): el *indeterminismo*. Dicha sentencia está implementada en el lenguaje CSP pero no en ADA; en éste lenguaje debe simularse.

NOTACIÓN 6.0 La notación original de Dijkstra es:

$$do\ b_1 \to S_1 \square \dots \square b_n \to S_n\ od$$

y la del lenguaje CSP es: $*[\![\, b_1 \to S_1 \square \dots \square b_n \to S_n \,]\!]$, que será la más utilizada; también utilizaremos las notaciones:

$$*[\![\ \square\ : 1 \le i \le n : b_i \to S_i \,]\!] \qquad *[\![\ \square_{1 \le i \le n} b_i \to S_i \,]\!] \qquad \boxed{\text{NOTACIÓN}}$$

EJEMPLO 6.1 (Un bucle indeterminista para el juego de Dijkstra) Tratemos de construir un programa para el siguiente *juego*, debido a [Dijkstra, 1981] (véase la Figura 6.0):

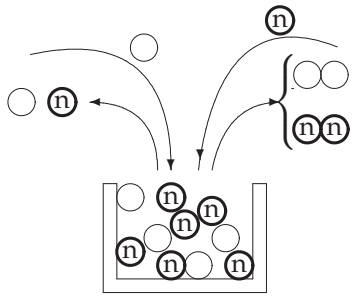


Figura 6.0: La urna de Dijkstra.

> Una urna contiene bolas blancas y negras; se extraen dos bolas, añadiéndose posteriormente una negra si éstas eran del mismo color o una blanca si eran de colores diferentes; este paso se repite cuantas veces sea posible.

Puesto que en cada paso decrece en una unidad el total de bolas (se extraen dos y se añade una) en un número finito de pasos quedará una sola bola; la pregunta es

¿de qué color es la última bola?

Para dar una respuesta al acertijo escribiremos un programa que simule el juego y estudiaremos sus propiedades. Usaremos dos variables enteras b y n que memoricen el estado del juego: número de bolas de cada color. Basta considerar una secuencia de inicialización para representar el estado inicial de la urna, y un bucle indeterminista (el juego) controlado por las variables b y n. Cada secuencia guardada del bucle deberá representar una de las acciones a realizar; su guarda capturará la posibilidad de realizarla. Es fácil ver que el bucle del siguiente programa *simula* el juego

$$
\begin{array}{l}
b, n :\in \mathbb{Z}; \\
b, n := 35, 14; \\
\qquad \begin{array}{lll}
\text{— guarda} & & \text{— extraer} \qquad\qquad\quad \text{— devolver} \\
*[\![\quad b > 1 & \rightarrow & b := b - 2; \qquad\quad n := n + 1 \\
\qquad n > 1 & \rightarrow & n := n - 2; \qquad\quad n := n + 1 \\
\qquad n > 0 \wedge b > 0 & \rightarrow & b, n := b - 1, n - 1; \quad b := b + 1 \,]\!]
\end{array}
\end{array}
$$

Obsérvese que la primera secuencia guardada modela la situación:

> ... si existen al menos dos bolas blancas, puedo extraer dos de ellas, y en ese caso añadiré una negra.

La tercera secuencia corresponde a extraer dos bolas de distinto color, y añadir una blanca. Por el Lema 4.7 (de sustitución) el bucle se simplifica como:

$$
\begin{array}{lll}
*[\![\quad b > 1 & \rightarrow & b, n := b - 2, n + 1 \\
\qquad n > 1 & \rightarrow & n := n - 1 \\
\qquad n > 0 \wedge b > 0 & \rightarrow & n := n - 1 \,]\!]
\end{array}
$$

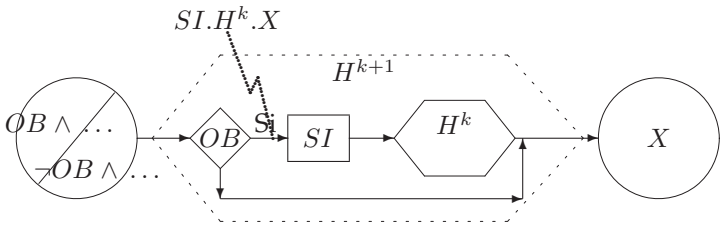


Figura 6.1: El transformador H^{k+1}.

A la vista de la sentencias guardadas vemos que cada acción, (1) disminuye en una unidad el total de bolas, y (2) *conserva la paridad de b*; de aquí conjeturamos: por (1) el bucle debe terminar, y por (2), termina con una única bola blanca, puesto que al principio hay un número impar de bolas blancas. En el resto de este capítulo *demostraremos* con rigor estas dos afirmaciones (termina y lo hace conservando la paridad de b). EJEMPLO

Semántica del bucle Tratemos la definición semántica de nuestra sentencia bucle $\mathcal{R} \doteq *[\![\Box : 1 \leq i \leq n : b_i \rightarrow S_i]\!]$. Es decir, $\mathcal{R}.X$. Ello lo haremos de dos formas: una inductiva y otra en términos de puntos fijos; la primera forma es la original de [Dijkstra, 1976]; la segunda la abordaremos en otro capítulo.

En primer lugar observamos que la sentencia $\mathcal{R}$ equivale a la ejecución secuencial de sentencias SI:

$$SI \doteq [\![\Box : 1 \leq i \leq n : b_i \rightarrow S_i]\!]$$

El principal problema es que no sabemos cuantas veces se va a repetir el cuerpo del bucle (la sentencia SI), y es más, éste número dependerá del estado inicial, e incluso podrá ser infinito. Si por ejemplo fueran n repeticiones, bastaría con ejecutar en forma secuencial n sentencias SI y la semántica de $\mathcal{R}$ sería simple.

Si $H^k.X$ representa la precondición más débil para que el cuerpo del bucle se ejecute a lo sumo k veces terminando verificando X, tenemos la definición:

$$[\mathcal{R}.X \doteq (\exists k : k \geq 0 : H^k.X)]$$

Para asegurar que la sentencia SI se ejecuta 0 veces (es decir ninguna) debemos asegurar que todas las guardas son falsas; es decir, $\neg OB$. Pero en ese caso debe darse la poscondición X; por tanto $[H^0.X \doteq \neg OB \wedge X]$. Por otro lado, para $k \geq 0$ tendremos:

$$[H^{k+1}.X \doteq H^0.X \vee SI.H^k.X]$$

ya que si alguna guarda es cierta y el cuerpo del bucle se ejecuta a lo sumo $k+1$ veces teniéndose al final X, o bien se tiene $H^0.X$ (se ejecuta 0 veces) o bien OB es cierta y en ese caso, después de ejecutar SI hay que asegurar que se ejecuta el bucle a lo sumo k veces (véase la Figura 6.1). Por consiguiente:

DEFINICIÓN 6.2 *Definimos la semántica del bucle en la forma*

$$[\mathcal{R}.X \doteq (\exists k : k \geq 0 : H^k.X)]$$

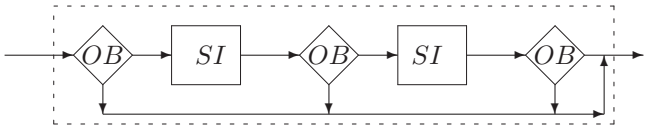


Figura 6.2: El transformador H^2.

donde

$$\begin{array}{rll} & [H^0.X & \doteq\ X \wedge \neg OB] \\ \forall k : k \geq 0 : & [H^{k+1}.X & \doteq\ H^0.X \vee SI.H^k.X] \end{array}$$

Obsérvese que cada H^k se corresponde con una secuenciación adecuada de selectivas. Por ejemplo, H^2 sería la sentencia de la Figura 6.2.

Los siguientes ejemplos y ejercicios muestran cómo utilizar la semántica inductiva de los bucles junto a mecanismo de inducción para probar las propiedades de algunos programas simples.

EJEMPLO 6.3 Si las guardas no se evalúan a falso en ningún paso del bucle, entonces el bucle no termina; por ejemplo, el bucle $*[\![\, b \to S \,\square\, \neg b \to T \,]\!]$ equivale a la sentencia $aborta$. Hay que probar $[OB \equiv C] \ \Rightarrow\ [\mathcal{R}.X \equiv Falso]$. Para ello, por la Definición 6.2, basta probar que se verifica $\forall k : k \geq 0 : [H^k.X \equiv F]$. Esto último lo demostramos por inducción; tenemos, *ptle*:

CASO BASE

$H^0.X$
$=$ $\quad \because$ definición
$\neg OB \wedge X$
$=$ $\quad \because [OB \equiv C]$
$Falso$

PASO INDUCTIVO

$H^k.X$
$=$ $\quad \because$ definición
$H^0.X \vee SI.H^{k-1}.X$
$=$ $\quad \because$ HI
$F \vee SI.F$
$=$ $\quad \because SI$ es estricta
F

EJEMPLO

6.4 *[265] Sea el bucle $\mathcal{R} \doteq\ *[\![\, b \to S \,]\!]$, de forma que $[S.\neg b \equiv Falso]$. Utilizando la Definición 6.2 probad que $[\mathcal{R}.X \equiv \neg b \wedge X]$.*

6.5 *[265] Probad (e interpretad) que si S es un programa que no asigna ninguna variable de Q, entonces, para cualquier predicado X, y $ptle$:*

$$Q \wedge S.X \equiv Q \wedge S.(Q \wedge X).$$

EJEMPLO 6.6 Consideremos el bucle $\mathcal{R} \doteq\ *[\![\, i < N \to i := i + 1 \,]\!]$. Demostremos utilizando la Definición 6.2 los siguientes resultados (plausibles), *ptle*

(a) $\mathcal{R}.C \equiv C$ (el bucle siempre termina).

(b) $\mathcal{R}.(i = N) \equiv (i \leq N)$.

(c) Por el contrario, $*[\![\, i \neq N \to i := i + 1 \,]\!].C \equiv i \leq N$.

La interpretación de (a) es que incrementando la variable i en algún momento se dejará de cumplir la guarda del bucle. La interpretación de (b) es que para obtener $i = N$ es necesario y suficiente partir de un estado verificando $i \leq N$, ya que de lo contrario podemos incrementar indefinidamente la variable i sin llegar a encontrar el valor $i = N$. Y la interpretación de (c) es que para que el bucle termine es necesario (y suficiente) que el estado inicial verifique $i \leq N$, ya que, para un estado inicial verificado $i > N$, incrementando la variable nunca podremos obtener el valor N.

Para probar (a), por la Definición 6.2, $[\mathcal{R}.C \doteq \exists n : n \geq 0 : H^n.C]$, y demostraremos por inducción que $\forall n : n \geq 0 : [H^n.C \equiv i + n \geq N]$

CASO BASE; *ptle*,

$$\begin{array}{ll} & H^0.C \\ = & \quad \because \text{definición} \\ & i \geq N \wedge C \\ = & \\ & i \geq N \end{array}$$

PASO INDUCTIVO; *ptle*, y para $n \geq 0$:

$$\begin{array}{ll} & H^{n+1}.C \\ = & \quad \because \text{definición} \\ & i \geq N \vee i < N \wedge [\![i < N \rightarrow i := i+1]\!].H^n.C \\ = & \quad \because [\![b \rightarrow S]\!].X \equiv b \wedge S.X \\ & i \geq N \vee i < N \wedge i := i+1.H^n.C \\ = & \quad \because \text{HI} \\ & i \geq N \vee i < N \wedge i + 1 + n \geq N \\ = & \\ & i + 1 + n \geq N \end{array}$$

y por tanto: $\mathcal{R}.C \equiv (\exists n : n \geq 0 : i + n \geq N) \equiv Cierto$. Para probar (b) probaremos $\forall n : n \geq 0 : [H^n.(i = N) \equiv N - n \leq i \leq N]$ por inducción sobre n. El caso base es trivial, siendo el paso inductivo, *ptle*,

$$\begin{array}{ll} & H^{n+1}.(i = N) \\ = & \quad \because \text{por definición} \\ & i = N \vee i < N \wedge i := i + 1.H^n.(i = N) \\ = & \quad \because \text{HI} \\ & i = N \vee i < N \wedge N - n \leq i + 1 \leq N \\ = & \\ & N - n - 1 \leq i \leq N \end{array}$$

y por tanto: $[\mathcal{R}.(i = N) \equiv (\exists n : n \geq 0 : N - n \leq i \leq N) \equiv i \leq N]$.
Para probar (c) basta probar por inducción

$$\forall n : n \geq 0 : [H^n.C \equiv N - n \leq i \leq N]$$

De nuevo el caso base es trivial, y el paso inductivo será:

$$\begin{array}{ll} & H^{n+1}.C \\ = & \quad \because \text{definición e HI} \\ & i = N \vee i \neq N \wedge N - n \leq i + 1 \leq N \\ = & \\ & N - n - 1 \leq i \leq N \end{array}$$

EJEMPLO

6.7 *[266]* **(Marzo, 94)** *Demostrad que la precondición más débil que asegura que el bucle*

$$*[\![x > 0 \rightarrow x := x - 1 \,\square\, x > 1 \rightarrow x := x - 2]\!]$$

se ejecute a lo sumo una vez es $x \leq 1$.

EJEMPLO 6.8 El siguiente ejemplo muestra una vez más cómo se utiliza directamente la definición semántica de los bucles para el análisis de éstos. Estudiemos pues el siguiente bucle

$$\mathcal{R} \doteq *[\![b \rightarrow x := x + 1 \,\square\, b \rightarrow b := Falso]\!]$$

donde x es entera y b booleana. Es plausible afirmar que el programa

$$x := 0; b := Cierto; \mathcal{R}$$

calcula un número natural arbitrario. Ya que el programa anterior está descrito con el lenguaje de Dijkstra es también plausible afirmar que no es posible asegurar la terminación del bucle en el entorno de la guarda. En efecto: es fácil probar por inducción que se verifica

$$\forall k : k \geq 0 : [H^k.C \equiv \neg b]$$

Para $k = 0$ es trivial; el paso inductivo es, *ptle*:

$$\begin{array}{ll} & H^{k+1}.C \\ = & \quad \because \text{definición} \\ & \neg b \vee [\![\, b \to x := x + 1 \,\square\, b \to b := Falso \,]\!].H^k.C \\ = & \quad \because \text{hipótesis inducción y semántica selectiva} \\ & \neg b \vee b \wedge (b \;\Rightarrow\; x := x + 1.\neg b) \wedge (b \;\Rightarrow\; b := Falso.\neg b) \\ = & \quad \because [b \wedge (b \;\Rightarrow\; Z) \equiv b \wedge Z] \\ & \neg b \vee b \wedge x := x + 1.\neg b \wedge b := Falso.\neg b \\ = & \quad \because \text{semántica asignación} \\ & \neg b \vee b \wedge \neg b \wedge \neg Falso \\ = & \quad \because \text{CP} \\ & \neg b \end{array}$$

Luego, $[\mathcal{R}.C \equiv \neg b]$, y por ello $x := 0; b := Cierto; \mathcal{R}$ equivale a $aborta$. Obsérvese (hágase como ejercicio) que el razonamiento anterior es válido si reemplazamos la sentencia $b := Falso$ por cualquier sentencia. EJEMPLO

6.9 *[267]* **(Julio, 94)** *Utilizando la semántica inductiva de los bucles, probad la terminación del bucle*

$$x :\in \mathbb{Z};\;\; *[\![\, b \to x, b := x + 1, x < 1 \,\square\, b \to b := Falso \,]\!]$$

6.1. Teoremas esenciales para los bucles

Una propiedad esencial es que un bucle con n guardas es equivalente a un bucle con una sola guarda pero sustituyendo el cuerpo por una sentencia selectiva. Ello es consecuencia del siguiente teorema fundamental

TEOREMA 6.10 *Los siguientes bucles $\mathcal{R}$ y $\mathcal{R}'$ son (semánticamente) equivalentes*

$$\begin{array}{lcl} \mathcal{R} & \doteq & *[\![\, \square\, i : 1 \leq i \leq n : b_i \to S_i \,]\!] \\ \mathcal{R}' & \doteq & *[\![\, OB \to SI \,]\!] \end{array}$$

donde $SI \doteq [\![\, \square\, i : 1 \leq i \leq n : b_i \to S_i \,]\!]$.

OBSERVACIÓN.– Del teorema podemos obtener consecuencias importantes. Aunque los bucles sean semánticamente equivalentes, no lo son operacionalmente: $\mathcal{R}'$ necesita una doble evaluación de las guardas; aún así, el teorema es útil desde el siguiente punto de vista teórico:

- ✓ Hubiera sido suficiente definir $\mathcal{R}.X$ para bucles con una sola guarda,

y desde un punto de vista práctico:

✓ Se puede *implementar* el indeterminismo de los bucles a través del indeterminismo de la sentencia selectiva.

Por ejemplo, en ADA se utilizará el bucle

$$
\begin{array}{l}
\mathit{while}\ OB\ \mathit{loop} \\
\quad \mathit{select} \\
\quad\quad \mathit{when}\ b_1 \Rightarrow \ldots \\
\quad\quad \ldots \\
\quad \mathit{end\ select}; \\
\mathit{end\ loop};
\end{array}
$$

ya que en ADA no existe la construcción bucle indeterminista. OBS

NOTA *6.11* *Señalemos una serie de aplicaciones importantes del teorema, como las del siguiente esquema general:*

> *La prueba de una propiedad* $\boxed{P}$ *para los bucles se reduce a la prueba de la misma propiedad para bucles con una sola guarda,*

que es particularmente interesante cuando P *es alguna de las siguientes propiedades:*

✓ *Salubridad: Teorema* **??**.

✓ *Determinismo: Teorema 6.18.*

✓ *Negación de guardas: Teorema 6.12*(i).

