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 7

Diseño de Programas con Invariantes

En este capítulo desarrollamos técnicas para la síntesis de programas con bucles; en éstas se trata de modificar la poscondición final del bucle con objeto de obtener un predicado candidato a invariante; a partir de éste se forzará el diseño para obtener la invariabilidad y la terminación. La corrección final se derivará del teorema de invariantes. Las técnicas que desarrollamos son esencialmente las expuestas en [Dijkstra, 1976]:

- ✓ SUSTITUCIÓN DE UNA CONSTANTE POR UNA VARIABLE, o más general,
- ✓ SUSTITUCIÓN DE UN TÉRMINO POR UNA VARIABLE, y
- ✓ DEBILITACIÓN DE LA POSCONDICIÓN.

7.0. Sustitución de una constante por una variable

EJEMPLO 7.0 Consideremos una tabla o función entera f con dominio el subconjunto de los naturales $[0, 1, \ldots, n-1]$; se trata de encontrar un programa para calcular el máximo de la función; es decir, diseñar un mecanismo tal que, al terminar su ejecución, haga cierta la siguiente poscondición:

$$R \doteq (0 \leq k < n) \wedge (\forall i : 0 \leq i < n : f.k \geq f.i)$$

Podemos intuir que tal mecanismo puede realizarse con un bucle para el cual trataremos de encontrar un invariante; una estrategia general en la búsqueda de invariantes es sustituir una constante por una variable y añadir condiciones adicionales a la variable introducida para obtener un predicado consistente. Así, en el segundo miembro de R aparecen dos constantes: 0 y n; la elección de la constante a reemplazar conduce a distintos invariantes y por tanto a distintos programas; por ejemplo, al sustituir la constante n por j, podemos considerar como invariante:

$$I \doteq (0 \leq k < j \leq n) \wedge (\forall i : 0 \leq i < j : f.k \geq f.i)$$

del cual obtenemos $[\, I \wedge j = n \;\Rightarrow\; R \,]$ y el problema se reduce a encontrar una sentencia S de forma que el siguiente esquema sea correcto:

$$\begin{array}{l} k, j := 0, 1; \{I\} \\ *[\![\, j \neq n \rightarrow \boxed{S} \,]\!] \\ \{I \wedge j = n\}\{\Rightarrow\}\{R\} \end{array}$$

Si recordamos el Teorema de Invariantes para bucles, la corrección es consecuencia de

$$\begin{array}{ll} [\, I \;\Rightarrow\; \mathcal{R}.C \,] & \text{— el bucle termina} \\ [\, I \wedge j \neq n \;\Rightarrow\; S.I \,] & \text{— } I \text{ es un invariante} \end{array}$$

Podemos aplicar el modelo de bucles con contadores y considerar el contador $\boxed{t \doteq n - j}$. Busquemos S; llama la atención la sentencia $j := j+1$ que verifica

$$\begin{array}{ll} & wdec(j := j+1, t) \\ = & \quad \text{⋱ Lema 6.43} \\ & (j := j+1.t) < t \\ = & \\ & n - j - 1 < n - j \\ = & \\ & Cierto \end{array}$$

Estudiemos la invariabilidad de I bajo la sentencia $j := j+1$; *ptle*

$$\begin{array}{ll} & j := j+1.I \\ = & \\ & (0 \leq k < j+1 \leq n) \wedge (\forall i : 0 \leq i < j+1 : f.k \geq f.i) \\ = & \quad \text{⋱ Busquemos el invariante} \\ & (0 \leq k < j+1 \leq n) \wedge (\forall i : 0 \leq i < j : f.k \geq f.i) \wedge (f.k \geq f.j) \\ \Leftarrow & \\ & I \wedge j \neq n \wedge f.k \geq f.j \end{array}$$

y obtenemos para el cuerpo de la sentencia S la secuencia con guarda:

$$[\![\, f.k \geq f.j \to j := j+1 \,\square\, \ldots \,]\!]$$

Esta única guarda no garantiza la terminación correcta del bucle; si ocurre $f.k \leq f.j$ debemos modificar el valor de k por j; es fácil comprobar:

$$[\, I \wedge j \neq n \wedge f.k \leq f.j \;\Rightarrow\; k, j := j, j+1.I \,]$$

y por tanto tenemos una primera versión de nuestro programa:

$$\begin{array}{l} k, j := 0, 1; \\ \quad *[\![\quad j \neq n \quad \to \quad [\![\quad f.k \geq f.j \quad \to \quad j := j+1 \\ \qquad\qquad\qquad\quad \square \quad f.k \leq f.j \quad \to \quad k, j := j, j+1 \,]\!] \;]\!] \end{array}$$

Aunque el programa anterior es correcto, no es eficiente: es necesario calcular $f.k$ y el correspondiente valor de $f.j$ dos veces en cada iteración; podemos formular una versión más eficiente si utilizamos variables adicionales:

$$\begin{array}{l} k, j, \text{máx} := 0, 1, f.0; \\ \quad *[\![j \neq n \to \quad h := f.j; \\ \qquad\qquad\qquad [\![\quad \text{máx} \geq h \quad \to \quad j := j+1 \\ \qquad\qquad\qquad \square \quad \text{máx} \leq h \quad \to \quad k, j, \text{máx} := j, j+1, f.j \,]\!] \\ \;]\!] \,\{\text{máx} = \text{ máximo } (f.0, \ldots, f.(n-1))\} \end{array}$$

EJEMPLO

7.1 *Si sustituimos en R la constante 0 por j obtenemos:*

$$I \;\doteq\; (0 \leq j \leq k < n) \wedge (\forall i : j < i \leq n : f.k \geq f.i)$$

Deducid un programa para tal candidato a invariante.

EJEMPLO 7.2 Tratemos de escribir un programa para comprobar si todos los elementos de una tabla A son iguales a 6; es decir, para la poscondición:

$$R \doteq t\_s \equiv (\forall i : 0 \leq i < n : A[i] = 6)$$

siendo t_s (todos seis) una variable booleana. De nuevo aparecen dos constantes en la poscondición: 0 y n (6 es constante pero no tiene sentido cambiarlo por una variable). La sustitución de una de ellas conduce a distintos programas; si sustituimos n se obtiene el candidato a invariante:

$$I \doteq t\_s \equiv (\forall i : 0 \leq i < j : A[i] = 6) \wedge 0 \leq j \leq n$$

El predicado I se puede satisfacer de forma simple tomando $j = 0$ y $t\_s = Cierto$ (ya que en ese caso la expresión cuantificada es cierta):

$$\begin{array}{ll} & j, t\_s := 0, Cierto.I \\ = & \quad \because \text{semántica asignación} \\ & Cierto \equiv Cierto \wedge 0 \leq 0 \leq n \\ = & \\ & 0 \leq n \end{array}$$

y el teorema de invariantes conduce al esquema:

$$\begin{array}{l} \{n \geq 0\} j, t\_s := 0, Cierto; \{I\} \\ *[\![j \neq n \rightarrow S]\!] \\ \{I \wedge j = n\} \ \{ \Rightarrow \} \ \{R\} \end{array}$$

La guarda puede ser también $j < n$ ya que $j \geq n \wedge I \ \Rightarrow \ j = n$. La sentencia $j := j + 1$ permite considerar el contador $t \doteq n - j$. Estudiemos la invariabilidad bajo $j := j + 1$

$$\begin{array}{ll} & j := j + 1.I \\ = & \quad \because \text{semántica asignación} \\ & t\_s \equiv (\forall i : 0 \leq i < j + 1 : A[i] = 6) \wedge 0 \leq j + 1 \leq n \\ = & \quad \because \text{descomponemos el campo } 0 \leq i < j + 1 \text{ para buscar el invariante} \\ & t\_s \equiv (A[j] = 6 \wedge \forall i : 0 \leq i < j : A[i] = 6) \wedge 0 \leq j + 1 \leq n \\ \Leftarrow & \\ & I \wedge A[j] = 6 \wedge j \neq n \end{array}$$

Un esquema (todavía incompleto) sería:

$$\begin{array}{l} j, t\_s := 0, Cierto; \\ \quad *[\![j \neq n \rightarrow \ [\![\quad A[j] = 6 \rightarrow j := j + 1 \\ \qquad\qquad\qquad\quad \square \quad A[j] \neq 6 \rightarrow \ldots]\!] \qquad\qquad]\!] \end{array}$$

Si $A[j] \neq 6$ debemos cambiar el valor de t_s a $Falso$ y forzar la terminación cambiando la variable de iteración j al último valor:

$$\begin{array}{l} j, t\_s := 0, Cierto; \\ \quad *[\![j \neq n \rightarrow \ [\![\quad A[j] = 6 \quad \rightarrow j := j + 1 \\ \qquad\qquad\qquad\quad \square \quad A[j] \neq 6 \quad \rightarrow t\_s := Falso; j := n]\!] \ ]\!] \end{array}$$

Solo tendremos que probar que la segunda sentencia guardada conserva la invariabilidad y decrementa el contador; en efecto:

$$wdec(t\_s := Falso; j := n \mid t)$$

$$
\begin{array}{ll}
= & \quad \because \text{Lema 6.43, t no depende de } t\_s \\
& j := n.(n-j) < n-j \\
= & \\
& j < n \\
\Leftarrow & \quad \because I \equiv 0 \leq j \leq n \wedge \ \ldots \\
& I \wedge j \neq n
\end{array}
$$

$$
\begin{array}{ll}
& t\_s := Falso; j := n.I \\
= & \\
& Falso \equiv (\forall i : 0 \leq i < n : A[i] = 6)) \wedge 0 \leq n \leq n \\
\Leftarrow & \\
& A[j] \neq 6 \wedge I
\end{array}
$$

Curiosamente podríamos haber tomado directamente dos asignaciones

$$
\begin{array}{l}
j := 0; t\_s := \text{ cierto;} \\
*[\![\, j \neq n \rightarrow t\_s := t\_s \wedge (A[j] = 6);\ j := j + 1 \,]\!]
\end{array}
$$

En efecto

$$
\begin{array}{ll}
& t\_s := t\_s \wedge (A[j] = 6); j := j + 1 \,.\, I \\
= & \\
& t\_s \wedge (A[j] = 6) \equiv (\forall i : 0 \leq i < j + 1 : A[i] = 6) \wedge 0 \leq j + 1 \leq n \\
= & \\
& t\_s \wedge (A[j] = 6) \equiv (\forall i : 0 \leq i < j : A[i] = 6) \wedge (A[j] = 6) \\
\wedge & \\
& 0 \leq j + 1 \leq n \\
\Leftarrow & \\
& t\_s \equiv (\forall i : 0 \leq i < j : A[i] = 6) \wedge 0 \leq j < n \\
\Leftarrow & \\
& I \wedge j < n
\end{array}
$$

Pero esta solución recorre toda la tabla incluso aunque se haya detectado un valor distinto de 6. EJEMPLO

EJEMPLO 7.3 Tratemos de escribir un programa para encontrar el n–ésimo número de Fibonacci; es decir, para la poscondición $R \doteq a = f_n$, donde

$$f_0 = 0, \qquad f_1 = 1, \qquad f_n = f_{n-1} + f_{n-2}\ (n \geq 2). \tag{$*$}$$

Puesto que el cómputo de f_n necesita dos valores previos, al menos son necesarias dos variables, y modificamos la poscondición

$$R \doteq a = f_n \wedge b = f_{n-1}$$

y consideramos la precondición $n > 0$. Cambiando la constante n por una variable se obtiene el candidato a invariante:

$$I \doteq 1 \leq i \leq n \wedge a = f_i \wedge b = f_{i-1}$$

y completaremos el esquema:

$$
\begin{array}{l}
\{n > 0\} \\
i, a, b, := 1, 1, 0; \{I\} \\
*[\![\, i < n \rightarrow S \,]\!] \\
\{a = f_n \wedge b = f_{n-1}\}
\end{array}
$$

(trivialmente se deduce $[n > 0 \ \Rightarrow \ i, a, b, := 1, 1, 0.I]$). La sentencia $i := i + 1$ asegura la terminación tomando el contador $t \doteq n - i$, pero las variables a y b

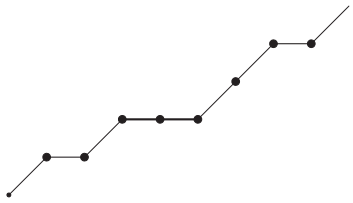

Figura 7.0: *Llanos* en una tabla ordenada.

deben ser modificadas conjuntamente con el incremento $i := i + 1$. Probemos que la sentencia $S \doteq i, a, b, := i + 1, a + b, a$ es suficiente:

$$
\begin{array}{ll}
 & i, a, b, := i + 1, a + b, a.I \\
= & \\
 & 1 \leq i + 1 \leq n \wedge a + b = f_{i+1} \wedge a = f_i \\
\Leftarrow & \quad \because \text{definición de sucesión de Fibonacci, } (*) \\
 & I \wedge i < n
\end{array}
$$

EJEMPLO

EJEMPLO 7.4 (El problema de los llanos) [Gries, 1981] Sea $b[0..n-1]$ una tabla de números enteros ordenada en orden ascendente; un *llano* (*plateau*) es una secuencia de elementos consecutivos iguales; puesto que la tabla está ordenada, $b[k..j]$ es un llano sii $b[k] = b[j]$. Se trata de escribir un programa para la poscondición:

$$R \doteq p \text{ es la longitud del llano más largo de } b[0..n-1]$$

Por ejemplo, si $b == [1, 2, 2, 3, 3, 3, 4, 5, 5, 6]$, entonces p será 3 (véase la Figura 7.0). R puede escribirse en la forma:

$$
\begin{array}{ll}
 & \text{— existe un llano de longitud } p\text{:} \\
 & \exists k : 0 \leq k \leq n - p : b[k] = b[k + p - 1] \\
\wedge & \\
 & \text{— no existen llanos de longitud } p + 1\text{:} \\
 & \forall k : 0 \leq k \leq n - p - 1 : b[k] \neq b[k + p]
\end{array}
$$

Si sustituimos n por i obtenemos el candidato a invariante

$$
\begin{array}{ll}
I \doteq & (\exists k : 0 \leq k \leq i - p : b[k] = b[k + p - 1]) \wedge \\
 & (\forall k : 0 \leq k \leq i - p - 1 : b[k] \neq b[k + p]) \wedge 0 \leq i \leq n
\end{array}
$$

Es decir, $I \Rightarrow$ 'p es la longitud del llano más largo de $b[0..i-1]$'. Por tanto completaremos el esquema:

$$
\begin{array}{l}
i, p := 1, 1; \{I\} \\
*[\![\, i < n \rightarrow \mathcal{S} \,]\!] \\
\{I \wedge i \geq n\}\ \{\Rightarrow\}\ \{R\}
\end{array}
$$

La sentencia $\mathcal{S}$ deberá *decrementar* $n - i$ *con invariabilidad de* I; tomemos la sentencia más simple

$$
\begin{array}{ll}
 & i := i+1.I \\
= & \\
 & (\exists k : 0 \le k \le i+1-p : b[k] = b[k+p-1]) \\
\wedge & \\
 & (\forall k : 0 \le k \le i-p : b[k] \ne b[k+p]) \qquad \wedge\, 0 \le i+1 \le n \\
= & \quad \because \text{busquemos el invariante} \\
 & b[i+1-p] = b[i] \vee (\exists k : 0 \le k \le i-p : b[k] = b[k+p-1]) \\
\wedge & \\
 & (b[i-p] \ne b[i] \wedge \forall k : 0 \le k \le i-p-1 : b[k] \ne b[k+p]) \wedge 0 \le i < n \\
\Leftarrow & \\
 & b[i-p] \ne b[i] \wedge I \wedge i < n
\end{array}
$$

Si $b[i-p] = b[i]$ se ha localizado un llano de longitud $p+1$; es fácil probar

$$i < n \wedge I \wedge b[i-p] = b[i] \;\Rightarrow\; p, i := p+1, i+1.I$$

de donde el programa:

$$
\begin{array}{lll}
i, p := 1, 1; & & \\
*[\![\, i < n \rightarrow & [\!] & b[i] = b[i-p] \rightarrow p, i := p+1, i+1 \\
 & \square & b[i] \ne b[i-p] \rightarrow i := i+1 \qquad\qquad]\!]\ ]\!]
\end{array}
$$

7.5 *[285] Escribid un programa para encontrar el número de llanos de una tabla ordenada.*

7.1. Debilitación de la poscondición

Otra estrategia para derivar invariantes consiste en eliminar parte de la poscondición R debilitándola; este método es particularmente útil si la poscondición R aparece descompuesta en forma natural como conjunción, $R \equiv R_1 \wedge R_2$, de forma que si queremos aplicar el Teorema de Invariantes, podemos tomar una de las partes como $\neg OB$ y la otra como el invariante: $R \equiv I \wedge \neg OB$.

Un ejemplo típico es la estrategia seguida en el Ejemplo 6.47 para la ordenación de cuatro constantes (Q_1, Q_2, Q_3 y Q_4) utilizando cuatro variables auxiliares (q_1, q_2, q_3 y q_4) de forma que la poscondición R se escribe en la forma $R_1 \wedge R_2$, siendo

$$
\begin{array}{ll}
R_1 \equiv (q_1, q_2, q_3, q_4) \text{ es una permutación de } (Q_1, Q_2, Q_3, Q_4) & \equiv I \\
R_2 \equiv q_1 \le q_2 \le q_3 \le q_4 & \equiv \neg OB
\end{array}
$$

y obtuvimos el programa de la página 109. La estrategia a tratar se basa en la descomposición 'natural' de la poscondición; a veces ésta no es tan simple.

EJEMPLO 7.6 (Cálculo del resto y cociente de una división entera) Sean a y d enteros verificando $a \ge 0 \wedge d > 0$; busquemos los números c (cociente) y r (resto) tales que:

$$a = dc + r, \qquad 0 \le r < d$$

Comencemos estudiando el cálculo del resto; tenemos como poscondición:

$$R \;\doteq\; d \mid (a-r) \wedge 0 \le r \wedge r < d$$

que puede ser debilitada fácilmente de dos formas: eliminando alguno de los últimos predicados. Si eliminamos el último obtenemos el candidato a invariante y la guarda siguientes

$$I \;\doteq\; d \mid (a-r) \wedge 0 \le r \qquad\qquad b \;\doteq\; r \ge d$$

y derivamos el siguiente esquema:

$$\begin{array}{l} r := a; \{I\} \\ *[\![\, r \geq d \rightarrow S \,]\!] \\ \{I \wedge \neg OB\}\ \{\equiv\}\ \{R\} \end{array}$$

Al definir la secuencia S debemos asegurar la invariabilidad de I tras S y la terminación; para utilizar el modelo de bucles con contador tomaremos como función monótona decreciente $\boxed{t \doteq r}$. Puesto que d es un entero positivo, éste puede servir para decrementar el valor de t:

$$\begin{array}{ll} & wdec(r := r - d, t) \\ = & \quad \because \text{Lema 6.43} \\ & r - d < r \\ = & \\ & d > 0 \end{array} \qquad \begin{array}{ll} & r := r - d.I \\ = & \quad \because \text{semántica} \\ & 0 \leq r - d \wedge d \mid (a - r + d) \\ \Leftarrow & \\ & I \wedge r \geq d \end{array}$$

Luego, si tomamos $S \doteq r := r - d$, la parte izquierda prueba la terminación del bucle y la derecha la invariabilidad.

7.7 *[285] El programa anterior es ineficiente si el cociente es grande; probad que:*

$$\begin{array}{l} \{a \geq 0 \wedge d > 0\} \\ r := a; \\ \quad *[\![\quad r \geq d \rightarrow \quad dd := d; \\ \qquad\qquad\qquad\quad *[\![\, r \geq dd \rightarrow \quad r := r - dd; \\ \qquad\qquad\qquad\qquad\qquad\quad dd := dd + dd \,]\!] \;]\!] \end{array}$$

es más eficiente y además es correcto (obsérvese que en el bucle interior se decrementa r en múltiplos de d).

Veamos ahora el cálculo del cociente. En la expresión $a = dc + r$ observamos que para el valor $r = a$, se tiene $c = 0$; por otro lado debe ocurrir:

$$\{a = d(c-1) + r\} r := r - d \{a = dc + r\}$$

Por consiguiente:

$$\{a = dc + r\} r, c := r - d, c + 1 \{a = dc + r\}$$

y podemos calcular el cociente modificando ligeramente el bucle anterior:

$$\begin{array}{l} \{a \geq 0 \wedge d > 0\} \\ r, c := a, 0 \\ \{a = dc + r, 0 \leq r\} \\ *[\![\, r \geq d \rightarrow r, c := r - d, c + 1 \,]\!] \end{array}$$

EJEMPLO

EJEMPLO 7.8 (Búsqueda lineal) Sea la tabla $b[0..m-1]$ y x un elemento suyo: $\exists i : 0 \leq i < m : x = b[i]$. Tratemos de escribir un programa para encontrar el menor índice i tal que $b[i] = x$. Si formulamos la precondición y la poscondición en la forma:

$$\begin{array}{l} P \doteq m > 0 \wedge x \in b[0..m-1] \\ R \doteq 0 \leq i < m \wedge x \notin b[0..i-1] \wedge x = b[i] \end{array}$$

vemos que la poscondición es conjunción de tres predicados. El problema esencial es seleccionar el predicado para debilitar R, ya que hay tres. Cualquier candidato a invariante que contenga el tercero ($x = b[i]$) necesitará unas sentencias de asignación que permitan su verificación y, curiosamente ¡éste es el problema inicial! En consecuencia si eliminamos éste obtenemos el candidato

$$I \doteq 0 \leq i < m \wedge x \notin b[0..i-1]$$

que, para el índice $i = 0$ es cierto; en consecuencia tenemos el esquema:

$$\begin{array}{l} \{P\} i := 0; \{I\} \\ *[\![\, x \neq b[i] \rightarrow \boxed{?} \,]\!] \\ \{I \wedge x = b[i]\}\{R\} \end{array}$$

El contador $t \doteq m - i$ es decrementado por $i := i + 1$; pero

$$i := i + 1.I \equiv 0 \leq i < m \wedge i + 1 < m \wedge x \notin b[0..i-1] \wedge x \neq b[i]$$

y no podemos probar $[\, I \wedge x \neq b[i] \Rightarrow i := i + 1.I \,]$ ya que el subpredicado $(i + 1 < m)$ no es deducible. Este inconveniente se solventa si incluimos la precondición en el invariante

$$I \doteq 0 \leq i < m \wedge x \notin b[0..i-1] \wedge x \in b[0..m-1]$$

ya que $\quad x \in b[0..m-1] \wedge x \neq b[i] \Rightarrow i + 1 < m.$ EJEMPLO

EJEMPLO 7.9 (Búsqueda lineal en una tabla ordenada) Sea una tabla $b[1..n]$ $(n > 1)$ y sea x tal que se satisface P,

$$P \doteq a \text{ ordenada } \wedge b[1] \leq x < b[n].$$

Trataremos de escribir un programa para encontrar el valor i tal que

$$R \doteq 1 \leq i < n \wedge b[i] \leq x < b[i+1].$$

La eliminación de algún término final ($b[i] \leq x$, o $x < b[i+1]$) conduce a un posible invariante. Si eliminamos el primero $I \doteq 1 \leq i < n \wedge x < b[i+1]$, y el programa:

$$\begin{array}{l} \{P\} i := n - 1; \{I\} \\ *[\![\, b[i] > x \rightarrow i := i - 1 \,]\!] \\ \{R\} \end{array}$$

cuya corrección es elemental:

$$\begin{array}{ll} & i := i - 1.I \\ = & \\ & 1 \leq i - 1 < n \wedge x < b[i] \\ \Leftarrow & \\ & x < b[i] \wedge I \wedge i \geq 2 \\ \Leftarrow & \quad \because b \text{ ordenada y } b[1] \leq x < b[n] \\ & x < b[i] \wedge I \end{array}$$

EJEMPLO

7.10 *[286] En el Ejemplo 7.8 buscamos la primera aparición de un item en una tabla ordenada $b[0..m-1]$ en el supuesto que $x \in b$. Modificad el programa si el item no necesariamente está en la tabla; es decir, escribid un programa para la precondición y poscondición siguientes:*

$$\begin{array}{lcll} P & \doteq & m > 0 & \\ R & \doteq & & x \in b[0..m-1] \wedge 0 \leq i < m \wedge x \notin b[0..i-1] \wedge x = b[i] \\ & & \vee & x \notin b[0..m-1] \wedge i = m \end{array}$$

EJEMPLO 7.11 Tratemos de escribir un programa para calcular en la variable x el cardinal del conjunto $\{i : 0 \leq i < n : par\ b[i]\}$:

$$R \doteq x = (Ni : 0 \leq i < n : par\ b[i])$$

donde la expresión $Ni : 0 \leq i < n : \alpha(i)$ representa el número de elementos del subcampo $0 \leq i < n$ de los naturales que verifican el predicado $\alpha(i)$. Si sustituimos la constante n por una variable obtenemos

$$I \doteq x = (Ni : 0 \leq i < j : par\ b[i]) \wedge 0 \leq j \leq n$$

Pero, $x, j := 0, 0.I$

$$\begin{array}{ll} & \\ = & 0 = (Ni : 0 \leq i < 0 : par\ b[i]) \wedge 0 \leq n \\ \Leftarrow & n \geq 0 \end{array}$$

de donde el esquema:

$$\begin{array}{l} \{n \geq 0\} x, j := 0, 0 \{I\}; \\ *[\![\, j < n \rightarrow S \,]\!] \\ \{I \wedge j \geq n\}\ \{\ \Rightarrow\ \}\ \{R\} \end{array}$$

La sentencia $j := j + 1$ conserva el invariante bajo cierta guarda:

$$\begin{array}{ll} & j := j + 1.I \\ = & x = (Ni : 0 \leq i < j + 1 : par\ b[i]) \wedge 0 \leq j < n \\ = & \quad \because \text{propiedades del cardinal} \\ & x = (Ni : 0 \leq i < j : par\ b[i]) \wedge impar\ b[j] \wedge 0 \leq j < n \\ \Leftarrow & I \wedge j < n \wedge impar\ b[j] \end{array}$$

Para la guarda $par\ b[j]$ debemos incrementar x:

$$\begin{array}{ll} & x, j := x + 1, j + 1.I \\ = & x + 1 = (Ni : 0 \leq i < j + 1 : par\ b[i]) \wedge 0 \leq j < n \\ = & x + 1 = (Ni : 0 \leq i < j : par\ b[i]) + 1 \wedge par\ b[j] \wedge 0 \leq j < n \\ \Leftarrow & I \wedge j < n \wedge impar\ b[j] \end{array}$$

de donde el programa:

$$\begin{array}{l} x, j := 0, 0; \\ \quad *[\![\, j < n \rightarrow \quad [\![\quad par\ b[j] \quad \rightarrow \quad x, j := x + 1, j + 1 \\ \qquad\qquad\quad \Box \quad impar\ b[j] \quad \rightarrow \quad j := j + 1 \,]\!] \qquad\qquad]\!] \end{array}$$

Veamos otros ejemplos de estrategia de *debilitación de la poscondición*.