✓ *Teorema de invariantes: Teorema 6.32.* OBS

Demostración.– Apliquemos la Definición 6.2: *ptle*

$$
\begin{array}{lll}
\mathcal{R}.X & \doteq & (\exists n : n \geq 0 : H^n.X), \text{ donde,} \\
H^0.X & \doteq & \neg OB \wedge X, \\
H^{n+1}.X & \doteq & H^0.X \vee SI.H^n.X,
\end{array}
$$

$$
\begin{array}{lll}
\mathcal{R}'.X & \doteq & (\exists n : n \geq 0 : J^n.X, \text{ donde,} \\
J^0.X & \doteq & \neg OB \wedge X, \\
J^{n+1}.X & \doteq & J^0.X \vee [\![OB \rightarrow SI]\!].J^n.X.
\end{array}
$$

Bastará probar por inducción, $\forall n : n \geq 0 : [H^n.X \equiv J^n.X]$. Para $n = 0$ es trivial, mientras el paso inductivo es, *ptle*

$$
\begin{array}{ll}
 & J^{n+1}.X \\
= & \quad \cdot\!:\cdot \text{ definición} \\
 & J^0.X \vee [\![OB \rightarrow SI]\!].J^n.X \\
= & \quad \cdot\!:\cdot \text{ definición de } J^0 \text{ y semántica de selección} \\
 & \neg OB \wedge X \vee OB \wedge (OB \ \Rightarrow\ SI.J^n.X) \\
= & \quad \cdot\!:\cdot \text{ CP} \\
 & \neg OB \wedge X \vee OB \wedge SI.J^n.X \\
= & \quad \cdot\!:\cdot \text{ idempotencia, ya que } SI.X = OB \wedge \ldots \\
 & \neg OB \wedge X \vee SI.J^n.X \\
= & \quad \cdot\!:\cdot \text{ HI} \\
 & \neg OB \wedge X \vee SI.H^n.X \\
= & \quad \cdot\!:\cdot \text{ definición} \\
 & H^{n+1}.X
\end{array}
$$

TEOREMA

El siguiente teorema afirma que al final de un bucle todas las guardas son falsas. Por el Teorema 6.10 basta probarlo para un bucle con una sola guarda.

TEOREMA 6.12 *Sea el bucle* $\mathcal{R} \doteq *[\![b \to S]\!]$*, donde* S *es sana; entonces*

(i) $\forall X :: [\mathcal{R}.X \equiv \mathcal{R}.(X \wedge \neg b)]$

(ii) *Si* $[b \equiv Cierto]$*, entonces,* $[\mathcal{R}.C \equiv F]$*; es decir: si la guarda siempre es cierta, el bucle* nunca termina.

NOTA ***6.13*** *Si el lector vuelve al Ejemplo 6.3, observará que lo allí probado es una conclusión directa del apartado* (ii) *del presente lema.*

Demostración.– Veamos (i),

$$
\begin{array}{ll}
 & [\mathcal{R}.X \equiv \mathcal{R}.(X \wedge \neg b])\\
= & \quad \because \text{semántica de los bucles (Definición 6.2)}\\
 & [\exists n : n \geq 0 : H^n.X \equiv \exists n : n \geq 0 : H^n.(X \wedge \neg b)]\\
\Leftarrow & \quad \because \text{CP}\\
 & \forall n : n \geq 0 : [H^n.X = H^n.(X \wedge \neg b)]
\end{array}
$$

y vemos esto último por inducción; para $n = 0$ es trivial; el paso inductivo es:

$$
\begin{array}{ll}
 & [H^{n+1}.X \equiv H^{n+1}.(X \wedge \neg b)]\\
= & \quad \because \text{definición}\\
 & [\neg b \wedge X \vee b \wedge S.H^n.X \equiv \neg b \wedge X \vee b \wedge S.H^n.(X \wedge \neg b)]\\
\Leftarrow & \quad \because \text{CP}\\
 & [S.H^n.X \equiv S.H^n.(X \wedge \neg b)]\\
\Leftarrow & \quad \because \text{regla de Leibniz}\\
 & [H^n.X \equiv H^n.(X \wedge \neg b)]
\end{array}
$$

Veamos (ii), calculando *ptle*, $\mathcal{R}.C$

$$
\begin{array}{ll}
= & \quad \because \text{Teorema 6.12}(i)\\
 & \mathcal{R}.\neg b\\
= & \quad \because \text{hipótesis: } [b \equiv Cierto], \text{ sustitutividad}\\
 & \mathcal{R}.F\\
= & \quad \because \text{si } S \text{ es sana, por el Teorema 6.15, } \mathcal{R} \text{ es sana, y por la LME}\\
 & F
\end{array}
$$

TEOREMA

El siguiente teorema afirma que si los cuerpos de dos bucles con la misma guarda tienen el mismo comportamiento en el entorno de la guarda, entonces los dos bucles también tienen el mismo comportamiento.

TEOREMA 6.14 *Si* $\forall Z :: [b \wedge S.Z \equiv b \wedge T.Z]$*, entonces* $*[\![b \to S]\!] = *[\![b \to T]\!]$.

Demostración.– Fijado el predicado X, sabemos que *ptle*

$$*[\![b \to S]\!].X \doteq \exists k : k \geq 0 : H^k, \qquad *[\![b \to T]\!].X \doteq \exists k : k \geq 0 : J^k,$$

donde, también *ptle*,

$$
\begin{array}{lcllcl}
H^0 & \doteq & \neg b \wedge X & J^0 & \doteq & \neg b \wedge X\\
H^{k+1} & \doteq & H^0 \vee [\![b \to S]\!].H^k & J^{k+1} & \doteq & J^0 \vee [\![b \to T]\!].J^k
\end{array}
$$

Hay que probar $[\ *[\![b \to S]\!].X \equiv *[\![b \to T]\!].X\ ]$; para ello es suficiente probar (por inducción) $\forall k : k \geq 0 : [H^k \equiv J^k]$. Para $k = 0$ es trivial; supuesto para valores $\leq k$, tenemos,

$$
\begin{array}{ll}
 & [H^{k+1} \equiv J^{k+1}] \\
= & \quad \text{⟨·⟩ definición} \\
 & [H^0 \vee [\![b \to S]\!].H^k \equiv J^0 \vee [\![b \to T]\!].J^k] \\
\Leftarrow & \quad \text{⟨·⟩ semántica selectiva: } [\![b \to S]\!].Z \equiv b \wedge S.Z;\ H^0 = J^0 \\
 & [b \wedge S.H^k \equiv b \wedge T.J^k] \\
\Leftarrow & \quad \text{⟨·⟩ HI, Leibniz} \\
 & [b \wedge S.J^k \equiv b \wedge T.J^k] \\
= & \quad \text{⟨·⟩} \forall Z :: [b \wedge S.Z \equiv b \wedge T.Z] \\
 & Cierto
\end{array}
$$

TEOREMA

TEOREMA 6.15 (Salubridad del bucle) *Sea el bucle* $\mathcal{R} \doteq *[\![b \to S]\!]$*, donde* S *es conjuntiva y estricta; entonces,*

(i) *Los transformadores de predicados* H^k *asociados son conjuntivos y estrictos.*

(ii) *La sucesión de transformadores* $\{H^k\}$ *es creciente en* $\mathcal{T}$*.*

(iii) $\mathcal{R}$ *es estricto y conjuntivo.*

Demostración.– Demostremos $\forall n : n \geq 0 : [H^n.Falso \equiv Falso]$, por inducción:

CASO BASE ($n = 0$), *ptle*,

$$
\begin{array}{ll}
 & H^0.F \\
= & \quad \text{⟨·⟩ definición} \\
 & \neg b \wedge F \\
= & \quad \text{⟨·⟩ CP} \\
 & Falso
\end{array}
$$

PASO INDUCTIVO; *ptle*,

$$
\begin{array}{ll}
 & H^{n+1}.F \\
= & \quad \text{⟨·⟩ definición} \\
 & \neg b \wedge F \vee b \wedge S.H^n.F \\
= & \quad \text{⟨·⟩ cálculo, HI} \\
 & b \wedge S.F \\
= & \quad \text{⟨·⟩ } S \text{ es estricto} \\
 & F
\end{array}
$$

luego, $\mathcal{R}.F \equiv (\exists i : i \geq 0 : F) \equiv F$, de donde $\mathcal{R}$ es estricto. Demostremos ahora por inducción $\forall n : n \geq 0 : [H^n.(X \wedge Y) \equiv H^n.X \wedge H^n.Y]$

CASO BASE ($n = 0$), *ptle*,

$$
\begin{array}{ll}
 & H^0.X \wedge H^0.Y \\
= & \quad \text{⟨·⟩ definición} \\
 & X \wedge \neg b \wedge Y \wedge \neg b \\
= & \quad \text{⟨·⟩ CP} \\
 & \neg b \wedge X \wedge Y \\
= & \quad \text{⟨·⟩ definición} \\
 & H^0.(X \wedge Y)
\end{array}
$$

PASO INDUCTIVO; *ptle*,

$$
\begin{array}{ll}
 & H^{n+1}.X \wedge H^{n+1}.Y \\
= & \quad \text{⟨·⟩ definición} \\
 & (\neg b \wedge X \vee b \wedge S.H^n.X) \\
\wedge & (\neg b \wedge Y \vee b \wedge S.H^n.Y) \\
= & \quad \text{⟨·⟩ CP} \\
 & \neg b \wedge X \wedge Y \vee b \wedge S.H^n.X \wedge S.H^n.Y \\
= & \quad \text{⟨·⟩ } S \text{ conjuntivo} \\
 & \neg b \wedge X \wedge Y \vee b \wedge S.(H^n.X \wedge H^n.Y) \\
= & \quad \text{⟨·⟩ HI: } H^n \text{ conjuntivo} \\
 & \neg b \wedge X \wedge Y \vee b \wedge S.H^n.(X \wedge Y) \\
= & \quad \text{⟨·⟩ definición} \\
 & H^{n+1}.(X \wedge Y)
\end{array}
$$

y esto prueba (i). Para probar la conjuntividad de $\mathcal{R}$ tenemos, *ptle*

$$
\begin{array}{ll}
 & \mathcal{R}.A \wedge \mathcal{R}.B \\
= & \\
 & (\exists i : i \geq 0 : H^i.A) \wedge (\exists j : j \geq 0 : H^j.B) \\
= & \quad \text{⟨·⟩ distributividad}
\end{array}
$$

$$
\begin{array}{ll}
 & \exists i,j : i,j \geq 0 : H^i.A \wedge H^j.B \\
= & \quad \because \text{reordenando, por distrib. e idempotencia} \\
 & \exists k : k \geq 0 : H^k.A \wedge (\exists j : 0 \leq j \leq k : H^j.B) \\
\vee & \exists k : k \geq 0 : H^k.B \wedge (\exists i : 0 \leq i \leq k : H^i.A) \\
= & \quad \because \text{por } (ii),\ (\exists j : 0 \leq j \leq k : H^j.B) \equiv H^k.B, \ldots \\
 & \exists k : k \geq 0 : H^k.A \wedge H^k.B \\
\vee & \exists k : k \geq 0 : H^k.B \wedge H^k.A \\
= & \quad \because \text{idempotencia} \\
 & \exists k : k \geq 0 : H^k.A \wedge H^k.B \\
= & \quad \because H^k \text{ es conjuntivo} \\
 & \exists k : k \geq 0 : H^k.(A \wedge B) \\
= & \quad \because \text{definición} \\
 & \mathcal{R}.(A \wedge B)
\end{array}
$$

Queda probar (ii); es decir, para cada X, $\forall n : n \geq 0 : [H^n.X \ \Rightarrow \ H^{n+1}.X]$. De nuevo por inducción; para $n = 0$ es trivial; el paso inductivo es:

$$
\begin{array}{ll}
 & [H^{n+1}.X \ \Rightarrow \ H^{n+2}.X] \\
= & \quad \because \text{definición} \\
 & [\neg b \wedge X \vee b \wedge S.H^n.X \ \Rightarrow \ \neg b \wedge X \vee b \wedge S.H^{n+1}.X] \\
\Leftarrow & \quad \because \text{CP} \\
 & [b \wedge S.H^n.X \ \Rightarrow \ b \wedge S.H^{n+1}.X] \\
\Leftarrow & \quad \because \text{cálculo} \\
 & [S.H^n.X \ \Rightarrow \ S.H^{n+1}.X] \\
\Leftarrow & \quad \because S \text{ monótona} \\
 & [H^n.X \ \Rightarrow \ H^{n+1}.X]
\end{array}
$$

TEOREMA

6.16 *[267]* **(Febrero, 93)** *Sean S y S' dos sentencias equivalentes en el entorno P; es decir: $\forall X :: [P \wedge S.X \equiv P \wedge S'.X]$. Demostrad que si P es un invariante de $\mathcal{R}$, entonces los bucles $\mathcal{R} \doteq *[\![b \to S]\!]$ y $\mathcal{R}' \doteq *[\![b \to S']\!]$ son equivalentes en el entorno P.*

COROLARIO 6.17 (Conservación de la salubridad) *Un bucle genérico con n guardas es sano si lo son todas las sentencias guardadas.*

Demostración.– Basta aplicar el Teorema 6.15 junto al Teorema 6.10. COROLARIO

Determinismo del bucle En el Teorema 6.18 veremos que para una guarda,

todo bucle hereda el determinismo de su cuerpo (her)

Por el Teorema 6.10, el cuerpo de un bucle $*SI$ con varias guardas es la sentencia SI, y por la propiedad (her), el bucle será determinista si lo es el cuerpo SI. Ahora aplicamos el Teorema 4.27, para deducir que SI es determinista si las guardas son excluyentes dos a dos y son deterministas todas las sentencias guardadas. En resumen, $*SI$ es determinista si las guardas son excluyentes dos a dos y son deterministas todas las sentencias guardadas.

Para probar (her) veamos que si el cuerpo S de un bucle es disyuntivo, lo son los transformadores H^n:

$$\forall n : n \geq 0 : [H^n.(X \vee Y) \equiv H^n.X \vee H^n.Y] \qquad (*)$$

Inducción sobre n; para $n = 0$ es trivial; el paso inductivo es

$$
\begin{array}{ll}
 & [H^{n+1}.(X \vee Y) \equiv H^{n+1}.X \vee H^{n+1}.Y] \\
= & \quad \because \text{definición} \\
\equiv & [\neg b \wedge (X \vee Y) \vee b \wedge S.H^n.(X \vee Y) \\
 & \neg b \wedge X \vee b \wedge S.H^n.X \vee \neg b \wedge Y \vee b \wedge S.H^n.Y] \\
\Leftarrow & \quad \because \text{distribuimos, reordenamos y eliminamos factores} \\
 & [S.H^n.(X \vee Y) \equiv S.(H^n.X \vee H^n.Y)] \\
\Leftarrow & \quad \because \text{regla de Leibniz} \\
 & [H^n.(X \vee Y) \equiv H^n.X \vee H^n.Y]
\end{array}
$$

que es la HI. A partir de la propiedad $(*)$, la prueba de la disyuntividad del bucle es fácil. En efecto, *ptle*

$$
\begin{array}{ll}
 & \mathcal{R}.A \vee \mathcal{R}.B \\
= & \quad \because \text{Definición 6.2} \\
 & (\exists i : i \geq 0 : H^i.A) \vee (\exists j : j \geq 0 : H^j.B) \\
= & \quad \because \text{disyuntividad de } \exists \\
 & \exists k : k \geq 0 : H^k.A \vee H^k.B \\
= & \quad \because H^k \text{ es disyuntiva} \\
 & \exists k : k \geq 0 : H^k.(A \vee B) \\
= & \quad \because \text{definición} \\
 & \mathcal{R}.(A \vee B)
\end{array}
$$

En consecuencia, hemos demostrado el siguiente

TEOREMA 6.18

(i) *Si S es determinista, también son deterministas los transformadores H^k y el bucle $*[\![b \to S]\!]$.*

(ii) *Para que el bucle genérico $*[\![\ \square_{1 \leq i \leq n} b_i \to S_i \ ]\!]$ sea determinista es suficiente con que lo sean las sentencias guardadas, y las guardas excluyentes dos a dos.*

NOTA 6.19 *Recuerde el lector la siguiente curiosidad, para un bucle $*[\![b \to S]\!]$:*

Los transformadores H^k heredan las siguientes propiedades del cuerpo S:
✓ *estrictez (LME)* ✓ *conjuntividadd* ✓ *disyuntividad*

que a su vez es trasladada al propio bucle. La razón es que los constructores elementales del lenguaje (salvo el constructor bucle) conservan tales propiedades, además de que H^k se puede construir con constructores básicos.

6.20 *[268] Sea el bucle $\mathcal{R} \doteq *[\![b \to S]\!]$. Probad e interpretad*

$$\mathcal{R}.C \equiv \neg b \vee S.\neg b \vee S^2.\neg b \vee \ \ldots S^n.\neg b \vee \ \ldots$$

donde S^n es el transformador S aplicado n veces.

6.21 *[268] Demostrad que ptle, $*[\![b \to S]\!].X \equiv (\exists k : k \geq 0 : J^k.X)$, con S determinista, y*

$$J^0.X \doteq \neg b \wedge X \qquad J^k.X \doteq b \wedge S.J^{k-1}.X$$

Justificad que $J^k.X$ es la precondición más débil para que el cuerpo del bucle se ejecute exactamente k veces y termine verificando el predicado X.