EJEMPLO 7.12 Sean tres tablas $a[0..m], b[0..n], c[0..p]$ ordenadas en orden ascendente. Se sabe que existe un item x común a las tres tablas. Se trata localizar la primera aparición de x. Si m_i, m_j y m_k son los menores índices tales que:

$$a[m_i] = b[m_j] = c[m_k]$$

el problema se escribe

$$\begin{array}{l} \{P\}(\ \doteq\ \exists m_i, m_j, m_k \wedge \text{ las tablas } a, b, c \text{ estan ordenadas}) \\ S \\ \{R\}(\ \doteq\ i = m_i \wedge j = m_j \wedge k = m_k) \end{array}$$

y tenemos la poscondición descompuesta en forma natural como conjunción, por lo que completaremos el esquema:

$$
\begin{array}{l}
\{P\} \\
i, j, k := 0, 0, 0; \\
\{I\}(\ \doteq\ 0 \leq i \leq m_i \wedge 0 \leq j \leq m_j \wedge 0 \leq k \leq m_k) \\
*[\!] \ i < m_i \vee j < m_j \vee k < m_k \rightarrow \\
\qquad \textit{decrementar } t\ (\ \doteq\ m_i + m_j + m_k - i - j - k) \\
\qquad \textit{con invariabilidad de } I \qquad\qquad [\!] \\
\{R\}
\end{array}
$$

La sentencias más simples que decrementan t son

$$i := i + 1 \qquad j := j + 1 \qquad k := k + 1$$

Por simetría es suficiente estudiar la primera

$$
\begin{array}{ll}
& i := i + 1.I \\
= & \\
& 0 \leq i + 1 \leq m_i \wedge 0 \leq j \leq m_j \wedge 0 \leq k \leq m_k \\
\Leftarrow & \\
& i < m_i \wedge I
\end{array}
$$

En el código del programa no deben aparecer referencias a m_i, m_j y m_k, ya que son los valores que queremos calcular. Por tanto la guarda $i < m_i$ es inadmisible. Pero

$$
\begin{array}{ll}
& a[i] < b[j] \wedge i \geq m_i \wedge j \leq m_j \\
\Rightarrow & \quad \because \text{la tabla } b \text{ es ascendente} \\
& a[i] < b[j] \wedge b[j] \leq b[m_j] = a[m_i] \leq a[i] \\
\Rightarrow & \\
& a[i] < a[i] \\
= & \\
& Falso
\end{array}
$$

de donde el predicado $a[i] < b[j] \wedge i \geq m_i \wedge j \leq m_j$ es $Falso$, y ya que I asegura $j \leq m_j$, debe tenerse $I \wedge a[i] < b[j] \ \Rightarrow\ i < m_i$, y de aquí:

$$a[i] < b[j] \wedge I \ \Rightarrow\ i := i + 1.I$$

y por simetría

$$b[j] < c[k] \wedge I \ \Rightarrow\ j := j + 1.I \qquad c[k] < a[i] \wedge I \ \Rightarrow\ k := k + 1.I$$

Pero la negación de las guardas es

$$
\begin{array}{ll}
& \neg(a[i] < b[j] \vee b[j] < c[k] \vee c[k] < a[i]) \\
= & \\
& a[i] \geq b[j] \geq c[k] \geq a[i] \\
\Rightarrow & \\
& a[i] = b[j] = c[k]
\end{array}
$$

de donde $[I \wedge \neg OB \ \Rightarrow\ R]$, y de aquí el siguiente programa correcto:

$$
\begin{array}{l}
\{P\} i, j, k := 0, 0, 0; \{I\} \\
\quad *[\!] \quad a[i] < b[j] \rightarrow i := i + 1 \\
\quad\ \square \quad b[j] < c[k] \rightarrow j := j + 1 \\
\quad\ \square \quad c[k] < a[i] \rightarrow k := k + 1\ [\!] \\
\{R\}
\end{array}
$$

EJEMPLO 7.13 (Cálculo de la raíz cuadrada) Consideremos el problema del cálculo de la raíz (cuadrada) entera de un entero $n(>0)$. La poscondición:

$$R \doteq a^2 \leq n \wedge n < (a+1)^2 \wedge n > 0$$

admite dos descomposiciones naturales, bien:

$$I \doteq a^2 \leq n \wedge n > 0 \qquad OB \doteq (a+1)^2 \leq n$$

o bien

$$I \doteq n < (a+1)^2 \wedge n > 0 \qquad OB \doteq a^2 > n$$

En ambas descomposiciones podemos establecer el invariante inicialmente vía una asignación; cada elección de la forma de debilitación conduce a un esquema distinto (suprimimos el predicado $n > 0$ para simplificar, aunque lo consideramos universalmente):

$\{n > 0\}$
$a := 0;$
$\{a^2 \leq n\}$
$*[\![(a+1)^2 \leq n \rightarrow S]\!]$
$\{a^2 \leq n \wedge (a+1)^2 > n\}$

$\{n > 0\}$
$a := n;$
$\{(a+1)^2 > n\}$
$*[\![a^2 > n \rightarrow S]\!]$
$\{a^2 \leq n \wedge n < (a+1)^2\}$

Completemos en primer lugar el esquema de la izquierda; se observa que partimos del menor valor de a y cada paso del bucle debe aumentarlo; algo que decrece cuando crece a es $\boxed{t \doteq K - a^2}$ donde K debe ser tal que

$$I \wedge OB \Rightarrow t > 0 \equiv K > a^2$$

y el propio n sirve como constante K; o sea, $t \doteq n - a^2$. Por otro lado es fácil comprobar

$$\begin{array}{lcll} wdec(a := a+1, t) & \equiv & n - (a+1)^2 < n - a^2 & \Leftarrow \quad a \geq 0 \\ a := a+1.I & \equiv & (a+1)^2 \leq n & \end{array}$$

y ambas condiciones son implicadas por $a \geq 0 \wedge (a+1)^2 \leq n$; luego una sentencia que asegura la terminación del esquema de la izquierda es $a := a + 1$.

Para el esquema de la derecha debemos decrementar el propio a, de donde el candidato a contador $\boxed{t \doteq a^2 + 1}$ trivialmente verifica $t > 0$; la sentencia más simple que decrementa el contador es $a := a - 1$; además

$$a := a - 1.I \Leftarrow a^2 > n$$

Por consiguiente podemos tomar $S == a := a - 1$. EJEMPLO

7.2. Sustitución de un término por una variable

EJEMPLO 7.14 (Cálculo de la raíz cuadrada) Los programas del Ejemplo 7.13 tienen una eficiencia pésima si n es grande. Obsérvese que da lo mismo comenzar por el mayor valor de a que por el menor. Si en la poscondición

$$R \doteq a^2 \leq n \wedge n < (a+1)^2 \wedge n > 0$$

cambiamos el término $a+1$ por otra variable b, podemos considerar dos variables a y b, una que aproxime la raíz por debajo y la otra por arriba:

$$I \doteq a^2 \leq n \wedge b^2 > n \wedge 0 \leq a < b \wedge n > 0$$

Tenemos la implicación $[I \wedge a+1 = b \Rightarrow R]$, de donde el esquema

$$\begin{array}{l} \{n>0\}a,b := 0, n+1; \{I\} \\ *[\![\, a+1 \neq b \rightarrow S \,]\!]\, \{R\} \end{array}$$

Realmente se trata de *estrechar* el intervalo $[a,b)$ (de los enteros) donde se encuentra la raíz. Para ello podemos, o aumentar a, o disminuir b, o ambas cosas; las sentencias más simples que podemos considerar son

$$a := a + d_1 \qquad\qquad b := b - d_2$$

Podemos usar un valor común $d_1 = d_2 = d$, y tendremos el esquema:

$$\begin{array}{l} \{n>0\}a,b := 0, n+1; \\ \quad *[\![\, a+1 \neq b \rightarrow \quad d := \boxed{?}; \\ \qquad\qquad\qquad\qquad [\![\quad \boxed{?} \rightarrow a := a+d \\ \qquad\qquad\qquad\qquad \square \quad \boxed{?} \rightarrow b := b-d \;]\!] \;]\!] \end{array}$$

La elección de d deberá verificar dos condiciones:

1. La terminación del bucle. Queda asegurada si $d > 0$ tomando $t \doteq b - a$.
2. La invariabilidad de I.

Veamos el comportamiento de I frente a tales sentencias

$$\begin{array}{lcl} a := a + d.I & \equiv & (a+d)^2 \leq n < b^2 \wedge 0 \leq a+d < b \\ b := b - d.I & \equiv & a^2 \leq n < (b-d)^2 \wedge 0 \leq a < b-d \end{array}$$

de lo cual

$$\begin{array}{l} I \wedge (a+d)^2 \leq n \;\Rightarrow\; a := a + d.I \\ I \wedge (b-d)^2 > n \;\Rightarrow\; b := b - d.I \end{array}$$

y obtenemos el nuevo esquema

$$\begin{array}{l} \{n>0\}a,b := 0, n+1; \\ \quad *[\![\, a+1 \neq b \rightarrow \quad d := \boxed{?}; \\ \qquad\qquad\qquad\qquad [\![\quad (a+d)^2 \leq n \rightarrow a := a+d \\ \qquad\qquad\qquad\qquad \square \quad (b-d)^2 > n \rightarrow b := b-d \;]\!] \;]\!] \end{array}$$

Para que la ejecución de la sentencia selectiva no aborte el programa debemos asegurar la disyunción de las guardas:

$$\begin{array}{ll} & (a+d)^2 \leq n \vee (b-d)^2 > n \\ = & \\ & n < (a+d)^2 \;\Rightarrow\; n < (b-d)^2 \\ \Leftarrow & \quad \because \text{transitividad de } < \\ & (a+d)^2 \leq (b-d)^2 \\ \Leftarrow & \\ & 0 \leq a+d \leq b-d \\ \Leftarrow & \\ & 2d \leq b-a \wedge I \end{array}$$

y el mejor valor de d es el mayor que verifica la desigualdad anterior; es decir, $d = (b-a) \div 2$. EJEMPLO

7.15 *[287] Si introducimos en el programa anterior la variable $c = b - a$, el hecho de tomar $d = c \div 2$ induce a tomar como valores de c las potencias de 2; utilizando el invariante:*

$$I \doteq a^2 \leq n < (a+c)^2 \wedge (\exists i : i \geq 0 : c = 2^i)$$

deducid el programa:

$$\begin{array}{l} \{n \geq 0\} a, c := 0, 1; \\ *[\![\, c^2 \leq n \rightarrow c := 2 * c \,]\!]; \\ *[\![\, c \neq 1 \rightarrow \quad c := c/2; \\ \qquad\qquad [\![\quad (a+d)^2 \leq n \rightarrow a := a + c \\ \qquad\qquad \square \quad (b-d)^2 > n \rightarrow \ nada \qquad]\!] \ ]\!] \end{array}$$

EJEMPLO 7.16 (Cálculo de raíces por el método de Newton-Raphson)

Caso real El cálculo aproximado de la raíz real $\sqrt[p]{N}$ (N entero > 1) por el método de Newton-Raphson (o de la tangente) consiste en obtener una sucesión $\{x_n\}$ monótona convergente al cero de la función

$$F(x) = x^p - N$$

donde x_0 es cierto valor inicial y, para $n > 0$, x_n es la abscisa del punto de corte de la tangente a la curva $y = F(x)$ en el punto x_{n-1}. La ecuación de la tangente en el punto $(a, F(a))$ es (si $q = p - 1$)

$$y - F(a) = (x - a)px^q$$

que corta al eje de abscisas en $x = \frac{1}{p}(\frac{N}{a^q} + qa)$, y por consiguiente, la sucesión se obtiene con la recurrencia

$$x_k \doteq f(x_{k-1}), k > 0 \qquad f(z) \doteq \frac{1}{p}(\frac{N}{a^q} + qz)$$

Puesto que

$$a - f(a) = \frac{a^p - N}{pa^q}$$

la sucesión es estrictamente decreciente si tomamos, por ejemplo, el valor inicial $x_0 = N$.

El método en el campo de los enteros Podemos intentar el cálculo de $\sqrt[p]{N}$ tomando la misma sucesión recurrente pero en el campo de los enteros:

$$x_0 = N \qquad\qquad x_k = f(x_{k-1}), k > 0$$

El siguiente lema permite tomar $f(z) = \lfloor \frac{1}{p}(\lfloor \frac{N}{a^q} \rfloor + qz) \rfloor$.

LEMA 7.17 *Si x es real y $n > 0$, se verifica*

$$\lfloor \frac{\lfloor x \rfloor + m}{n} \rfloor = \lfloor \frac{x+m}{n} \rfloor$$

Demostración.– Si $y = \lfloor \frac{\lfloor x \rfloor + m}{n} \rfloor$, entonces,

$$\lfloor x \rfloor + m = yn + r, \text{ con } 0 \leq r < n, \qquad x = \lfloor x \rfloor + \alpha, \text{ con } 0 \leq \alpha < 1$$

luego

$$\frac{x+m}{n} = y + \frac{r+\alpha}{n}$$

y además

$$0 \leq \frac{r+\alpha}{n} \leq \frac{n-1+\alpha}{n} = 1 - \frac{1-\alpha}{n} < 1$$

y por tanto

$$\lfloor \frac{x+m}{n} \rfloor = y$$

LEMA

Al tomar abscisas enteras surge un problema: la sucesión no necesariamente es monótona; por ejemplo, para $N = 7$ y $p = 3$ se obtiene una sucesión oscilante a partir del tercer término

$$7, 4, 2, 1, 3, 2, 1, 3, 2, 1, \ldots$$

También es un hecho conocido que podemos considerar solamente aquel tramo de la sucesión hasta el primer valor que verifique $x_n^p \leq N$; en ese caso la secuencia ¡finita! de términos $x_0 = N, x_1, \ldots, x_n$ es estrictamente decreciente:

$$x_n^p \leq N < x_{n-1}^p < \ldots < x_1 < x_0 = N$$

ya que $a - f(a) \geq \frac{1}{pa^q}(a^p - N)$, de donde el siguiente bucle termina

$$a := N;$$
$$*[\![\, N < a^p \rightarrow a := \lfloor \tfrac{1}{p}(\lfloor \tfrac{N}{a^q} \rfloor + qz) \rfloor \,]\!]$$

Por otro lado, no es trivial que tal bucle calcule la raíz entera, es decir, que

$$x_n^p \leq N < (x_n + 1)^p$$

Demostraremos un hecho más sorprendente:

LEMA 7.18 *Sean p, N y a enteros, $p > 1, N > 1, a > 0$ y sea*

$$b = \lfloor \frac{1}{p}(\lfloor \frac{N}{a^q} \rfloor + qa) \rfloor$$

donde $q = p - 1$; entonces $N < (b+1)^p$.

OBSERVACIÓN.– El lema prueba que el predicado $I \doteq N < (a+1)^p$ es un invariante del bucle anterior, y de aquí la corrección.

OBS

Demostración.–

$$\begin{array}{ll} & (b+1)^p > N \\ = & \quad \because \exists k : 0 \leq k < a^q : N = \lfloor N/a^q \rfloor a^q + k \\ & (b+1)^p > \lfloor N/a^q \rfloor a^q + k \\ = & \quad \because q = p - 1 \\ & qa^p + (b+1)^p > (\lfloor N/a^q \rfloor + qa)a^q + k \end{array}$$

$$
\begin{array}{ll}
= & \because \exists j : 0 \leq j < p : pb + j = \lfloor N/a^q \rfloor + qa \\
& qa^p + (b+1)^p > (pb+j)a^q + k \\
= & \\
& qa^p - p(b+1)a^q + (b+1)^p > (j-p)a^q + k \\
\Leftarrow & \\
& qa^p - p(b+1)a^q + (b+1)^p \geq 0 \wedge 0 > (j-p)a^q + k
\end{array}
$$

Pero

$$
\begin{array}{ll}
& 0 > (j-p)a^q + k \\
\Leftarrow & \quad \because p - j \geq 1 \\
& k - a^q < 0 \\
= & \\
& Cierto
\end{array}
$$

y queda probar

$$qa^p - p(b+1)a^q + (b+1)^p \geq 0$$

o lo que es igual, que $p(a) \geq 0$, siendo p el polinomio $qx^p - p\alpha x^q + \alpha^p$, con $\alpha = b + 1$. Pero este polinomio admite la raíz doble $x = \alpha$

$$p(x) = (x - \alpha)^2 s(x)$$

obteniéndose por la regla de Ruffini:

$$
\begin{array}{ll}
s(x) = qx^{q-1} + \alpha^2(q-1)x^{q-2} + \ldots + \alpha^{q-1}, & \text{para } p > 2, q > 1 \\
s(x) = 1, & \text{para } p = 2, q = 1
\end{array}
$$

El resultado sigue trivialmente si observamos que todos los coeficientes del polinomio $s(x)$ son positivos. LEMA

EJEMPLO 7.19 (Búsqueda binaria) En el Ejemplo 7.9 hemos resuelto el problema de la búsqueda lineal en una tabla $b[1..n]$ ordenada; trataremos ahora de encontrar con una complejidad $\theta(\log n)$ la misma poscondición

$$R \doteq 1 \leq i < n \wedge b[i] \leq n < b[i+1]$$

La poscondición puede debilitarse con la introducción de una variable j:

$$I \doteq 1 \leq i < j \leq n \wedge b[i] \leq x < b[j]$$

de forma que obtenemos $[I \wedge i = j - 1 \Rightarrow R]$, y el esquema:

$$
\begin{array}{l}
i, j := 1, n; \{I\} \\
*[\![\, i \neq j - 1 \rightarrow \textit{decrementar } j - i \textit{ con invariabilidad de } I \,]\!]
\end{array}
$$

Para obtener un programa con complejidad logarítmica necesitamos decrementar rápidamente $j - i$; la solución es clásica:

$$
\begin{array}{lll}
*[\![& i \neq j - 1 \rightarrow & c := (i + j) \div 2; \\
& & \{I \wedge i \neq j - 1 \wedge c = (i + j) \div 2\} \\
& & [\![\quad x < b[c] \quad \rightarrow j := c \\
& & \quad \square \quad x \geq b[c] \quad \rightarrow i := c]\!] \quad]\!]
\end{array}
$$

La terminación viene asegurada por el contador $t \doteq j - i$ ya que, por ejemplo, para la sentencia $j := c$ tendremos:

$$
\begin{array}{ll}
 & (j := c.t) < t \\
= & \\
 & c - i < j - i \\
= & \\
 & c < j \\
\Leftarrow & \\
 & I \wedge i \neq j - 1 \wedge c = (i + j) \div 2
\end{array}
$$

La invariabilidad también es fácil

$$
\begin{array}{ll}
 & j := c.I \\
= & \\
 & 1 \leq i < c \leq n \wedge b[i] \leq x < b[c] \\
\Leftarrow & \\
 & I \wedge i \neq j - 1 \wedge c = (i + j) \div 2 \wedge x < b[c]
\end{array}
$$

EJEMPLO

7.3. Problemas de recuento

En esta sección analizaremos una estrategia genérica para resolver problemas de conteo o recuento. Supongamos que queremos determinar el número de elementos de un conjunto A vía cierto bucle:

$$
\begin{array}{l}
\ldots \\
*[\![\, ? \rightarrow ? \,]\!] \\
\{k = \text{ card } A\}
\end{array}
$$

Seguiremos la siguiente variante de la estrategia de *sustitución de variables*,

AÑADIR A LA VARIABLE QUE CUENTA PARTE DEL PROBLEMA

de forma que podemos derivar el invariante:

$$I \doteq k + \boxed{\text{card } A'} = \text{ card } A \qquad \wedge \qquad Q$$

siendo Q un predicado que conserva la consistencia, y $A' \subseteq A$. Supongamos además que cierta sentencia $Init$ permite inferir $\{\ldots\}Init\{A' = A\}$ mientras que cierto predicado $\neg OB$ asegura que el conjunto A' es vacío. En ese caso seguiremos el siguiente esquema

$$
\begin{array}{l}
Init; \{A' = A\} \\
k := 0; \{k + \text{ card } A' = \text{ card } A\}\{I\} \\
*[\![\, OB \rightarrow \boxed{?} \,]\!] \\
\{\neg OB\}\ \{\Rightarrow\}\ \{A' = \emptyset\} \\
\{I \wedge A' = \emptyset\}\ \{\Rightarrow\}\ \{k = \text{ card } A\}
\end{array}
$$

Aplicaremos este esquema a un par de ejemplos típicos.

EJEMPLO 7.20 (Búsqueda en una tabla bidimensional ordenada) Sea una tabla $a[0..m-1, 0..n-1]$, $(n, m > 0)$ ordenada, es decir:

$$a[i, j] < a[i, j + 1], \qquad\qquad a[i, j] < a[i + 1, j]$$

Tratemos de contar el número de veces que aparece x en la tabla. Sea

$$
\begin{array}{rll}
n\_o(p, q, r, s) & \doteq & \textit{número de apariciones de } x \textit{ en la subtabla} \\
 & & a[p..q-1, r..s-1]
\end{array}
$$

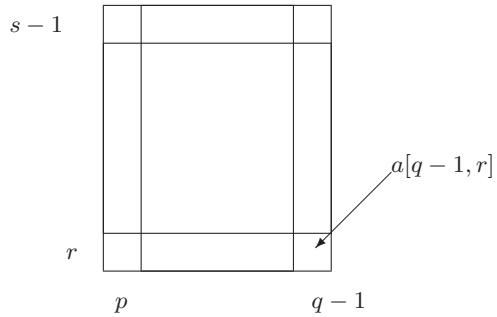

Figura 7.1: Localización del elemento $a[q-1,r]$ a estudiar

O lo que es igual:

$$n\_o(p,q,r,s) \doteq (Ni,j : p \leq i < q, r \leq j < s : x = a[i,j])$$

De la poscondición $R \doteq k = n\_o(0,m,0,n)$ inferimos un invariante

AÑADIENDO A LA VARIABLE QUE CUENTA PARTE DEL PROBLEMA

$$I \doteq k + \boxed{n\_o(p,q,r,s)} = n\_o \ \wedge \ 0 \leq p \leq q \leq m \wedge 0 \leq r \leq s \leq n$$

donde $n\_o \doteq n\_o(0,m,0,n)$ para simplificar. Si $p = q$ la tabla $a[p..q-1, r..s-1]$ es vacía, e igualmente para $r = s$. Por tanto

$$p = q \vee r = s \ \Rightarrow \ n\_o(p,q,r,s) = 0$$

y tomaremos $OB = q \neq p \wedge r \neq s$. El invariante puede obtenerse con una inicialización simple, de donde el esquema

$$\begin{array}{l} p,q,r,s,k := 0,m,0,n,0; \{I\} \\ *[\![\, q \neq p \wedge r \neq s \rightarrow S \,]\!] \\ \{R\} \end{array}$$

La subtabla $a[p..q-1, r..s-1]$ debemos *estrecharla* hasta hacerla vacía, tal como muestra la Figura 7.1. La sentencia S deberá disminuir el valor del contador $\boxed{t \doteq q-p+s-r}$ y tomamos para ello las sentencias más elementales

$$q := q-1 \qquad s := s-1 \qquad p := p+1 \qquad r := r+1$$

Como es de esperar la elección de una de ellas condiciona el resto del diseño; por ejemplo, si estudiamos la última

$$\begin{array}{l} r := r+1.I \\ = \\ k + n\_o(p,q,r+1,s) = n\_o \wedge 0 \leq p \leq q \leq m \wedge 0 \leq r+1 \leq s \leq n \end{array}$$

$\Leftarrow$ $\quad\because$ busquemos el invariante
$I \wedge r \neq s \wedge n\_o(p,q,r+1,s) = n\_o(p,q,r,s)$

Pero la tabla $a[p..q-1, r..s-1]$ coincide con la tabla $a[p..q-1, r+1..s-1]$ salvo la fila r–ésima:

$$a[p,r], a[p+1,r], \ldots, a[q-1,r]$$

cuyo máximo es $a[q-1,r]$; por tanto

$$a[q-1,r] < x \;\Rightarrow\; n\_o(p,q,r+1,s) = n\_o(p,q,r,s)$$

de donde

$$I \wedge q \neq p \wedge r \neq s \wedge a[q-1,r] < x \;\Rightarrow\; r := r+1.I$$

y tenemos el esquema incompleto

$$\begin{array}{lllll} *[\![\, q \neq p \wedge r \neq s \rightarrow & [\![& a[q-1,r] < x & \rightarrow & r := r+1 \\ & \square & a[q-1,r] = x & \rightarrow & \boxed{?} \\ & \square & a[q-1,r] > x & \rightarrow & \boxed{?} \;]\!]\;]\!] \end{array}$$

Si $a[q-1,r] = x$ entonces, por la misma razón, podemos abandonar la fila r-ésima incrementando el valor de k. Además, ya que en cada columna no hay elementos repetidos, podemos abandonar la columna $(q-1)$–ésima ya que $a[q-1,r]$ es el menor de esta columna. Es decir

$$\begin{array}{ll} & k,q,r := k+1, q-1, r+1.I \\ = & \\ & k+1+n\_o(p,q-1,r+1,s) = n\_o \\ \wedge & \\ & 0 \leq p \leq q-1 \leq m \wedge 0 \leq r+1 \leq s \leq n \\ \Leftarrow & \quad \because \text{ busquemos el invariante} \\ & I \wedge q \neq p \wedge r \neq s \wedge n\_o(p,q-1,r+1,s) = n\_o(p,q,r,s) - 1 \\ \Leftarrow & \\ & I \wedge q \neq p \wedge r \neq s \wedge a[q-1,r] = x \end{array}$$

De la misma forma

$$I \wedge q \neq p \wedge r \neq s \wedge a[q-1,r] > x \;\Rightarrow\; q := q-1.I$$

y el programa final será

$$\begin{array}{l} p,q,r,s,k := 0,m,0,n,0; \\ \quad *[\![\, q \neq p \wedge r \neq s \rightarrow \;\; [\![\quad a[q-1,r] < x \rightarrow r := r+1 \\ \qquad\qquad\qquad\quad \square \quad a[q-1,r] = x \rightarrow r,k := r+1, k+1 \\ \qquad\qquad\qquad\quad \square \quad a[q-1,r] > x \rightarrow q := q-1 \,]\!] \qquad\qquad]\!] \end{array}$$

Si hubiéramos elegido inicialmente la sentencia $p := p+1$ el elemento a estudiar hubiera sido el de la esquina superior izquierda: $a[p, s-1]$. Por el contrario, el lector puede comprobar que el estudio de las esquinas $a[p,r]$ y $a[q-1, s-1]$ no conduce a ningún programa eficiente. EJEMPLO

EJEMPLO 7.21 Sean $a[0..m-1]$ y $b[0..n-1]$ dos tablas ordenadas en sentido estricto. Calculemos el número de elementos comunes; es decir, la poscondición:

$$R \doteq k = (Ni, j : 0 \leq i < m \wedge 0 \leq j < n : a[i] = b[j])$$

Conjeturamos un invariante utilizando la técnica anterior:

AÑADIR A LA VARIABLE QUE CUENTA PARTE DEL PROBLEMA

Para ello introducimos los conjuntos

$$\begin{array}{lcl} A(p,q) & \doteq & \{(i,j) \mid 0 \leq i < p \wedge 0 \leq j < q \wedge a[i] = b[j]) \\ A & \doteq & A(m,n) \end{array}$$

de forma que la poscondición se escribe $k = \text{ card } A$ y añadimos el subproblema '$Car\ A(p,q)$' a k para obtener el candidato a invariante

$$I \doteq k + \boxed{\text{card}\, A(p,q)} = \text{ card } A \ \wedge \ 0 \leq p \leq m \wedge 0 \leq q \leq n$$

junto al esquema

$$\begin{array}{l} k,p,q := 0,m,n;\{I\} \\ *[\![p \neq 0 \wedge q \neq 0 \rightarrow \mathcal{S}]\!] \\ \{p = 0 \vee q = 0\}\ \{ \Rightarrow \}\ \{A(p,q) = \emptyset\}\ \{ \Rightarrow \}\ \{k = \text{ card } A\} \end{array}$$

De nuevo se trata de reducir la cardinalidad de A, y las sentencias más simples son $q := q - 1$ y $p := p - 1$. Pero

$$\begin{array}{ll} & p := p - 1.I \\ = & \\ & k + \text{ card } A(p-1,q) = \text{ card } A \ \wedge \ 0 \leq p-1 \leq m \wedge 0 \leq q \leq n \\ \Leftarrow & \quad \because \text{buscamos el invariante} \\ & p \neq 0 \wedge I \wedge A(p-1,q) = A(p,q) \\ \Leftarrow & \quad \because A(p,q) = A(p-1,q) \cup \{(p-1,j) \mid 0 \leq j < q \wedge a[p-1] = b[j])\} \\ & \quad \text{siendo } b[0..n-1] \textit{estrictamente creciente} \\ & p \neq 0 \wedge I \wedge a[p-1] > b[q-1] \end{array}$$

y por tanto la sentencia $\mathcal{S}$ tendrá la forma

$$\begin{array}{ll} [\![& a[p-1] > b[q-1] \rightarrow p := p-1 \\ \square & a[p-1] = b[q-1] \rightarrow \boxed{?} \\ \square & a[p-1] < b[q-1] \rightarrow \boxed{?} \ ]\!] \end{array}$$

7.22 *Completad la sentencia anterior.*

7.4. El conjunto de Dijkstra

Enunciado del problema Sea $\mathcal{D}$ el menor subconjunto de $\mathbb{N}$ verificando los axiomas:

$$\begin{array}{ll} Ax1: & 1 \in \mathcal{D} \\ Ax2: & h(\mathcal{D},\mathcal{D}) \subseteq \mathcal{D}, \textit{ es decir}, \\ & \forall x,y : x,y \in \mathcal{D} : h(x,y) \in \mathcal{D} \end{array}$$

donde la función $h : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ verifica las propiedades

$$\begin{array}{ll} p1: & \forall x,y : x,y \in \mathbb{N} : h(x,y) > x \\ p2: & h \textit{ es monótona creciente en el segundo argumento; es decir,} \\ & \forall x,y,y' : x,y,y' \in \mathbb{N} : y > y' \ \Rightarrow \ h(x,y) > h(x,y') \end{array}$$

Se trata de escribir un programa para calcular $a[0..N-1]$: los N primeros elementos de la sucesión creciente de elementos de $\mathcal{D}$.