EJEMPLO 6.22 Apliquemos el resultado del Ejercicio 6.21, tenemos,

$$*[\![\, i < N \to i := i+1 \,]\!].C \equiv (\exists k : k \geq 0 : J^k.C)$$

Pero es fácil probar por inducción,

$$\forall k : k \geq 1 : [J^k.C \equiv i = N - k] \tag{1}$$

CASO BASE ($k = 1$):

$$\begin{array}{ll} & J^1.C \\ = & \quad \because \text{definición} \\ & i < N \wedge i := i+1 . i \geq N \\ = & \quad \because \text{semántica} \\ & i < N \wedge i + 1 \geq N \\ = & \\ & i = N - 1 \end{array}$$

PASO INDUCTIVO:

$$\begin{array}{ll} & J^{k+1}.C \\ = & \quad \because \text{definición de } J^k \text{, e HI} \\ & i < N \wedge i := i+1 . (i = N - k) \\ = & \quad \because \text{sustitución, } i, N, k \text{ enteros, CP} \\ & i = N - (k+1) \end{array}$$

En resumen,

$$\begin{array}{ll} & *[\![\, i < N \to i := i+1 \,]\!].C \\ = & \\ & i \geq N \vee i = N-1 \vee i = N-2 \vee \ldots \\ = & \quad \because (1) \\ & Cierto \end{array}$$

Compárese el resultado con el obtenido en el Ejemplo 6.6. De la misma forma se obtiene, $\forall k : k \geq 1 : [J^k.(i = N) \equiv i \leq N - k]$, de donde, *ptle*, $*[\![\, i < N \to i := i+1 \,]\!].(i = N) \equiv i \leq N$. Para $\mathcal{R}' \doteq *[\![\, i \neq n \to i := i+1 \,]\!]$, se tiene $\forall k : k \geq 1 : [J^k.C \equiv i = N - k]$, de donde $[\, \mathcal{R}'.C \equiv i \leq N \,]$. EJEMPLO

Contextos y sustitutividad del lenguaje

Si el símbolo $\langle\,\rangle$ denota un *hueco*, un contexto es una sentencia en la cual se sustituye alguna sentencia distinguida por el hueco $\langle\,\rangle$. Por ejemplo,

$$\begin{array}{lcl} C\langle\,\rangle & \doteq & x := x+1; \langle\,\rangle; y := z+2 \\ C'\langle\,\rangle & \doteq & x := x-1; [\![\, x > 1 \to \langle\,\rangle \,\square\, x > 3 \to \langle\,\rangle; x := x-1 \,]\!] \end{array}$$

El último contexto puede obtenerse al reemplazar en la sentencia

$$x := x-1; [x > 1 \to y := 1 \,\square\, x > 3 \to y := 1; x := x-1]$$

la sentencia $y := 1$ por el hueco $\langle\,\rangle$. El conjunto $\mathcal{C}\langle\,\rangle$ de contextos con hueco $\langle\,\rangle$ se puede definir en forma inductiva

$$\begin{array}{lcl} & & \langle\,\rangle, nada, aborta, x := E \in \mathcal{C}\langle\,\rangle \\ C_1, C_2 \in \mathcal{C}\langle\,\rangle & \Rightarrow & C_1; C_2 \in \mathcal{C}\langle\,\rangle \\ \forall i : 1 \leq i \leq n : C_i \in \mathcal{C}\langle\,\rangle & \Rightarrow & [\![\, \square : 1 \leq i \leq n : b_i \to C_i \,]\!] \in \mathcal{C}\langle\,\rangle \\ C \in \mathcal{C}\langle\,\rangle & \Rightarrow & *[\![\, b \to C \,]\!] \in \mathcal{C}\langle\,\rangle \end{array}$$

y en consecuencia, el conjunto $\mathcal{C}\langle\,\rangle$ es un conjunto inductivo, sobre el que podemos aplicar el principio de inducción.

Al reemplazar en un contexto $K\langle\,\rangle$ el hueco $\langle\,\rangle$ por una sentencia S se obtiene otra sentencia que será denotada con $K\langle S\rangle$; por ejemplo, para los contextos anteriores tenemos

$$\begin{array}{lcl} C\langle a := a+1001\rangle & \equiv & x := x+1; a := a+1001; y := z+2 \\ C'\langle aborta\rangle & \equiv & x := x-1; \\ & & [\![\, x>1 \to aborta \,\square\, x>3 \to aborta; x := x-1 \,]\!] \end{array}$$

Así $K\langle S\rangle$ captura la sustitución, y ésta puede definirse por inducción

$$\begin{array}{lcl} \langle S\rangle & \doteq & S \\ nada\langle S\rangle & \doteq & nada \\ aborta\langle S\rangle & \doteq & aborta \\ x := E\langle S\rangle & \doteq & x := E \\ (C_1;C_2)\langle S\rangle & \doteq & C_1\langle S\rangle; C_2\langle S\rangle \\ [\![\, \square : 1\le i\le n : b_i \to S_i \,]\!]\,\langle S\rangle & \doteq & [\![\, \square : 1\le i\le n : b_i \to S_i\langle S\rangle \,]\!] \\ *[\![\, b\to C \,]\!]\,\langle S\rangle & \doteq & *[\![\, b\to C\langle S\rangle \,]\!] \end{array}$$

6.23 *Dad una demostración de la propiedad de sustitutividad del lenguaje de Dijkstra. Es decir, si $K\langle\,\rangle$ es un contexto arbitrario, demostrad:*

$$S_1 = S_2 \;\Rightarrow\; K\langle S_1\rangle = K\langle S_2\rangle$$

AYUDA.- Pruebe que la aplicación $X \mapsto K\langle X\rangle$ es una función probando que $K\langle X\rangle$ es una fórmula del cálculo de predicados. Aplique después la regla de Leibniz.

DEFINICIÓN 6.24 (Inversión de sentencias) *Se dice que S' es una inversa (a la derecha) de S si se verifica $S; S' = nada$. $S^{-1} \backsimeq T$ se lee: T es una inversa de S.*

La sentencia $aborta$ no es inversible ya que $aborta; T = aborta$. Además:

$$nada^{-1} \backsimeq nada \qquad\qquad (S;T)^{-1} \backsimeq T^{-1}; S^{-1}$$

6.25 *Aplicando la Definición 6.24,*

- (i) *Calculad $(x := \alpha x + \beta y)^{-1}$. AYUDA.- buscad una inversa que tenga la forma $x := \alpha' x + \beta' y$; use el lema de sustitución (Lema 4.7, página 59).*
- (ii) *Dad ejemplos no triviales de sentencias que coincidan con su inversa.*

6.26 *Probad con ejemplos que la propiedad de inversión no es sustitutiva; es decir, puede ocurrir que S tenga inversa ($S; S^{-1} = nada$), pero, para cierto contexto $K\langle\,\rangle$, la sentencia $K\langle S^{-1}\rangle$ no sea inversa de $K\langle S\rangle$; por ejemplo:*

- ✓ *Una inversa de $x, y := y, x$ es ella misma.*
- ✓ *Para el contexto*

$$K\langle\,\rangle \equiv [\![\, C \to \langle\,\rangle; z := 1 \,\square\, C \to \langle\,\rangle; z := 2 \,]\!]$$

se verifica $(K\langle x, y := y, x\rangle)^{-1} \not\backsimeq K\langle x, y := y, x\rangle$.

6.27 **(Propiedades Hereditarias)** *El símbolo de composición de sentencias ";" es visto como un operador binario entre sentencias:*

$$;.S_1.S_2 \;\doteq\; S_1; S_2$$

mientras que el operador selectivo $selec(b_1, b_2, ..., b_n)$ definido

$$selec(b_1, b_2, ..., b_n).(S_1, S_2, ..., S_n) \equiv [\![\, \square : 1\le i\le n : b_i \to S_i \,]\!]$$

es un operador n–ario. En caso de una guarda simplificamos la notación en la forma

$$b.S \doteq [\![b \to S]\!]$$

Igualmente definimos el operador unario "$*b$" como $*b.S \doteq *[\![b \to S]\!]$. El operador de alternancia ∇, definido en la forma:

$$\nabla.S.T \doteq [\![Cierto \to S \,\square\, Cierto \to T]\!]$$

resulta ser también un operador binario. Sea ahora f un operador n–ario y p una propiedad entre sentencias. Se dice que f conserva la propiedad p si se cumple

$$p.S_1 \wedge p.S_2 \wedge \ \ldots \ \wedge p.S_n \ \Rightarrow \ p.f.(S_1, S_2, ..., S_n)$$

Completad los cuadros siguientes (colocando en cada casilla SI, NO, o la condición necesaria) sobre conservación de propiedades para los distintos operadores del lenguaje de Dijkstra:

	$;$	$selec(b_1, b_2, ..., b_n)$	b	∇	$*b$
Ley del milagro excluido					
Conjuntividad					
Determinismo					
Terminación: $[S.C \equiv C]$					
$\{x = a\}S\{x > a\}$					
Conserva I*:* $\{I\}S\{I\}$					
Invertibilidad					

6.28 **(Propiedades Distributivas)** *Estudiad si el operador "b" verifica las siguientes propiedades distributivas, dando contraejemplos en caso de que no se cumplan y condiciones adicionales para que se cumplan:*

(A) $b.(S;T) = b.S;b.T$

(B) $b.(S;T) = (b.S);T$

(C) $b.(S;T) = S;b.T$

(D) Generalizad al caso de varias guardas; es decir, estudiad las propiedades

$$[\![b \to S;T \,\square\, b' \to S';T]\!] = [\![b \to S \,\square\, b' \to S']\!];T$$
$$[\![b \to T;S \,\square\, b' \to T;S']\!] = T;[\![b \to S \,\square\, b' \to S']\!]$$

Estudiad si el operador "$*b$" verifica las siguientes propiedades distributivas, dando contraejemplos o condiciones adicionales para que se cumplan:

(E) $*b.(S;T) = *b.S;*b.T$

(F) $*b.(S;T) = *b.S;T$

6.29 *Utilizando Ejercicio 6.28 y Ejercicio 6.23 probad la igualdad*

$$[\![b \to x := x+1 \,\square\, b' \to x := x+2]\!];x := x-1 \ = \ [\![b \to nada \,\square\, b' \to x := x+1]\!].$$

6.30 *[269]* **(Conmutatividad de sentencias)** *S_1 y S_2 conmutan si $S_1;S_2 = S_2;S_1$. Por ejemplo, $nada$ y $aborta$ conmutan con cualquier sentencia.*

(A) Dad ejemplos no triviales de sentencias que conmuten.

(B) ¿Cuándo conmutan las asignaciones $x := E$ y $x := F$?

(C) Dad condiciones para la conmutatividad de $x := E$ con otra sentencia S.

6.31 *[271]* **(Setiembre, 99)** *Se consideran dos variables $x, y :\in \mathcal{A}$, siendo $\mathcal{A}$ un tipo con una relación de orden total.*

(A) Escribid los predicados: $Z_1 \doteq x = max(a,b), \qquad Z_2 \doteq y = min(a,b)$.

(B) Siendo $Z \doteq Z_1 \wedge Z_2$*, calculad los valores* $\mathcal{S}.Z$ *y* $\mathcal{S}'.Z$*, donde*

$$\begin{array}{lcl} \mathcal{S} & \doteq & [\![\, a \geq b \to x, y := a, b \,\square\, a \leq b \to x, y := b, a \,]\!] \\ \mathcal{S}' & \doteq & x, y := a, b; *[\![\, y > x \to x, y := y, x \,]\!] \end{array}$$

(C) Concluid $\{Cierto\}\mathcal{S}\{Z\}$ *y además* $\{Cierto\}\mathcal{S}'\{Z\}$.

6.2. El Teorema de Invariantes

Veremos en este apartado y en el siguiente dos resultados esenciales: el *Teorema de Invariantes* y el *Teorema de los Contadores*. Nuestras demostraciones están profundamente influenciadas por las presentadas en [Dijkstra, 1976], y utilizan la semántica inductiva de los bucles (Definición 6.2). En el Capítulo 8 veremos demostraciones más sutiles que utilizan la semántica en términos de puntos fijos. Aunque daremos la equivalencia de ambas semánticas, la razón de exponer las demostraciones en este orden es puramente histórica. Pondremos de manifiesto la necesidad de un razonamiento inductivo guiado por la definición inductiva de los transformadores H^k.

Concepto de Invariante Recordemos el Ejemplo 6.1, donde obtuvimos el siguiente programa que simula el juego de la urna de Dijkstra:

$$\begin{array}{l} b, n :\in \mathbb{Z}; \\ b, n := 35, 14; \\ \quad *[\![\quad \begin{array}{lcl} b > 1 & \to & b, n := b-2, n+1 \\ n > 1 & \to & n := n-1 \\ n > 0 \wedge b > 0 & \to & n := n-1 \,]\!] \end{array} \end{array}$$

Cada acción del cuerpo del bucle *conserva la paridad de b*. Conclusión: el predicado $I \doteq impar\ b$ es invariante: se conserva bajo cada ejecución del cuerpo del bucle, i.e., $[I \wedge OB \;\Rightarrow\; SI.I]$. Si I es cierto al principio del bucle y si el bucle termina, es plausible afirmar que el bucle terminará verificando I, y por el Teorema 6.12, también terminará verificando $I \wedge \neg OB$. A la vista de la definición de I, no podemos deducir que al final quede una única bola. Podemos modificar el predicado I en la forma

$$I \doteq impar\ b \wedge b, n \geq 0$$

con lo que es fácil obtener: $[I \wedge \neg OB \;\Rightarrow\; n = 0 \wedge b = 1]$. Esta idea condujo a Robert [Floyd, 1967] a formalizar el concepto de invariante y dar, en términos de tripletes, una regla de derivación para el bucle.

TEOREMA 6.32 (Teorema de Invariantes para bucles) *Sea $\mathcal{R}$ el bucle $*[\![\, \square : 1 \leq j \leq n : b_j \to S_j \,]\!]$, y sea I un predicado verificando:*

$$[I \wedge OB \;\Rightarrow\; SI.I] \qquad (a)$$

entonces, se verifica

$$[I \wedge \mathcal{R}.C \;\Rightarrow\; \mathcal{R}.(I \wedge \neg OB)] \qquad (b)$$

En términos de tripletes

$$\{I \wedge OB\}SI\{I\} \;\Rightarrow\; \{I \wedge \mathcal{R}.C\}\mathcal{R}\{I \wedge \neg OB)\}$$

OBSERVACIÓN.– La condición (a) significa que I es un *invariante*; es decir, después de la ejecución del cuerpo del bucle se cumple I, si se cumplía antes. La tesis (b) del teorema tiene la siguiente interpretación: si partimos de un estado para el cual el invariante I es cierto y el bucle termina, terminará cumpliéndose I, y por el Teorema 6.12, $\neg OB$. OBS

OBSERVACIÓN.– Para el bucle de PASCAL $while\ B\ do\ S$ tenemos el bucle equivalente $*[\![\, b \to S \,]\!]$. Si recordamos la regla dada por [Hoare, 1969]

$$\frac{\{I \wedge b\}S\{I\}}{\{I\}\ while\ b\ do\ S\ \{I \wedge \neg b\}}$$

encontramos una diferencia fundamental: no se especifica que el bucle debe terminar, y la regla solamente infiere tripletes *correctos parcialmente*. Por consiguiente la regla no infiere tripletes correctos en sentido *total*. Por ejemplo, el predicado $n > 0$ es invariante del bucle $while\ n > 0\ do\ n := n + 1$, pero éste no termina para los estados iniciales tales que $n > 0$. OBS

Demostración.– Según el Teorema 6.10[1] es suficiente considerar una guarda y tomamos $\mathcal{R} == *[\![\, b \to S \,]\!]$. Además, por el Teorema 6.12, $[\ \mathcal{R}.I \equiv \mathcal{R}.(I \wedge \neg b)\ ]$, y por tanto, para probar el teorema es suficiente probar la siguiente implicación:

$$[I \wedge b \;\Rightarrow\; S.I] \;\Rightarrow\; [I \wedge \mathcal{R}.C \;\Rightarrow\; \mathcal{R}.I]$$

Procedemos a partir del consecuente,

$$\begin{array}{ll}
 & [I \wedge \mathcal{R}.C \;\Rightarrow\; \mathcal{R}.I] \\
= & \quad \because \text{semántica de bucles} \\
 & [I \wedge (\exists k : k \geq 0 : H^k.C) \;\Rightarrow\; (\exists k : k \geq 0 : H^k.I)] \\
= & \quad \because \text{distributividad de } \exists \\
 & [(\exists k : k \geq 0 : I \wedge H^k.C) \;\Rightarrow\; (\exists k : k \geq 0 : H^k.I)] \\
\Leftarrow & \quad \because \text{CP} \\
 & \forall k : k \geq 0 : [I \wedge H^k.C \;\Rightarrow\; H^k.I] \\
= & \quad \because \text{principio de inducción} \\
 & [I \wedge H^0.C \;\Rightarrow\; H^0.I] \qquad \text{— trivial} \\
\wedge & \forall k : k \geq 1 : [I \wedge H^{k-1}.C \;\Rightarrow\; H^{k-1}.I] \;\Rightarrow\; [I \wedge H^k.C \;\Rightarrow\; H^k.I]
\end{array}$$

Pero, el consecuente de la anterior implicación se manipula fácilmente

$$\begin{array}{ll}
 & [I \wedge H^k.C \;\Rightarrow\; H^k.I] \\
= & \quad \because \text{definición} \\
 & [I \wedge (\neg b \vee b \wedge S.H^{k-1}.C) \;\Rightarrow\; \neg b \wedge I \vee b \wedge S.H^{k-1}.I] \\
\Leftarrow & \quad \because \text{CP}
\end{array}$$

[1] Esta es otra de las aplicaciones interesantes anunciadas en la Nota 6.11.