El problema anterior aparece propuesto en [Dijkstra, 1976]:134 y también en [Dijkstra y Feijen, 1988]:170; en el primero se formula como una curiosa modificación de un problema atribuido a R.W. Hamming (véase Ejercicio 7.35). Dijkstra dice, pero no justifica, que el problema puede no tener solución si no se imponen ciertas condiciones a las funciones que generan el conjunto $\mathcal{D}$. Veremos que las propiedades $p1$ y $p2$ son suficientes para resolverlo.

Si se considera $(\mathbb{P}(\mathbb{N}), \leq)$ el dominio de las partes de $\mathbb{N}$ con la relación parcial $A \leq B$ sii $A \subseteq B$, entonces, para cada función $h : \mathbb{N}^2 \to \mathbb{N}$, la función inducida $g : \mathbb{P}(\mathbb{N}) \to \mathbb{P}(\mathbb{N})$ dada por

$$g(X) \doteq \{h(x, x') \,|\, x, x' \in X\} \doteq h(X, X)$$

es monótona y, según el Teorema de Kleene admite un mínimo punto fijo; es más, éste es el conjunto vacío. Es decir, el menor subconjunto de $\mathbb{N}$ verificando el axioma $Ax2$ es el conjunto vacío. Por otro lado, dado un natural a_0, la función inducida

$$g(X) = h(X, X) \cup \{a_0\}$$

también es monótona y el mínimo punto fijo es precisamente el conjunto de Dijkstra $\mathcal{D}$. En consecuencia $\mathcal{D}$ es la menor solución de la ecuación

$$\mathcal{D} = h(\mathcal{D}, \mathcal{D}) \cup \{a_0\}$$

Una propiedad esencial del conjunto de Dijkstra

LEMA 7.23 *Sea h una función verificando $(p1)+(p2)$ y sea $\mathcal{D} = \{a_n\}_{n\geq 0}$ la sucesión que describe el conjunto de Dijkstra correspondiente a h; entonces:*

(i) $\forall p, q : p \geq 0, q \geq 1 : a_{p+q} \leq h(a_p, a_{q-1})$.

(ii) *Si consideremos la sucesión creciente de conjuntos $\mathcal{D}_n$,*
$$\mathcal{D}_n \doteq \{h(a_i, a_j) \,|\, i + j \leq n\} \cup \{a_0\}$$
entonces, $\forall n : n > 0 : a_n \in \mathcal{D}_{n-1}$.

(iii) $\forall n : n > 0 : n + 2 \leq \operatorname{card} \mathcal{D}_n \leq 1 + (n+1)(n+2)/2$.

(iv) $(\lim \mathcal{D}_n \equiv) \cup_{n\geq 0} \mathcal{D}_n = \mathcal{D}$.

Demostración.– Partiendo de las propiedades $(p1) + (p2)$ obtenemos:

$$a_p < h(a_p, a_0) < h(a_p, a_1) < \ldots < h(a_p, a_{q-1}) < h(a_p, a_q)$$

luego $h(a_p, a_{q-1})$ tiene delante como mínimo $p+q$ elementos de la sucesión, de donde (i). Ya que $\mathcal{D} = h(\mathcal{D}, \mathcal{D}) \cup \{a_0\}$, entonces, para $n > 0$, existen términos a_p y a_q tales que $a_n = h(a_p, a_q)$, y por $(i) + (p2)$:

$$\begin{array}{ll} & a_{p+q} \leq h(a_p, a_{q-1}) < h(a_p, a_q) = a_n \\ \Rightarrow & \quad \because \text{transitividad de } \leq \\ & a_{p+q} < a_n \\ \Rightarrow & \quad \because a_n \text{ es creciente} \\ & p + q < n \\ \Rightarrow & \quad \because p, q, n \in \mathbb{N} \\ & p + q \leq n - 1 \\ \Rightarrow & \quad \because \text{definición de } \mathcal{D}_{n-1} \end{array}$$

$a_n = h(a_p, a_q) \in \mathcal{D}_{n-1}$

La segunda desigualdad de (iii) es trivial puesto que

$$(Ni, j : 0 \leq i, j \leq n : i + j \leq n) = (n+1)(n+2)/2$$

La primera desigualdad sigue por tener la fila $\{h(a_0, a_i) : 0 \leq i \leq n\}(\subseteq \mathcal{D}_n)$ $n+1$ elementos distintos (por ser h creciente en el segundo argumento).
(iv) se demuestra por doble inclusión:
(1) $\cup_{n\geq 0}\mathcal{D}_n \subseteq \mathcal{D}$, ya que $(\forall n : n \geq 0 : \mathcal{D}_n \subseteq \mathcal{D})$ ($\mathcal{D}$ verifica $Ax1 + Ax2$)
(2) $\cup_{n\geq 0}\mathcal{D}_n$ verifica los axiomas $Ax1 + Ax2$; y por ser $\mathcal{D}$ el menor conjunto verificando tales axiomas se tiene $\mathcal{D} \subseteq \cup_{n\geq 0}\mathcal{D}_n$. LEMA

NOTA 7.24 $\operatorname{card}\mathcal{D}_n$ *puede tomar el valor mínimo* $n+2$*; por ejemplo para la función* $h(x, y) = x + y$*, ya que en ese caso* $h(a_i, a_j) = i + j + 2$*. Es más, para tal función el conjunto de Dijkstra* $\mathcal{D}$ *es* $\mathbb{N}^*$*; en efecto, se tiene trivialmente* $\mathcal{D} \subseteq \mathbb{N}^*$*; por otro lado,*

$a \in \mathcal{D}$
$\Rightarrow$ $\quad \because 1 \in \mathcal{D}$
$a, 1 \in \mathcal{D}$
$\Rightarrow$ $\quad \because Ax1$
$h(a, 1) \in \mathcal{D}$
$=$ $\quad \because$ *definición de* h
$a + 1 \in \mathcal{D}$

de donde $\mathbb{N}^* \subseteq \mathcal{D}$*, por definición de* $\mathbb{N}^*$*: el menor subconjunto verificando* $1 \in \mathbb{N}^* \wedge \forall n : n \in \mathbb{N}^* : n + 1 \in \mathbb{N}^*$.

Un algoritmo para el cómputo de $\mathcal{D}$ Las propiedades (ii)-(iv) del lema permiten dar un método para el cómputo de $\mathcal{D}$. En efecto; puesto que $a_0 \in \mathcal{D}_0$ y ya que $a_1 \in \mathcal{D}_0, a_1 \neq a_0$,

$$a_1 = \text{mín}(\mathcal{D}_0 - \{a_0\})$$

De hecho $\mathcal{D}_0 = \{a_0, h(a_0, a_0)\}$ y $h(a_0, a_0) > a_0$. De aquí se concluye que

$$a_2 = \text{mín}(\mathcal{D}_1 - \{a_0, a_1\})$$

En general, se prueba por inducción que

$$a_{n+1} = \text{mín}(\mathcal{D}_n - \{a_0, a_1, \ldots, a_n\}) \qquad (*)$$

EJEMPLO 7.25 Veamos con un ejemplo que la cardinalidad de los conjuntos $\mathcal{D}_n$ puede ser maximal, es decir:

$$\operatorname{card}\mathcal{D}_n = 1 + (n+1)(n+2)/2$$

Para ello basta encontrar una función h tal que $\mathcal{D}_n$ contenga todos los números naturales positivos hasta el $1 + (n+1)(n+2)/2$; así, si

$$\mathcal{D}_n = \{1, 2, \ldots, 1 + (n+1)(n+2)/2\}$$

de aquí se obtendría, por $(*)$, $a_n = n + 1$. Una posible función es la correspondiente a la tabla

	1	2	3	4	5	…
1	2	3	5	8	12	…
2	4	6	9	13		j-1
3	7	10	14	…	j	
4	11	15	…	j+1		
5	16		⋱			
⋮						

que viene dada por

$$h(n,k) = \frac{(n+k)(n+k-1)}{2} + 2 - k$$

De aquí es fácil probar

$$n \geq 1 \;\Rightarrow\; h(n,k) - h(n-1,k+1) = 1$$
$$h(1,n+1) - h(n,1) = 1$$

Probemos por inducción que, en efecto, se tiene

$$\mathcal{D}_n = \{1, 2, \ldots, 1 + (n+1)(n+2)/2\}$$

Para $n = 0$ es trivial; supongamos

$$\forall k : 0 \leq k \leq n-1 : \mathcal{D}_k = \{1, 2, \ldots, 1 + (k+1)(k+2)/2\}$$

De aquí se concluye, por $(*)$,

$$\forall k : 0 \leq k \leq n, a_k = k + 1$$

y en ese caso

$$\begin{array}{ll} & \mathcal{D}_n \\ = & \quad \because \text{definición} \\ & \mathcal{D}_{n-1} \cup \{h(a_i, a_j) \,|\, i + j = n, i, j \geq 0\} \\ = & \quad \because \text{HI} \\ & \{1, 2, \ldots, 1 + n(n+1)/2\} \cup \{h(i+1, j+1) \,|\, i + j = n, i, j \geq 0\} \\ = & \quad \because \text{definición de } h \\ & \{1, 2, \ldots, 1 + n(n+1)/2\} \cup \{2 - (j+1) + (n+2)(n+1)/2 \,|\, 0 \leq j \leq n\} \\ = & \\ & \{1, 2, \ldots, 1 + n(n+1)/2\} \cup \{2 + n(n+1)/2, \ldots, 1 + (n+1)(n+2)/2\} \end{array}$$ EJEMPLO

Por la ecuación $(*)$ podemos definir en forma recurrente la sucesión solución a través de una sucesión de conjuntos $\mathcal{U}_n$:

$$\begin{cases} a_1 &= h(a_0, a_0) \\ \mathcal{U}_1 &= \{a_1\} \end{cases}$$

$$\begin{cases} a_{n+1} &= \text{mín}\, \mathcal{U}_n \\ \mathcal{U}_{n+1} &= (\mathcal{U}_n - \{a_{n+1}\}) \cup \{h(a_i, a_j) \,|\, i + j = n + 1\} \end{cases}$$

Obsérvese que $\mathcal{U}_n = \mathcal{D}_n - \{a_0, a_1, \ldots, a_n\}$. Para generar a_{n+1} necesitamos conocer $\mathcal{U}_n$, y para ello debemos conocer $\mathcal{U}_{n-1}$ y los elementos $a_0, \ldots, a_n$ para con ellos formar la diagonal:

$$\{h(a_0, a_n), h(a_1, a_{n-1}), \ldots, h(a_n, a_0)\}$$

Ilustremos el método para la función:

$$h(x,y) = \begin{cases} 3x+2y, & \text{si} \quad x \neq 13 \vee y \neq 1 \\ 14, & \text{si} \quad x = 13 \wedge y = 1 \end{cases}$$

Ya que $\mathcal{U}_1 = \{h(a_0, a_0)\}$, en nuestro ejemplo la situación de partida es:

	1
1	5

y el mínimo será 5 (a_1), de forma que $\mathcal{U}_2$ son los elementos de la siguiente tabla:

	1	5
1	[5]	13
5	17	

Aparece en un cuadro el elemento eliminado. El siguiente mínimo es 13, que lo encuadramos y añadimos la diagonal:

	1	5	13
1	[5]	[13]	29
5	17	25	
13	14		

Obsérvese que en el siguiente paso se extrae el mínimo de la tercera fila:

	1	5	13	14
1	[5]	[13]	29	31
5	17	25	41	
13	[14]	49		
14	44			

OBSERVACIÓN.– [UNA CURIOSIDAD] En general $a_n \notin \mathcal{D}_{n-2}$ y es necesario utilizar todos los términos de $\mathcal{D}_{n-1}$ para computar a_n; es más, dado un N arbitrario, existe una función tal que el término a_{N+1} se calcula precisamente en la iteración N (es decir, no había sido calculado anteriormente). Por ejemplo, sea, para N natural > 1, la función

$$h(x,y) = \begin{cases} 2(x+y), & \text{si} \quad x \neq 6N-2 \vee y \neq 1 \\ 6N-1, & \text{si} \quad x = 6N-2 \wedge y = 1 \end{cases}$$

Si la función fuera $h(x,y) = 2(x+y)$, vamos a demostrar que

$$\forall n : 1 \leq n \leq N : a_n = 6n-2$$

por inducción. Para $n = 1$ es trivial; supongamos

$$\forall k : 1 \leq k \leq n < N : a_k = 6k-2$$

entonces estudiemos el mínimo del conjunto:

$$\mathcal{D}_n - \{a_0, \ldots, a_n\}$$

Si $i > 1, j \geq 1$ tenemos, por hipótesis de inducción que

$$h(a_i, a_j) = h(a_{i-1}, a_{j+1}) = 12(i+j) - 8$$

luego:

$$\begin{array}{ll} & \{h(a_i,a_j) \,|\, i+j \leq n\} \\ \subseteq & \{h(a_0,a_j) \,|\, 0 \leq j \leq n\} \cup \{h(a_1,a_j) : 0 \leq j \leq n-1\} \\ = & \{4\} \cup \{h(a_0,a_j \,|\, 1 \leq j \leq n\} \cup \{10\} \cup \{h(a_1,a_j) \,|\, 1 \leq j \leq n-1\} \\ = & \{12j-2 : 1 \leq j \leq n\} \cup \{12j+4 : 0 \leq j \leq n-1\} \end{array}$$

de donde:

$$\mathcal{D}_n = \{1\} \cup \{b_j \equiv 12j-2 : 1 \leq j \leq n\} \cup \{c_j \equiv 12j+4 : 0 \leq j \leq n-1\}$$

y teniendo en cuenta que

$$c_{j-1} < b_j < c_j < b_{j+1}$$

la sucesión ordenada de elementos de $\mathcal{D}_n$ es:

$$a_0, c_0, b_1, c_1, b_2, c_2, \ldots, b_{n-1}, c_{n-1}, b_n$$

y de aquí sigue fácilmente que,

$$\begin{array}{llllllll} \text{si } n \text{ par} > 1, & a_{n+1} & = & c_{n/2} & = & 12(n/2)+4 & = & 6(n+1)-2 \\ \text{si } n \text{ impar}, & a_{n+1} & = & b_{(n+1)/2} & = & 12(n+1)/2-2 & = & 6(n+1)-2 \end{array}$$

y hemos probado $a_n = 6n-2$ $(1 \leq n \leq N)$. Pero el valor $6N-2$ solamente se alcanza en el término a_n, de donde también es cierto lo anterior para la función modificada $h(6N-2, 1) = 6N-1$. Ahora bien,

$$\begin{array}{ll} & \mathcal{D}_n \\ = & \{h(a_i,a_j) \,|\, i+j \leq N\} \\ = & \{h(a_i,a_j) \,|\, i+j < N\} \cup \{h(a_i,a_j) \,|\, i+j = N\} \\ = & \{h(a_i,a_j) \,|\, i+j < N\} \cup \{12N-8, 12N-2, 6N-1\} \\ = & \{a_0, c_0, b_1, c_1, b_2, c_2, \ldots, b_{N-1}, c_{N-2}, b_{N-1}\} \cup \{6N-1\} \\ = & \{1, 4, \ldots, 6N-2, 6N+4, 6N+10, \ldots, 12N-14\} \cup \{6N-1\} \end{array}$$

de donde

$$\begin{array}{l} \mathcal{D}_n - \{a_0, \ldots, a_n\} = \{6N-1, 6N+4, 6N+10, \ldots, 12N-14\} \\ a_{N+1} = 6N-1 \end{array}$$

y también

$$\begin{array}{l} \mathcal{D}_{N+1} = \mathcal{D}_n \cup \{12N, 12N+4\} \\ \mathcal{D}_{N+2} = \mathcal{D}_n \cup \{12N, 12N+4, 12N+10\} \\ \vdots \\ a_{2N+2} = 12N \end{array}$$

OBS

Un programa para el cómputo de $\mathcal{D}$ Utilizaremos tres tablas

$$a[0..n] \qquad f[0..n, 0..n] \qquad m[0..n]$$

La primera memoriza los elementos de la sucesión: $a[k] = a_k$. La segunda memoriza las imágenes de h de los ya calculados: $f[k, l] = h(a_k, a_l)$. La tercera memoriza qué elementos de la tabla ya aparecieron en la sucesión, de forma que cada $m[k]$ indica el índice del primer elemento de la fila k–ésima

$$f[k, 0], f[k, 1], \ldots, f[k, i - k]$$

que no ha sido incluido todavía en a. Con tales tablas podemos formar el predicado *invariante*:

$$I.i \equiv A.i \wedge F.i \wedge M.i$$

donde

$$\begin{array}{lcl} A.i & = & i \leq n \wedge (\forall k : 0 \leq k \leq i : a[k] = a_k) \\ F.i & = & i \leq n \wedge (\forall k, l \geq 0 : 0 \leq k + l \leq i : f[k, l] = h(a_k, a_l)) \\ M.i & = & i \leq n \wedge (\forall k : 0 \leq k \leq i : m[k] = \text{mín}\{j | j \geq 0 \wedge a_i < f[k, j]\}) \end{array}$$

y el siguiente esquema resolverá el problema

$$\begin{array}{l} a[0] := a0; \{A\} \\ f[0, 0] := h(a0, a0); \{F\} \\ m[0] := 0; \{M\} \\ i := 0; \{I\} \\ *[\![\, i < n \rightarrow \textit{incrementar } i \textit{ con invariabilidad de } I \,]\!] \end{array}$$

Al estudiar la sentencia $i := i + 1$, obtenemos

$$\begin{array}{lcl} i := i + 1.A & = & A \wedge i + 1 \leq n \wedge a[i + 1] = a_i \\ i := i + 1.F & \Rightarrow & F \wedge i < n \wedge (\forall k, l \geq 0 : k + l = i + 1 : f[k, l] = h(a_k, a_l)) \end{array}$$

pero:

$$I \wedge i < n \;\Rightarrow\; a_{i+1} = \text{ mínimo } \{f[0, m[0]], \ldots, f[i, m[i]]\}$$

de donde:

$$\begin{array}{l} \{I \wedge i < n\} \\ \text{mín} := f[0, m[0]]; \\ k := 1; *[\![\, k \leq i \rightarrow \text{mín} := \textit{mínimo}(f[k, m[k]], \text{mín}) \,]\!]; \\ \{\text{mín} = \text{ mínimo } \{f[0, m[0]], \ldots, f[i, m[i]]\} \\ a[i + 1] := \text{mín}; \\ \{\textit{AÑADIENDO LA NUEVA DIAGONAL}\} \\ k := 0; *[\![\, k \leq i + 1 \rightarrow f[k, i + 1 - k] := h(a[k], a[i + 1 - k]) \,]\!]; \\ \{A.i + 1 \wedge F.i + 1\} \\ i := i + 1; \\ \{A.i \wedge F.i\} \end{array}$$

Finalmente queda restablecer el predicado M; para ello, naturalmente, hay que reajustar la tabla de mínimos $m[k]$; comenzamos viendo $i := i + 1.M$, que se descompone en tres predicados

$$M.(i + 1)(\equiv i := i + 1.M)$$

$$
\begin{array}{ll}
= & \\
 & i+1 \leq n \wedge (\forall k : 0 \leq k \leq i+1 : m[k] = \text{mín}\{j | j \geq 0 \wedge a_{i+1} < f[k,j] \\
= & \\
 & i+1 \leq n \wedge & (1) \\
 & (\forall k : 0 \leq k \leq i : m[k] = \text{mín}\{j | j \geq 0 \wedge a_{i+1} < f[k,j]\} \wedge & (2) \\
 & m[i+1] = \text{mín}\{j | j \geq 0 \wedge a_{i+1} < f[i+1,j]\} & (3)
\end{array}
$$

Para restablecer (1) basta con $i \leq n \wedge i < n$; por las propiedades de la función h,

$$a_{i+1} < h(a_{i+1}, a_0)$$

pero

$$F.(i+1) \;\Rightarrow\; h(a_{i+1}, a_0) = f[i+1, 0]$$

de donde tendremos

$$F.(i+1) \;\Rightarrow\; \text{mín}\{j | j \geq 0 \wedge a_{i+1} < f[i+1,j]\} = 0$$

y es necesario que $m[i+1] = 0$ para restablecer (3); es decir

$$\{F.(i+1)\}\; m[i+1] := 0; \{(3)\}$$

El predicado (2) es más complicado; sea k fijo $(\leq i)$, y

$$J_{i+1} = \{j | j \geq 0 \wedge a_{i+1} < f[k,j]\}$$

y comparemos con el conjunto que aparece en el predicado $M.i$

$$J_i = \{j | j \geq 0 \wedge a_i < f[k,j]\}$$

Basta con que los dos conjuntos sean los mismos para no tener que alterar el valor $m[k]$; por ser la sucesión a_i creciente tenemos $J_{i+1} \subseteq J_i$, de donde $\text{mín}\, J_{i+1} \geq \text{mín}\, J_i$. Además

$$
\begin{array}{ll}
 & f[k, m[k]] \neq a_{i+1} \\
\Rightarrow & \quad \because a_i \text{ creciente} \\
 & m[k] \in J_{i+1} \\
\Rightarrow & \quad \because \text{propiedades del mínimo} \\
 & m[k] \geq J_{i+1}
\end{array}
$$

y de aquí

$$m[k] = \text{mín}\, J_i \wedge f[k, m[k]] \neq a_{i+1} \;\Rightarrow\; m[k] = \text{mín}\, J_{i+1}$$

y en definitiva tenemos el siguiente cuerpo para el bucle (casi completo)

$$
\begin{array}{l}
\{A.(i+1) \wedge F.(i+1) \wedge M.i \wedge i < n\} \\
m[i+1] := 0; \\
\{(3) \wedge (1)\} \\
\{i < n \wedge M.i \wedge a[i+1] = a_{i+1}\} \\
k := 0; \\
\quad *[\![\, k \leq i \rightarrow \\
\qquad\qquad [\!] \quad f[k, m[k]] = a[i+1] \;\rightarrow\; \boxed{?} \\
\qquad\qquad \square \quad f[k, m[k]] \neq a[i+1] \;\rightarrow\; nada \;]\!]; \\
\qquad\qquad \{m[k] = \text{mín}\, J_{i+1}\} \\
\qquad\qquad \{(1) \wedge (2) \wedge (3) \equiv M.i+1\} \\
\qquad\qquad i := i+1 \{M.i\} \;]\!]
\end{array}
$$

y solo queda encontrar la sentencia $\boxed{?}$; tenemos:

$$\begin{array}{l} A.(i+1) \wedge a[i+1] = f[k, m[k]] \\ \Rightarrow \\ a[i] < a[i+1] = f[k, m[k]] < f[k, m[k]+1] \\ \Rightarrow \\ m[k] + 1 = J_{i+1} \end{array}$$

de donde

$$\begin{array}{l} \{A.(i+1) \wedge M.i \wedge a[i+1] = f[k, m[k]]\} \\ \{m[k] + 1 = J_{i+1}\} \\ m[k] := m[k] + 1; \\ \{m[k] = J_{i+1}\} \end{array}$$

y finalmente nuestro programa será:

$$\begin{array}{l} a[0] := a0; m[0] := 0; f[0,0] := h(a0, a0); \\ i := 0; \\ *[\![\, i < n \rightarrow \quad \text{mín} := f[0, m[0]]; \\ \qquad\qquad k := 1; *[\![\, k \leq i \rightarrow \text{mín} := \textit{mínimo}(f[k, m[k]], \text{mín}) \,]\!]; \\ \qquad\qquad a[i+1] := \text{mín}; m[i+1] := 0; \\ \qquad\qquad k := 0; *[\![\, k \leq i+1 \rightarrow f[k, i+1-k] := h(a[k], a[i+1-k]) \,]\!]; \\ \qquad\qquad k := 0; \\ \qquad\qquad *[\![\, k \leq i \rightarrow \quad [\![\quad f[k, m[k]] = a[i+1] \rightarrow m[k] := m[k] + 1 \\ \qquad\qquad\qquad\qquad \square \quad f[k, m[k]] \neq a[i+1] \rightarrow nada \,]\!] \,]\!]; \\ \qquad\qquad i := i + 1 \qquad]\!] \end{array}$$

7.5. La criba de Eratóstenes

Se trata de obtener el conjunto de números primos no superiores a N:

$$primos.N \doteq \{k \mid k \text{ es primo} \leq N\}$$

a través del método más antiguo que se conoce y descubierto por Eratóstenes. Utilizaremos una variable (de tipo conjunto) $criba$ que verifique el siguiente predicado P invariante:

$$P \doteq primos.N \subseteq criba \subseteq [2..N]$$

Una primera acción a definir es: cribar tal conjunto quitando en cada paso los múltiplos de cierto elemento. Nos interesarán los siguientes predicados:

$cri\_con.q \doteq \forall n : n \in criba \wedge q \mid n : n = q$
— el conjunto $criba$ ha sido *cribado* con el número q

$cri\_d\_de.p \doteq \forall q : q \in primos.(p-1) : cri\_con.q$
— el conjunto $criba$ ha sido *cribado*
con todos los primos menores que p

LEMA 7.26 $P \wedge p^2 > N \wedge cri\_d\_de.p \Rightarrow criba = primos.N$

Demostración.– Si $s \in criba \backslash primos.N$ y q es el menor factor primo de s,

$$\begin{array}{l} q^2 \leq s \leq N < p^2 \\ \Rightarrow \\ q \leq p - 1 \end{array}$$

$\Rightarrow$
$q \in primos.(p-1)$
$\Rightarrow$
$cri\_con.q$

y por tanto $s = q$, que es imposible; luego $criba = primos.N$. LEMA

El lema anterior *conjetura* la existencia de un bucle cuyo invariante se obtiene por debilitación de la poscondición, de forma que tengamos:

$$\{I\}(\ \doteq\ I \wedge cri\_d\_de.p)$$
$$*[\![\, p^2 \leq N \rightarrow \ldots]\!]\, \{I \wedge p^2 > N\}\ \{\Rightarrow\}\ \{criba = primos.N\}$$

Para que el predicado $P \wedge cri\_d\_de,2$ sea cierto es suficiente que el conjunto inicial $criba$ contenga únicamente números impares junto con el 2; es decir, podemos escribir:

$$criba, k := \{2\}, p;$$
$$*[\![\, p \leq N \rightarrow criba := criba \cup \{p\}; p := p + 2]\!]$$
$$\{criba = \{2\} \cup \{k \mid 2 < k \leq N, k\ impar\}\}$$

Un invariante para el bucle anterior puede ser:

$$A\ \doteq\ criba = \{2\} \cup \{k \mid 2 < k < p, k\ impar\} \wedge p \leq N + 2 \wedge p\ impar$$

Partimos entonces del invariante $I\ \doteq\ P \wedge cri\_d\_de.p$ y consideremos una operación: $cribar.p$ que verifique la poscondición:

$$\{I\}\ cribar.p\ \{cri\_con.p\}$$

LEMA 7.27 $p^2 \leq N \wedge cri\_d\_de.p \wedge cri\_con.p\ \Rightarrow\ cri\_de\_d.(spa.p)$, *donde* $spa.p$ *es el siguiente primo a* p.