$$
\begin{array}{ll}
& [I \wedge b \wedge S.H^{k-1}.C \;\Rightarrow\; S.H^{k-1}.I] \\
= & \quad \because \text{por } (a) \text{ y la regla de oro: } [I \wedge b \equiv I \wedge b \wedge S.I] \\
& [I \wedge b \wedge S.I \wedge S.H^{k-1}.C \;\Rightarrow\; S.H^{k-1}.I] \\
= & \quad \because S \text{ es conjuntivo} \\
& [I \wedge b \wedge S.(I \wedge H^{k-1}.C) \;\Rightarrow\; S.H^{k-1}.I] \\
\Leftarrow & \quad \because S \text{ es monótono, cálculo} \\
& [I \wedge H^{k-1}.C \;\Rightarrow\; H^{k-1}.I] \\
= & \\
& HI
\end{array}
$$

TEOREMA

6.33 *[271] Probad que si I y J son predicados invariantes de cierto bucle $\mathcal{R}$, también los es el predicado $I \wedge J$, así como el predicado $I \vee J$.*

6.34 *[272] Siendo $\mathcal{R} \doteq *[\![\, b \to [\![\, f \to S \,\Box\, \neg f \to T \,]\!] \,]\!]$, probad que I es un invariante de $\mathcal{R}$ sii $[I \wedge b \wedge f \;\Rightarrow\; S.I] \wedge [I \wedge b \wedge \neg f \;\Rightarrow\; T.I]$.*

6.35 *[272] Probad que el bucle $\mathcal{R}$ del Ejercicio 6.34 equivale al bucle:*

$$
\begin{array}{lll}
*[\![& b \wedge f & \to S \\
\Box & b \wedge \neg f & \to T \,]\!]
\end{array}
$$

EJEMPLO 6.36 Tratemos de probar la corrección del siguiente programa

$$
\begin{array}{l}
a, b, c := A, B, C; \\
\quad *[\![\qquad a > b \quad \to a, b := b, a \\
\qquad\quad \Box \quad b > c \quad \to b, c := c, b \,]\!] \\
\{a = \text{mín}(A, B, C) \wedge c = \text{máx}(A, B, C)\}
\end{array}
$$

Que el predicado $I \doteq {}'(a, b, c)$ es una permutación de $(A, B, C)'$ es un invariante es fácilmente demostrable utilizando el Teorema 5.16:80, tomando $Q == I$. De aquí, si el bucle termina, por aplicación del teorema de invariantes, terminará verificando

$$
\begin{array}{ll}
& I \wedge \neg OB \\
= & \\
& I \wedge a \leq b \leq c \\
\Rightarrow & \quad \because \text{propiedades de mín y de máx} \\
& a = \text{mín}(A, B, C) \wedge c = \text{máx}(A, B, C)
\end{array}
$$

y el programa *ordena* los valores iniciales. La terminación la demostraremos de dos formas. En la primera, utilizaremos directamente la semántica inductiva de los bucles. La segunda utiliza el *Teorema de los contadores* (Teorema 6.38). Concretamente probaremos, $[H^3.C \equiv C]$, de donde, por ser la sucesión de transformadores H^k una sucesión creciente — Teorema 6.15(ii) — deducimos, *ptle*

$$\mathcal{R}.C(\equiv \exists k : k \geq 0 : H^k.C) \equiv C$$

Trivialmente (simplificamos la notación en la forma $H^k.C == H^k$),

$$[\, H^0 = a \leq b \leq c \,]$$

de donde, *ptle*, H^1

$$
\begin{array}{ll}
= & \quad \because \text{definición} \\
& H^0 \vee OB \wedge (a > b \;\Rightarrow\; a, b := b, a \,.\, H^0) \wedge (b > c \;\Rightarrow\; b, c := c, b \,.\, H^0)
\end{array}
$$

$=$ $\quad \because$ semántica asignación
$H^0 \vee OB \wedge (a > b \ \Rightarrow \ b \leq a \leq c) \wedge (b > c \ \Rightarrow \ a \leq c \leq b)$
$=$ $\quad \because a > b \ \Rightarrow \ b \leq a, \ldots$
$H^0 \vee OB \wedge (a > b \ \Rightarrow \ a \leq c) \wedge (b > c \ \Rightarrow \ a \leq c)$
$=$ $\quad \because$ CP
$H^0 \vee OB \wedge (a > b \vee b > c \ \Rightarrow \ a \leq c)$
$=$ $\quad \because [a \leq c \wedge H^0 \equiv \neg OB]$, $[OB \equiv a > b \vee b > c]$, CP
$\neg OB \wedge a \leq c \vee OB \wedge a \leq c$
$=$ $\quad \because$ CP
$a \leq c$

es decir: $[\ H^1.C \equiv a \leq c\ ]$, lo que es lógico ya que el bucle termina a lo sumo en un 1 paso si se tiene tal relación. En efecto; si $a \leq c$, b puede ocupar tres posiciones:

1. $b < a$, y con un intercambio quedan ordenadas las tres variables,
2. $a \leq b \leq c$, y ya están ordenadas, y finalmente
3. $c < b$, y con un intercambio quedan ordenadas.

Un cálculo similar conduce a $[SI.H^1.C \equiv OB \wedge (a \leq b \vee b \leq c)]$, luego *ptle*,

$H^2.C$
$=$ $\quad \because$ definición
$\neg OB \vee OB \wedge (a \leq b \vee b \leq c)$
$=$ $\quad \because [\neg OB \ \Rightarrow \ a \leq b \vee b \leq c]$, regla de oro, tercio excluido
$a \leq b \vee b \leq c$

es decir: $[\ H^2.C \equiv a \leq b \vee b \leq c\ ]$. Y finalmente:

$SI.H^2.C$
$=$ $\quad \because$ definición y cálculo
OB

de donde, *ptle*, $H^3.C \equiv OB \vee \neg OB \equiv C$. EJEMPLO

6.37 *Estudiad los predicados $H^k.C$, para $0 \leq k \leq 4$, y para el bucle*

$$
\begin{array}{l}
q_1, q_2, q_3, q_4 := Q_1, Q_2, Q_3, Q_4; \\
\quad *[\![\qquad q_1 > q_2 \rightarrow q_1, q_2 := q_2, q_1 \\
\qquad\quad \square \quad q_2 > q_3 \rightarrow q_2, q_3 := q_3, q_2 \\
\qquad\quad \square \quad q_3 > q_4 \rightarrow q_3, q_4 := q_4, q_3 \,]\!] \\
\{q_1 = \text{mín}(Q_1, Q_2, Q_3, Q_4) \wedge q_4 = \text{máx}(Q_1, Q_2, Q_3, Q_4)\}
\end{array}
$$

6.3. El Teorema de los Contadores

El teorema de invariantes asegura que si I es un invariante se verifica:

$$[I \wedge \mathcal{R}.C \ \Rightarrow \ \mathcal{R}.(I \wedge \neg OB)]$$

Como muestra el ejemplo anterior, en la práctica es difícil encontrar $\mathcal{R}.C$ y sería deseable estudiar condiciones para que $[I \ \Rightarrow \ \mathcal{R}.C]$. La idea original de [Floyd, 1967] es asignar un *contador entero al bucle*; es decir, una función $t : \mathcal{E} \rightarrow$

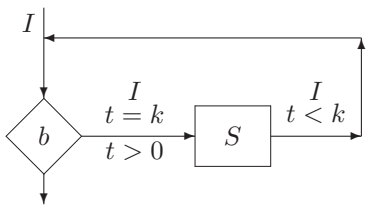


Figura 6.3: Interpretación del Teorema de los Contadores.

$\mathbb{Z}$ que se decremente en cada paso del bucle conservándose positiva $t > 0$ al comienzo de cada paso del bucle.

Enunciamos y probamos el teorema de los contadores para bucles con una guarda. La validez para varias guardas queda asegurada por el Teorema 6.10.

TEOREMA 6.38 (Teorema de los Contadores) *Sea $t : \mathcal{E} \to \mathbb{Z}$ un contador entero del bucle $\mathcal{R} \doteq *[\![b \to S]\!]$ asociado al invariante I; es decir, verificando las siguientes condiciones:*

(a) $[I \wedge b \Rightarrow S.I]$
(b) $[I \wedge b \Rightarrow t > 0]$
(c) $\forall k : k \in \mathbb{Z} : [I \wedge b \wedge t = k \Rightarrow S.(t < k)]$

Entonces $[I \Rightarrow \mathcal{R}.C]$, y por el Teorema de Invariantes $[I \Rightarrow \mathcal{R}.(I \wedge \neg b)]$.

OBSERVACIÓN.– (c) significa que el *contador entero* t es decrementado como mínimo en una unidad cada vez que se ejecuta el cuerpo S. Entonces, la sucesión de valores de t inmediatamente antes de la ejecución de S (véase la Figura 6.3) es estrictamente decreciente, y por (b) está acotada inferiormente por el natural 0. Luego la guarda b debe ocurrir solamente en un número finito de pasos. Es decir, si I siempre ocurre, $t \leq 0$ llegará a ocurrir, y entonces alguna guarda llegará a fallar, y el bucle terminará en un número finito de pasos, acotado por el valor inicial de t. OBS

NOTA 6.39 *Sabemos que el Cálculo de Hoare de la Figura 5.0 no es correcto; la razón es que con la regla del bucle*

$$\frac{\{I \wedge b\} S \{I\}}{\{I\} while\ b\ do\ S \{I \wedge \neg b\}}\ (rep)$$

es fácil construir bucles que terminan en la $\mathcal{LH}$, pero que no terminan en la $\mathcal{LD}$. Podemos obtener una $\mathcal{LH}$ correcta cra *$\mathcal{LD}$ si modificamos la regla del bucle en la forma*

$$\frac{\{I \wedge b \wedge t = k\} S \{I \wedge t < k\} \qquad [I \wedge b \Rightarrow t > 0]}{\{I\} while\ b\ do\ S \{I \wedge \neg b\}}\ (rep)$$

Mientras I se llama la parte invariante, algunos autores denominan a t la parte variante.

Demostración.– Para simplificar la prueba probaremos antes la equivalencia

$$\forall k : k \in \mathbb{Z} : t \leq k \Rightarrow S.(t < k) \equiv \forall k : k \in \mathbb{Z} : t = k \Rightarrow S.(t < k) \qquad (*)$$

La prueba de $\Rightarrow$ es inmediata:

$$
\begin{array}{ll}
 & \forall k : k \in \mathbb{Z} : t \leq k \ \Rightarrow \ S.(t < k) \\
\Rightarrow & \quad \because \quad \Uparrow, \text{transitividad de } \Rightarrow \\
 & \forall k : k \in \mathbb{Z} : t = k \ \Rightarrow \ S.(t < k)
\end{array}
$$

Para probar la recíproca razonamos en la forma siguiente, para cada $k \in \mathbb{Z}$:

$$
\begin{array}{ll}
 & t \leq k \\
= & \quad \because t \text{ y } k \text{ enteras} \\
 & \ldots \vee t = -1 \vee t = 0 \vee t = 1 \vee \ \ldots \ \vee t = k \\
\Rightarrow & \quad \because \text{si ocurre } \forall k' : t = k' \ \Rightarrow \ S.(t < k') \\
 & \ldots \vee S.(t < -1) \vee S.(t < 0) \vee S.(t < 1) \vee \ \ldots \ \vee S.(t < k) \\
\Rightarrow & \quad \because S \text{ es monótona, y } \ldots \ \Rightarrow \ t < -1 \ \Rightarrow \ t < 0 \ \Rightarrow \ \ldots \ \Rightarrow \ t < k \\
 & S.(t < k)
\end{array}
$$

Ahora procedemos a la prueba de nuestro teorema. Podemos debilitar la tesis del teorema en la forma siguiente

$$
\begin{array}{ll}
 & [I \ \Rightarrow \ \mathcal{R}.C] \\
= & \quad \because t \text{ entero, de donde } [(\exists k : k \geq 0) : t \leq k \equiv Cierto\,]; \text{ definición de } \mathcal{R}.C \\
 & [I \wedge (\exists k : k \geq 0 : t \leq k) \ \Rightarrow \ (\exists k : k \geq 0 : H^k.C)] \\
\Leftarrow & \quad \because \text{CP} \\
 & \forall k : k \geq 0 : [I \wedge t \leq k \ \Rightarrow \ H^k.C]
\end{array}
$$

La implicación $[I \wedge t \leq k \ \Rightarrow \ H^k.C]$ se interpreta en la forma siguiente: si $t \leq k$, entonces el cuerpo del bucle se ejecuta a lo sumo k veces, y el valor inicial de t establece una cota del número de pasos. La prueba por inducción del último predicado sería:

CASO BASE ($k = 0$):

$$
\begin{array}{ll}
 & [I \wedge t \leq 0 \ \Rightarrow \ H^0.C] \\
= & \quad \because H^0.C \equiv \neg b \\
 & [I \wedge t \leq 0 \ \Rightarrow \ \neg b] \\
= & \quad \because \text{regla de intercambio} \\
 & [I \wedge b \ \Rightarrow \ t > 0] \\
= & \\
 & (b)
\end{array}
$$

PASO INDUCTIVO;

$$
\begin{array}{ll}
 & [I \wedge t \leq k+1 \ \Rightarrow \ H^{k+1}.C] \\
= & \quad \because \text{definición} \\
 & [I \wedge t \leq k+1 \ \Rightarrow \ \neg b \vee b \wedge S.H^k.C] \\
= & \quad \because \text{regla de intercambio} \\
 & [I \wedge b \wedge t \leq k+1 \ \Rightarrow \ S.H^k.C] \\
\Leftarrow & \quad \because \quad \Downarrow (c), (*), \text{transitividad} \\
 & [I \wedge b \wedge S.(t < k+1) \ \Rightarrow \ S.H^k.C] \\
\Leftarrow & \quad \because \quad \Downarrow (a), \text{transitividad} \\
 & [S.I \wedge S.(t < k+1) \ \Rightarrow \ SI.H^k.C] \\
\Leftarrow & \quad \because S \text{ es conjuntiva y monótona} \\
 & [I \wedge t < k+1 \ \Rightarrow \ H^k.C] \\
= & \quad \because t, k \text{ enteros} \\
 & [I \wedge t \leq k \ \Rightarrow \ H^k.C]
\end{array}
$$

TEOREMA

NOTA 6.40 *Podemos reemplazar* (b) *por la condición* $[I \wedge b \ \Rightarrow \ t > K]$, *para cierto* K *prefijado. En este caso,* $t - K$ *es un contador.*

6.41 *[272] Demostrad el Teorema 6.38 utilizando directamente* (c)*. Es decir, sin utilizar la forma equivalente dada por* $(*)$*.*

Estudio práctico de contadores A la vista de la condición (c) del Teorema 6.38, nos interesa introducir la siguiente definición:

DEFINICIÓN 6.42 *Siendo $t : \mathcal{E} \to \mathcal{C}$, definimos la funcional $wdec$ como*

$$wdec(S,t) \doteq \forall k : k \in \mathcal{C} : t = k \Rightarrow S.(t < k)$$

En el caso $\mathcal{C} = \mathbb{Z}$, $wdec(S_j, t)$ denota la precondición más débil que garantiza que S termina decrementando t en una unidad como mínimo. El conjunto $\mathcal{C}$ será normalmente $\mathbb{Z}$ (en este capítulo), pero cuando describamos contadores generalizados, t podrá tomar valores en otras estructuras.

Vía $wdec$, la condición (c) del Teorema 6.38 se rescribe, tomando $\mathcal{C} == \mathbb{Z}$:

$(c) \quad [I \wedge b \Rightarrow wdec(S,t)]$

Debemos pues realizar un estudio mínimo de la funcional $wdec$. Normalmente en el cuerpo de un bucle aparecen asignaciones y selectivas, por lo que el siguiente lema será de gran utilidad.

LEMA 6.43 *La función $wdec$ verifica las siguientes propiedades, ptle*

$$\begin{array}{lll}
(i) & wdec(x := E, t) & \equiv \quad (x := E.t) < t \\
(i') & wdec(x_1, x_2 := E_1, E_2 \mid t) & \equiv \quad (x_1, x_2 := E_1, E_2 \,.\, t) < t \\
(ii) & wdec(x := E; y := F \ , \ t) & \equiv \quad (x := E.y := F.t) < t \\
(iii) & wdec([\![\ \Box_{i \in \mathcal{I}} \ b_i \to S_i \ ]\!], t) & \equiv \quad OB \wedge \forall i : i \in \mathcal{I} : b_i \Rightarrow wdec(S_i, t)
\end{array}$$

Demostración.– Veamos (i); tenemos, *ptle*:

$$\begin{array}{ll}
 & wdec(x := E, t) \\
= & \quad \cdot\!:\cdot \text{ definición} \\
 & \forall k : k \in \mathcal{C} : t = k \Rightarrow x := E.(t < k) \\
= & \quad \cdot\!:\cdot \text{ definición de sustitución, } k \text{ es una variable } \textit{muda} \\
 & \forall k : k \in \mathcal{C} : t = k \Rightarrow (x := E.t) < k \\
= & \quad \cdot\!:\cdot \text{ intercambio en rango} \\
 & \forall k : t = k : k \in \mathcal{C} \Rightarrow (x := E.t) < k \\
= & \quad \cdot\!:\cdot \text{ regla puntual – Definición 1.16} \\
 & t \in \mathcal{C} \Rightarrow (x := E.t) < t \\
= & \quad \cdot\!:\cdot \, t \in \mathcal{C} \\
 & (x := E.t) < t
\end{array}$$

La prueba de (ii) es parecida. Probemos (iii); *ptle*

$$\begin{array}{ll}
 & wdec([\![\ \Box_{i \in \mathcal{I}} b_i \to S_i \ ]\!], t) \\
= & \quad \cdot\!:\cdot \text{ definición, suponiendo } k \in \mathcal{C} \text{ incluido en } t = k \\
 & \forall k : t = k : [\![\ \Box_{i \in \mathcal{I}} b_i \to S_i \ ]\!] \,.\, (t < k) \\
= & \quad \cdot\!:\cdot \text{ semántica selección} \\
 & \forall k : t = k : OB \wedge (\forall i : i \in \mathcal{I} : b_i \Rightarrow S_i.(t < k)) \\
= & \quad \cdot\!:\cdot \text{ CP} \\
 & OB \wedge \forall k : (\forall i : i \in \mathcal{I} : b_i \Rightarrow (t = k \Rightarrow : S_i.(t < k))) \\
= & \quad \cdot\!:\cdot \text{ intercambio de cuantificadores} \\
 & OB \wedge \forall i : i \in \mathcal{I} : b_i \Rightarrow (\forall k : t = k : S_i.(t < k)) \\
= & \quad \cdot\!:\cdot \text{ definición} \\
 & OB \wedge \forall i : i \in \mathcal{I} : b_i \Rightarrow wdec(S_i, t)
\end{array}$$

LEMA

La interpretación del apartado (iii) es simple: para que una sentencia selectiva decremente t, debemos asegurar que se tenga OB (para que termine correctamente) y además, cada guarda debe asegurar el decremento de t vía la sentencia que guarda.

Como aplicación del lema anterior, si $t \doteq x + y$, obtenemos, *ptle*

$$\begin{array}{lcll} wdec(x := x - 2, t) & \equiv & x - 2 + y < x + y & \equiv \quad Cierto \\ wdec(x := x + 1, t) & \equiv & x + 1 + y < x + y & \equiv \quad Falso \end{array}$$

Veamos un ejemplo más elaborado:

EJEMPLO 6.44 Sea $S \doteq [\![x > 0 \to x := x - 1 \,\square\, x > 6 \to x := x + 2]\!]$. Calculemos $wdec(S, x)$:

$$\begin{array}{ll} & wdec(S, x) \\ = & \quad \because \text{Lema 6.43}(iii) \\ & OB \wedge (x > 0 \;\Rightarrow\; wdec(x := x - 1, x)) \wedge (x > 6 \;\Rightarrow\; wdec(x := x + 2, x)) \\ = & \quad \because \text{Lema 6.43}(i) \\ & OB \wedge (x > 0 \;\Rightarrow\; x - 1 < x) \wedge (x > 6 \;\Rightarrow\; x + 2 < x) \\ = & \quad \because \text{CP} \\ & OB \wedge (x > 0 \;\Rightarrow\; C) \wedge (x > 6 \;\Rightarrow\; F) \\ = & \quad \because \text{CP} \\ & 0 < x \leq 6 \end{array}$$

que tiene tiene una interpretación sencilla: solo es posible asegurar el decremento de x seleccionando siempre la primera guarda. EJEMPLO

6.45 *[273]* **(Enero, 96)** *Calculad*

$$wdec([\![x > 1 \to x := x - 1 \,\square\, x > 0 \to x := x + 2]\!], 3x)$$

e interpretad el resultado.

6.46 **(Diciembre, 96)** *Calculad* $wdec(S, t)$ *en los casos siguientes*

S	t
$[\![x > 0 \to x := x + 1 \,\square\, x > 1 \to x := x - 2]\!]$	x
$[\![x > 1 \to x := x + 1 \,\square\, x > 0 \to x := x - 2]\!]$	$2x$
$[\![x > 0 \to y := y + 1; x := x - 1 \,\square\, x > 3 \to x := x - 1]\!]$	$40 - y$

Veamos la verificación práctica de las condiciones del Teorema 6.38 si consideramos bucles con varias guardas $*[\![\,\square\, : b_i \to S_i]\!]$:

$$[I \wedge OB \;\Rightarrow\; t > 0 \wedge SI.I \wedge wdec(SI.t < k)]$$

Vamos a describir la condición anterior en función de guardas y sentencias guardadas; por el Teorema 5.16 (fundamental de la selectiva) (página 80), los dos primeros términos se rescriben en la forma:

$$\forall j :: I \wedge b_j \;\Rightarrow\; t > 0 \wedge S_j.I$$

El tercero también es fácil:

$$\begin{array}{ll} & I \wedge OB \;\Rightarrow\; SI.t < k \\ = & \quad \because \text{Lema 6.43}(iii) \end{array}$$

$$
\begin{array}{ll}
 & I \wedge OB \;\Rightarrow\; OB \wedge \forall j : 1 \le j \le n : b_j \;\Rightarrow\; wdec(S_j, t) \\
= & \quad \because \text{CP} \\
 & \forall j : 1 \le j \le n : I \wedge b_j \;\Rightarrow\; wdec(S_j, t)
\end{array}
$$

En consecuencia, las condiciones del Teorema 6.38 para varias guardas quedan resumidas en, $\forall j$, y *ptle*:

$$
\boxed{I \wedge b_j \;\Rightarrow\; \left\{ \begin{array}{ll} S_j.I & \text{— invariabilidad} \\ wdec(S_j, t) & \text{— decremento} \\ t > 0 & \text{— terminación} \end{array} \right.}
$$

EJEMPLO 6.47 Volvamos al bucle del Ejercicio 6.37 que permite calcular el máximo y el mínimo de cuatro números enteros ordenándolos:

$$
\begin{array}{l}
q_1, q_2, q_3, q_4 := Q_1, Q_2, Q_3, Q_4; \\
\quad *[\!] \quad q_1 > q_2 \to q_1, q_2 := q_2, q_1 \\
\qquad \Box \quad q_2 > q_3 \to q_2, q_3 := q_3, q_2 \\
\qquad \Box \quad q_3 > q_4 \to q_3, q_4 := q_4, q_3 \,]\!] \\
\{q_1 = \text{mín}(Q_1, Q_2, Q_3, Q_4) \wedge q_4 = \text{máx}(Q_1, Q_2, Q_3, Q_4)\}
\end{array}
$$

Es trivial que el siguiente predicado es un invariante

$$I \;\doteq\; (q_1, q_2, q_3, q_4) \text{ es una permutación de } (Q_1, Q_2, Q_3, Q_4)$$

Luego, para probar la corrección basta probar la terminación. Como hicimos en el Ejercicio 6.37, podemos calcular los predicados $H^k.C$ y comprobar que efectivamente $H^4.C \equiv C$. Esto es muy engorroso. Otra forma más elegante usa el Teorema de los Contadores. Sea

$$t = \sum_{1 \le i < j \le 4} \delta(q_i, q_j), \qquad \text{donde} \quad \delta(x, y) \;= \left\{ \begin{array}{ll} 1, & \text{si } x > y \\ 0, & \text{si } x \le y \end{array} \right.$$

Estudiemos por ejemplo $wdec(q_1, q_2 := q_2, q_1 | t)$[2]

$$
\begin{array}{ll}
= & \quad \because \text{Lema 6.43} \\
 & (q_1, q_2 := q_2, q_1.t) < t \\
= & \quad \because \text{definición de } t \\
 & q_1, q_2 := q_2, q_1 \,. \\
 & (\delta(q_1, q_2) + \delta(q_1, q_3) + \delta(q_1, q_4) + \delta(q_2, q_3) + \delta(q_2, q_4) + \delta(q_3, q_4)) \\
< & \\
 & \delta(q_1, q_2) + \delta(q_1, q_3) + \delta(q_1, q_4) + \delta(q_2, q_3) + \delta(q_2, q_4) + \delta(q_3, q_4) \\
= & \quad \because \text{definición sustitución} \\
 & \delta(q_2, q_1) + \delta(q_2, q_3) + \delta(q_2, q_4) + \delta(q_1, q_3) + \delta(q_1, q_4) + \delta(q_3, q_4) \\
< & \\
 & \delta(q_1, q_2) + \delta(q_1, q_3) + \delta(q_1, q_4) + \delta(q_2, q_3) + \delta(q_2, q_4) + \delta(q_3, q_4) \\
= & \quad \because \text{simplificando términos iguales} \\
 & 0 < \delta(q_1, q_2) - \delta(q_2, q_1) \\
= & \quad \because \text{definición de } \delta \\
 & q_1 > q_2
\end{array}
$$

[2] $wdec(S|t)$ denota lo mismo que $wdec(S, t)$; utilizaremos esta segunda forma cuando pueda haber confusión con la coma.

y en definitiva

$$[\ q_1 > q_2 \equiv wdec(q_1, q_2 := q_2, q_1 | t)\ ]$$

e igualmente para las restantes sentencias. Lo que prueba que t es un contador, ya que trivialmente se verifica $I \wedge b_j \ \Rightarrow\ t > 0$.

Podemos aprovechar que las variables son enteras para estudiar otro contador más simple combinando de forma apropiada las variables, como por ejemplo, en la forma

$$t = 4q_1 + 3q_2 + 2q_3 + q_4$$

Es fácil ver que la función anterior es un contador; por ejemplo, estudiemos la variación de t para alguna de las sentencias

$$\begin{array}{ll} & wdec(q_1, q_2 := q_2, q_1, t) \\ = & \quad \because \text{Lema 6.43} \\ & (q_1, q_2 := q_2, q_1 . (4q_1 + 3q_2 + 2q_3 + q_4)\ ) < 4q_1 + 3q_2 + 2q_3 + q_4 \\ = & \quad \because \text{definición sustitución} \\ & 4q_2 + 3q_1 + 2q_3 + q_4 < 4q_1 + 3q_2 + 2q_3 + q_4 \\ = & \\ & q_2 < q_1 \end{array}$$

EJEMPLO

6.48 *Escribid en el lenguaje* PASCAL *un programa equivalente al bucle del Ejemplo 6.47 y demostrad su corrección.*

6.49 **(Diciembre, 01)** *Consideremos el siguiente juego: Una urna contiene inicialmente 7 bolas rojas y 7 blancas; si el número de bolas de la urna es inferior a tres, termina el juego; si es mayor que dos, se extraen tres bolas, y posteriormente se realizan las siguientes acciones.*

a.– si son del mismo color se añade una de las bolas extraídas.

b.– si son de distinto color, añadimos la bola extraída de color diferente.

Escribid un programa que simule el juego y con el que podamos probar: (1) el juego termina, y (2) termina con una bola de cada color.

6.50 *[273] Probad que* $[\ A \ \Rightarrow\ wdec(S; T,\ t)\ ]$ *es consecuencia de las propiedades*

(a) $\{A\}S\{B\}$.

(b) $[A \ \Rightarrow\ wdec(S, t)]$.

(c) $\forall t_0 : t_0 \in \mathbb{Z} : [B \wedge t < t_0 \ \Rightarrow\ T.(t < t_0)]$.

6.4. Ejemplos de diseño con contadores

EJEMPLO 6.51 (Cálculo del Máximo Común Divisor)

El MCD está definido para valores enteros $(x, y) \neq (0, 0)$ y verifica las siguientes propiedades:

(a) $MCD(x, y) = MCD(y, x)$,
(b) $MCD(x, y) = MCD(-x, y)$,
(c) $MCD(x, y) = MCD(x + y, y) = MCD(x - y, y)$,
(d) Si $x = y$ entonces $MCD(x, y) = |x|$.

Tratemos de escribir un algoritmo para calcular el MCD si la única información que tenemos de la función MCD son las propiedades $(a) - (d)$.

Si X e Y son constantes enteras, consideremos dos variables x, y con valores iniciales $(x, y) = (X, Y)$. El predicado

$$I \doteq (MCD(X,Y) = MCD(x,y))$$

es invariante para las operaciones $(a) - (c)$. Puesto que el propósito es llegar a $x = y$ para después aplicar la propiedad (d), podemos considerar varias funciones $t(x, y)$, encontrar las guardas b_j y las correspondientes secuencias guardadas S_j tales que se verifique:

$$\forall j : 1 \leq j \leq n : [I \wedge b_j \Rightarrow S_j.I \wedge wdec(S_j, t) \wedge t > 0] \qquad (d)$$

ya que de aquí concluiremos, por el Teorema 6.38 (de los contadores):

$$[I \Rightarrow \mathcal{R}.(I \wedge \neg OB)]$$

Si además $[I \wedge \neg OB \Rightarrow I \wedge x = y]$, entonces, por monotonía de $\mathcal{R}$,

$$[I \Rightarrow \mathcal{R}.(I \wedge x = y)]$$

Buscaremos un bucle con $\neg OB \equiv (x = y)$. Para simplificar el método, podemos suponer que partimos de valores no negativos, y considerar el invariante:

$$I \doteq MCD(X,Y) = MCD(x,y) \wedge x, y > 0$$

y por la propiedad (d),

$$[\, I \wedge x = y \Rightarrow x = MCD(X,Y) \,]$$

Consideremos como *candidato a contador* la función $\boxed{t = x + y}$ (al final del ejemplo se proponen como ejercicios otros contadores). Por las propiedades (a) y (c), las asignaciones:

$$\begin{array}{lll} x, y := y, x & & \\ x := x - y & x := x + y & x := y - x \\ y := x - y & y := x + y & y := y - x \end{array}$$

conservan el MCD. Pero,

$$wdec(x, y := y, x|t) \equiv y + x < x + y \equiv F$$

y la sentencia $x, y := y, x$ no garantiza el decremento de t. Por otro lado:

$$wdec(x := x + y, x + y) \equiv x + y + y < x + y \equiv y < 0$$

y la operación $x := x + y$ la excluimos ya que no podemos inferir $I \wedge b_j \Rightarrow wdec(x := x + y, t)$; por simetría también excluimos $y := x + y$. Sin embargo:

$$\begin{array}{lcl} wdec(x := x - y, t) & \equiv & x - y + y < x + y \quad \equiv \quad y > 0 \\ wdec(y := y - x, t) & \equiv & x > 0 \end{array}$$

Estudiemos la invariabilidad de I bajo estas operaciones: x:=x-y.I

$=$ $\because$ semántica

$$\begin{array}{l} \Leftarrow \\ \end{array} \begin{array}{l} MCD(X,Y) = MCD(x-y,y) \wedge x-y>0 \wedge y>0 \\ I \wedge x > y \end{array}$$

de donde, la operación $x := x - y$ debe estar protegida por la guarda $x > y$ si queremos mantener la invariabilidad de I; razonando en forma similar para la operación $y := y - x$, obtendremos el esquema

$$\begin{array}{lll} *[\!] & x > y & \rightarrow x := x - y \\ \Box & x < y & \rightarrow y := y - x \,]\!] \end{array}$$

Debemos asegurar que el invariante es cierto antes del bucle: $x, y := X, Y.I$

$$\begin{array}{l} = \\ \end{array} \begin{array}{l} MCD(X,Y) = MCD(X,Y) \wedge X, Y > 0 \\ = \\ X, Y > 0 \end{array}$$

de donde nuestro programa será:

$$\begin{array}{l} \{X, Y > 0\} \\ x, y := X, Y; \\ \quad *[\!] \quad x > y \quad \rightarrow x := x - y \\ \quad \Box \quad x < y \quad \rightarrow y := y - x \,]\!] \\ \{I \wedge x = y\}\{\Rightarrow\}\{x = MCD(X,Y)\} \end{array}$$

La elección del contador t es crucial en este desarrollo. Es fácil ver que con otra elección la síntesis del programa es diferente, y por ello también el programa final. Consideremos a modo de ejemplo el siguiente contador $\boxed{t \doteq x + 2y}$. Tenemos, *ptle*

$$\begin{array}{ll} & wdec(x, y := y, x|t) \\ = & \quad \because \text{Lema 6.43} \\ & (x, y := y, x.t) < t \\ = & \quad \because \text{sustitución} \\ & y + 2x < x + 2y \\ = & \quad \because \text{CP} \\ & x < y \end{array}$$

y la sentencia $x, y := y, x$ no altera el invariante, por lo que tenemos también el programa

$$\begin{array}{l} \{X, Y > 0\} \\ x, y := X, Y; \\ \quad *[\!] \quad x > y \quad \rightarrow x := x - y \\ \quad \Box \quad x < y \quad \rightarrow x, y := y, x \,]\!] \\ \{x = MCD(X,Y)\} \end{array}$$

que siendo menos eficiente que el anterior también es correcto. Consideremos finalmente el contador $\boxed{t \doteq \text{máx}(x, y)}$. Es fácil demostrar que en efecto es un contador ya que se cumple

$$[\, wdec(x := x - y, t) \equiv \text{máx}(x - y, y) < \text{máx}(x, y) \,]$$

y de aquí el esquema $*[\!] \, y < x \rightarrow x := x - y \,\Box\, \ldots \,]\!]$. Quedaría por demostrar que se verifica:

$$[y < x \qquad \Rightarrow \qquad \text{máx}(x - y, y) < \text{máx}(x, y) \qquad (*)]$$

Esto último es consecuencia de la siguiente implicación:

$$[a < a' \wedge a > b \;\Rightarrow\; max(a,b) \leq max(a',b)]$$

(pruébese esto último como ejercicio). Obsérvese que se cumple $[a < a' \;\Rightarrow\; max(a,b) \leq max(a',b)]$, pero en el consecuente puede darse la igualdad; por ejemplo $max(3,7) = max(4,7)$.