Demostración.– Sea $q \in primos.(spa.p - 1)$. Entonces puede ocurrir:

— $q \leq p - 1$; luego por ser cierto $cri\_d\_de.p.\ \Rightarrow\ cri\_con.q$.

— $q = p$; $cri\_con.q \equiv cri\_con.p$.

— $q > p$ es imposible, ya que se tendría $p < q \leq spa.p - 1 < spa.p$, que es absurdo por ser q es primo. LEMA

Por otro lado tenemos el siguiente lema:

LEMA 7.28 $I \wedge p^2 \leq N\ \Rightarrow\ spa.p = s(p, criba)$, *donde* $s(p, criba)$ *es el menor elemento del conjunto* $criba$ *mayor que* p.

Demostración.– Sea $p' = s(p, criba)$; es suficiente probar que p' es primo, ya que

$$p < p' \leq spa.p \qquad\qquad primos.N\ \subseteq\ criba$$

Sea q el menor factor primo de p. Si $q < p' \leq spa.p$, entonces se da $cri\_con.q$, de donde debe tenerse $q = p'$ y p' es primo. LEMA

Por consiguiente tenemos el siguiente esquema *correcto*

$$
\begin{array}{l}
\{criba = \{2\} \cup \{k \mid 2 < k \leq N, k\ impar\}\ \} \\
p := 3; \{I\} \\
\quad *[\![\, p^2 \leq N \ \rightarrow \quad \{I \wedge p^2 \leq N\} \\
\qquad\qquad\qquad\ \ cribar.p \\
\qquad\qquad\qquad\ \ \{cri\_d\_de.p. \wedge cri\_con.p\} \\
\qquad\qquad\qquad\ \ \{cri\_d\_de.(spa.p)\} \\
\qquad\qquad\qquad\ \ p := spa.p \\
\qquad\qquad\qquad\ \ \{I \wedge cri\_d\_de.p\}(\ \Rightarrow\ I)\ ]\!] \\
\{criba = primos.N\}
\end{array}
$$

Falta especificar dos operaciones: $cribar$ y spa; la operación $p := spa.p$ se obtiene calculando el primer elemento impar de $criba$ mayor que p:

$$
\begin{array}{l}
p := p + 2; \\
*[\![\, p \notin criba \rightarrow p := p + 2\,]\!]
\end{array}
$$

¿Existe $s(p, criba)$? Ello se deduce de ser $p^2 \leq N$ y de que entre un primo p y su cuadrado p^2 existe siempre un número primo.

Queda por especificar la operación $cribar.p$. Si se verifica el invariante I, se tiene $cri\_d\_de.p$, y por tanto, de entre los múltiplos de p:

$$p, 2p, 3p, 4p, \ldots, (p-1)p, pp, (p+1)p, (p+2)p, \ldots, (p+\alpha)p, \ldots$$

no es necesario suprimir los múltiplos kp, con $k < p$; por otro lado, al ser p impar, $p + \alpha$ es par cuando α sea impar, por lo que solamente tenemos que suprimir los múltiplos de la forma $(p + \alpha)p$, con α par, y hemos demostrado el siguiente lema:

LEMA 7.29 *Si $q \mid p$ con $q > p > 2$, y además se cumple $cri\_con.p$, entonces las dos condiciones siguientes son equivalentes:*

$$
\begin{array}{ll}
(a) & q \bmod 2p = p \\
(b) & q = (p + \alpha)p,\ \textit{con } \alpha \textit{ impar } \geq 0.
\end{array}
$$

Por consiguiente tenemos el siguiente esquema para la operación $cribar.p$

$$
\begin{array}{l}
\{I \equiv cri\_d\_de.p \wedge I \wedge p > 2\} \\
dosp, q := p + p, p * p; \\
\{A\}(\ \doteq\ q \bmod 2p = p \wedge I \wedge p \mid k > p \wedge (k \in criba\ \Rightarrow\ k \geq q)\ ) \\
\quad *[\![\, q \leq N \rightarrow \quad criba := criba \backslash \{q\}; \\
\qquad\qquad\qquad\ \ q := q + dosp \qquad\ ]\!] \\
\{cri\_con.p\}
\end{array}
$$

para lo cual es suficiente probar que A es un invariante (hágase como ejercicio). En definitiva el programa completo es:

$$
\begin{array}{l}
criba, k := \{2\}, p; \\
*[\![\, p \leq N \rightarrow criba := criba \cup \{p\}; p := p + 2 \,]\!] \\
\{criba = \{2\} \cup \{k \mid 2 < k \leq N, k\ impar\}\ \} p := 3; \\
\{I\}(\ \doteq\ cri\_d\_de.p \wedge p > 2 \wedge\ ) \\
\quad *[\![\, p^2 \leq N \rightarrow \quad dosp, q := p + p, p * p; \\
\qquad\qquad\qquad\quad *[\![\, q \leq N \rightarrow criba := criba \backslash \{q\}; q := q + dosp \,]\!] \\
\qquad\qquad\qquad\quad p := p + 2; \\
\qquad\qquad\qquad\quad *[\![\, p \notin criba \rightarrow p := p + 2 \,]\!] \qquad\quad]\!] \\
\{criba = primos.N\}
\end{array}
$$

Ejercicios

7.30 *[288] Escribid un programa para invertir una tabla $a[1..n]$; es decir, para la precondición y poscondición:*

$$
\begin{array}{l}
P\ \doteq\ \forall i : 1 \leq i \leq n : a[i] = A_i \\
Q\ \doteq\ \forall i : 1 \leq i \leq n : a[i] = A_{n-i+1}
\end{array}
$$

7.31 **(El problema de la Partición)** *Sea una tabla $b[m..n-1]$ con valores iniciales $B[m..n-1]$; escribid un programa para la poscondición:*

$$
\begin{array}{rl}
R\ \doteq & m \leq p < n \wedge b \text{ contiene una permutación de los valores } B \wedge \\
& b[p] = B[m] \wedge \\
& (m \leq i \leq p\ \Rightarrow\ b[i] \leq B[m]) \wedge (p < i \leq n-1\ \Rightarrow\ b[i] > B[m])
\end{array}
$$

7.32 **(La regla de Horner)** *Demostrad la corrección del siguiente programa que calcula el valor de un polinomio $\sum_{0 \leq i \leq n-1} a_i z^i$ en el punto x por la regla de Horner:*

$$
\begin{array}{l}
i, y, z := 1, a_0, x; \\
*[\![\, i \neq n \rightarrow i, y, z := i + 1, y + a_i z, zx \,]\!]
\end{array}
$$

7.33 *Escribid un programa para encontrar las formas posibles en que puede descomponerse un entero $r \geq 0$ como suma de dos cuadrados: $x^2 + y^2 = r$ (que genere las soluciones enteras de $x^2 + y^2 = r \wedge 0 \leq y \leq x$).*

7.34 *[288]* **(El problema de la secuencia ascendente más larga)** [Dijkstra, 1976] *Dada una tabla $a[0..n-1]$, encontrad la longitud de la secuencia $a[i..j-1]$ ascendente más larga. Así, si definimos el predicado:*

$$
as(i, j)\ \doteq\ a[i..j-1] \text{ es ascendente } (\equiv \forall r : i < r < j : a[r-1] \leq a[r])
$$

la poscondición vendrá dada por $m = Msa(n)$ donde

$$
Msa(n)\ \doteq\ \text{máx}\{j - i \mid 0 \leq i < j \leq n \wedge as(i, j)\}
$$

o lo que es igual

$$
m = \text{máx}\{M_j \mid 0 < j \leq n\}, \qquad \text{donde } M_j\ \doteq\ \text{máx}\{j - i \mid i < j, as(i, j)\}.
$$

7.35 *[289]* **(El problema de Hamming)** *Escribid un programa para generar una tabla ordenada $a[0..n-1]$ que contenga el 1 y los primeros números divisibles solamente por los primos $2, 3$ y 5. Es decir, los n primeros términos de la secuencia $\mathcal{H}$ que verifica los únicos axiomas:*

(a) $1 \in \mathcal{H}$,

(b) $x \in \mathcal{H} \;\Rightarrow\; 2x, 3x, 5x \in \mathcal{H}$.

7.36 *[290]* **(El problema de McCarthy)** *Demostrad que el programa*

$$
\begin{array}{l}
u, v := x, 1; \\
\quad *[\![\, u \leq 100 \vee v \neq 1 \rightarrow \quad [\![\quad u > 100 \quad \rightarrow u, v := u - 10, v - 1 \\
\qquad\qquad\qquad\qquad\quad \square \quad u \leq 100 \quad \rightarrow u, v := u + 11, v + 1 \,]\!] \qquad]\!]\,; \\
z := u - 10
\end{array}
$$

calcula la función 91 de McCarthy: $z = \; if\; x > 100\; then\; x - 10\; else\; 91$.
AYUDA.- si consideramos el caso $x \leq 100$ *(si* $x > 100$ *es trivial) se observa que el número de pasos es* $2(101 - x)$*. Podemos pues considerar tres variables auxiliares*

$$
\begin{array}{lcl}
np & \equiv & \textit{número de pasos que realiza el bucle} \\
k & \equiv & \textit{número de veces que se ejecuta } u, v := u + 11, v + 1 \\
q & \equiv & \textit{número de veces que se ejecuta } u, v := u - 10, v - 1
\end{array}
$$

e introducir así mismo sentencias apropiadas para mantener tales variables. Demostrad que el siguiente predicado es un invariante del bucle:

$$
\begin{array}{rl}
I \;\doteq & x \leq 100 \wedge np = 2(101 - x) - k - q \geq 0 \wedge \\
& u = x + 11k - 10q \wedge v = 1 + k - q
\end{array}
$$

y que además se verifica $I \wedge u > 100 \wedge v = 1 \;\Rightarrow\; u = 101$.

7.37 *[291] Sea* $\{f(k) : 0 \leq k < n\}$ *una tabla de enteros positivos, es decir, verificando la precondición* $P \doteq \forall k : 0 \leq k < n : 0 \leq f(k) \in Z$*. Escribid un programa* $\mathcal{S}$ *con complejidad mínima verificando el triplete* $\{I\}\mathcal{S}\{t = (\sum_{0 \leq i < n} f(i) < 1000)\}$.

7.38 *Sean* $a[0..m-1]$ *y* $b[0..n-1]$ *dos tablas de números enteros estrictamente crecientes con algún elemento común; es decir, verificando*

$$P \;\doteq\; \exists p, q : 0 \leq p < m, 0 \leq q < n : a[p] = b[q]$$

Encontrad un programa para el triplete

$$\{P\}\mathcal{S}\{x \textit{ es el mayor entero común a las tablas } a \textit{ y } b\}$$

7.39 *Sean* $a[0..m-1]$ *y* $b[0..n-1]$ *dos tablas de números enteros estrictamente crecientes siendo* $a[0] = b[0] = 0$*; escribid un programa para encontrar el menor natural que no está en ninguna de las tablas.*

7.40 *[292] Se define la sucesión de Simon en la forma*

$$a_0 = a_1 = a_2 = 1, \qquad \forall k : k \geq 0 : a_{k+3} = a_{k+1}.a_{k+2} + a_k.$$

Escribid un programa para calcular el término a_{1000}.

7.41 *Un número natural es perfecto si es la suma de sus divisores propios. Por ejemplo,* 6 *y* 28 *son perfectos. Por consiguiente los números perfectos verifican el predicado*

$$n = \sum_{1 \leq x < n, x | n} x$$

(A) Escribid un programa para la poscondición $R \;\doteq\; p \equiv n$ *es un número perfecto.*

(B) Escribid un programa parcialmente correcto para encontrar el menor entero perfecto de tres cifras.

7.42 *[293] Sea $a[0..n-1]$ una tabla creciente de enteros verificando*

$$P \doteq n > 1 \wedge a[0..n-1]\uparrow \wedge a(n-1) = 300.$$

Escribid un programa para localizar el menor múltiplo de tres.

7.43 *[294] Sean $a[0..m-1]$, $b[0..n-1]$ y $c[0..s-1]$ tres tablas ordenadas en sentido estrictamente ascendente; escribid un programa que cuente los elementos comunes.*

7.44 *[294]* **(Febrero, 96)** *Probad la corrección del programa*

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

7.45 *[294]* **(Setiembre, 95)** *Sea $a[0..n-1]$ una tabla creciente de enteros verificando la precondición:*

$$P \doteq n > 1 \wedge a[0..n-1]\uparrow \wedge a(n-1) = 200$$

Escribid un programa para localizar el menor número par de la tabla.
AYUDA.- Sea $\mathcal{P}$ el conjunto de enteros pares. Escribamos la precondición en la forma

$$P \equiv a[0..n-1] \subseteq \mathbb{Z} \wedge a\uparrow \wedge a(n-1) \in \mathcal{P} \wedge n > 0.$$

y la poscondición en la forma $R \doteq a(x) \in \mathcal{P} \wedge (a[0..x-1] \cap \mathcal{P}) = \emptyset \wedge 0 \leq x < n \wedge P$, de donde $R \Rightarrow a(x)$ es el menor par de $a[0..n-1]$.

7.46 *[295]* **(Diciembre, 94)** *Dada una tabla de enteros $a[0..n-1]$, con al menos dos elementos distintos, escribid un programa para encontrad un elemento distinto de su mínimo.*

7.47 *[296]* **(Diciembre, 92)** *Sea $a[0..n-1]$ una tabla de n números enteros no necesariamente ordenada; escribid un programa para encontrar el segundo menor elemento.*
AYUDA.- Considere la poscondición

$$Q \doteq n \geq 2 \wedge x = \min a[0..n-1] \wedge y = \min(a[0..n-1] - \{x\})$$

de la cual se puede obtener un invariante introduciendo una variable en lugar de n.

7.48 *[297]* **(Enero, 96)** *Sea $a[0..n-1]$ una tabla de números enteros. Escribid un programa EFICIENTE para la poscondición $R \doteq q =$ algún elemento de la tabla es 6.*

7.49 *[297]* **(Diciembre, 00)** *Sea $n > 0$, $A[0..n-1]$ una tabla de números enteros no decreciente, $B(p,q) \doteq \{i \mid p \leq i < q, A[i] = 6\}$, y supongamos el predicado $R \doteq k = Card\, B(0,n)$. Deducid un programa correcto para la poscondición R. AYUDA.- Se trata de un problema de conteo, por lo que podemos utilizar la técnica añadir a la variable que cuenta parte del problema vista en la Sección 7.3.*

7.50 *[298]* **(Febrero, 92)** *Sean $a[0..m-1]$ y $b[0..n-1]$ dos tablas de números enteros, siendo la tabla a estrictamente decreciente y la tabla b estrictamente creciente; sea el predicado*

$$P \doteq \exists p, q : 0 \leq p < m, 0 \leq q < n : a[p] = b[q]$$

Escribid un programa $\mathcal{S}$ tal que $\{P\}\mathcal{S}\{x$ es el mayor entero común a las tablas a y $b\}$.

7.51 *[298]* **(Diciembre, 96)** *Sea $a[0..n-1]$ una tabla de valores sobre un conjunto donde hay definida una igualdad; escribid un programa para comprobar si existen objetos diferentes en la tabla; por ejemplo, el programa devolverá en una variable q el test: existen dos elementos de la tabla que son distintos.*

7.52 *[299]* **(Julio, 94)** *Sea $a[0..n-1]$ una tabla creciente de enteros verificando la precondición: $P \doteq n > 1 \wedge a[0..n-1]\uparrow \wedge a(0) = 2$. Escribid un programa para la poscondición $R \doteq i$ es el mayor número par de la tabla.*

Capítulo 8

Continuidad, Puntos Fijos y Semántica de Bucles

8.0. La propiedad de continuidad

En un retículo completo D (Definición 2.0, página 27), una función $f : D \to D$ se dice continua si lo es por cadenas; es decir, si para toda cadena $x_n \uparrow$ (sucesión no decreciente) se tiene: $f(\sup\{x_n\}) = \sup\{f(x_n)\}$ (véase Teorema 2.11, página 28). Consideraremos el retículo completo $\mathcal{P}$ de los predicados sobre un espacio de estados, con la relación de orden $P \leq Q \doteq [P \Rightarrow Q]$. En ese caso, una sucesión $\{C_k\}$ es una cadena si

$$\forall i : i \geq 0 : [C_i \Rightarrow C_{i+1}] \qquad (C_n \uparrow)$$

Entonces, la continuidad y la $\vee$–continuidad coinciden ya que el supremo de una colección de predicados $\{C_k\}$ es la disyunción $(\exists k : k \geq 0 : C_k)$.

DEFINICIÓN 8.0 *Un transformador de predicados S se dice continuo si para toda cadena $C_n \uparrow$ se verifica*

$$[S.(\exists k : k \geq 0 : C_k) \equiv \exists k : k \geq 0 : S.C_k] \qquad (cont)$$

En este capítulo veremos que la continuidad de los transformadores de las sentencias de un lenguaje tiene numerosas aplicaciones teóricas y prácticas.

TEOREMA 8.1 *El lenguaje de Dijkstra $\mathcal{D}$ es continuo.*

Demostración.– Probaremos la propiedad $(cont)$ de la Definición 8.0 por inducción estructural sobre la sentencia S. Si recordamos la Sección 4.3 (página 67) y el Teorema 1.22 (página 22), sabemos que para demostrar que $\mathcal{D}$ es continuo basta demostrar que lo son las sentencias básicas y que los constructores del lenguaje conservan la continuidad. Probaremos en primer lugar $(cont)$ para las sentencias básicas $(nada, aborta, x := E)$; tenemos, *ptle*

$$x := E.(\exists i : i \geq 0 : P_i) \overset{\text{definición sustitución}}{\equiv} \exists i : i \geq 0 : x := E.P_i$$

$$\begin{array}{ll} & nada.(\exists i : i \geq 0 : P_i) \\ = & \quad \because \text{definición de } nada \\ & \exists i : i \geq 0 : P_i \\ = & \quad \because \text{definición de } nada \\ & \exists i : i \geq 0 : nada.P_i \end{array} \qquad \begin{array}{ll} & aborta.(\exists i : i \geq 0 : P_i) \\ = & \quad \because \text{definición de } aborta \\ & F \\ = & \quad \because \text{CP} \\ & \exists i : i \geq 0 : F \\ = & \quad \because \text{definición de } aborta \\ & \exists i : i \geq 0 : aborta.P_i \end{array}$$

— COMPOSICIÓN.- Sean ahora A y B continuos, B monótono, y $P_n \uparrow$; veamos que la composición conserva la continuidad, viendo *ptle*

$$
\begin{array}{ll}
 & A;B.(\exists i : i \geq 0 : P_i) \\
= & \quad \because \text{definición composición} \\
 & A.B.(\exists i : i \geq 0 : P_i) \\
= & \quad \because P_i \uparrow, B \text{ es continuo} \\
 & A.(\exists i : i \geq 0 : B.P_i) \\
= & \quad \because A \text{ continuo, } B \text{ monótono, y por ello } B.P_i \uparrow \\
 & \exists i : i \geq 0 : A.B.P_i \\
= & \quad \because \text{definición composición} \\
 & \exists i : i \geq 0 : A;B.P_i
\end{array}
$$

— SELECTIVA.- Sea $SI == [\!] \ \Box \, j : 1 \leq j \leq n : b_j \rightarrow S_j \ [\!]$ donde cada S_j es sana y continua. Razonamos en la forma siguiente. Si $n = 1$ trivialmente SI es continua ya que

$$[\![b \rightarrow S]\!].X = b \wedge S.X$$

y $[\![b \rightarrow S]\!]$ es continua de serlo S; para $n \geq 2$, sabemos — por Teorema 4.24: pág. 65 — que SI se puede escribir como una selección con dos guardas; por consiguiente bastará demostrar la conservación de la continuidad en el caso de dos guardas y proceder después por inducción sobre el número de guardas. Sean ahora f y g dos transformadores continuos y monótonos; entonces, para cada cadena $P_k \uparrow$,

$$
\begin{array}{ll}
 & f.(\exists i : i \geq 0 : P_i) \wedge g.(\exists j : j \geq 0 : P_j) \\
= & \quad \because f \text{ y } g \text{ continuas} \\
 & (\exists i : i \geq 0 : f.P_i) \wedge (\exists j : j \geq 0 : g.P_j) \\
= & \quad \because \text{cálculo} \\
 & \begin{array}{l} \exists i : i \geq 0 : f.P_i \wedge (\exists j : 0 \leq j \leq i : g.P_j) \\ \vee \\ \exists j : j \geq 0 : g.P_j \wedge (\exists i : 0 \leq i \leq j : f.P_i) \end{array} \\
= & \quad \because P_j \text{ creciente, } f \text{ y } g \text{ monótonas} \\
 & \exists i : i \geq 0 : f.P_i \wedge g.P_i
\end{array}
$$

y el transformador $(f \wedge g).X \doteq f.X \wedge g.X$ es continuo. Por otro lado es fácil ver que el transformador $f.X \doteq (b \Rightarrow S.X)$ es continuo y monótono cuando lo sea S; veamos por ejemplo la monotonía

$$
\begin{array}{ll}
 & [f.X \ \Rightarrow \ f.Y] \\
= & \quad \because \text{definición} \\
 & [(b \ \Rightarrow \ S.X) \ \Rightarrow \ (b \ \Rightarrow \ S.Y)] \\
\Leftarrow & \quad \because \text{CP} \\
 & [S.X \ \Rightarrow \ S.Y]
\end{array}
$$

De ello se concluye que el transformador $[\![b \rightarrow S \,\Box\, c \rightarrow T]\!]$ es continuo cuando S y T sean continuos y monótonos, ya que, *ptle*

$$
\begin{array}{ll}
 & [\![b \rightarrow S \,\Box\, c \rightarrow T]\!].X \\
= & \quad \because \text{para } f.X := b \ \Rightarrow \ S.X, \text{ y para } g.X := c \ \Rightarrow \ T.X \\
 & (b \vee c) \wedge (f \wedge g).X
\end{array}
$$

Para probar que la construcción bucle conserva la continuidad, basta verlo para una guarda, ya que todo bucle se rescribe como un bucle con una guarda

y cuerpo una selecctiva (Teorema 6.10:pág. 92), y la construcción selectiva conserva la continuidad. Sea pues el bucle $\mathcal{R} \doteq *[\![b \to S]\!]$, donde S es monótona y continua. Veamos en primer lugar que la continuidad de $\mathcal{R}$ es consecuencia de la continuidad de los predicados asociados H^k; en efecto:

$$
\begin{array}{ll}
 & [\ \mathcal{R}.(\exists i : i \geq 0 : P_i) \equiv (\exists i : i \geq 0 : \mathcal{R}.P_i)\ ] \\
= & \quad \because \text{semántica de } \mathcal{R} \\
 & [\ (\exists k : k \geq 0 : H^k.(\exists i : i \geq 0 : P_i)) \equiv (\exists i : i \geq 0 : \exists k : k \geq 0 : H^k.P_i)\ ] \\
= & \quad \because \text{cada } H^k \text{ es continuo} \\
 & [\ (\exists k : k \geq 0 : \exists i : i \geq 0 : H^k.P_i) \equiv (\exists i : i \geq 0 : \exists k : k \geq 0 : H^k.P_i)\ ] \\
= & \quad \because \text{intercambio de cuantificadores} \\
 & Cierto
\end{array}
$$