Obviamente también podemos considerar la guarda simétrica, de donde el programa

$$\begin{array}{l} \{X, Y > 0\} \\ x, y := X, Y; \\ \quad *[\!] \quad x > y \quad \rightarrow \quad x := x - y \\ \quad\;\; \square \quad x < y \quad \rightarrow \quad y := y - x \;[\!] \end{array}$$

ya que $\neg OB \;\Rightarrow\; x = y$. EJEMPLO

6.52 *[274] Estudiad la invariabilidad de I y la funcional $wdec(S,t)$ para las operaciones descritas por las propiedades (a)–(c) y para las siguientes funciones enteras:*

$$|x - y| \qquad x \cdot y \qquad x^y \qquad \text{mín}(x,y)$$

EJEMPLO 6.53 (Cálculo simultáneo del MCD y del mcm) Ligeros cambios en el programa anterior permiten calcular también el mínimo común múltiplo. Demostraremos la corrección del siguiente programa

$$\begin{array}{l} \{X, Y > 0\} \\ x, y, u, v := X, Y, Y, X; \\ \quad *[\!] \qquad x > y \rightarrow x, v := x - y, u + v \\ \qquad\quad \square \quad x < y \rightarrow y, u := y - x, u + v \;[\!] \\ \{x = y = MCD(X,Y) \wedge u + v = 2mcm(X,Y)\} \end{array}$$

Las nuevas variables u y v no afectan a x e y, y el predicado

$$I \;\doteq\; MCD(x,y) = MCD(X,Y) \wedge x, y > 0$$

sigue siendo invariante y el bucle termina (tal como probamos anteriormente). La idea es fortalecer el predicado I para obtener un nuevo invariante que sea función también de las variables u y v. Si trazamos las variables con la tabla:

	x	u	y	v
S_1	$x - y$	u	y	$u + v$
S_2	x	$u + v$	$y - x$	v

podemos observar que el predicado $Q \;\doteq\; (xu + yv = \text{ cte})$ es invariante bajo las sentencias guardadas ya que, *ptle*:

$$\begin{array}{ll} & x, v := x - y, u + v.(xu + yv = \text{ cte}) \\ = & \\ & (x - y)u + y(u + v) = \text{ cte} \\ = & \\ & xu + yv = \text{ cte} \end{array}$$

e igual para la otra sentencia. Por consiguiente el predicado $I \wedge Q$ es un invariante del bucle (véase Ejercicio 6.33). Para que se tenga:

$$\begin{array}{l} \{X, Y > 0\} \\ x, y, u, v := X, Y, Y, X; \\ \{xu + yv = \text{ cte }\} \end{array}$$

el valor de la constante debe ser $2XY$. Por otro lado

$$\begin{array}{ll} & I \wedge Q \wedge x = y \\ \Rightarrow & x = MCD(X,Y) \wedge x(u+v) = 2XY \\ \Rightarrow & \quad \because MCD(X,Y) * mcm(X,Y) = XY \\ & x = MCD(X,Y) \wedge u + v = 2mcm(X,Y) \end{array}$$

EJEMPLO

EJEMPLO 6.54 (Cálculo del mcm a partir de sus propiedades) Partimos de las siguientes propiedades del mcm:

$$\begin{array}{ll} (1) & a, b \leq mcm(a,b) \\ (2) & mcm(a,b) = mcm(b,a) \\ (3) & \alpha a < \beta b \leq mcm(a,b) \;\Rightarrow\; (\alpha+1)a \leq mcm(a,b) \\ (4) & \alpha a = \beta b \leq mcm(a,b) \;\Rightarrow\; \alpha a = mcm(a,b) \end{array}$$

válidas para a, b, α, β naturales, con $a, b > 0$. En el supuesto de que las propiedades anteriores constituyen la única información sobre la función mcm, vamos a escribir un programa para su cálculo. La propiedad (4) conduce a

$$\begin{array}{ll} & a \mid x \wedge b \mid y \wedge x, y \leq mcm(a,b) \wedge x = y \\ \Rightarrow & \\ & x = mcm(a,b) \end{array}$$

Si comparamos el antecedente de la implicación con la expresión $I \wedge x = y$ podríamos buscar un bucle cuyo invariante sea el predicado

$$I \doteq a \mid x \wedge b \mid y \wedge x, y \leq mcm(a,b) \wedge a, b, x, y > 0$$

de donde el esquema:

$$\begin{array}{l} \{I\} \\ *[\![\, x \neq y \rightarrow S \,]\!] \\ \{I \wedge x = y\}\{\Rightarrow\}\{R\} \end{array}$$

(1) asegura la validez del invariante tras la sentencia $x, y := a, b$. Las propiedades (3) y (2) dan lugar a las implicaciones:

$$\begin{array}{lll} a \mid x \wedge b \mid y \wedge x < y \leq mcm(a,b) & \Rightarrow & x + a \leq mcm(a,b) \\ a \mid x \wedge b \mid y \wedge y < x \leq mcm(a,b) & \Rightarrow & y + b \leq mcm(a,b) \end{array}$$

Por consiguiente:

$$\begin{array}{l} x < y \wedge I \;\Rightarrow\; x := x + a.I \\ y < x \wedge I \;\Rightarrow\; y := y + b.I \end{array}$$

y S puede ser la sentencia $[\![\, x < y \rightarrow x := x + a \,\square\, x > y \rightarrow y := y + b \,]\!]$. Pero, según el Teorema 6.10, un programa equivalente es

$$\begin{array}{ll} x, y := a, b; & \\ *[\![& x < y \rightarrow x := x + a \\ \square & y < x \rightarrow y := y + b \,]\!] \end{array}$$

y solo queda probar que el bucle termina. Por ser x e y positivos, al incrementar x o y, la suma $x + y$ se incrementa, y buscaremos un contador de la forma:

$$K - (x + y)$$

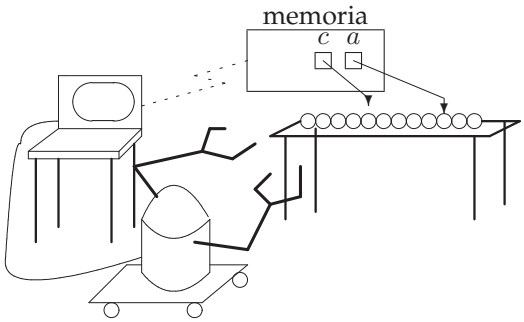


Figura 6.4: El robot ordena las bolas según los colores de la bandera nacional holandesa.

Tenemos que seleccionar K de forma que $[I \wedge x \neq y \ \Rightarrow \ t > 0]$. Teniendo en cuenta que $mcm(a,b) \leq ab$, basta tomar $t \doteq 2ab - (x+y) + 1$. En efecto; *ptle*

$$
\begin{array}{ll}
 & wdec(x := x + a, t) \\
= & \quad \because \text{Lema 6.43} \\
 & x := x + a.t < t \\
= & \quad \because \text{definición de } t \\
 & 2ab - (x + a + y) + 1 < 2ab - (x + y) + 1 \\
= & \\
 & a > 0
\end{array}
$$

y de la misma forma $[\ wdec(y := y + b, t) \ \Leftarrow \ b > 0\ ]$, luego:

$$[\ I \ \Rightarrow \ wdec(SI, t) \wedge t > 0\ ]$$

y según el Teorema de los Contadores (Teorema 6.38), $[I \ \Rightarrow \ \mathcal{R}.C]$. EJEMPLO

EJEMPLO 6.55 (El problema de la Bandera Nacional Holandesa) [3]
Sea una colección de n bolas de los colores R (Rojo), B (Blanco) y A (Azul) colocadas sobre una tabla con n agujeros alineados tal como la Figura 6.4. Un robot que puede manipular las bolas con dos brazos articulados está controlado por un computador; se trata de escribir un programa para que el robot ordene las bolas según los colores de la bandera nacional holandesa; es decir, aparecerán ocupando las primeras posiciones las rojas, a continuación las blancas y posteriormente las azules.

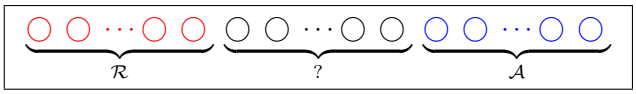
Supondremos que nuestro lenguaje tiene dos primitivas especiales (numeramos las bolas de izquierda a derecha con los números $0, 1, \ldots, n-1$):

- ✓ $color(i)$: un ojo móvil del robot se coloca frente a la bola que ocupa el lugar i-ésimo y detecta su color; tal expresión será una *función* de nuestro lenguaje y devuelve un dato de tipo *Color*.
- ✓ $inter(i,j)$: dos brazos mecánicos intercambian las bolas situadas sobre las posiciones i y j (si éstas son de distinto color).

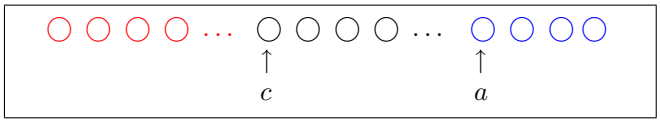
[3] Propuesto por W. Feijen, aparece analizado en [Dijkstra y Feijen, 1988].

Se supondrá que nuestro computador no tiene memoria suficiente para memorizar los colores de las bolas y no podemos resolver el problema ordenando un array de colores. Mostraremos que se puede resolver el problema con solo tres variables.

Una simplificación del problema Consideremos en primer lugar que las bolas son de dos colores solamente, R y A y queremos colocarlas en el orden $R \ldots RA \ldots A$ (las rojas seguidas de las azules); podemos considerar la ristra dividida en tres grupos: $\mathcal{R}$ (ya estudiadas como rojas), $\boxed{?}$ (desconocidas) y $\mathcal{A}$ (ya estudiadas como azules):



Inicialmente los grupos $\mathcal{R}$ y $\mathcal{A}$ son vacíos y la sección central ocupará toda la tabla; en cada paso tomamos una bola del grupo central y, una vez analizado su color, la añadimos al grupo correspondiente. Según las limitaciones de nuestro robot, añadir una bola a un grupo será fácil si tal operación es controlable por el programa con pocas sentencias (de otra forma habría que realizar muchos movimientos de bolas); es curioso observar que las tres secciones están perfectamente determinadas por dos variables c y a que indican las posiciones de la primera bola del centro y la primera de la sección de las azules:



Así, si consideramos el predicado:

$$\mathcal{C}(p,q,K) \doteq (\forall i : p \leq i \leq q : color(i) = K)$$

la situación de la figura anterior se captura con el predicado

$$I \doteq 0 \leq c \leq a \leq n \wedge \mathcal{C}(0, c-1, R) \wedge \mathcal{C}(a, n-1, A)$$

Tal predicado queremos que sea invariante de un bucle que en cada paso reduce el número $a - c$ de bolas centrales en al menos una unidad. Si consideramos el contador $\boxed{t \doteq a - c}$ y observamos que las secciones $\mathcal{R}$ y $\mathcal{A}$ son vacías para los valores $a = n$ y $c = 0$, un esquema de nuestro programa será:

$$\begin{array}{l} \{n \geq 0\} c, a := 0, n; \{I\} \\ *[\![\, c \neq a \rightarrow \textit{decrementar } a - c \textit{ con invariabilidad de } I \,]\!] \\ \{I \wedge c = a\} \end{array}$$

de forma que $I \wedge c = a$ resuelve el problema: si $c = a$ la sección $\boxed{?}$ es vacía.

Las operaciones más simples que decrementan el contador en al menos una unidad son $c := c + 1$ y $a := a - 1$. Tenemos, *ptle*

$$\begin{array}{l} c := c + 1.I \\ = \\ 0 \leq c + 1 \leq a \leq n \wedge \mathcal{C}(0, c, R) \wedge \mathcal{C}(a, n-1, A) \end{array}$$

$\Leftarrow$ $\quad\because$ busquemos el invariante
$0 \leq c \leq a \leq n \wedge c \neq a \wedge \mathcal{C}(0, c-1, R) \wedge color(c) = R \wedge \mathcal{C}(a, n-1, A)$
$\Leftarrow$
$I \wedge c \neq a \wedge color(c) = R$

y por tanto $[\ I \wedge c \neq a \wedge color(c) = R \ \Rightarrow\ c := c + 1.I\ ]$. Si el cuerpo del bucle es una sentencia selectiva, tendrá el aspecto:

$$*[\![\, c \neq a \rightarrow \quad [\!] \quad color(c) = R \quad \rightarrow \quad c := c + 1 \quad [\!] \quad color(c) = A \quad \rightarrow \quad ?\,]\!]$$

Es decir: el robot estudia la bola situada en la posición c y el programa incrementará el valor de c si la bola está bien colocada. Si la bola es azul tendrá que colocarla en la cabecera de las azules. Veremos que se verifica

$$[\ I \wedge c \neq a \wedge color(c) = A \qquad \Rightarrow \qquad a := a - 1; inter(c, a).I\ ]$$

El transformador de predicados de la sentencia $inter(c, a)$ Con objeto de probar la implicación anterior analicemos el transformador de $inter(c, a)$. Tal sentencia solo afecta al mundo exterior: no afecta a las variables internas del programa, pero puede alterar los predicados que hagan referencia al color de las bolas situadas en las posiciones a y c. Así, debemos admitir:

$$(v = v_0 \wedge color(c) = X) \ \equiv\ inter(c, a).(v = v_0 \wedge color(a) = X)$$

donde v es cualquier variable y X cualquier color. Dicho esto, tendremos, *ptle*

$a := a - 1; inter(c, a)\,.\,I$
$=$ $\quad\because$ definición de I
$a := a - 1.inter(c, a).$
$\quad 0 \leq c \leq a \leq n \wedge \mathcal{C}(0, c-1, R) \wedge color(a) = A \wedge \mathcal{C}(a+1, n-1, A)$
$=$
$a := a - 1.$
$\quad 0 \leq c \leq a \leq n \wedge \mathcal{C}(0, c-1, R) \wedge color(c) = A \wedge \mathcal{C}(a+1, n-1, A)$
$=$
$0 \leq c \leq a - 1 \leq n \wedge \mathcal{C}(0, c-1, R) \wedge color(c) = A \wedge \mathcal{C}(a, n-1, A)$
$\Leftarrow$
$I \wedge c \neq a \wedge color(c) = A$

Así, el programa siguiente es correcto y resuelve el problema para dos colores:

$$\begin{array}{l} c, a := 0, n; \\ \{I\} \\ \quad *[\![\, c \neq a \rightarrow \quad [\!] \quad color(c) = R \quad \rightarrow \quad c := c + 1 \\ \qquad\qquad\quad [\!] \quad color(c) = A \quad \rightarrow \quad a := a - 1; inter(c, a)\,]\!] \\ \quad]\!]\,\{I \wedge c = a\} \end{array}$$

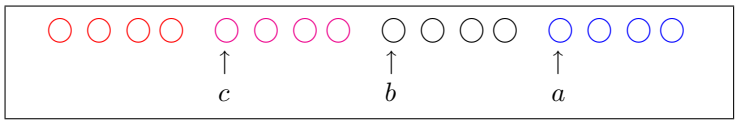
La sentencia selectiva no aborta el programa ya que estamos suponiendo que solamente existen bolas de dos colores; en cualquier caso podría añadirse tal condición al invariante I; no lo hacemos para simplificar la descripción.

El problema para tres colores Consideremos ahora tres colores para las bolas (el problema original). La estrategia seguida para dos colores puede generalizarse considerando cuatro secciones de bolas: $\mathcal{R}, \mathcal{B}, \mathcal{A}$ (las ya estudiadas rojas, blancas y azules, resp.) y $\boxed{?}$ (desconocidas); sabemos que el orden entre las

secciones $\mathcal{R}, \mathcal{B}$ y $\mathcal{A}$ debe ser RBA, pero la colocación de la sección $\boxed{?}$ entre las tres puede hacerse de cuatro formas diferentes:

$$\boxed{?}RBA \qquad R\boxed{?}BA \qquad RB\boxed{?}A \qquad RBA\boxed{?}$$

¿Cuál de ellas es la mejor? La primera y la cuarta son simétricas y por lo tanto tendrán asignados el mismo promedio de operaciones, al igual que la segunda y la tercera. Estudiemos en primer lugar la distribución $R\boxed{?}BA$; seleccionamos entonces tres variables para el comienzo de cada sección:



y consideramos el invariante:

$$I \doteq 0 \leq c \leq b \leq a \leq n \wedge \mathcal{C}(0, c-1, R) \wedge \mathcal{C}(b, a-1, B) \wedge \mathcal{C}(a, n-1, A)$$

que es cierto después de la asignación inicial $c, b, a := 0, n, n$ (las tres secciones $\mathcal{R}, \mathcal{B}$ y $\mathcal{A}$ son vacías y la sección $\boxed{?}$ es toda la ristra). En cada paso del bucle trataremos de decrementar el número de bolas $b - c$ de la sección $\boxed{?}$; por tanto tenemos el esquema:

$$\begin{array}{l} c, b, a := 0, n, n; \{I\} \\ *[\![\, c \neq b \rightarrow \textit{decrementar } t = b - c \textit{ con invariabilidad de } I \,]\!] \\ \{I \wedge b = c\} \end{array}$$

Entre todas las bolas ¿cuál estudiaremos en cada paso? Una respuesta la encontraremos analizando las sentencias más sencillas que decrementan el contador:

$$b := b - 1 \qquad\qquad c := c + 1$$

Tomemos la primera

$$\begin{array}{ll} & b := b - 1.I \\ = & 0 \leq c \leq b - 1 \leq a \leq n \wedge \\ & \mathcal{C}(0, c-1, R) \wedge \mathcal{C}(b-1, a-1, B) \wedge \mathcal{C}(a, n-1, A) \\ = & \quad \because \text{busquemos el invariante} \\ & 0 \leq c \leq b - 1 \leq a \leq n \wedge \\ & \mathcal{C}(0, c-1, R) \wedge \mathcal{C}(b, a-1, B) \wedge \mathit{color}(b-1) = B \wedge \mathcal{C}(a, n-1, A) \end{array}$$

de donde:

$$[\, I \wedge c \neq b \wedge \mathit{color}(b-1) = B \qquad \Rightarrow \qquad b := b - 1.I \,]$$

y tenemos una guarda para una sentencia selectiva que constituye el cuerpo del bucle; es fácil ver que las restantes sentencias y guardas son

$$\begin{array}{lll} [\!] & \mathit{color}(b-1) = B & \rightarrow \quad b := b - 1 \\ [\!] & \mathit{color}(b-1) = A & \rightarrow \quad a, b := a - 1, b - 1; \mathit{inter}(a, b) \\ [\!] & \mathit{color}(b-1) = R & \rightarrow \quad \mathit{inter}(b-1, c); c := c + 1 \,]\!] \end{array}$$

y de aquí el programa final. Si hubiésemos estudiado la sentencia $c := c + 1$ habríamos obtenido:

$$\begin{array}{ll} & c := c + 1.I \\ = & \\ & 0 \leq c + 1 \leq b \leq a \leq n \wedge \mathcal{C}(0, c, R) \wedge \\ & \mathcal{C}(b, a - 1, B) \wedge \mathcal{C}(a, n - 1, A) \\ = & \\ & 0 \leq c + 1 \leq b \leq a \leq n \wedge \mathcal{C}(0, c - 1, R) \wedge \\ & color(c) = R \wedge \mathcal{C}(b, a - 1, B) \wedge \mathcal{C}(a, n - 1, A) \end{array}$$

de donde $[\ I \wedge c \neq b \wedge color(c) = R \quad \Rightarrow \quad c := c + 1.I\ ]$ y es fácil ver que el cuerpo del bucle es la sentencia:

$$\begin{array}{lll} [\![& color(c) = B & \rightarrow \quad b := b - 1; inter(b, c) \\ \square & color(c) = A & \rightarrow \quad a, b := a - 1, b - 1; inter(a, b); inter(c, a) \\ \square & color(c) = R & \rightarrow \quad c := c + 1\ ]\!] \end{array}$$

Si la distribución inicial de colores es aleatoria, la sentencia anterior realiza (en media) un intercambio adicional. EJEMPLO

6.56 *Derivar un programa si seleccionamos la distribución* $\boxed{?}RBA$.