Para probar la continuidad de los transformadores H^k probaremos, por inducción sobre k, que cada H^k es continuo:

$$\forall k : k \geq 0 : [H^k.(\exists i : i \geq 0 : P_i) \equiv (\exists i : i \geq 0 : H^k.P_i)]$$

El caso base ($k = 0$) es consecuencia inmediata de la distributividad; el paso inductivo es consecuencia de ser H^{k+1} composición de operaciones continuas

$$H^{k+1}.X = H^0.X \vee [\![b \to S]\!].H^k.X$$

Es decir, el transformador dado por el miembro derecho de la ecuación anterior es continuo si lo son H^0, H^k y S. TEOREMA

EJEMPLO 8.2 (Un transformador no continuo: $desastre$) Estudiado en Ejemplo 3.8:45; definido como $[\ desastre.X \doteq [X]\ ]$ no es continuo; en efecto,

$$
\begin{array}{ll}
 & desastre.(\exists i : i \geq 0 : x \leq i) \\
= & [\exists i : i \geq 0 : x \leq i] \\
= & \quad \because x \text{ natural} \\
 & Cierto
\end{array}
\qquad
\begin{array}{ll}
 & \exists i : i \geq 0 : desastre.(x \leq i) \\
= & \exists i : i \geq 0 : [x \leq i] \\
= & \quad \because \text{ya que } x \text{ puede ser } i+1 \\
 & Falso
\end{array}
$$

y en consecuencia *desastre* no es continuo. Obsérvese que la incorporación de tal sentencia asignaría a x un valor indeterminismo no acotado. EJEMPLO

8.1. Consecuencias de la propiedad de continuidad

Primera consecuencia: Implementación recursiva de los bucles

TEOREMA 8.3 *Sea $\mathcal{R}$ el bucle $*[\![b \to S]\!]$, entonces $[\ b \wedge \mathcal{R}.X \equiv b \wedge S; \mathcal{R}.X\ ]$.*

NOTA 8.4 *Para un bucle genérico $*[\![\ \Box\ : 1 \leq i \leq n : b_i \to S_i]\!]$ aplicamos el Teorema 6.10:92 y el Teorema 8.3 para obtener $[\ OB \wedge \mathcal{R}.X \equiv OB \wedge S; \mathcal{R}.X\ ]$.*

Demostración.– Tenemos, *ptle*

$$
\begin{array}{ll}
 & b \wedge S; \mathcal{R}.X \\
= & \quad \because \text{semántica de ; y de } \mathcal{R} \\
 & b \wedge S.(\exists k : k \geq 0 : H^k.X) \\
= & \quad \because S \text{ continua, y } \{H^k.X\} \text{ es una cadena} \\
 & b \wedge \exists k : k \geq 0 : S.H^k.X
\end{array}
$$

$$
\begin{array}{ll}
= & \quad \because \text{distributividad} \\
& \exists k : k \geq 0 : b \wedge S.H^k.X \\
= & \quad \because [b \wedge H^0.X = F] \\
& \exists k : k \geq 0 : b \wedge (H^0.X \vee b \wedge S.H^k.X) \\
= & \quad \because \text{definición de } H^k \\
& \exists k : k \geq 0 : b \wedge H^{k+1}.X \\
= & \quad \because [b \wedge H^0.X = F] \\
& \exists k : k \geq 0 : b \wedge H^k.X \\
= & \quad \because \text{semántica de } \mathcal{R} \text{ y distributiva} \\
& b \wedge \mathcal{R}.X
\end{array}
$$

TEOREMA

El teorema anterior tiene la siguiente interpretación interesante:

> EN EL ENTORNO DE ALGUNA GUARDA LA EJECUCIÓN DE UN BUCLE EQUIVALE A LA EJECUCIÓN DEL CUERPO SEGUIDA DEL BUCLE.

que permite implementar los bucles vía la recursión ya que,

COROLARIO 8.5 *Todo bucle verifica su ecuación recursiva:*

$$\mathcal{R} = [\![\, b \to S; \mathcal{R} \,\Box\, \neg b \to nada \,]\!] \qquad (*)$$

Demostración.–

$$
\begin{array}{ll}
& [\![\, b \to S; \mathcal{R} \,\Box\, \neg b \to nada \,]\!].X \\
= & \quad \because \text{semántica selección binaria} \\
& b \wedge S; \mathcal{R}.X \vee \neg b \wedge X \\
= & \quad \because \text{Teorema 8.3} \\
& b \wedge \mathcal{R}.X \vee \neg b \wedge X \\
= & \quad \because [\, \mathcal{R}.X \equiv \neg b \wedge X \vee b \wedge \ \ldots \,] \text{ y por absorción} \\
& b \wedge \mathcal{R}.X \vee \neg b \wedge \mathcal{R}.X \\
= & \\
& \mathcal{R}.X
\end{array}
$$

COROLARIO

2ª consecuencia: Equivalencia entre indeterminismo acotado y continuidad

EJEMPLO 8.6 (La sentencia $Azar_x$) Esta asigna a la variable x un número natural arbitrario. Tal sentencia puede definirse con el transformador:

$$Azar_x.Z \;\doteq\; \forall k : k \in \mathbb{N} : x := k.Z$$

y puede describirse con la selectiva no acotada $[\![\ \Box\ : i \geq 0 : C \to x := i \,]\!]$, y también puede ser caracterizada por las siguientes propiedades,

(a) $[Azar_x.(x \in \mathbb{N})\,]$

(b) $\forall k : k \geq 0 : [\, Azar_x.(0 \leq x \leq k) \equiv Falso \,]$

De (a) se desprende que la sentencia termina siempre, y asignando a x un número natural; la condición (b) expresa que el valor final de x no está acotado por ningún valor prefijado, y por tanto la sentencia presenta indeterminismo

no acotado (*unbounded nondeterminacy*). Veamos que las dos condiciones (a) y (b) son incompatibles con la continuidad:

$$
\begin{array}{ll}
 & Cierto \\
= & \quad \because (a) \\
 & Azar_x.(x \in \mathbb{N}) \\
= & \\
 & Azar_x.(\exists k : k \geq 0 : x \in \mathbb{N} \wedge 0 \leq x \leq k) \\
= & \quad \because Azar_x \text{ es continua, } 0 \leq x \leq k \text{ es una cadena} \\
 & \exists k : k \geq 0 : Azar_x.(x \in \mathbb{N} \wedge 0 \leq x \leq k) \\
= & \quad \because (b) \\
 & \exists k : k \geq 0 : Falso \\
= & \\
 & Falso
\end{array}
$$

EJEMPLO

Por tanto concluimos que la continuidad implica el indeterminismo acotado (esta conclusión aparece ampliamente discutida en [Dijkstra, 1976], páginas 76–78 y 204–207). Sorprendentemente, como muestra [Dijkstra, 1982]:358–359, el indeterminismo acotado es equivalente a la continuidad. El argumento de Dijkstra está basado en la siguiente propiedad

> A PARTIR DE UN PROGRAMA NO CONTINUO PODEMOS CONSTRUIR UN PROGRAMA CON INDETERMINISMO NO ACOTADO.

Para probarlo, sea S un programa no continuo; entonces existe una sucesión $\{C_k\} \uparrow$ de predicados tal que el siguiente predicado I no es idénticamente falso

$$
\begin{array}{ll}
 & I \\
\doteq & \quad \because \\
 & \exists k : k \geq 0 : S.C_k \not\equiv S.(\exists k : k \geq 0 : C_k) \\
= & \quad \because \text{denominamos } M \text{ y } N \text{ los predicados de la desigualdad} \\
 & M \not\equiv N \\
= & \quad \because \text{definición} \\
 & \neg(M \;\Rightarrow\; N \wedge N \;\Rightarrow\; M) \\
= & \quad \because \text{por ser } S \text{ monótona, } M \;\Rightarrow\; N \\
 & N \wedge \neg M \\
= & \\
 & S.(\exists k : k \geq 0 : C_k) \wedge \neg(\exists k : k \geq 0 : S.C_k) \qquad (*)
\end{array}
$$

Sea ι un estado verificando I y sea el programa

$$S' \;\doteq\; S; x := \text{mín}\{k : k \geq 0 \wedge C_k\}$$

que puede implementarse en la forma $S; k := 0; *[\![\neg C_k \;\rightarrow\; k := k + 1]\!]$. Si S' fuera indeterminista acotado, para cierto K tendríamos $\{I\}S'\{x \leq K\}$, de donde, por $(*)$

$$\{I\}S\{\exists k : k \geq 0 : C_k\}; x := \text{mín}\{k : k \geq 0 \wedge C_k\}\{x \geq 0\}$$

y por ser $\{C_k\}$ creciente, partiendo del estado ι, S termina verificando C_K, que es incompatible con el segundo miembro de la conjunción $(*)$; de aquí que el indeterminismo es acotado.

De todo ello se deduce que la inclusión de una sentencia indeterminista no acotada (tal como $Azar_x$) conlleva la no-continuidad, y viceversa. Es decir,

TEOREMA 8.7 *En el lenguaje de Dijkstra, la continuidad equivale a que todo programa proporciona indeterminismo acotado.*

8.2. Semántica de los bucles vía puntos fijos

Una consecuencia importante de la propiedad de continuidad (Teorema 8.3) es que todo bucle $\mathcal{R} \doteq *[\![b \to S]\!]$ verifica *su ecuación recursiva*

$$\mathcal{R} = [\![\neg b \to nada \,\square\, b \to S; \mathcal{R}]\!] \qquad (*)$$

Dicho de otro modo, $\mathcal{R}.X$ es solución de la ecuación en Y

$$Y : [\, Y \equiv \neg b \wedge X \vee b \wedge S.Y \,] \qquad (**)$$

La ecuación $(**)$ se llama la *ecuación característica* de $\mathcal{R}.X$. Recordemos que la ecuación $(*)$ permite dar una posible implementación de los bucles. Sin embargo, tal ecuación no puede servir como definición semántica de los bucles ya que puede tener varias soluciones, como muestra el siguiente ejemplo.

EJEMPLO 8.8 La sentencia $nada$ verifica la ecuación en Υ

$$\Upsilon : \Upsilon = [\![\neg b \to nada \,\square\, b \to nada; \Upsilon]\!] \qquad (*')$$

ya que, para cada X, $nada.X$ es solución de la ecuación en Y

$$Y : [\, Y = \neg b \wedge X \vee b \wedge nada.Y \,] \qquad (**')$$

Pero también es solución de $(**')$ el valor $*[\![b \to nada]\!].X$, que equivale a $\neg b \wedge X$. Luego la ecuación $(*')$ tiene al menos dos soluciones: $nada$ y el bucle $*[\![b \to nada]\!]$. Es fácil ver que cualquier solución de $(**')$ no es menor que $\neg b \wedge X$

$$\begin{array}{ll} & [\, Y \equiv \neg b \wedge X \vee b \wedge nada.Y \,] \\ = & \quad \because \text{regla de oro} \\ & [\, \neg b \wedge X \;\Rightarrow\; Y \,] \end{array}$$

y por tanto $*[\![b \to nada]\!]$ es la menor solución de $(*')$. De la misma forma, para la ecuación

$$\mathcal{R} = [\![Falso \to nada \,\square\, Cierto \to S; \mathcal{R}]\!] \qquad (*'')$$

la menor solución es $*[\![Cierto \to S]\!]$, que equivale a $aborta$. Por tanto, parece lógico pensar que es la condición de minimalidad junto a la ecuación $(*)$ lo que caracteriza la semántica de los bucles, como veremos en este capítulo. EJEMPLO

8.9 *Encontrad una sentencia S tal que la ecuación $(*)$ tenga varias soluciones.*

Existencia de soluciones extremas Si quisiéramos definir la semántica del bucle como la solución mínima debemos antes asegurar la existencia de tales soluciones extremas. Sea entonces $f : \mathcal{P} \to \mathcal{P}$ un transformador de predicados; denotemos con $\mu Y : f.Y$ la solución más fuerte de

$$Y : [Y = f.Y]$$

Es decir, el mínimo punto fijo de f. El teorema de Tarski-Knaster (véase Teorema 2.13:29) asegura la existencia de tal solución y la caracteriza:

TEOREMA 8.10 (Tarski–Knaster, 1955) *Sea $f : D \to D$ una función monótona y D un retículo completo; entonces:*

(i) *existe* $\mu Y : f.Y$.
(ii) *Si* $f.Z \leq Z$ *entonces* $(\mu Y : f.Y) \leq Z$.

OBSERVACIÓN.– (ii) caracteriza al mínimo punto fijo ya que si Y_1 es un punto fijo, satisface $Y_1 \leq f.Y_1$, y por (ii), $(\mu Y : f.Y) \leq Y_1$. OBS

Olvidemos provisionalmente la definición de semántica de los bucles vista hasta el momento y demos una nueva definición; al final del capítulo veremos que coinciden (Teorema 8.23). Queremos caracterizar el valor $\mathcal{R}.X$ como el mínimo punto fijo del transformador de predicados

$$Y \longmapsto g.X.Y(\ \doteq\ \neg b \wedge X \vee b \wedge S.Y)$$

La existencia del menor punto fijo viene asegurada por el teorema de Tarski-Knaster (o T–K para simplificar). Por ello es suficiente demostrar que el transformador de predicados $g.X.Y$ es monótono en Y. Es más, es conjuntivo en ambas variables:

$$\forall P, Q, M, N :: [g.P.M \wedge g.Q.N \equiv g.(P \wedge Q).(M \wedge N)]$$

siempre que S sea conjuntivo, lo cual ocurre si S es *sano*. En particular, $g.X$ es monótono por ser conjuntivo.

DEFINICIÓN 8.11 (Semántica del bucle en términos de puntos fijos) *Sea el bucle $\mathcal{R} \doteq *[\![b \to S]\!]$, con S sano; se define su semántica como*

$$[\ \mathcal{R}.X \doteq \mu Y : \neg b \wedge X \vee b \wedge S.Y\ ]$$

Para un bucle con varias guardas $*SI$, siendo SI una selectiva genérica (con varias guardas), definimos su semántica como la semántica del bucle con una sola guarda $*[\![OB \to SI]\!]$. Por consiguiente, se seguirá satisfaciendo la equivalencia $*SI = *[\![OB \to SI]\!]$.

NOTA 8.12 *De la Definición 8.11 anterior deducimos que*

SI $\neg b \wedge X$ ES SOLUCIÓN DE LA ECUACIÓN $(**)$, ENTONCES TAMBIÉN ES LA MENOR SOLUCIÓN, Y POR TANTO $[\mathcal{R}.X \equiv \neg b \wedge X]$.

Este es el caso del Ejemplo 8.8, y del siguiente.

EJEMPLO 8.13 Sea el bucle estudiado en el Ejemplo 6.8

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

para el que queremos probar $[\ \mathcal{R}.X \equiv \neg b \wedge X\ ]$. Según la Nota 8.12 bastará demostrar que $\neg b \wedge X$ es solución de la ecuación $(**)$. En efecto.

$$\begin{array}{ll} & \neg b \wedge X \vee b \wedge [\![b \to x := x + 1 \,\square\, b \to b := Falso]\!].(\neg b \wedge X) \\ = & \quad \because \text{semántica selección} \\ & \neg b \wedge X \vee b \wedge (b \ \Rightarrow\ x := x + 1.(\neg b \wedge X)) \wedge (b \ \Rightarrow\ b := Falso.(\neg b \wedge X)) \\ = & \quad \because \text{CP} \end{array}$$

$$
\begin{array}{ll}
& \neg b \wedge X \vee b \wedge x := x+1.(\neg b \wedge X) \wedge b := Falso.(\neg b \wedge X) \\
= & \quad \because \text{def. sustitución} \\
& \neg b \wedge X \vee b \wedge \neg b \wedge \ \ldots \\
= & \quad \because \text{CP} \\
& \neg b \wedge X
\end{array}
$$

Observe el lector que no hemos utilizado la segunda guarda ni la segunda sentencia guardada. Por consiguiente, el bucle $*[\![b \rightarrow x := x+1 \,\square\, \ldots]\!]$ será equivalente a la sentencia $[\![\neg b \rightarrow nada]\!]$, siempre que la disyunción de las guardas sea b. EJEMPLO

8.14 *[300] Sea el bucle $\mathcal{R} \doteq *[\![b \rightarrow S]\!]$, donde $[S.\neg b \equiv Falso]$. Probad utilizando la Definición 8.11, que $[\mathcal{R}.X \equiv \neg b \wedge X]$ ¿Qué interpretación tiene?*

8.15 *[300]* **(Febrero, 96)** *Sea el bucle $\mathcal{R} \doteq *[\![b \rightarrow nada]\!]$ ¿Cuándo se da $\{b\}\mathcal{R}\{Cierto\}$? ¿Qué interpretación tiene?*

8.3. Salubridad de los bucles, determinismo y teorema de invariantes en términos de puntos fijos

Los resultados aquí descritos están formulados para bucles con una sola guarda; para varias guardas recordemos la equivalencia $*SI = *[\![OB \rightarrow SI]\!]$.

Para que el lenguaje obtenido con la Definición 8.11 sea sano hemos de probar que si S es estricto (LME) y conjuntivo, entonces $\mathcal{R}$ es estricto y conjuntivo. La LME es fácil ya que por ser S estricto, $F = b \wedge S.F$, luego F es el menor punto fijo de $g.F$. La conjuntividad de $\mathcal{R}.X$ será consecuencia del siguiente lema extraído de [Morris, 1990]:90.

LEMA 8.16 *Sea g un transformador conjuntivo en sus dos argumentos, por lo que está determinado el transformador $h.X \doteq \mu Y : g.X.Y$. Se verifican,*

(i) *h es conjuntivo.*

(ii) *Si g es disyuntivo, entonces h es disyuntivo.*

En particular, tomando $g.X.Y \doteq \neg b \wedge X \vee b \wedge S.Y$, siendo S sano, y tomando el bucle $\mathcal{R} \doteq *[\![b \rightarrow S]\!]$,

(iii) *$\mathcal{R}$ es sano.*

(iv) *Si S es disyuntivo (determinista), entonces $\mathcal{R}$ es disyuntivo.*

Demostración.– Por Lema 1.9:15, basta probar $[h.(P \overset{\wedge}{\vee} Q) \equiv h.P \overset{\wedge}{\vee} h.Q]$ por doble implicación, donde $\overset{\wedge}{\vee}$ puede ser la conjunción ($\wedge$) o la disyunción ($\vee$). La implicación $\Rightarrow$ la hacemos en forma simultánea:

$$
\begin{array}{ll}
& [h.(P \overset{\wedge}{\vee} Q) \ \Rightarrow \ h.P \overset{\wedge}{\vee} h.Q] \\
= & [\mu Y : g.(P \overset{\wedge}{\vee} Q).Y \ \Rightarrow \ h.P \overset{\wedge}{\vee} h.Q] \\
\Leftarrow & \quad \because \text{Teorema } 8.10(ii) \\
& [g.(P \overset{\wedge}{\vee} Q).(h.P \overset{\wedge}{\vee} h.Q) \ \Rightarrow \ h.P \overset{\wedge}{\vee} h.Q] \\
= & \quad \because g \text{ es conjuntivo/disyuntivo} \\
& [g.P.(h.P) \overset{\wedge}{\vee} g.Q.(h.Q) \ \Rightarrow \ h.P \overset{\wedge}{\vee} h.Q]
\end{array}
$$

$=$ $\quad \because h.P$ y $h.Q$ son puntos fijos
$\quad [h.P \overset{\wedge}{\vee} h.Q \;\Rightarrow\; h.P \overset{\wedge}{\vee} h.Q]$
$=$
$\quad Cierto$

La prueba anterior, así como las siguientes, son válidas por una colección arbitraria de predicados $P \overset{\wedge}{\vee} Q \overset{\wedge}{\vee} \ldots$. Ahora terminamos de probar (i).

$\quad [h.P \wedge h.Q \;\Rightarrow\; h.(P \wedge Q)]$
$=$ $\quad \because$ regla de intercambio
$\quad [h.P \;\Rightarrow\; \neg h.Q \vee h.(P \wedge Q)]$
$\Leftarrow$ $\quad \because$ Teorema 8.10(ii), $(\mu Y : g.P.Y) \doteq h.P$
$\quad [g.P.(\neg h.Q \vee h.(P \wedge Q)) \;\Rightarrow\; \neg h.Q \vee h.(P \wedge Q)]$
$=$ $\quad \because$ regla de intercambio
$\quad [h.Q \wedge g.P.(\neg h.Q \vee h.(P \wedge Q)) \;\Rightarrow\; h.(P \wedge Q)]$
$=$ $\quad \because h.Q$ es punto fijo
$\quad [g.Q.(h.Q) \wedge g.P.(\neg h.Q \vee h.(P \wedge Q)) \;\Rightarrow\; h.(P \wedge Q)]$
$=$ $\quad \because g$ es conjuntiva
$\quad [g.(P \wedge Q).(h.Q \wedge h.(P \wedge Q)) \;\Rightarrow\; h.(P \wedge Q)]$
$=$ $\quad \because h.(P \wedge Q)$ es punto fijo
$\quad [g.(P \wedge Q).(h.Q \wedge h.(P \wedge Q)) \;\Rightarrow\; g.(P \wedge Q).h.(P \wedge Q)]$
$=$ $\quad \because g$ es monótona en segundo argumento por ser conjuntiva
$\quad Cierto$

Por ser h conjuntivo, también es monótono, y la implicación

$$[h.(P \vee Q \vee \ldots) \;\Leftarrow\; h.P \vee h.Q \ldots]$$

es trivial. Esto prueba (ii). La prueba de (iv) sigue de (ii) teniendo en cuenta que el transformador $\neg b \wedge X \vee b \wedge S.Y$ en disyuntivo en X e Y. Demos finalmente, otra demostración directa de (iv). Por ser $\mathcal{R}.(X \vee X' \vee \ldots)$ la menor solución de la ecuación:

$$Y : [\; Y = \neg b \wedge (X \vee X' \vee \ldots) \vee b \wedge S.Y \;]$$

basta probar que $\mathcal{R}.X \vee \mathcal{R}.X' \vee \ldots$ verifica la ecuación anterior:

$\quad \mathcal{R}.X \vee \mathcal{R}.X' \vee \ldots$
$=$ $\quad \because$ semántica de bucles según p.f.
$\quad \neg b \wedge X \vee b \wedge S.\mathcal{R}.X \vee \neg b \wedge X' \vee b \wedge S.\mathcal{R}.X' \vee \ldots$
$=$ $\quad \because S$ determinista, cálculo de predicados
$\quad \neg b \wedge (X \vee X' \vee \ldots) \vee b \wedge S.(\mathcal{R}.X \vee \mathcal{R}.X' \vee \ldots)$ LEMA

Luego hemos caracterizado el bucle en términos de puntos fijos y hemos justificado que coincide con la semántica operacional.

Veamos para finalizar algunas conclusiones que se deducen fácilmente de la semántica según puntos fijos, algunas de las cuales las hemos demostrado ya anteriormente utilizando la semántica tradicional (inductiva). Las demostramos nuevamente para que el lector compruebe la elegancia de las demostraciones. Los enunciamos para una sola guarda ya que se verifica la equivalencia $*SI = *[\![OB \to SI]\!]$.

TEOREMA 8.17 *Sea* $\mathcal{R} \doteq *[\![b \to S]\!]$. *Se verifican*

(i) $[\mathcal{R}.X \equiv \mathcal{R}.(X \wedge \neg b)]$.
(ii) $[Q \;\Rightarrow\; \neg b] \;\Rightarrow\; [Q \wedge \mathcal{R}.X = Q \wedge X]$.
(iii) *Sea otro bucle* $\mathcal{R}' \doteq *[\![b' \to S']\!]$; *entonces* $[b' \;\Rightarrow\; b] \;\Rightarrow\; [\mathcal{R}; \mathcal{R}' = \mathcal{R}]$.

NOTA *8.18* *(i) afirma que al terminar el bucle la guarda es falsa; (ii) dice que si en el entorno Q la guarda es falsa, $\mathcal{R}$ equivale a la sentencia $nada$; (iii) tiene una interpretación operacional simple: puesto que al terminar $\mathcal{R}$ la guarda es falsa, será también falsa la guarda de $\mathcal{R}'$ al comenzar éste, y por tanto $\mathcal{R}'$ no ejecutará su cuerpo.*

Demostración.– (i).– Tenemos que $g.X = g.(\neg b \wedge X)$, luego tienen los mismos puntos fijos.

(ii).– si $[Q \;\Rightarrow\; \neg b]$ entonces,
$[Q \wedge b \equiv F]$, y $[Q \wedge \neg b \equiv Q]$,
luego, *ptle*:

$$
\begin{array}{ll}
 & Q \wedge \mathcal{R}.X \\
= & \quad \because \mathcal{R}.X \text{ es punto fijo} \\
 & Q \wedge (\neg b \wedge X \vee b \wedge S.\mathcal{R}.X) \\
= & \quad \because \text{CP} : [Q \;\Rightarrow\; \neg b] \\
 & Q \wedge X
\end{array}
$$

(iii).–

$$
\begin{array}{ll}
 & \mathcal{R}; \mathcal{R}'.X \\
= & \quad \because \text{semántica composición} \\
 & \mathcal{R}.(\mathcal{R}'.X) \\
= & \quad \because \text{por } (i) \\
 & \mathcal{R}.(\neg b \wedge \mathcal{R}'.X) \\
= & \quad \because (ii) \text{ con } \mathcal{R}, Q == \mathcal{R}', \neg b \\
 & \mathcal{R}.(\neg b \wedge X) \\
= & \quad \because \text{por } (i) \\
 & \mathcal{R}.X
\end{array}
$$

TEOREMA

Los siguientes ejemplos ilustran de nuevo cómo usar la técnica de los puntos fijos para mostrar la equivalencia semántica de bucles.