6.5. Algunos ejemplos de verificación

Analicemos otros ejemplos de verificación en los cuales el problema esencial es la terminación. Comenzamos con uno sencillo.

EJEMPLO 6.57 Queremos verificar el triplete

$$\begin{array}{l} m := a; \{m = a\} \\ \quad *[\![\quad b > m \rightarrow m := b \\ \quad\ \square \quad c > m \rightarrow m := c\ ]\!] \\ \{m = \text{ máximo}(a, b, c)\} \end{array}$$

La poscondición se escribe:

$$\begin{array}{ll} & (m = a \vee m = b \vee m = c) \wedge a \leq m \wedge b \leq m \wedge c \leq m \\ = & \\ & (m = a \vee m = b \vee m = c) \wedge a \leq m \wedge \neg(b > m \vee c > m) \\ = & \\ & (m = a \vee m = b \vee m = c) \wedge a \leq m \wedge \neg OB \end{array}$$

que coincide con el predicado $I \wedge \neg OB$ si tomamos el candidato a invariante:

$$I \doteq (m = a \vee m = b \vee m = c) \wedge a \leq m$$

que trivialmente es cierto al principio. Habrá que demostrar la invariabilidad de I y la terminación; tenemos, *ptle*, $m := b\,.\,I$

$$\begin{array}{ll} = & \quad \because \text{semántica} \\ & (b = a \vee b = b \vee b = c) \wedge a \leq b \\ = & \quad \because \text{cálculo} \\ & a \leq b \\ \Leftarrow & \\ & I \wedge m < b \end{array}$$

y de la misma forma [$I \wedge m < c \;\Rightarrow\; m := c.I$]. Luego I es un invariante. Si las variables son enteras, podemos tomar como contador

$$t = |a| + |b| + |c| - m + 1$$

que es trivialmente positivo en el entorno $m = a \vee m = b \vee m = c$, y además, $wdec(m := b, t)$

$$\begin{array}{ll} = & \quad \because \text{Lema 6.43} \\ & (m := b.t) < t \\ = & \quad \because \text{definición de } t \\ & |a| + |b| + |c| - b < |a| + |b| + |c| - m \\ = & \\ & b > m \end{array}$$

e igualmente, por simetría, [$c > m \;\Rightarrow\; wdec(m := c, t)$]. Si las variables no fueran enteras podemos tomar,

$$t = \delta(b, m) + \delta(c, m), \qquad \text{donde } \delta(x, y) = \begin{cases} 1, & \text{si } x > y \\ 0, & \text{si } x \leq y \end{cases}$$

Obsérvese que t mide el grado de *desorden* o la *distancia* al predicado $b, c \leq m$. La siguiente tabla contiene los valores de t en función de los valores de las variables para un caso concreto:

	b	c	m	t
	6	3	1	2
si tomamos la 1ª guarda	6	3	6	0
si tomamos la 2ª guarda	6	6	3	1

y observamos que la primera sentencia guardada decrementa t en dos unidades. Probemos el decremento de t:

$$\begin{array}{ll} & b > m \wedge (m := b.t) < t \\ = & \\ & b > m \wedge \delta(b, b) + \delta(c, b) < \delta(b, m) + \delta(c, m) \\ = & \quad \because \text{definición de } \delta \\ & b > m \wedge \delta(c, b) < 1 + \delta(c, m) \\ = & \quad \because \text{Si } c \leq b, \text{ el 2º término es trivialmente cierto, y si } c > b,\ \delta(c, m) = 1 \\ & b > m \end{array}$$

Es decir, $[b > m \equiv b > m \wedge (m := b.t) < t]$, y por la regla de oro, $[b > m \;\Rightarrow\; wdec(m := b, t)]$. EJEMPLO

En el ejemplo siguiente conjeturamos la función que calcula cierto algoritmo, para después estudiar un posible invariante que pruebe la conjetura.

EJEMPLO 6.58 ¿Qué función de X e Y calcula el siguiente programa

$$\begin{array}{l} \{Y \geq 0\} \\ x, y, z := X, Y, 1; \\ \quad *[\!] \quad y > 0 \wedge y \; par \quad \rightarrow y, x := y/2, x * x \\ \quad\; [\!] \quad y > 0 \qquad\qquad\;\; \rightarrow y, z := y - 1, z * x \;]\!] \end{array}$$

en el cual todas las variables son enteras?

Trazándolo se observa que $x^y z$ es constante. En efecto, veamos la variación de los valores de las variables según las sentencias ejecutadas:

	x	y	z	$x^y z$	
$y, x := y/2, x * x$	x^2	$y/2$	z	$x^{2(y/2)} z$	$\equiv x^y z$
$y, z := y - 1, z * x$	x	$y - 1$	zx	$x^{y-1} zx$	$\equiv x^y z$

Dicha constante viene determinada por los valores iniciales

$$x, y, z := X, Y, 1.(x^y z) \qquad \equiv \qquad X^Y$$

Ahora es fácil probar que el predicado

$$I \doteq x^y z = X^Y \wedge y \geq 0$$

es cierto después de la asignación inicial y por tanto I es cierto antes del bucle. Si el bucle termina e I es invariante tendremos, *ptle*

$$I \wedge y \leq 0 \Rightarrow I \wedge y = 0 \Rightarrow z = X^Y$$

y el programa calcula sobre la variable z la potencia X^Y. Queda probar que cada sentencia guardada conserva la invariabilidad:

$$\begin{array}{ll} = & y, x := y/2, x * x.I \\ \Leftarrow & (x \cdot x)^{y/2} z = X^Y \wedge y/2 \geq 0 \\ & I \wedge y > 0 \wedge y \; par \end{array} \qquad \begin{array}{ll} = & y, z := y - 1, z * x.I \\ \Leftarrow & x^{y-1} \cdot z \cdot x = X^Y \wedge y - 1 \geq 0 \\ & I \wedge y > 0 \end{array}$$

La terminación del bucle viene asegurada por ser y un contador. EJEMPLO

EJEMPLO 6.59 Tratemos de probar la corrección del siguiente esquema, en el cual todas las variables son reales:

$$\begin{array}{l} \{0 < x < 1 \wedge 0 < \epsilon < 1\} \\ a, c := 1, 1 - x; \\ \quad *[\![c > \epsilon \rightarrow \quad \{ax = 1 - c, 0 < c < 1, 0 < \epsilon < 1\} \\ \qquad\qquad\quad a := a * (1 + c); c := c^2 \,]\!] \\ \{(1 - \epsilon)/x \leq a < 1/x\} \end{array}$$

En primer lugar probaremos la invariabilidad de

$$I \doteq ax = 1 - c \wedge 0 < c < 1 \wedge 0 < x < 1 \wedge 0 < \epsilon < 1$$

Tenemos, *ptle*:

$$\begin{array}{ll} \Leftarrow & a := 1; c := 1 - x.I \\ & 0 < x < 1 \wedge 0 < \epsilon < 1 \end{array}$$

$$\begin{array}{ll} & a := a * (1 + c).c := c^2.I \\ = & a(1 + c)x = 1 - c^2 \wedge 0 < c^2 < 1 \wedge 0 < x < 1 \wedge 0 < \epsilon < 1 \\ = & \quad \because 1 - c^2 = (1 + c)(1 - c) \\ & ax = 1 - c \wedge 0 < c^2 < 1 \wedge 0 < x < 1 \wedge 0 < \epsilon < 1 \\ \Leftarrow & I \end{array}$$

$$\begin{array}{ll} & |c| < \epsilon \wedge ax = 1 - c \wedge 0 < c < 1 \wedge 0 < x < 1 \\ \Rightarrow & 0 < 1 - c < 1 \wedge ax = 1 - c < 1 \wedge x > 0 \wedge 1 - \epsilon \leq 1 - c = ax \\ \Rightarrow & \end{array}$$

$(1-\epsilon)/x \leq a < 1/x$

lo que prueba la corrección parcial. Para probar la corrección total buscamos un contador; de ser $c > c^2 > c^4 > \ldots$, obtenemos

$$\log c > \log c^2 > \log c^4 > \ldots$$

y por ello

$$1/\log c < 1/\log c^2 < 1/\log c^4 < 0$$

Pero al ser $\log \epsilon < 0$, tenemos

$$\log \epsilon/\log c > \log \epsilon/\log c^2 > \log \epsilon/\log c^4 > \ldots$$

Es decir, si consideramos la función $\boxed{t = \lfloor \log \epsilon/\log c \rfloor}$, tenemos:

$$\begin{array}{ll} & (c := c^2.t) < t \\ = & \lfloor \log \epsilon/\log c^2 \rfloor < \lfloor \log \epsilon/\log c \rfloor \\ = & \quad \because \text{pongamos } u \doteq \log \epsilon/\log c \\ & \lfloor (1/2)u \rfloor < \lfloor u \rfloor \\ \Leftarrow & \quad \because x < 1 \;\Rightarrow\; \lfloor xu \rfloor \leq x \lfloor u \rfloor \\ & \lfloor u \rfloor < 2\lfloor u \rfloor \\ \Leftarrow & \lfloor u \rfloor > 0 \\ \Leftarrow & I \wedge |c| > \epsilon \end{array}$$

EJEMPLO

6.60 *Probad la corrección del siguiente esquema, en el cual todas las variables son reales:*

$$\begin{array}{l} \{0 < x < 1 \wedge 0 < \epsilon < 1\} \\ a, c := 1, 1 - x; \\ *[\![\, |c| > \epsilon \to a := a * (1 + 0{,}5 * c); c := c^2 * (0{,}75 + 0{,}25 * c) \,]\!] \\ \{(x * (1 - \epsilon) \leq a^2 < x\} \end{array}$$

Vemos a continuación el *bien conocido* algoritmo de Euclides que calcula el MCD; su corrección es simple, pero su complejidad es más complicada.

EJEMPLO 6.61 (El Algoritmo de Euclides) Probemos la corrección de:

$$\begin{array}{l} \{X > Y > 0\} \\ x, y := X, Y; \\ \{MCD(x, y) = MCD(X, Y)\} \\ *[\![\, y \neq 0 \to x, y := y, x \bmod y \,]\!] \\ \{x = MCD(X, Y)\} \end{array}$$

La precondición $X > Y > 0$ facilita la prueba de la terminación; después veremos como debilitarla. Si consideramos las propiedades

$$\begin{array}{ll} MCD(x, 0) = x, & \text{si } x > 0 \\ MCD(x, y) = MCD(y, x \bmod y), & \text{si } x \geq y > 0 \end{array}$$

es trivial que $I \doteq MCD(x, y) = MCD(X, Y) \wedge x \geq y \geq 0$ en un invariante y además $[I \wedge y = 0 \;\Rightarrow\; MCD(X, Y) = x]$. Queda probar que el programa termina. Un contador puede ser $t \doteq y$ ya que

$$
\begin{array}{ll}
 & wdec(x, y := y, x \bmod y \mid t) \\
= & \quad \because \text{Lema 6.43} \\
 & (x, y := y, x.x \bmod y.t) < t \\
= & \\
 & x \bmod y < y \\
\Leftarrow & \quad \because \text{el resto es menor que el divisor} \\
 & x \geq y > 0
\end{array}
$$

Sin embargo, consideraremos ahora el siguiente contador $\boxed{t \doteq x + y}$. Para éste tenemos:

$$
\begin{array}{ll}
 & wdec(x, y := y, x \bmod y \mid t) \\
= & \\
 & y + x \bmod y < x + y \\
= & \\
 & x \bmod y \leq x - 1
\end{array}
$$

¿Bajo qué condiciones es cierto el predicado anterior? Ello será consecuencia del siguiente resultado:

LEMA 6.62 *Si* $a \geq b \geq 1$, *entonces* $a \bmod b \leq (a-1)/2$

Demostración.–

$$
\begin{array}{ll}
 & Cierto \\
= & \quad \because a \geq b \geq 1 \\
 & a \bmod b \leq b - 1 \\
\Rightarrow & \quad \because \text{si } a \geq b \geq 1, \lfloor a/b \rfloor \geq 1, \text{ de donde: } a \bmod b = a - \lfloor a/b \rfloor b \leq a - b \\
 & a \bmod b \leq \text{mín}(b - 1, a - b) \\
\Rightarrow & \quad \because \text{ya que } \text{mín}(p, q) \leq (p + q)/2 \\
 & a \bmod b \leq (a-1)/2
\end{array}
$$

LEMA

Si consideramos los números a y b en representación binaria, el lema anterior dice que después de la asignación $a, b := b, a \bmod b$ la variable b tiene cuando menos un bit menos que a, por lo que en dos pasos se decrementa cuando menos el número de bits de cada argumento en una unidad. Así el número de operaciones no debe superar al doble del número de bits del mayor de los iniciales; más exactamente:

TEOREMA 6.63 *Dados dos enteros positivos, el algoritmo de Euclides calcula el* MCD *con a lo sumo* $\lfloor 2 \log(M+1) \rfloor$ *divisiones, donde* $M = \text{máx}(a, b)$.

Demostración.– Supongamos en primer lugar $a \geq b$; en ese caso el algoritmo calcula la sucesión:

$$a_0 = a, \;\; a_1 = b, \;\; a_2 = a_0 \bmod a_1, \ldots, \;\; a_j = a_{j-2} \bmod a_{j-1}$$

Por el lema sabemos que $a_j \leq a_{j-2}/2 - 1/2$, para $j \geq 2$, de donde

$$a_j \leq a_{j-2k}/2^k - \sum_{1}^{k} 1/2^i = (a_{j-2k} + 1)/2^k - 1$$

y por tanto:

$$a_j \leq (M+1)/2^{mj} - 1, \text{ donde } mj = \lfloor j/2 \rfloor$$

El valor máximo de j se obtiene cuando $a_j < 1$; pero tenemos:

$$(M+1)/2^{mj} < 2 \text{ sii } \lfloor 2\log(M+1)\rfloor - 2 < j$$

de donde,

- si j es par, $(M+1)/2^{mj} < 2$ sii $\lfloor 2\log(M+1)\rfloor - 2 < j$
- si j es impar, $(M+1)/2^{mj} < 2$ sii $\lfloor 2\log(M+1)\rfloor - 2 < j-1$

Luego, para cualquier j se tiene:

$$\lfloor 2\log(M+1)\rfloor - 1 < j \;\Rightarrow\; a_j < 1$$

Es decir, si $a \geq b$ existen $\lfloor 2\log(M+1)\rfloor - 1$ divisiones como máximo; si $a < b$ entonces en la primera división se llega a $a \geq b$. TEOREMA

NOTA 6.64 *La cota anterior no es muy buena. Para comprobar $MCD(5351, 8658) = 1$ se realizan 19 divisiones, y la cota del teorema es 26. Por otro lado observemos que tampoco es $\tan$ mala; si consideramos la sucesión de Fibonacci:*

$$x_1 = 1, \qquad x_2 = 2, \qquad x_{n+1} = x_n + x_{n-1},$$

entonces el número de divisiones necesario para el cálculo de MCD(x_n, x_{n+1}) es $n+1$ ya que x_{n+1} mod $x_n = x_{n-1}$, y por ello las asignaciones:

$$x_{n+1}, x_n := x_n, x_{n+1} \text{ mod } x_n$$

equivalen a $x_{n+1}, x_n := x_n, x_{n-1}$, y las n parejas

$$(x_{n+1}, x_n), (x_n, x_{n-1}), \ldots, (2, 1)$$

son parejas de números primos entre sí. Por ejemplo:

n	$MCD(x_n, x_{n+1})$	divisiones	cota del teorema
7	$MCD(21, 34)$	*8*	*10*
21	$MCD(17711, 28657)$	*22*	*29*

En todo caso ¿es mejorable tal cota? EJEMPLO

Ejercicios

6.65 *[274]* **(El Algoritmo de Euclides Extendido)** *Consideremos el problema de encontrar dos números p y q tales que se tenga:*

$$MCD(X, Y) = pX + qY$$

Si tenemos en cuenta el Ejemplo 6.61, se observa que para $y = 0$ se debe cumplir $x = pX + qY = MCD(X, Y)$, por lo que introduciremos nuevas variables y añadiremos al invariante el predicado $x = pX + qY$. Por simetría, introduciremos también dos nuevas variables r y s, para expresar y como combinación lineal de X e Y; en definitiva, consideraremos el invariante más fuerte:

$$I \doteq MCD(X, Y) = MCD(x, y) \wedge x = pX + qY \wedge y = rX + sY \wedge x \geq y \geq 0$$

y el esquema:

$$\begin{array}{l} \{X \geq Y \geq 0\} x, y := X, Y; p, q, r, s := 1, 0, 0, 1; \{I\} \\ \quad *[\![\, y \neq 0 \rightarrow \quad S; \quad x, y := y, x \text{ mod } y \,]\!] \\ \{x = MCD(X, Y) = pX + qY\} \end{array}$$

Demostrad que si S no altera las variables x e y entonces el bucle termina en, a lo sumo, $\lfloor 2\log(M+1)\rfloor$ pasos. Encontrad la sentencia S para que el programa sea correcto.