EJEMPLO 8.19 En Teorema 6.14 vimos que los bucles $\mathcal{R} \doteq *[\![b \to S]\!]$ y $\mathcal{R}' \doteq *[\![b \to T]\!]$ tienen el mismo comportamiento si los cuerpos son equivalentes en el entorno de la guarda

$$\forall X :: [\; b \wedge S.X \equiv b \wedge T.X \;] \qquad (*)$$

Veamos una demostración de esto último en términos de puntos fijos (en la solución del Ejercicio 8.20 aparece otra demostración alternativa en el caso en que S y T sean equivalentes en el entorno de un predicado invariante):

$$
\begin{array}{ll}
 & *[\![b \to S]\!].X \\
= & \quad \because \text{semántica (mínimo punto fijo)} \\
 & \mu Y :: \neg b \wedge X \vee b \wedge S.Y \\
= & \quad \because (*) \\
 & \mu Y :: \neg b \wedge X \vee b \wedge T.Y \\
= & \quad \because \text{semántica (mínimo punto fijo)} \\
 & *[\![b \to T]\!].X
\end{array}
$$

EJEMPLO

8.20 *[300]* **(Febrero, 93)** *Resuelva el Ejercicio 6.16 utilizando puntos fijos. AYUDA.- en el Ejemplo 8.19 se prueba que los bucles tienen el mismo comportamiento si $\forall X :: [b \wedge S.X \equiv b \wedge T.X]$, pero en el presente ejercicio se prueba que tienen el mismo comportamiento en presencia del invariante P; el lector deberá observar la diferencia importante entre los dos conceptos.*

TEOREMA 8.21 (Teorema de Invariantes) *Para todo bucle* $\mathcal{R} \doteq *[\![b \to S]\!]$,

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

Demostración.– Adaptamos una prueba de [van Gasteren, 1990]:

$$
\begin{array}{ll}
 & [\ P \wedge \mathcal{R}.C \ \Rightarrow\ \mathcal{R}.P\ ] \\
= & \quad \because \text{regla de intercambio} \\
 & [\ \mathcal{R}.C \ \Rightarrow\ \neg P \vee \mathcal{R}.P\ ] \\
\Leftarrow & \quad \because \text{Teorema 8.10}(ii),\ \mathcal{R}.C = \mu Y : f.Y \\
 & [\ \neg b \vee b \wedge S.(\neg P \vee \mathcal{R}.P) \ \Rightarrow\ \neg P \vee \mathcal{R}.P\ ] \\
= & \quad \because \text{regla de intercambio} \\
 & [\ \neg b \wedge P \vee b \wedge P \wedge S.(\neg P \vee \mathcal{R}.P) \ \Rightarrow\ \mathcal{R}.P\ ] \\
= & \quad \because \text{cálculo} \\
 & \wedge\ [\ \neg b \wedge P \ \Rightarrow\ \mathcal{R}.P\ ] \\
 & \quad\ [\ b \wedge P \wedge S.(\neg P \vee \mathcal{R}.P) \ \Rightarrow\ \mathcal{R}.P\ ] \\
\Leftarrow & \quad \because [\ \mathcal{R}.P = \neg b \wedge P \vee b \wedge S.\mathcal{R}.P\ ], [\ P \wedge b \ \Rightarrow\ S.P\ ], \text{transitividad de } \Rightarrow \\
 & \wedge\ Cierto \\
 & \quad\ [\ b \wedge S.P \wedge S.(\neg P \vee \mathcal{R}.P) \ \Rightarrow\ \mathcal{R}.P\ ] \\
= & \quad \because \text{conjuntividad de } S \\
 & [\ b \wedge S.(P \wedge \mathcal{R}.P) \ \Rightarrow\ \mathcal{R}.P\ ] \\
\Leftarrow & \quad \because \text{monotonía de } S, \text{ transitividad de } \Rightarrow \\
 & [\ b \wedge S.\mathcal{R}.P \ \Rightarrow\ \mathcal{R}.P\ ] \\
= & \quad \because [\mathcal{R}.P = \neg b \wedge P \vee b \wedge S.\mathcal{R}.P] \\
 & Cierto
\end{array}
$$

TEOREMA

Continuidad de la construcción bucle Con la nueva semántica de bucles, nuestro lenguaje sigue siendo continuo. Para probarlo bastará razonar por inducción sobre las sentencias del lenguaje, y probar además que los bucles introducidos con la Definición 8.11 siguen conservando la continuidad.

LEMA 8.22 *Si S es continuo, entonces también lo es $\mathcal{R}(\ \doteq\ * [\![\ b \rightarrow S\ ]\!])$.*

Demostración.– Sea una cadena de predicados C_k. Hay que probar, *ptle*

$$\mathcal{R}.(\exists k : k \geq 0 : C_k) \equiv \exists k : k \geq 0 : \mathcal{R}.C_k$$

La implicación $\Leftarrow$ es consecuencia de la monotonía. Veamos $\Rightarrow$:

$$
\begin{array}{ll}
 & [\ \mathcal{R}.(\exists k : k \geq 0 : C_k) \ \Rightarrow\ \exists k : k \geq 0 : \mathcal{R}.C_k\ ] \\
\Leftarrow & \quad \because \text{Teorema 8.10}(ii) \\
 & [\ \neg b \wedge (\exists k : k \geq 0 : \mathcal{R}.C_k) \vee b \wedge S.(\exists k : k \geq 0 : \mathcal{R}.C_k) \ \Rightarrow\ \exists k : k \geq 0 : \mathcal{R}.C_k\ ] \\
= & \quad \because S \text{ es continuo} \\
 & [\ \neg b \wedge (\exists k : k \geq 0 : \mathcal{R}.C_k) \vee b \wedge (\exists k : k \geq 0 : S.\mathcal{R}.C_k) \ \Rightarrow\ \exists k : k \geq 0 : \mathcal{R}.C_k\ ] \\
= & \quad \because \text{distrib., y conmutativa} \\
 & [\ \exists k : k \geq 0 : \neg b \wedge \mathcal{R}.C_k \vee b \wedge S.\mathcal{R}.C_k \ \Rightarrow\ \exists k : k \geq 0 : \mathcal{R}.C_k\ ] \\
= & \quad \because \mathcal{R}.C_k \text{ es un punto fijo} \\
 & [\ \exists k : k \geq 0 : \mathcal{R}.C_k \ \Rightarrow\ \exists k : k \geq 0 : \mathcal{R}.C_k\ ] \\
= & \quad \because \text{CP} \\
 & Cierto
\end{array}
$$

TEOREMA

TEOREMA 8.23 (Equivalencia entre las dos semánticas) *La semántica de los bucles en términos inductivos dada por la Definición 6.2 coincide con la semántica en términos de puntos fijos de la Definición 8.11.*

Demostración.– Tendremos que probar, *ptle*

$$\mu Y : g.X.Y \equiv \exists n : n \geq 0 : H^n.X$$

Ya que S es continuo, por el Lema 8.22, el transformador $g.X$ es continuo, y por el teorema de Kleene (Teorema 2.14):

$$\mu Y : g.X.Y \equiv \exists n : n \geq 0 : (g.X)^n.F$$

ya que $Falso$ es el ínfimo en el retículo $\mathcal{P}$, y siendo

$$(g.X)^0.F = F \qquad (g.X)^{n+1}.Z = (g.X)^n.((g.X).Z)$$

Demostremos por inducción que $[(g.X)^{n+1}.F \equiv H^n.X]$; para $n = 0$ es trivial; por otro lado, *ptle*, $H^n.X$

$=$ $\quad \because$ definición y semántica de la selectiva
$\neg b \wedge X \vee b \wedge S.H^{n-1}.X$
$=$ $\quad \because$ definición de $g.X$
$g.X.(H^{n-1}.X)$
$=$ $\quad \because$ hipótesis de inducción
$g.X.((g.X)^n.X)$
$=$ $\quad \because$ definición
$(g.X)^{n+1}.X$ TEOREMA

Ejercicios

8.24 *[301] Probad la igualdad* $*[\![b \to nada]\!] \;=\; *[\![b \to aborta]\!]$.

8.25 *[301] Utilizando la semántica según puntos fijos, probar* $\mathcal{S} = aborta$, *siendo*

$$\begin{array}{lcl} \mathcal{R} & \doteq & *[\![b \to x := x + 1 \,\Box\, b \to b := Falso]\!] \\ \mathcal{S} & \doteq & b := Cierto; \mathcal{R} \end{array}$$

8.26 *[301] Sea el operador infijo '*$*$*' definido en la forma* $b * S \doteq *[\![b \to S]\!]$. *Probad o refutad las siguientes propiedades:*

$$\begin{array}{lll} (a) & (b \vee c) * S & = \quad b * S; c * (S; b * S) \\ (b) & (b \wedge c) * S; b * S & = \quad b * S \end{array}$$

8.27 *[301] Sea el bucle* $\mathcal{R} \doteq *[\![q \to nada \,\Box\, q \to q := \neg q]\!]$

(A) Demostrad $[\mathcal{R}.C \equiv \neg q]$ *utilizando puntos fijos.*

(B) Demostrad $[\mathcal{R}.C \equiv \neg q]$ *utilizando la semántica inductiva.*

(C) Utilizando lo anterior probad $\mathcal{R} = [\![\neg q \to nada]\!]$.

8.28 *Probad las siguientes equivalencias utilizando puntos fijos y semántica inductiva:*

$$\begin{array}{lcl} *[\![Cierto \to S]\!] & = & aborta \\ *[\![b \to S \,\Box\, \neg b \to T]\!] & = & aborta \end{array}$$

8.29 *[301] Sea* $\mathcal{R} \doteq *[\![b \to S]\!]$ *un bucle arbitrario. Probad* $[\mathcal{R}.C \;\Rightarrow\; \mathcal{R}.\neg b]$ *utilizando la semántica inductiva y la semántica según puntos fijos.*

8.30 *[302] Probad e interpretad, utilizando la semántica en términos de puntos fijos*

$$[A \;\Rightarrow\; \mathcal{R}.C] \;\Rightarrow\; [S.A \;\Rightarrow\; \mathcal{R}.C]$$

8.31 *[302]* **(Marzo, 94)** *Probad que los bucles* $*[\![\, b \to S \,]\!]$ *verifican la ecuación*

$$\mathcal{X} = [\![\, \neg b \to nada \,\Box\, b \to S; \mathcal{X} \,]\!] \qquad (*)$$

pero que tal ecuación no sirve como definición semántica de los bucles. AYUDA.- encontrad sentencias tales que la ecuación $(*)$ *tenga varias soluciones.*

8.32 *[302]* **(Marzo, 94)** *Sea el bucle* $\mathcal{R} \doteq *[\![\, b \to S \,]\!]$. *La siguiente es una prueba de* $\{b\}S\{\neg b\} \;\Rightarrow\; [\mathcal{R}.C]$. *Completad los comentarios que faltan*

$$\begin{array}{ll} & \{b\}S\{\neg b\} \\ = & \quad \because \text{\_\_\_\_\_} \\ & [b \;\Rightarrow\; S.\neg b] \\ \Rightarrow & \quad \because \text{\_\_\_\_\_} \\ & [b \;\Rightarrow\; S.\mathcal{R}.C] \\ = & \quad \because \text{\_\_\_\_\_} \\ & [b \wedge S.\mathcal{R}.C \equiv b] \\ \Rightarrow & \quad \because \text{\_\_\_\_\_} \\ & [\mathcal{R}.C \equiv \neg b \vee b] \\ = & \quad \because \text{\_\_\_\_\_} \\ & [\mathcal{R}.C] \end{array}$$

8.33 *[302]* **(Julio, 94)** *Probad* $[\mathcal{R}.C \equiv x \leq 1]$ *utilizando puntos fijos, donde*

$$\mathcal{R} \;\doteq\; *[\![\, x > 1 \to x := x + 1 \,\Box\, x > 2 \to x := x - 4 \,]\!]$$

¿Qué interpretación tiene? ¿Cambia el resultado si tomamos el bucle

$$\mathcal{R}' \;\doteq\; *[\![\, x > 1 \to x := x + 1 \,\Box\, x > 2 \to \mathcal{S} \,]\!] \;\; ?$$

8.34 *[303]* **(Diciembre, 94)** *Sea el bucle* $\mathcal{R} \doteq *[\![\, b \to S \,]\!]$.

(A) Probad, utilizando puntos fijos, $\forall X :: [\mathcal{R}.X \equiv \mathcal{R}.(\neg b \wedge X)]$.

(B) Utilizando (A), probad la igualdad $\mathcal{R}; [\![\, b \to A \,\Box\, \neg b \to B \,]\!] \;=\; \mathcal{R}; B$.

8.35 *[303]* **(Enero, 95)** *Sean* S *y* T *dos sentencias deterministas,* p *y* q *dos predicados incompatibles (*$p \wedge q = Falso$*) y la sentencia* $\mathcal{R} \doteq *[\![\, p \to S \,\Box\, q \to T \,]\!]$.

(A) Probad que $\mathcal{R}$ *es determinista, utilizando la semántica según puntos fijos.*

(B) Probad que si $[p \vee q = Cierto]$*, entonces* $\mathcal{R} \equiv aborta$.

(C) Calculad e interpretad el triple $\{C\}\mathcal{R}\{C\}$.

8.36 *[304]* **(Noviembre, 96)** *Sea el bucle* $\mathcal{R} \doteq *[\![\, i \neq N \to i := i + 1 \,]\!]$.

(A) ¿Cual es la ecuación en términos de puntos fijos para la semántica de $\mathcal{R}.C$*?*

(B) Probad que $i \leq N$ *es un punto fijo.*

(C) Probad, por inducción, que para cualquier punto fijo Y_1 *se tiene*

$$\forall k : 0 \leq k : [\; N - k \leq i \leq N \;\Rightarrow\; Y_1 \;]$$

(D) Concluid que $i \leq N$ *es el menor punto fijo, y por tanto* $[\mathcal{R}.C \equiv i \leq N]$.

8.37 *[304]* **(Setiembre, 02)** *Sea el bucle* $\mathcal{R} \doteq *[\![\, b \to S \,]\!]$.

(A) *Probad* $\{b\}S\{\neg b\} \;\Rightarrow\; [\mathcal{R}.C]$ *utilizando la semántica en términos de puntos fijos. AYUDA.- Usad* $[\mathcal{R}.C \equiv \mathcal{R}.(\neg b)]$ *y un esquema como el siguiente,* $ptle$*,*

$$
\begin{array}{ll}
 & \mathcal{R}.C \\
= & \quad \because [\mathcal{R}.C \equiv \mathcal{R}.(\neg b)] \\
 & \mathcal{R}.(\neg b) \\
= & \quad \because \textit{semántica en términos de puntos fijos} \\
 & \neg b \vee b \wedge S.\mathcal{R}.(\neg b) \\
= & \quad \because \underline{\qquad\qquad\qquad\qquad} \\
 & \neg b \vee b \wedge S.(\neg b \vee b \wedge S.\mathcal{R}.(\neg b)) \\
\Leftarrow & \quad \because \underline{\qquad\qquad\qquad\qquad} \\
 & \neg b \vee b \wedge S.\neg b \\
= & \quad \because \underline{\qquad\qquad\qquad\qquad} \\
 & \neg b \vee b \\
= & \quad \because \underline{\qquad\qquad\qquad\qquad} \\
 & Cierto
\end{array}
$$

(B) *¿Qué interpretación tiene la implicación anterior?*

(C) *Dad un contraejemplo para el cual la implicación recíproca sea falsa.*

8.38 **(Diciembre, 99)** *Sea el bucle* $\mathcal{R} \doteq *[\![x < N \to x := x+1 \,\square\, x < N \to x := x+2]\!]$.

(A) *Encontrar un contador del bucle.*

(B) *Utilizando tal contador, probad,* $[\mathcal{R}.C \equiv C]$.

(C) *Probad, utilizando puntos fijos, que* $[\mathcal{R}.(x = N) \equiv (x = N)]$*, y dad una interpretación de esto último.*

(D) *Probad e interpretad que* $[\mathcal{R}'.C \equiv x \geq N]$ *si modificamos el bucle en la forma*

$$\mathcal{R}' \doteq *[\![x < N \to x := x+1 \,\square\, x < N \to x := x-1]\!]$$

8.39 *[305]* **(Diciembre, 00)** *Sea* $\mathcal{S}$ *el cuerpo del bucle*

$$\mathcal{R} \doteq *[\![z \neq 0 \to z := z+1 \,\square\, z \neq 0 \to z := z+2]\!]$$

(A) *Justificad (con una frase) que* $[(z = 0) \equiv \mathcal{R}.C]$

(B) *Probad, y justificad que, para cualquier* a, $[\mathcal{S}.(z = a) \equiv Falso]$.

(C) *Demostrad que* $\mathcal{S}$ *es indeterminista. AYUDA.- Calculad* $\mathcal{S}.(z = 2 \vee z = 3)$.

(D) *Escribid la ecuación que determina la semántica en TPF de* $\mathcal{R}.C$.

(E) *Demostrad que* $z = 0$ *es el menor punto fijo de la ecuación del apartado* (D).

(F) *Calculad* k *para que se verifique el triplete* $\{z = k\}\mathcal{R}\{z = 0\}$.

8.40 *[305]* **(Febrero, 01)** *Siendo* x *es una variable entera, sea el bucle*

$$\mathcal{R} \doteq *[\![x > 0 \to x := x-1 \,\square\, x > 0 \to x := x-2]\!]$$

(A) *Utilizando la semántica inductiva de los bucles, prueba que* $[C \equiv \mathcal{R}.C]$.

(B) *Usando el apartado A, prueba* $[C \equiv \mathcal{R}.(x \leq 0)]$.

(C) *Utilizando PF prueba* $[x = 0 \equiv \mathcal{R}.(x = 0)]$*, así como* $[x < 0 \equiv \mathcal{R}.(x < 0)]$.

(D) *Deduce de los apartados* (B) *y* (C) *que* $\mathcal{R}$ *es indeterminista.*

8.41 *[306]* **(Junio, 99)** *Sea la sentencia* $desastre$ *definida en la forma* $[\; desastre.X \doteq [X] \;]$.

(A) Estudiad bajo qué condiciones es indeterminista.

(B) Probad que si es indeterminista, tiene indeterminismo no acotado.

(C) Probad que el programa $x := 8; *[\![x > 3 \to desastre]\!]$ *nunca termina.*

(D) Probad que el bucle $*[\![x > 1 \to x := x - 1; Azar_y \ \square \ y > 1 \to y := y - 1]\!]$ *siempre termina.*

(E) ¿Qué ocurre si cambiamos en el bucle anterior la sentencia $Azar_y$ *por la sentencia* $desastre$*?*

8.42 **(Enero, 95)** *Sea* x *una variable entera, y el bucle*

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

Utilizando la semántica vía PF probad el triplete $\{x \neq 0\}\mathcal{R}\{x = 1\}$.

8.4. El Teorema de los Contadores Generalizados

Concepto de contador generalizado Estudiemos la terminación de

$$\begin{array}{l} \{m > 0 \wedge n > 0\} \\ i, j := m - 1, n - 1; \\ \quad *[\![\quad j \neq 0 \qquad\quad \to j := j - 1 \\ \quad\ \square \quad j = 0 \wedge i \neq 0 \quad \to i, j := i - 1, n - 1 \,]\!] \end{array}$$

Se observa que el predicado $I \doteq n, m > 0 \wedge i, j \geq 0$ verifica $\{m > 0 \wedge n > 0\}\ i, j := m - 1, n - 1\ \{I\}$, siendo invariante,

$$\begin{array}{ll} \Rightarrow & I \wedge j \neq 0 \\ & n, m > 0 \wedge i \geq 0 \wedge j > 0 \\ = & j := j - 1.I \end{array} \qquad \begin{array}{ll} \Rightarrow & I \wedge (j = 0 \wedge i \neq 0) \\ & n, m > 0 \wedge i > 0 \\ = & i, j := i - 1, n - 1.I \end{array}$$

Veamos que los valores de las parejas (i, j) generados por el programa constituyen una sucesión decreciente para la relación de orden lexicográfica:

$$(a, b) < (c, d) \text{ sii } a < c \vee a = c \wedge b < d$$

En efecto: es fácil ver que $I \wedge (i, j) = (a, b) \ \Rightarrow \ SI.(i, j) < (a, b)$ ya que

$$\begin{array}{ll} = & j := j - 1.(i, j) < (a, b) \\ & (i, j - 1) < (a, b) \\ \Leftarrow & \because \text{orden lexicográfico} \\ & (i, j) = (a, b) \end{array} \qquad \begin{array}{ll} = & i, j := i - 1, n - 1.(i, j) < (a, b) \\ & (i - 1, n - 1) < (a, b) \\ \Leftarrow & \because \text{orden lexicográfico} \\ & (i, j) = (a, b) \end{array}$$

y por tanto el programa genera una sucesión estrictamente decreciente. Pero en el conjunto $\mathcal{C}(= \mathbb{N}^2)$ cualquier sucesión decreciente es finita. Además, de ser

$$[\ I \wedge OB \ \Rightarrow \ t \in \mathcal{C}\ ]$$

concluimos que el bucle debe terminar ya que si OB se da indefinidamente se obtendría una sucesión estrictamente decreciente e infinita de elementos de $\mathcal{C}$,

lo que es imposible. Por el Teorema 1.23:23, un conjunto donde cada sucesión estrictamente decreciente debe ser finita es *bien construido* (*well-founded set*).

Sea ahora t la función con origen el espacio de estados y destino el conjunto parcialmente ordenado $\mathbb{Z}^2$ ($t : \xi \to \mathbb{Z}^2$) definida en la forma $t \doteq (i, j)$. Diremos que t es una estructura de tipo $\mathbb{Z}^2$. Entonces, hemos probado que para nuestro programa se verifican las propiedades

$$\begin{array}{ll} (a') & [I \wedge OB \Rightarrow SI.I] \qquad \text{— } I \text{ es un invariante} \\ (b') & [I \wedge OB \Rightarrow t \in \mathcal{C}] \\ (c') & \forall t_0 :: [I \wedge OB \wedge t = t_0 \Rightarrow SI.t < t_0] \end{array}$$

Resumimos las tres condiciones anteriores diciendo que

> t ES UN $\mathbb{Z}^2$–CONTADOR (O CONTADOR GENERALIZADO DE TIPO $\mathbb{Z}^2$) ASOCIADO AL CONJUNTO BIEN CONSTRUIDO $\mathcal{C}$ Y RELATIVO AL INVARIANTE I.

Compárese (b')-(c') con las condiciones del *Teorema de los Contadores* (Teorema 6.38:105): un contador en sentido tradicional no es más que un $\mathbb{Z}$–contador. Por tanto, los contadores que estudiaremos en las secciones siguientes son una generalización de los contadores enteros.

Parece justificado conjeturar que si un bucle admite un contador generalizado relativo al invariante I entonces se tiene $[I \Rightarrow \mathcal{R}.C]$. En muchos casos es posible construir un contador entero a partir de uno generalizado; por ejemplo, para el programa anterior tenemos que $\boxed{t \doteq in + j}$ es un contador entero y por el teorema de los contadores el bucle termina. En general esto no es posible como veremos con algunos ejemplos en los que interviene el indeterminismo no acotado.

El Teorema central de los bucles Un programa que termina a partir de cierto estado ι se dice que termina fuertemente si existe una cota del tiempo de ejecución (función del estado inicial ι). La demostración de la terminación fuerte de un bucle suele hacerse mediante el uso de contadores enteros ya que el valor inicial de un contador da una cota del número de pasos del bucle. Por el contrario, como veremos seguidamente, existen bucles que terminan pero no fuertemente. Para ellos, en lugar de buscar contadores enteros vamos a demostrar un teorema para contadores generalizados.

TEOREMA 8.43 (Teorema Central de los bucles) *Sea $\mathcal{R} \doteq *[\![b \to S]\!]$ y sea t un $\mathcal{D}$-contador asociado al conjunto bien construido $\mathcal{C}$ y relativo al invariante I; es decir, t es una aplicación $t : \xi \to \mathcal{D}$ y $\mathcal{C} \subseteq \mathcal{D}$ verificando*

$$\begin{array}{ll} (bc) & \mathcal{C} \text{ es un subconjunto bien construido,} \\ (f) & [I \wedge b \Rightarrow t \in \mathcal{C}], \\ (id) & \forall x : x \in \mathcal{C} : [I \wedge b \wedge t = x \Rightarrow S.(I \wedge t < x)]. \end{array}$$

Entonces en el entorno de I el bucle termina verificando I; es decir, $[I \Rightarrow \mathcal{R}.I]$.

NOTA *8.44 En la literatura existen numerosas demostraciones que podemos clasificar en dos grupos, según utilicen el teorema de invariantes o no. Aquí veremos una prueba que unifica las anteriores y resulta ser una adaptación de la demostración de Dijkstra, Feijen y van Gasteren (1986).*

Demostración.– Consideremos que queremos probar el predicado más general $[I \ \Rightarrow \ Q]$, donde Q es un punto fijo $[Q \equiv g.X.Q]$ de un transformador monótono en su segundo argumento. En particular, por la semántica en términos de puntos fijos de los bucles, $\mathcal{R}.X$ es el menor punto fijo del transformador (en Y),

$$g.X.Y \ \doteq \ \neg b \wedge X \vee b \wedge S.Y \qquad (*)$$

Veamos qué condiciones debe cumplir g; tenemos

$$\begin{array}{ll} & [I \ \Rightarrow \ Q] \\ = & \quad \because \text{tercio excluido} \\ & [I \wedge t \in \mathcal{C} \ \Rightarrow \ Q] \qquad (1) \\ \wedge & [I \wedge t \notin \mathcal{C} \ \Rightarrow \ Q] \qquad (2) \end{array}$$

(2) es inmediata si tomamos $I \leq X$ para el caso particular $(*)$ con $Q \equiv \mathcal{R}.X$,

$$\begin{array}{ll} & [I \wedge t \notin \mathcal{C} \ \Rightarrow \ Q] \\ = & \quad \because Q \equiv \neg b \wedge X \vee b \wedge S.\mathcal{R}.X \\ & [I \wedge t \notin \mathcal{C} \ \Rightarrow \ \neg b \wedge X \vee b \wedge S.Q] \\ \Leftarrow & \quad \because \text{regla de intercambio dos veces, transitividad} \\ & [I \wedge (b \vee \neg X) \ \Rightarrow \ t \in \mathcal{C}] \\ = & \quad \because \text{para } I \leq X \text{ se cumple } I \wedge b \equiv I \wedge (b \vee \neg X) \\ & (f) \end{array}$$

Ahora vamos a utilizar la hipótesis '$\mathcal{C}$ es un conjunto bien construido'; recordemos que por el Teorema 1.22:22, ello equivale a decir que es inductivo, es decir, para cada predicado p, y *ptle*, se verifica:

$$\begin{array}{ll} & \forall x : x \in \mathcal{C} : p.x \\ = & \quad \because \text{Definición 1.21:22} \qquad (ci) \\ & \forall x : x \in \mathcal{C} : (\forall y : y \in \mathcal{C} \wedge y < x : p.y) \ \Rightarrow \ p.x \end{array}$$

Para utilizar (ci) en la demostración de (1) debemos introducir el cuantificador $\forall x$. Para ello utilizaremos la *regla puntual* vista como axioma del cuantificador $\forall$ en la Sección 1.4. Tomando $A == \neg I \vee Q$, por la regla de intercambio modificamos (1) en la forma $[t \in \mathcal{C} \ \Rightarrow \ A]$, que se debilita

$$\begin{array}{ll} & [t \in \mathcal{C} \ \Rightarrow \ A] \\ = & \quad \because \text{regla puntual –Definición 1.16 – } x \notin A \\ & \forall x : x = t : [x \in \mathcal{C} \ \Rightarrow \ A] \\ = & \quad \because \text{intercambio en rango} \\ & \forall x : x \in \mathcal{C} : [x = t \ \Rightarrow \ A] \\ = & \quad \because \mathcal{C} \text{ es inductivo, } (ci) \\ & \forall x : x \in \mathcal{C} : (\forall y : y \in \mathcal{C} \wedge y < x : [y = t \ \Rightarrow \ A]) \ \Rightarrow \ [x = t \ \Rightarrow \ A] \\ \Leftarrow & \quad \because \text{intercambio cuantificadores} \\ & \forall x : x \in \mathcal{C} : [\forall y : y \in \mathcal{C} \wedge y < x : y = t \ \Rightarrow \ A] \ \Rightarrow \ [x = t \ \Rightarrow \ A] \\ \Leftarrow & \quad \because \text{intercambio rango} \\ & \forall x : x \in \mathcal{C} : [\forall y : y = t : y \in \mathcal{C} \wedge y < x \ \Rightarrow \ A] \ \Rightarrow \ [x = t \ \Rightarrow \ A] \\ = & \quad \because \text{regla puntual} \\ & \forall x : x \in \mathcal{C} : [t \in \mathcal{C} \wedge t < x \ \Rightarrow \ A] \ \Rightarrow \ [x = t \ \Rightarrow \ A] \\ = & \quad \because \text{definición de } A, \text{ regla de intercambio} \\ & \forall x : x \in \mathcal{C} : [I \wedge t \in \mathcal{C} \wedge t < x \ \Rightarrow \ Q] \ \Rightarrow \ [I \wedge x = t \ \Rightarrow \ Q] \\ = & \quad \because \text{por } (2), [I \wedge t \notin \mathcal{C} \wedge t < x \ \Rightarrow \ Q], \text{cálculo} \end{array}$$

$$\forall x : x \in \mathcal{C} : [I \wedge t < x \ \Rightarrow \ Q] \ \Rightarrow \ [I \wedge x = t \ \Rightarrow \ Q] \tag{3}$$

Para probar la implicación (3), sea $x \in \mathcal{C}$; entonces,

$$\begin{array}{ll} & [I \wedge t < x \ \Rightarrow \ Q] \\ \Rightarrow & \quad \because g \text{ es monótona en el segundo argumento} \\ & [g.X.(I \wedge t < x) \ \Rightarrow \ g.X.Q] \\ \Rightarrow & \quad \because \qquad\qquad \Downarrow\text{, transitividad} \\ & [g.X.(I \wedge t < x) \ \Rightarrow \ Q] \\ \Rightarrow & \quad \because \Uparrow\text{, transitividad} \\ & [I \wedge t = x \ \Rightarrow \ Q] \end{array}$$

En el último paso necesitamos $[I \wedge t = x \ \Rightarrow \ g.X.(I \wedge t < x)]$. Obsérvese que g puede ser más general y hemos demostrado el siguiente resultado.

LEMA 8.45 *Sea g un transformador de dos argumentos, y supongamos que se verifican las siguientes condiciones, para ciertos predicados X, I y Q:*

$$\begin{array}{ll} (ac) & I \leq X \\ (bc) & \mathcal{C} \text{ es un conjunto bien construido,} \\ (f') & [I \wedge t \notin \mathcal{C} \ \Rightarrow \ g.X.Z], \\ (id') & \forall x : x \in \mathcal{C} : [I \wedge t = x \ \Rightarrow \ g.X.(I \wedge t < x)], \\ (pf) & [g.X.Q \ \Rightarrow \ Q], \\ (m) & g \text{ es monótona en el segundo argumento.} \end{array}$$

Entonces $[I \ \Rightarrow \ Q]$.

Para terminar la prueba del Teorema Central utilizando el lema anterior solamente hemos de comprobar la condición segunda para el caso particular de g dado por $(*)$:

$$\begin{array}{ll} & [I \wedge t = x \ \Rightarrow \ g.X.(I \wedge t < x)] \\ = & \quad \because \text{caso particular } (*) \\ & [I \wedge t = x \ \Rightarrow \ \neg b \wedge X \vee b \wedge S.(I \wedge t < x)] \\ = & \quad \because \text{regla de intercambio} \\ & [I \wedge (b \vee \neg X) \wedge t = x \ \Rightarrow \ S.(I \wedge t < x)] \\ = & \quad \because I \leq X \\ & [I \wedge b \wedge t = x \ \Rightarrow \ S.(I \wedge t < x)] \end{array}$$ TEOREMA

Si introducimos la funcional $wdec$ definida en Definición 6.42, así como el Teorema 5.16 fundamental de la sentencia selectiva, podemos rescribir el enunciado del Teorema Central para varias guardas en la forma siguiente:

COROLARIO 8.46 *Sea un bucle genérico $\mathcal{R} \doteq \ * [\![\ \Box \ : b_i \ \rightarrow \ S_i \ ]\!]$ y sea t una aplicación $t : \xi \rightarrow \mathcal{D}$ y $\mathcal{C} \subseteq \mathcal{D}$ verificando*

$$\begin{array}{ll} (bc) & \mathcal{C} \text{ es un subconjunto bien construido,} \\ (f) & \forall i :: [I \wedge b_i \ \Rightarrow \ t \in \mathcal{C}], \\ (id) & \forall i :: [I \wedge b_i \ \Rightarrow \ S_i.I \wedge wdec(S_i, t)]. \end{array}$$

Entonces $[I \ \Rightarrow \ \mathcal{R}.I]$.

Si alguna sentencia guardada S_i es composición de asignaciones, podemos utilizar el Lema 6.43, como mostramos en el siguiente ejemplo.

EJEMPLO 8.47 Volvamos al bucle del Ejemplo 6.47 que calcula los valores máximo y mínimo de cuatro enteros a través de un algoritmo de ordenación:

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

Vimos que $I \doteq {}'(q_1, q_2, q_3, q_4)$ es una permutación de $(Q_1, Q_2, Q_3, Q_4)'$ es invariante. Consideremos ahora el conjunto

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

que resulta ser un subconjunto bien construido (por ser finito) de $\mathbb{Z}^4$; tomaremos el orden lexicográfico. Sea además $c : \xi \to \mathbb{Z}^4$ la cuaterna definida por $c \doteq (q_1, q_2, q_3, q_4)$. Probaremos que c es un $\mathbb{Z}^4$-contador asociado a $\mathcal{C}$ y relativo a I. De entre las condiciones del Corolario 8.46, queda probar,

$$\begin{array}{ll} (f) & \forall i :: [I \wedge q_i > q_{i+1} \;\Rightarrow\; c \in \mathcal{C}] \\ (id') & \forall i :: [I \wedge q_i > q_{i+1} \;\Rightarrow\; wdec(q_i, q_{i+1} := q_{i+1} \mid c)] \end{array}$$

(f) es trivial; veamos (id') para el caso $i = 1$; $wdec(q_1, q_2 := q_2, q_1 \mid c)$

$$\begin{array}{ll} = & \because \text{definición de } c\text{, Lema 6.43}(i') \\ & (q_2, q_1, q_3, q_4) < (q_1, q_2, q_3, q_4) \\ \Leftarrow & \because \text{orden lexicográfico} \\ & q_1 > q_2 \end{array}$$

EJEMPLO

EJEMPLO 8.48 (Un ejemplo de terminación débil) Estamos ahora en condiciones de probar que existen programas que terminan pero no fuertemente; es decir, se sabe que terminan, pero no es posible acotar el número de pasos del programa. El ejemplo que sigue está extraído de [Dijkstra, 1982]:355–357, *On weak and Strong Termination*.

OBSERVACIÓN.– La prueba del Teorema 8.43 no hace uso de la continuidad. Por tanto, el teorema sigue siendo válido si incorporamos a nuestro lenguaje la sentencia no continua $Azar_y$, que asigna a la variable y un número natural arbitrario con indeterminismo no acotado — véase Ejemplo 8.6:162.

OBS

Sea el programa

$$\begin{array}{l} x, y := X, Y; \\ *[\!] \quad x > 0 \to x := x - 1; Azar_y \\ \Box \quad y > 0 \to y := y - 1 \,[\!] \end{array}$$

Consideramos $\mathcal{D} = \mathbb{N}^2 = \mathcal{C}$, que es un conjunto bien construido para el orden lexicográfico. Sea el predicado $I \doteq x, y \in \mathbb{N}$, que es un invariante. Sea además el candidato a contador $t \doteq (x, y)$. Para probar la terminación probaremos las condiciones del Corolario 8.46; la condición (f) es trivial y queda probar (id); para ello probaremos

$$\begin{array}{ll} (id_1) & I \wedge x > 0 \;\Rightarrow\; wdec(x := x - 1; Azar_y \mid t) \\ (id_2) & I \wedge y > 0 \;\Rightarrow\; wdec(y := y - 1 \mid t) \end{array}$$

La implicación (id_1) es inmediata ya que $wdec(y := y - 1 \mid t)$

$=$ $\quad \because$ definición de t, Lema 6.43(i')
$(x, y-1) < (x, y)$
$=$ $\quad \because$ orden lexicográfico
$Cierto$

Para probar la implicación (id_1) es necesario usar la semántica de la sentencia $Azar_y$. Según vimos en Ejemplo 8.6:162, sabemos que, *ptle*

$$Azar_y.Z \doteq \forall k : k \in \mathbb{N} : y := k.Z$$

Entonces, $x := x - 1; Azar_y.(t < t_0)$

$=$ $\quad \because$ semántica de $Azar_y$, definición de t
$x := x - 1.\forall k : k \geq 0 : y := k.((x, y) < t_0)$
$=$ $\quad \because$ sustitución
$\forall k : k \geq 0 : (x-1, k) < t_0$
$\Leftarrow$ $\quad \because$ orden lexicográfico
$(x, y) = t_0$

de donde obtenemos directamente $[wdec(x := x-1; Azar_y \mid t)]$. La terminación es débil ya que es imposible establecer una cota del número de pasos, debido al uso de la sentencia $Azar_y$. En efecto; para cualquier K entero existe una ejecución con más de K pasos: elegir en el primer ciclo del bucle la primera sentencia guardada que suponemos termina asignando a y el valor $K + 1$, y después, en sucesivos ciclos, elegir la segunda sentencia guardada.

Si cambiamos la sentencia $Azar_y$ por $Azar_{y+}$ cuya semántica es:

$$Azar_{y+}.Z \doteq \forall k : k \in \mathbb{N} : y := y + k.Z$$

la prueba de la terminación es muy parecida. Nótese que tal sentencia permite incrementar una variable con indeterminismo no acotado. Obsérvese además que el bucle resultante puede utilizarse para simular el siguiente juego

> Una urna contiene inicialmente bolas rojas y blancas; si el número de bolas de la urna es inferior a uno termina el juego. En otro caso se extrae una bola; si es roja añadimos a la urna un número arbitrario de bolas blancas, pero no se añade nada si la bola extraída es blanca; a continuación volvemos al primer paso.

(x captura el número de bolas rojas, e y las bolas blancas). EJEMPLO

Por consiguiente, hemos demostrado que existen programas no continuos con terminación débil. Por otro lado, dado un programa que termine pero no fuertemente, podemos derivar un cómputo de $Azar_y$ (por ejemplo, el contador del número de pasos). Por consiguiente, hemos demostrado:

TEOREMA 8.49 *Las siguientes propiedades son equivalentes*

- (i) *continuidad*,
- (ii) *indeterminismo acotado*,
- (iii) *terminación* $\Rightarrow$ *terminación fuerte*.

Ejercicios

8.50 *[307] Sea el programa*

$$x, b := 100, Cierto; *[\![b \to x, b := x - 1, x > 1 \,\square\, b \to b := Falso]\!]$$

(A) Probad, utilizando la semántica inductiva de los bucles, que siempre termina.

(B) Justificad que tal programa asigna a x un entero indeterminista, pero con indeterminismo acotado; justificad lo anterior, utilizando la propiedad de continuidad.

(C) Probad que siempre termina utilizando el teorema de los contadores y el teorema de invariantes; deducid de aquí que el indeterminismo es acotado.

8.51 **(Enero, 91)** *Para el bucle $\mathcal{R}$ del Ejemplo 8.13, razonad los siguientes afirmaciones:*

(a) $\mathcal{R}$ no termina en presencia de b porque nuestro lenguaje no permite milagros.

(b) Según la propiedad de continuidad nuestro lenguaje no permite el indeterminismo no acotado, y por tanto, $\mathcal{R}$ no termina por ser continuo.

8.52 *[309]* **(Febrero, 95)** *Escribid un programa para encontrar el segundo menor elemento del conjunto de constantes $\{A, B, C, D, E\}$.*

8.53 *[309]* **(Febrero, 92)** *Escribid un programa para calcular la mediana del conjunto de constantes $\{A, B, C, D, E\}$ (si $a, b \leq c \leq d, e$ entonces c es la mediana de $\{a, b, c, d, e\}$).*

8.54 *[311]* **(Febrero, 97)** *Sean A, B, C, D, E y F seis valores de un conjunto totalmente ordenado $\mathcal{D}$. Escribid un programa, con ayuda de un bucle, para encontrar el tercer menor elemento del conjunto (es decir, un elemento no menor que dos de ellos y no mayor que los tres restantes).*

8.55 *[311] Probad la corrección total del esquema*

$$\begin{array}{l} x, y, :\in \mathbb{Z}; \\ x, y := 1000, 2001; \\ \quad *[\![\quad y > 2 \qquad\quad \to y := y - 2 \\ \quad \square \quad x > 1 \wedge y \leq 2 \quad \to x, y := x - 2, x + 1 \,]\!] \\ \{x = 0 \wedge y = 1\} \end{array}$$

Cambiad alguna sentencia para transformarlo en un programa que termine débilmente.

8.56 *[313] Probad la corrección total del esquema*

$$\begin{array}{l} x, y, z :\in \mathbb{Z}; \\ Azar_z; \\ x, y := 2 * z, 2 * z + 1; \\ \quad *[\![\quad y > 2 \qquad\quad \to y := y - 2 \\ \quad \square \quad x > 1 \wedge y \leq 2 \quad \to x, y := y - 1, x + 1 \,]\!] \\ \{x = 0 \wedge y = 1\} \end{array}$$

Cambiad alguna sentencia para obtener un programa que termine fuertemente.

8.57 *[313] Probad la corrección total del esquema*

$$\begin{array}{l} x, y :\in \mathbb{Z}; \\ x, y := 10, 10; \\ \quad *[\![\quad x > y \quad \to \quad x, y := y, x \\ \quad \square \quad y > 1 \quad \to \quad y := y - 1 \\ \quad \square \quad x > 2 \quad \to \quad x, y := x - 2, x^y \,]\!] \\ \{x = 1 \wedge y = 1\} \end{array}$$

8.58 *[314] Probad la corrección total del esquema*

$$
\begin{array}{l}
x, y :\in \mathbb{Z}; \\
x, y := 1000, 1000; \\
*[\![\quad y > 2 \qquad\qquad \rightarrow y := y - 2 \\
\ \ \square \quad x \neq 1 \wedge y \leq 2 \quad \rightarrow x, y := x - 1, 2 * x \,]\!] \\
\{y = 0 \vee y = 2\}
\end{array}
$$

Demostrad que termina fuertemente y dad una cota del número de operaciones aritméticas que realiza.

8.59 *[314] Sea el bucle* $\mathcal{R} \doteq *\mathcal{S}$*, donde* $\mathcal{S} \doteq [\![x > 0 \rightarrow x := x - 1 \,\square\, x < 0 \rightarrow x := x + 1]\!]$.

(A) Probad que $t(x) = |x|$ *es un contador. AYUDA.- probad* $[x \neq 0 \ \Rightarrow\ wdec(\mathcal{S}, t)]$.

(B) Probad que el bucle siempre termina, y fuertemente.

(C) Probad $\forall k : k \geq 0 : [H^k.C \equiv -k \leq x \leq k]$.

(D) Concluir de (C) que $[\mathcal{R}.C \equiv C]$.

(E) Modificad el bucle anterior en la forma

$$
\begin{array}{ll}
*[\![& x > 0 \rightarrow x := x - 1; \boxed{?} \\
\square & x < 0 \rightarrow x := x + 1; \boxed{?} \\
\square & y > 0 \rightarrow y := y - 1 \,]\!]
\end{array}
$$

de tal manera que el bucle termine sólo débilmente.

8.60 *Se considera la sentencia* $x :- x + Impar$ *que incrementa la variable* x *con un número natural impar con indeterminismo no acotado.*

(A) Definid su transformador de predicados y justificad que al añadir al lenguaje de Dijkstra tal sentencia resulta un lenguaje no continuo.

(C) Utilizando la sentencia anterior y un bucle, escribid un programa para simular el siguiente juego:

> *Una urna contiene inicialmente 3 bolas rojas y 3 blancas; si el número de bolas de la urna es inferior a dos, termina el juego; si es mayor que uno, se extraen dos bolas; si son de distinto color, añadimos a la urna un número arbitrario pero impar de bolas blancas; no se añade nada si son del mismo color, repitiendo en cualquier caso estos últimos pasos (hasta conseguir que el número de bolas sea menor que dos).*

(D) Probad, utilizando el teorema de invariantes, que si el programa anterior termina (o sea el juego termina), entonces, al final del juego la urna contiene exactamente una única bola blanca.

(E) Justificad, utilizando conjuntos bien construidos, que el juego termina. Es decir, que efectivamente el programa también termina, pero solo débilmente.

8.61 *[316]* **(Febrero, 01)** *Siendo* b *una variable entera, se considera la sentencia* inc_b *con el siguiente transformador de predicados,* $ptle$*:*

$$inc_b.Z \doteq \forall i : i \geq 0 : b := b + 2i.Z$$

(A) Prueba alguno de los siguientes tripletes, indicando a la derecha su significado con una pequeña frase:

$\{C\} inc_b \{C\}$ *significa: . . .*
$\{b = k\} inc_b \{(b - k)$ *par* $\geq 0\}$ *significa: . . .*

(B) Prueba que inc_b *tiene indeterminismo no acotado, y por tanto no es continua.*

(C) Utilizando la sentencia inc_b y un bucle, escribe un programa para simular el siguiente juego. Una urna contiene inicialmente 3 bolas rojas y 3 blancas; si el número de bolas de la urna es inferior a dos, termina el juego; si es mayor que uno, se extraen dos bolas, y posteriormente se realizan las siguientes acciones, hasta conseguir que el número de bolas sea menor que dos:

a.– si son de distinto color, añadimos a la urna un número impar de bolas blancas.

b.– no se añade nada si son del mismo color.

(D) Utilizando el teorema de los contadores, prueba que el programa anterior termina sólo débilmente (o sea, que el juego termina) y al final del juego, la urna contiene exactamente una única bola blanca.

8.62 *[317]* **(Setiembre, 01)** *Utilizando un contador generalizado, probad (todas las variables son naturales)*

$$*[\![\quad x > 0 \to Azar_{012}; y := y^x \\ \quad\; y > 0 \to y := y - 1]\!] \qquad \{x = y = 0\}$$

donde $Azar_{012} \doteq [\![x > 0 \to x := 0 \,\square\, x > 1 \to x := 1 \,\square\, x > 2 \to x := 2]\!]$.

8.63 *[317] Dad un ejemplo de un programa que termine débilmente pero no fuertemente, utilizando únicamente sentencias del lenguaje de Dijkstra (asignaciones, bucles y selectivas indeterministas)*

8.64 *[317]* **(Setiembre, 95)** *Probad la corrección parcial débil del programa:*

$$\begin{array}{l} x, y, z :\in \mathbb{Z}; \\ Azar_z; \\ x, y := 2z + 1, 2|z| + 1; \\ \quad *[\![\quad x < 0 \qquad\quad \to x := -x \\ \quad\;\; \square \quad y > 2 \qquad\quad \to y := y - 2 \\ \quad\;\; \square \quad x > 1 \wedge y \leq 2 \quad \to x, y := x - 2, y + 2]\!] \\ \{x = 1 \wedge y = 1\} \end{array}$$

8.65 *[318]* **(Enero, 96)**

(A) ¿Qué se entiende por $terminación\ débil$? Poned un ejemplo.

(B) ¿Es posible escribir un programa con transformador de predicados $continuo$ que termine débilmente pero no fuertemente?

(C) Consideremos el siguiente juego:

> *Una urna contiene bolas blancas, verdes o rojas. Se extrae una bola; si es blanca se añaden bolas rojas o verdes en número arbitrario; si es roja se añaden verdes en un numero arbitrario; si es verde no se añade nada. Se repite el paso anterior hasta que la urna quede vacía.*

Partiendo de una urna inicial con 20 bolas de cada color, escribid un programa que simule el juego y probad que termina débilmente pero no fuertemente.

¿Existe alguna contradicción con el apartado (B)?

8.66 *[319]* **(Enero, 91)** *Probad o refutad las siguientes afirmaciones:*

(A) Toda sentencia sana es continua.

(B) Toda sentencia con indeterminismo no acotado es continua.

(C) Toda sentencia estricta es conjuntiva.

(D) Si S es sana y disyuntiva, entonces dado un estado inicial para el que S termina, el estado final es único.

(E) Existe una sentencia indeterminista verificando $\{C\}S\{x = 1\}$.

8.67 **(Febrero, 93)** *Sea el programa*

$$\begin{array}{l} x, y :\in \mathbb{Z}; \\ x, y := 1000, 1001; \\ \quad *[\!] \quad y > 3 \quad \rightarrow \quad y := y - 2 \\ \quad\ \square \quad x \neq 0 \wedge y \leq 3 \quad \rightarrow \quad x, y := x - 2, x + 1\,]\!] \\ \{y = 1 \vee y = 3\} \end{array}$$

(A) Probad que termina fuertemente.

(B) Si cambiamos la segunda sentencia guardada por $x := x-2; y : -Impar$*, siendo esta última una sentencia que asigna a* y *un natural impar con indeterminismo no acotado, probad que entonces solo termina débilmente.*

8.68 *[319]* **(Enero, 91)** *Escribid, con la ayuda de la sentencia indeterminista no acotada* $Azar_x$ *un programa que termine débilmente ¿Está en contradicción esto último con el hecho de que, en ausencia de continuidad*

$$\textit{terminación} \not\Rightarrow \textit{terminación fuerte}\ ?$$

8.69 *[319]* **(Febrero, 93)** *Dad ejemplos de cada uno de los casos siguientes:*

(A) Una sentencia que no sea conjuntiva.

(B) Una sentencia indeterminista verificando $\{Cierto\}S\{Cierto\}$.

(C) Una sentencia $\mathcal{A}$ *no continua verificando* $\{Cierto\}\mathcal{A}\{x$ *es par*$\}$.

8.70 **(Febrero, 98)** *Sea el programa* $\mathcal{S} \doteq x, y := 10, 10; \mathcal{R}$*, donde*

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

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

(B) Encontrad un valor de α *para que* $t \doteq \alpha x + y$ *sea un contador entero; demostrad que en ese caso* $\{C\}\mathcal{S}\{x = 1 \wedge y = 1\}$*, y además,* $\mathcal{S}$ *termina fuertemente.*

(C) Cambiad la sentencia $x, y := y, x$ *por otra de forma que el bucle resultante termine solo débilmente.*

8.71 *[320]* **(Febrero, 97)** *Definimos la relación* $S =_p S'$ *en la forma:*

$$S =_p S' \doteq \forall X :: [p \wedge S.X \equiv p \wedge S'.X]$$

que se lee: S *y* S' *tienen el mismo comportamiento en el entorno* p.

(A) Utilizando la semántica de los bucles en términos de puntos fijos, probad

$$S =_p S' \;\Rightarrow\; *[\!]\, p \rightarrow S\,]\!] = *[\!]\, p \rightarrow S'\,]\!]$$

(B) Probad $S =_p S' \wedge T =_q T' \;\Rightarrow\; [\!]\, p \rightarrow S \,\square\, q \rightarrow T\,]\!] = [\!]\, p \rightarrow S' \,\square\, q \rightarrow T'\,]\!]$.

(C) Probad que si $\forall i : 1 \leq i \leq n : S_i =_{b_i} T_i$*, entonces*

$$[\!]\, \square\ : 1 \leq i \leq n : b_i \rightarrow S_i\,]\!] = [\!]\, \square\ : 1 \leq i \leq n : b_i \rightarrow T_i\,]\!]$$

(D) Probad $S =_p S' \wedge T =_q T' \;\Rightarrow\; *[\!]\, p \rightarrow S \,\square\, q \rightarrow T\,]\!] = *[\!]\, p \rightarrow S' \,\square\, q \rightarrow T'\,]\!]$.

(E) Utilizando los apartados anteriores probad

$$x := x + 1 =_{x>0} [\!]\, x > 0 \rightarrow x := x + 1 \,\square\, x < -3 \rightarrow x := x - 1\,]\!]\,.$$

(F) Simplificad la sentencia

$$[\![x > 0 \to x := x + 1 \,\square\, x \leq 0 \to x := x - 1]\!];$$
$$[\![x > 0 \to x := x - 1 \,\square\, x < 0 \to x := x + 1]\!].$$

(G) Probad que la sentencia anterior equivale a la sentencia $nada$ calculando directamente su transformador de predicados.

(H) Probad lo anterior utilizando el cálculo de Hoare.

(I) Sea ahora el bucle $\mathcal{R} \doteq *SI$, siendo

$$SI \doteq [\![x > 0 \to x := x - 1 \,\square\, x < 0 \to x := x + 1]\!].$$

Probad: $x := x - 1 =_{x>0} SI$, así como $x := x + 1 =_{x<0} SI$, y, siendo

$$\mathcal{R}_1 \doteq *[\![x > 0 \to x := x - 1]\!] \qquad \mathcal{R}_2 \doteq *[\![x < 0 \to x := x + 1]\!]$$

demostrad $\mathcal{R} = [\![x \neq 0 \to nada \,\square\, x > 0 \to \mathcal{R}_1 \,\square\, x < 0 \to \mathcal{R}_2]\!]$ para concluir finalmente, que el bucle $\mathcal{R}$ siempre termina.

8.72 *[321]* **(Diciembre, 96)** *Sea el bucle $\mathcal{R} \doteq *[\![i < N \to i := i + 2]\!]$*

(A) ¿Cual es la ecuación en términos de puntos fijos que define a $\mathcal{R}.C$?

(B) Probad, por inducción, que para cualquier solución Y_1 de la ecuación anterior se tiene $\forall k : k \geq 0 : [N - 2k \leq i \;\Rightarrow\; Y_1]$

(C) Concluid de lo anterior que $[\mathcal{R}.C \equiv C]$.

(D) Si modificamos el bucle en la forma $\mathcal{R}' \equiv *S'$, siendo el cuerpo

$$S' = [\![i < N \to i := i + 2 \,\square\, i < N \to i := i - 1]\!]$$

probad, a partir de la ecuación que determina la semántica en términos de PF de $\mathcal{R}'.C$, que $i \geq N$ es el menor punto fijo ¿Qué interpretación tiene?