6.66 *[275] Demostrad la corrección del programa siguiente:*

$$\begin{array}{l}\{X, Y > 0\}\\ x, y, u, v := X, Y, Y, X;\\ \quad *[\![\, x \neq y \rightarrow \quad *[\![\, x > y \rightarrow x := x - y; v := u + v \,]\!];\\ \qquad\qquad\quad\; *[\![\, x < y \rightarrow y := y - x; u := u + v \,]\!]\ ]\!]\\ \{u + v = 2mcm(X, Y), x = y = MCD(X, Y)\}\end{array}$$

6.67 *[276] Probad, siendo n una variable entera, y $ptle$:*

$$*[\![\, n\ par\ \rightarrow n := n/2 \,]\!].C \ \equiv\ \exists k, p : k, p \geq 0 : n = 2^k(2p + 1)$$

y por tanto, $[n \in \mathbb{Z} \wedge n \geq 1\ \Rightarrow\ *[\![\, n$ par $\rightarrow n := n/2 \,]\!].C]$. Es decir, para $n \geq 1$ entero, el bucle siempre termina.

6.68 *Siendo S el siguiente programa*

$$\begin{array}{l}i, suma := n, n;\\ *[\![\, i > 1 \rightarrow i := i - 1; suma := suma + i \,]\!]\end{array}$$

demostrad el triplete $\{n > 0\}\ S\ \{suma = 1 + 2 + \ldots + n\}$.
AYUDA.- considere el invariante $P\ \doteq\ suma = i + \ldots + n, i \geq 1, n > 0$.

6.69 *[276]* **(El problema de la Gasolinera)** *En una carretera circular aparecen n estaciones $e_0, \ldots, e_{n-1}$ numeradas en orden creciente y sentido horario. Cada estación e_i dispone de d_i litros de gasolina y un vehículo que recorre tal carretera gasta g_i litros en el tramo $e_i e_{i+1}$; supongamos que $\sum g_i = \sum d_i$; escribid un programa que encuentre (si existe) la estación inicial desde la cual un vehículo pueda recorrer el ciclo completo si comienza repostando desde esta estación.*

6.70 *Bajo qué condiciones es correcto el siguiente esquema*

$$\begin{array}{l}x, y, z := a, b, c;\\ \quad *[\![\quad x < b + c \quad \rightarrow x := x + 1\\ \qquad \square \quad y < a + c \quad \rightarrow y := y + 1\\ \qquad \square \quad z < a + b \quad \rightarrow z := z + 1 \,]\!]\\ \{x = b + c, y = a + c, z = a + b\}\end{array}$$

6.71 *[277]* **(Junio, 00)** *Verificad el siguiente programa para multiplicar números:*

$$\begin{array}{l}\{x, y > 0\}\\ z, u := 0, x;\\ *[\![\, u \neq 0 \rightarrow z, u := z + y, u - 1 \,]\!]\\ \{z = xy\}\end{array}$$

AYUDA.- Búsquese un invariante de la forma $I \equiv z + g = xy \wedge x, y > 0 \wedge u \geq 0$, donde g es cierta función a precisar.

6.72 *[277] Probad la corrección del siguiente programa (todas las variables son enteras):*

$$\begin{array}{l}\{n \geq k \geq 0\}\\ x, y, b := n, 1, 1;\\ *[\![\, x \neq k \rightarrow b := b * x \div y;\ x, y := x - 1, y + 1 \,]\!]\\ \{b = \binom{n}{k}\}\end{array}$$

6.73 *[278] ¿Cuándo se da $\{b\} * [\![\, b \rightarrow nada \,]\!] \{Cierto\}$? ¿Qué interpretación tiene?*

6.74 *[278] Dad un ejemplo de sentencia $\mathcal{S}$ indeterminista que no termine en el entorno del predicado b; probad entonces que la sentencia $[\![b \to \mathcal{S}]\!]$ es equivalente a $*[\![b \to \mathcal{S}]\!]$ en el entorno b.*

6.75 *[279]* **(Enero, 95)** *Sea x una variable entera, y el bucle*

$$\begin{array}{rlll} \mathcal{R} \doteq & *[\![& x<0 & \to \quad x := -x \\ & \square & x>1 & \to \quad x := x-1 \\ & \square & x>2 & \to \quad x := x \div 2 \,]\!] \end{array}$$

(A) Probad la corrección del esquema $\{Cierto\}\mathcal{R}\{x = 0 \vee x = 1\}$.
AYUDA 1.- Si x_0 es el valor inicial de x, probad la invariabilidad del predicado $I \doteq |x| < |x_0| + 1$, y construir con su ayuda un contador entero en la forma

$$t(x) = \begin{cases} |x_0|+1 & \text{, si } x < 0 \\ \ldots & \text{, si } x \geq 0. \end{cases}$$

AYUDA 2.- Probad $\{x < 0\}\mathcal{R}\{x = 0 \vee x = 1\}$, $\{x \geq 0\}\mathcal{R}\{x = 0 \vee x = 1\}$, utilizando la invariabilidad de $x \geq 0$.

(B) Probad que la sentencia $\mathcal{R}$ es determinista.

6.76 **(Noviembre, 96)** *Dad un ejemplo de sentencia determinista compuesta únicamente de sentencias indeterministas.*

6.77 **(Diciembre, 98)** *Sea el programa $\mathcal{S} \doteq x, y := 10, 10; \mathcal{R}$, donde $\mathcal{R}$ es el bucle*

$$\mathcal{R} \doteq *[\![x > y \to x, y := y, x \,\square\, y > 1 \to y := y - 1 \,]\!]$$

(A) Buscad un invariante I para probar $\{I \wedge \mathcal{R}.C\}\mathcal{R}\{x = 1 \wedge y = 1\}$.

(B) Encontrad α de forma que la función $t \doteq \alpha x + y$ sea un contador entero. Demostrad $\{C\}\mathcal{S}\{x = y \wedge y = 1\}$.

6.78 *[281] Sea $\mathcal{S}$ el siguiente programa para calcular un número entero aleatorio en el rango $[0..N]$ (siendo N y x variables enteras)*

$$\begin{array}{l} x := 0; f := Cierto; \\ *[\![\quad f \to \quad f := Falso \\ \square \quad f \to \quad x := x+1; \\ \qquad\qquad [\![\quad x = N \to f := Falso \\ \qquad\qquad \square \quad x \neq N \to nada \,]\!] \qquad]\!] \end{array}$$

(A) Probad la corrección total del esquema $\{N > 0\}\mathcal{S}\{0 \leq x \leq N\}$.

(B) Probad que el programa puede no terminar si $N \leq 0$. Es más, probad que $[\, S.Cierto \equiv N > 0 \,]$.

(C) En el supuesto de que nuestro mecanismo seleccione cada guarda con idéntica probabilidad ¿cual es la distribución de probabilidades de la variable aleatoria x? es decir, calcular, $\forall k, P[x = k]$.

(D) Escribid un programa para el mismo propósito pero que calcule x con una distribución binomial.

Bibliografía

[Alagic y Arbib, 1978] Alagic, S. y Arbib, M. (1978). *The Design of Well-Structured and Correct Programs*. Springer-Verlag, New-York.

[ANSI-83, 1983] ANSI-83 (1983). Reference Manual for the Ada Programming Language. U.S. Government (Ada Joint Program Office). Reimpreso en [Horowitz, 1983].

[Apt, 1988] Apt, K. R. (1988). Proving Correctness of Concurrent Programs: A Quick Introduction. En Börger, E. (ed.), *Trends in Theoretical Computer Science*, pp. 305–345. Computer Science Press.

[Arsac, 1985] Arsac, J. (1985). Teaching Programming. En Griffiths, M. y Tagg, E. (eds.), *The role of programming in teaching Informatics. Proc. IFIP, TC3, Working Conference on Teaching Programming, París, 7–9 mayo'84*, pp. 3–6. Elsevier Science Pbl., Amsterdam.

[Babbage, 1864] Babbage, C. (1864). De la Máquina Analítica. En *Perspectives on Computer Revolution*. Prentice-Hall, New Jersey. Traducción al castellano, Alianza, Madrid (1975) de la del inglés (1970).

[Barendregt, 1984] Barendregt, H. P. (1984). *The Lambda Calculus, Its Syntax and Semantics*, volumen 103 de *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam. Edición revisada de la primera (1981).

[Berg y o., 1982] Berg, H. y o. (1982). *Formal Methods of Program Verification and Specification*. Prentice-Hall, New Jersey.

[Bird y Wadler, 1988] Bird, R. y Wadler, P. (1988). *Introduction to Functional Programming*. Pretice–Hall.

[Cauchy, 1821] Cauchy, A. L. (1821). Cours d'analyse. En *Oeuvres Complétes (II^e Série)*, volumen 3. École Royale Polytechnique. Reeditado en forma facsimilar por SAEM Thales (1999).

[Dijkstra, 1981] Dijkstra, E. (1981). Why correctness must be a mathematical concern. En Boyer, R. y Moore, J. S. (eds.), *The correctness problem in computer science*. Academic Press, London.

[Dijkstra, 1982] Dijkstra, E. (1982). The equivalence of bounded nondeterminacy and continuity. En Dijkstra, E. (ed.), *Selected Writings on Computing: A personal Perspective*, pp. 358–359. Springer-Verlag.

[Dijkstra, 1990] Dijkstra, E. (ed.) (1990). *Formal Development of Programs and Proofs*. Addison-Wesley. The Year of Programming.

[Dijkstra y Feijen, 1984] Dijkstra, E. y Feijen, W. (1984). *Een methode van programmeren*. The Hague: Academic Service. traducido al inglés en [Dijkstra y Feijen, 1988].

[Dijkstra, 1976] Dijkstra, E. W. (1976). *A Discipline of Programming*. Prentice-Hall.

[Dijkstra y Feijen, 1988] Dijkstra, E. W. y Feijen, W. (1988). *A Method of Programming*. Addison-Wesley, Massachusetts.

[Dijkstra y Scholten, 1990] Dijkstra, E. W. y Scholten, C. S. (1990). *Predicate Calculus and Program Semantics*. Springer-Verlag, New York.

[Field y Harrison, 1988] Field, A. y Harrison, P. (1988). *Functional Programming*. Addison-Wesley.

[Floyd, 1967] Floyd, R. W. (1967). Assigning Meanings to Programs. En Schwartz, J. T. (ed.), *Mathematical Aspects of Computer Science*, volumen 19 de *Symposia in Applied Mathematics*, pp. 19–32. American Mathematical Society, Providence, RI.

[Gehani y McGettrick, 1988] Gehani, N. y McGettrick, A. (1988). *Concurrent Programming*. Addison-Wesley.

[Gries, 1981] Gries, D. (1981). *The Science of Programming*. Springer-Verlag, New-York.

[Hebenstreit, 1985] Hebenstreit, J. (1985). Teaching programming to everybody, why? to whom? what? En Griffiths, M. y Tagg, E. (eds.), *The role of programming in teaching Informatics, Proceed. IFIP, TC3, Working Conference on Teaching Programming, París, 7–9 mayo, 1984*, pp. 17–21. Elsevier Science Pbl., Amsterdam.

[Hehner, 1984] Hehner, E. (1984). *The Logic of Programming*. Prentice-Hall, New Jersey.

[Hennessy, 1990] Hennessy, M. (1990). *The Semantics of Programming Languajes; An Elementary Introduction using Structural Operational Semantics*. Wiley.

[Hoare, 1969] Hoare, C. (1969). An Axiomatic Basis for Computer Programming. *Communications of the ACM*, 12(10):576–580. Reimpreso en *C.ACM*, 26(1):53-56, 1983, y también en [Hoare y Jones, 1989]:45-58.

[Hoare, 1971] Hoare, C. (1971). Computer Science. *New Lectures Series*, 62. reimpreso en [Hoare y Jones, 1989]:89–101.

[Hoare, 1978] Hoare, C. (1978). Communicating Sequential Processes. *Communications of the ACM*, 21(8). Reimpreso en [Gehani y McGettrick, 1988]:278-308, y también en [Horowitz, 1983]:311-322.

[Hoare, 1985] Hoare, C. (1985). *Communicating Sequential Processes*. Prentice–Hall, New Jersey.

[Hoare y Jones, 1989] Hoare, C. y Jones, C. (1989). *Essays in Computing Science*. Prentice-Hall.

[Horowitz, 1983] Horowitz, E. (1983). *Programming Languages. A grand Tour*. Computer Science Press.

[Horowitz y Sahni, 1978] Horowitz, E. y Sahni, S. (1978). *Fundamentals of Computer Algorithms*. Comp. Science Press.

[Huet, 1990] Huet, G. P. (1990). A Uniform approach to Type Theory. En Huet, G. (ed.), *Logical Foundations of Functional Programming*, pp. 337–397. Addison-Wesley.

[Knuth, 1968] Knuth, D. E. (1968). *The Art of Computer Programming. Vol. 1: Fundamental Algorithms*. Addison-Wesley, Massachusetts. Segunda edición (1973). Traducido al castellano en Ed. Reverté, Barcelona.

[Kowalski, 1979] Kowalski, R. (1979). *Logic for Problem Solving*. Elsevier Sc. Publ. Co. Traducción al castellano en Díaz de Santos, Madrid (1986), con el título *Lógica, Programación e Inteligencia Artificial*.

[Liskov y Zilles, 1974] Liskov, B. y Zilles, S. (1974). Programming with abstract data types. En *Proc. ACM SIGPLAN Conference on Very High Level Languages*, volumen 9, 4, pp. 50–59.

[Manna, 1974] Manna, Z. (1974). *Mathematical Theory of Computation*. McGraw-Hill.

[Meyer, 1988] Meyer, B. (1988). *Object-Oriented Software Construction*. Prentice–Hall.

[Morris, 1990] Morris, J. (1990). Programs from Specifications. En Dijkstra, E. (ed.), *Formal Development of Programs and Proofs*, pp. 81–115. Addison-Wesley. The Year of Programming.

[Nielson y Nielson, 1992] Nielson, H. y Nielson, F. (1992). *Semantics with Applications*. Wiley.

[Popek y Horning, 1977] Popek, G. y Horning, J. (1977). Notes on the Design of Euclid. *ACM SIGPLAN Notices*, 12(3):11–19.

[Ruiz Jiménez et al., 2000] Ruiz Jiménez, B. C., Gutiérrez López, F., Guerrero García, P., y Gallardo Ruiz, J. E. (2000). *Razonando con Haskell. Una Introducción a la Programación Funcional*. José E. Gallardo Ruiz (editor).

[Schmidt, 1988] Schmidt, D. (1988). *Denotational Semantics*. Allyn and Bacon.

[Shapiro, 1987] Shapiro, E. (1987). *Concurrent Prolog. Collected Papers*. MIT Press, Cambridge. Dos volúmenes.

[Sperschneider y Antoniou, 1991] Sperschneider, V. y Antoniou, G. (1991). *LOGIC. A Foundation for Computer Science*. Addison Wesley.

[Ueda, 1985] Ueda, K. (1985). Guarded Horn Clauses. Informe Técnico núm. 103, ICOT, Tokyo. También en [Shapiro, 1987]:(Vol.1,140-156).

[van Gasteren, 1990] van Gasteren, A. (1990). On the Formal Derivation of a Proof of the Invariance Theorem. En Dijkstra, E. (ed.), *Formal Development of Programs and Proofs*, pp. 49–54. Addison-Wesley. The Year of Programming.

[Wegner, 1984] Wegner, P. (1984). Capital-intensive software technology. *IEEE Software*, pp. 7–45.

[Wirth, 1973] Wirth, N. (1973). *Systematic Programming*. Prentice-Hall, New Jersey. traducción al castellano en Ed. El Ateneo, Buenos Aires (1982).

[Wirth, 1976] Wirth, N. (1976). *Algorithms + Data Structures = Programs*. Prentice-Hall, New York. traducción al castellano en Ed. del Castillo, Madrid, 1980.

[Wirth, 1983] Wirth, N. (1983). On the Design of Programming Languages. En *IFIP, 1974*, pp. 386–393. North-Holland Pub. Comp. reimpreso en [Horowitz, 1983]:23–30.

[Wirth y Hoare, 1973] Wirth, N. y Hoare, C. (1973). An Axiomatic Definition of the Programming Language PASCAL. *Acta Informatica*, 2(4):335–355. Reimpreso en [Hoare y Jones, 1989]:153–169.