8.73 *[322]* **(Junio, 99)** *Sea $\mathcal{R}$ el bucle*

$$*[\![\quad b \to \quad x := x + 1; b := (x < 10)$$
$$\square \quad b \to \quad x := x - 1; b := (x > 0)]\!]$$

(A) Probad $[\mathcal{R}.C \equiv \neg b]$ utilizando la semántica inductiva de los bucles.

(B) ¿Qué interpretación tiene lo anterior?

(C) ¿Se obtiene el mismo resultado si reemplazamos la sentencia $x := x + 1$ por una sentencia de la forma $x := x + N$?

(D) Probad, a partir de la ecuación que determina la semántica en términos de puntos fijos de $\mathcal{R}.C$, que $\neg b$ es el menor punto fijo.

8.74 *[323]* **(Diciembre, 98)**

(A) Escribid una sentencia indeterminista $Extrae$ que verifique

$$\{C\}Extrae\{x = 1 \vee x = 3 \vee x = 5\}$$

y que solamente realice asignaciones sobre la variable x. Probad que efectivamente es indeterminista.

(B) Utilizando la sentencia anterior y un bucle, escribid un programa para simular el siguiente juego:

Una urna contiene inicialmente 3 bolas rojas y 3 blancas; si el número de bolas de la urna es inferior a dos, termina el juego; si es mayor que uno, se extraen dos bolas; si son de distinto color, añadimos a la urna un número impar de bolas blancas menor que 6; no se añade nada si son del mismo color, repitiendo en cualquier caso estos últimos pasos (hasta conseguir que el número de bolas sea menor que dos).

(C) Probad, utilizando el Teorema de Invariantes, que si el programa anterior termina (o sea el juego termina), entonces, al final del juego la urna contiene exactamente una única bola blanca.

(D) Encontrad un contador entero para el bucle y probad que el programa termina fuertemente.

(E) Utilizando conjuntos bien construidos, justificad que el juego termina.

8.75 *[323]* **(Setiembre, 99)** *Se considera el bucle*

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

(A) Escribid la ecuación que determina la semántica en términos de PF de $\mathcal{R}.C$.

(B) Vía el teorema de Tarski–Knaster, probad que el predicado $Z \doteq (x-y)\ par$ verifica $[\mathcal{R}.C \Rightarrow Z]$.

(C) Demostrad que si se cumple el triplete $\{A\}\mathcal{R}\{C\}$, entonces $[A \Rightarrow Z]$.

(D) Probad que Z es un invariante del bucle.

(E) Probad que $t \doteq |x-y|$ es un contador relativo al invariante Z.

(F) Concluid de los apartados anteriores que, $ptle$, $\mathcal{R}.C \equiv (x-y)\ par$.

(G) Interpretad lo anterior.

8.76 **(Febrero, 99)**

(A) Describid la sentencia $Nat\_impar\ y$ en términos de transformadores de predicados si su funcionamiento informal es: 'asignar a la variable y un numero natural impar con indeterminismo no acotado'.

(B) Se considera el bucle $\mathcal{R}$

$$\begin{array}{lll} *[\![& x<0 & \to \quad x := -x \\ \square & y>2 & \to \quad y := y-2 \\ \square & x>1 & \to \quad x := x-2; Nat\_impar\ y \,]\!] \end{array}$$

donde todas las variables son enteras. Encontrad un invariante I que permita probar $\{I \wedge \mathcal{R}.C\}\mathcal{R}\{x=1 \wedge y=1\}$.

(C) Si se considera $\mathbb{Z}^2$ con el orden lexicográfico ¿la función $u(x,y) \doteq (x,y)$ puede ser un contador?

(D) Describid una función de la forma $t(x,y) = \begin{cases} (x,y), & \text{si } x>0 \\ (?,y), & \text{si } x \le 0 \end{cases}$, y un subconjunto $\mathcal{C}$ (de $\mathbb{Z}^2$) bien construido, de forma que t sea un contador.

(E) Utilizando lo anterior probad el triplete $\{impar\ x\}y := 1001; \mathcal{R}\{x=1 \wedge y=1\}$, pero la terminación es débil, y por tanto el programa no es continuo.

8.77 *Sea el bucle $*[\![x > y \to x, y := x-1, y+1 \,\square\, x < y \to x, y := x+1, y-2]\!]$. Aplicando el teorema de los contadores, probad que el bucle termina siempre. AYUDA.- Encontrad un contador de la forma $t = \alpha x + \beta y$. ¿Qué se puede decir del valor final de x si al principio y=x+3p+2?*

Capítulo 9

Recursión y Procedimientos

9.0. Ecuaciones, Recursión y Puntos Fijos

Consideremos el siguiente lenguaje $\mathcal{L}$ con programas variables en sus frases

$$\begin{array}{rcll} P & ::= & x := e \mid nada \mid P; P \mid [\!] \ \square \ : 1 \leq i \leq n : b_i \rightarrow P_i \ ]\!] \\ & \mid & V & \text{– programas variables} \end{array}$$

donde cada variable V representa un programa a determinar. No aparecen bucles ni la sentencia $aborta$ (esta es $[\![F \rightarrow nada]\!]$).

Sea ϕ un programa escrito con el lenguaje anterior. Si ϕ no tiene programas variables su semántica esta bien definida. Si ϕ tiene programas variables, es necesario asignar un transformador de predicados a cada programa variable. Usualmente, ello se consigue a través de ecuaciones.

EJEMPLO 9.0 Sea el par de ecuaciones *formales*:

$$\begin{array}{rcll} T & = & [\!] & n < 6 \rightarrow n := n + 1; S; n := n + 1 \\ & & \square & n \geq 6 \rightarrow n := 1 \qquad\qquad]\!] \\ S & = & nada & \end{array}$$

S es una variable programa, cuyo significado viene dado por la segunda ecuación. Es obvio que la semántica de T se obtiene reemplazando en ϕ la letra S por el transformador $nada$; en términos de sustituciones, $[S := nada].\phi$. EJEMPLO

EJEMPLO 9.1 Sea ahora la ecuación:

$$\begin{array}{rcll} S & = & [\!] & n < 6 \rightarrow n := n + 1; S; n := n + 1 \\ & & \square & n \geq 6 \rightarrow n := 1 \ ]\!] \end{array}$$

S aparece descrita a partir de S. Es decir, existe recursión. EJEMPLO

El bucle $*[\![b \rightarrow T]\!]$ puede ser representado en nuestro lenguaje a través de la ecuación:

$$S = [\![b \rightarrow T; S \ \square \ \neg b \rightarrow nada]\!]$$

EJEMPLO 9.2 Sean las ecuaciones:

$$\begin{array}{rcll} S & = & [\!] & n < 6 \rightarrow n := n + 1; T; n := n + 1 \\ & & \square & n \geq 6 \rightarrow n := 1 \ ]\!] \\ T & = & [\!] & n < 8 \rightarrow n := n + 1; S \\ & & \square & n \geq 8 \rightarrow n := 1 \ ]\!] \end{array}$$

Existe recursión doble. Su semántica será estudiada más tarde. EJEMPLO

Sea $\mathcal{T}$ el conjunto de transformadores continuos. Cada ecuación $S = \phi$, donde en ϕ aparece S como único programa variable, determina la transformación continua $\phi : \mathcal{T} \to \mathcal{T}$ dada por $\phi.t \doteq [S := t]\phi$. El mínimo punto fijo de ϕ, si existe, lo denominaremos $\mu S : \phi$, y determina la semántica de S. Para asegurar la existencia basta aplicar el teorema de Kleene; i.e., que las frases de $\mathcal{L}$ son funciones contínuas de sus programas variables.

TEOREMA 9.3 *Sea ϕ un programa con una variable S y sea $\phi.t \doteq [S := t]\phi$. Entonces ϕ hereda las siguientes propiedades.*

(i) *t conjuntivo $\Rightarrow$ $\phi.t$ conjuntivo,*
t estricto $\Rightarrow$ $\phi.t$ estricto,
t monótono $\Rightarrow$ $\phi.t$ monótono,
t continua $\Rightarrow$ $\phi.t$ continua.

(ii) *$\phi : \mathcal{T} \to \mathcal{T}$ es monótono.*

(iii) *ϕ es continuo.*

NOTA 9.4 *Para el dominio de los predicados sobre un espacio de estados $(\mathcal{P}, \leq)$ la relación de orden es $P \leq Q \doteq [P \Rightarrow Q]$. En el conjunto $\mathcal{T}$ la relación de orden es*

$$t \leq r \doteq \forall X : X \in \mathcal{P} : t.X \leq r.X$$

No confunda el lector la tercera implicación de (i) con la propiedad (ii), que afirma: si $t \leq q$ entonces $\phi.t \leq \phi.q$.

Demostración.– Las implicaciones de (i) se demuestran por inducción estructural sobre ϕ; veamos la primera. Sea t conjuntivo. Probaremos

$$\forall \phi : \phi \in \mathcal{L} : \phi.t \text{ conjuntivo}$$

(a) Los casos base corresponden a $\phi \equiv nada$ y $\phi \equiv x := e$, que al ser constantes (no tienen programas variables) son conjuntivos; por otro lado, si ϕ es la propia variable S, entonces $\phi.t \equiv t$ y $\phi.t$ es conjuntivo si lo es t.

(b) Hay dos pasos inductivos. Sea la HI: $\forall \psi : \psi \leq \phi : \psi.t$ es conjuntivo (donde $\leq$ se lee: 'es una subfrase de '); entonces

✓ Si $\phi \equiv \psi; \xi$

$(\psi; \xi).t.(X \wedge Y)$
$=$ ∵ definición de sustitución y semántica composición
$(\psi.t).(\xi.t.(X \wedge Y))$
$=$ ∵ por HI, $\xi.t$ es conjuntivo
$(\psi.t).(\xi.t.X \wedge \xi.t.Y)$
$=$ ∵ por HI, $\psi.t$ es conjuntivo
$(\psi.t).(\xi.t.X) \wedge (\psi.t).(\xi.t.Y)$
$=$ ∵ definición
$(\psi; \xi).t.X \wedge (\psi; \xi).t.Y$

✓ Si $\phi \equiv [\![\Box b_i \to \phi_i]\!]$, definamos

$$(\Box b_i \to S_i).X \doteq (\exists i :: 1 \leq i \leq n : b_i) \wedge (\forall i : 1 \leq i \leq n : b_i \Rightarrow S_i.X)$$

$$\begin{array}{ll} & ([\![\Box b_i \to \phi_i]\!].t).(X \wedge Y) \\ = & \quad \because \text{definición de sustitución} \\ & [\![\Box b_i \to \phi_i.t]\!].(X \wedge Y) \\ = & \quad \because \text{semántica de la condicional} \\ & (\Box b_i \to \phi_i.t.(X \wedge Y)) \\ = & \quad \because \text{HI} \\ & (\Box b_i \to \phi_i.t.X \wedge \phi_i.t.Y) \\ = & \quad \because \Rightarrow \text{ es conjuntiva en el consecuente} \\ & (\Box b_i \to \phi_i.t.X) \wedge (\Box b_i \to \phi_i.t.Y) \\ = & \quad \because \text{definición} \\ & ([\![\Box b_i \to \phi_i]\!].t).X \wedge ([\![\Box b_i \to \phi_i]\!].t).Y \end{array}$$

Para demostrar (ii) procedemos también por inducción estructural sobre ϕ:

(a) Los programas $x := e$ y $nada$ son constantes (no tienen programas variables) y por tanto son monótonos: si $\phi == S$ es la variable,

$$t \leq r \Rightarrow (\phi.t \equiv) t \leq r (\equiv \phi.r)$$

(b) Supongamos $(\forall \psi : \psi \leq \phi : \psi \text{ es monótona en } S) \wedge t \leq r$ en $\mathcal{T}$; entonces,

✓ Si $\phi \equiv \psi; \xi$

$$\begin{array}{ll} & (\psi;\xi).t.X \leq (\psi;\xi).r.X \\ = & \quad \because \text{definición de sustitución y semántica composición} \\ & \psi.t.(\xi.t.X) \leq \psi.r.(\xi.r.X) \\ \Leftarrow & \quad \because \text{transitiva} \\ & \psi.t.(\xi.t.X) \leq \psi.r.(\xi.t.X) \leq \psi.r.(\xi.r.X) \\ \Leftarrow & \quad \because \psi \text{ monótona (HI) y } \phi.r \text{ monótona } ((i)) \\ & t \leq r \wedge \xi.t.X \leq \xi.r.X \\ \Leftarrow & \quad \because \xi \text{ monótona (HI)} \\ & t \leq r \end{array}$$

✓ Si $\phi \equiv [\![\Box b_i \to \phi_i]\!]$

$$\begin{array}{ll} & [\![\Box b_i \to \phi_i]\!].t.X \leq [\![\Box b_i \to \phi_i]\!].r.X \\ = & \quad \because \text{definición y semántica} \\ & (\Box b_i \to \phi_i.t.X) \leq (\Box b_i \to \phi_i.r.X) \\ \Leftarrow & \quad \because \Rightarrow \text{ es conjuntiva en el consecuente} \\ & \forall i : 1 \leq i \leq n : \phi_i.t.X \leq \phi_i.r.X \\ \Leftarrow & \quad \because \text{HI} \\ & t \leq r \end{array}$$

Probemos finalmente (iii) también por inducción estructural:

(a) $nada$ y $x := e$ son constantes (no tienen programas variables) y son continuas; si $\phi == S$ es la variable, $\phi.t(\equiv t)$ es continua en t (ya que es la identidad).

(b) Supongamos $(\forall \psi : \psi \leq \phi : \psi$ es continua en $S)$ y sea $\{t_k\}$ una cadena en $\mathcal{T}$; entonces,

✓ Si $\phi \equiv \psi; \xi$

$\lim(\psi;\xi).t_k.X$

$=$ $\quad \because$ definición de sustitución y semántica composición

$\lim \psi.t_k.(\xi.t_k.X)$

$=$ $\quad \because$ propiedad del limite doble – véase después (*)

$\psi.\lim t_k.\lim(\xi.t_k.X)$

$=$ $\quad \because \xi$ es continua

$\psi.\lim t_k.(\xi.\lim t_k.X)$

$=$ $\quad \because$ definición de sustitución y semántica composición

$\psi;\xi.\lim t_k.X$

✓ Si $\phi \equiv [\![\ \Box\, b_i \to \phi_i \ ]\!]$

$\lim[\![\ \Box\, b_i \to \phi_i \ ]\!].t_k.X$

$=$ $\quad \because$ definición y semántica

$\lim(\ \Box\, b_i \to \phi_i.t_k.X)$

$=$ $\quad \because$ propiedad de la condicional – véase después (**)

$(\ \Box\, b_i \to \lim \phi_i.t_k.X)$

$=$ $\quad \because$ cada ϕ_i es continua

$(\ \Box\, b_i \to \phi_i.\lim t_k.X)$

$=$ $\quad \because$ definición y semántica

$[\![\ \Box\, b_i \to \phi_i \ ]\!].\lim t_k.X$

Queda por demostrar $(*)$ y $(**)$; probaremos que si ψ es continua, $\{t_k\} \subseteq \mathcal{T}$ y $\{X_k\} \subseteq \mathcal{P}$, entonces:

$$\lim \psi.t_k.X_k = \psi.\lim t_k.\lim X_k$$

La desigualdad $\leq$ es trivial por monotonía; veamos la otra desigualdad solo en el caso de que cada t_k sea continuo; tenemos, $\psi.\lim t_k.\lim X_k \leq \lim \psi.t_k.X_k$

$\Leftarrow$ $\quad \because \psi$ continua

$\forall k :: \psi.t_k.\lim X_k \leq \lim \psi.t_k.X_k$

$\Leftarrow$ $\quad \because \psi.t_k$ es continua

$\forall k, n :: \psi.t_k.X_n \leq \lim \psi.t_k.X_k$

$\Leftarrow$ $\quad \because \psi.t_k.X_n \leq \psi.t_l.X_l$, donde $l = \max(k, n)$

$\forall k :: \psi.t_k.X_k \leq \lim \psi.t_k.X_k$

$\Leftarrow$ $\quad \because$ por monotonía

$Cierto$

9.5 *¿Es cierto el razonamiento anterior si los t_k no son necesariamente continuos?*

Veamos ahora $(**)$; se demuestra fácilmente (por doble implicación), si $A_k \uparrow$, $B_k \uparrow, \ldots, C_k \uparrow$ son N cadenas, entonces $(A_k \wedge B_k \wedge \ldots C_k) \uparrow$, y además:

$$\lim(A_k \wedge B_k \wedge \ \ldots \ \wedge C_k) = \lim A_k \wedge \lim B_k \wedge \ \ldots \ \wedge \lim C_k$$

Ahora basta observar que $(\ \Box\, b_i \rightarrow \lim \phi_i.t_k.X)$

$=$ $\quad \because$ definición
$(\exists i : 1 \leq i \leq n : b_i) \wedge (\forall i : 1 \leq i \leq n.(b_i \ \Rightarrow \ \lim \phi_i.t_k.X))$
$=$ $\quad \because$ por lo anterior, ya que para cada i, $(b_i \ \Rightarrow \ \lim \phi_i.t_k.X) \uparrow$
$(\exists i : 1 \leq i \leq n : b_i) \wedge \lim(\forall i : 1 \leq i \leq n.(b_i \ \Rightarrow \ \phi_i.t_k.X))$
$=$
$\lim(\ \Box\, b_i \rightarrow \phi_i.t_k.X)$ TEOREMA

El Teorema 9.3 tiene varias consecuencias; (1) el *modelado de la recursión*; y (2) dado un programa con una sentencia *distinguida* u, ésta es una función monótona de u:

$$t \leq t' \ \Rightarrow \ [u := t]\phi \leq [u := t']\phi$$

Es decir, si reemplazamos u por una sentencia u' más determinista ($u \leq u'$) entonces se obtiene una programa más determinista. De aquí tendremos:

$$\{P\}[u := t]\phi\{R\} \ \Rightarrow \ \{P\}[u := t']\phi\{R\}$$

Por ejemplo $aborta \ \leq \ nada \ \ \Rightarrow \ [\![b \rightarrow aborta]\!] \ \leq \ [\![b \rightarrow nada]\!]$. Esto tiene aplicaciones en *síntesis de programas por refinamientos*.

9.1. Entornos y Semántica de la Recursión

Sea la ecuación *formal* $S = \phi$, donde en ϕ aparece S. El mínimo punto fijo de ϕ (que existe por el teorema de Tarski–Knaster), $\mu S : \phi$, es límite de la sucesión creciente de transformadores ϕ^k dada por

$$\phi^0 \ \doteq \ aborta \qquad\qquad \phi^k \ \doteq \ \phi.\phi^{k-1}$$

NOTA 9.6 *Si no incluimos continuidad, sino solamente la monotonía (es decir, $\mathcal{T} = \{t : \mathcal{P} \rightarrow \mathcal{P} \mid t$ monótono$\}$) entonces ϕ solamente es monótono, existe el mínimo punto fijo, y existe también $\lim \phi^k$, pero solamente podemos afirmar, $\lim \phi^k \leq \phi.\lim \phi^k$.*

Al admitir las ecuaciones podemos considerar el nuevo lenguaje

$$\begin{array}{lll} P & ::= & x := e \mid nada \mid P; P \mid \mu S : P \\ & \mid & [\![\ \Box : 1 \leq i \leq n : b_i \rightarrow P_i \ ]\!] \\ & \mid & V \qquad\qquad\qquad\qquad \text{– programas variables} \end{array}$$

Veamos qué ocurre en el caso de varias definiciones mutuamente recursivas. Sea $\mathcal{D}$ un conjunto finito de definiciones formales

$$\mathcal{D} = \{S_1 = \phi_1, \ldots, S_n = \phi_n\}$$

donde en cada ϕ_i solamente aparecen las variables $S_1, \ldots, S_n$; en otro caso, para cada variable X que no aparezca se considerará la definición $X = X$;

esto, como se verá más tarde, equivaldrá a asociar a cada variable indefinida el transformador $aborta$. Ahora identificamos cada ϕ_i con una función sobre tuplas de transformadores de predicados y escribimos:

$$\phi_i(t_1, \ldots, t_n) \equiv [S_1, \ldots, S_n := t_1, \ldots, t_n]\phi_i$$

Veamos que es posible definir la semántica de cualquier sentencia en el sistema de definiciones $\mathcal{D}$; antes hay que dar la semántica de los programas variables de $\mathcal{D}$. Para ello hay que construir un punto fijo de la transformación $\phi : \mathcal{T}^n \to \mathcal{T}^n$ definida en la forma

$$\phi(t_1, \ldots, t_n) \doteq (\phi_1(t_1, \ldots, t_n), \ldots, \phi_n(t_1, \ldots, t_n))$$

donde $\mathcal{T}^n$ es el dominio producto con el orden producto:

$$(t_1, \ldots, t_n) \leq (s_1, \ldots, s_n) \doteq (\forall i : 1 \leq i \leq n : t_i \leq s_i)$$

y con elemento mínimo $(\bot, \ldots, \bot) \equiv (aborta, \ldots, aborta)$. En ese caso ϕ es continua y tendrá un mínimo punto fijo, que denotaremos con

$$\mu S_1, \ldots, S_n : \phi$$

Este punto fijo es límite ($\lim \phi^k$) de la sucesión de transformadores

$$\begin{array}{lcl} \phi^0 & \doteq & (\bot, \ldots, \bot) \\ \phi^k & \doteq & (\phi_1 \circ \phi^{k-1}, \ldots, \phi_n \circ \phi^{k-1}) \end{array}$$

donde indicamos, para simplificar $\phi^i \circ \phi \doteq \phi^i(p_1 \circ \phi, p_2 \circ \phi, \ldots, p_n \circ \phi)$ (p_i indica la proyección i–ésima). Ahora basta asignar la proyección i–ésima para obtener la semántica de cada variable S_i:

$$S_i \doteq p_i \circ \mu S_1, \ldots, S_n : \phi$$

Para denotar la dependencia del entorno podemos escribir $S_i[\mathcal{D}] \doteq p_i \circ \mu\mathcal{D}$. Si ahora P es un programa con variables $S_1, \ldots, S_n$, su significado se obtiene al cambiar los valores de las variables por los valores dados por el entorno

$$P[\mathcal{D}] \doteq P(S_1[\mathcal{D}], \ldots, S_n[\mathcal{D}])$$

Así, por un teorema análogo al Teorema 9.3, obtenemos que los transformadores $P[\mathcal{D}]$ son estrictos, conjuntivos, continuos, etc., siempre que lo sean los transformadores $S_1[\mathcal{D}], \ldots, S_n[\mathcal{D}]$; pero esto es también cierto: los programas sin variables son estrictos, conjuntivos y continuos, y al ser los transformadores de las variables también sanos, lo serán todos los programas.

EJEMPLO 9.7 Sea el sistema de definiciones:

$$\begin{array}{rlcl} \mathcal{D} = \{ & A & = & n := n + 1, \\ & B & = & x := 1, \\ & C & = & y := 0, \\ & S & = & [\![b \to A; T \,\square\, \neg b \to B]\!] \\ & T & = & S; C \\ & U & = & U \} \end{array}$$

entonces

$$\begin{array}{lcl} \phi^0 & = & (\bot, \bot, \bot, \bot, \bot, \bot) \\ \phi^1 & = & (A, B, C, [\![\, b \to \bot \ \square \neg b \to \bot \,]\!], \bot; C, \bot) \end{array}$$

A partir de ϕ^1, las tres primeras componentes no cambian –son los transformadores $n := n + 1$, $x := 1$ y $y := 0$ – y la última que corresponde a la definición $U = U$ tampoco cambia y siempre es $aborta$. Por ejemplo,

$$\begin{array}{lcl} \phi^2 & = & (A, B, C, [\![\, \neg b \to B \,]\!], \bot, \bot) \\ \phi^3 & = & (A, B, C, [\![\, b \to A; \bot \ \square \neg b \to B], [\![\, \neg b \to B; C \,]\!], \bot) \end{array}$$

COROLARIO 9.8 (El problema de Turing) *Sea T un transformador de predicados con una variable P, verificando para todo programa p*

$$[P := p]T.(u \equiv p.Cierto).$$

Entonces T no es un programa.

NOTA 9.9 *Obsérvese que si T fuera un programa implementable resolvería el famoso problema de la parada: existe un algoritmo que tiene como entrada un programa p y responde si tal programa es útil; es decir, si es posible garantizar que siempre termina.*

Demostración.– Al igual que en [Hehner, 1984]:148, razonamos por reducción al absurdo. Supongamos que T sea un programa; en particular será monótono en la variable P, conjuntivo y estricto para cada programa estricto y conjuntivo. Entonces:

$$\begin{array}{ll} & [P := nada]T.(u \equiv nada.Cierto) \\ = & \quad \because \text{semántica de } nada, \text{ CP} \\ & [P := nada]T.(u) \end{array} \tag{1}$$

$$\begin{array}{ll} & [P := aborta]T.(u \equiv aborta.Cierto) \\ = & \quad \because \text{semántica de } aborta, \text{ CP} \\ & [P := aborta]T.(\neg u) \\ \Rightarrow & \quad \because \text{monotonía, } aborta \ \leq nada \\ & [P := nada]T.(\neg u) \end{array} \tag{2}$$

y de aquí llegamos a un absurdo en la forma que sigue

$$\begin{array}{ll} & Cierto \\ = & \quad \because (1), (2) \text{ y conjuntividad de } [P := nada]T \\ & [P := nada]T.(u \wedge \neg u) \\ = & \quad \because \text{CP} \\ & [P := nada]T.Falso \\ = & \quad \because \text{ley del milagro excluido} \\ & Falso \end{array}$$

COROLARIO

9.2. Ejemplos de Procedimientos sin parámetros

EJEMPLO 9.10 (Setiembre, 99) Consideremos el procedimiento recursivo

$$m = [\![\, x > 10 \to x := x - 1; m; x := x - 2 \,\square\, x \leq 10 \to nada \,]\!]$$

Un primer estudio consiste en *trazar* las ejecuciones para valores próximos a $x = 10$; completemos pues la siguiente tabla de valores de x después de la ejecución del procedimiento m, en función de los valores iniciales

valor inicial de x	...	9	10	11	12	13	14	...
valor después de ejecutar m	...	9	10	8				...

Tracemos el procedimiento para el valor inicial $x = 11$, escribiendo en una tabla los sucesivos valores de x en una columna y la sentencia que queda pendiente de ejecutar en otra columna:

x	sentencia pendiente de ejecutar
11	m
11	$x := x - 1; m; x := x - 2$
10	$m; x := x - 2$
10	$x := x - 2$
8	$\Box$

La línea '$11 \mid m$' indica indica que para $x = 11$, queremos ejecutar m; en la siguiente línea sustituimos m por la sentencia guardada que corresponde a $x > 10$. La siguiente línea indica que hemos ejecutado $x := x - 1$ quedando pendiente de ejecutar la sentencia $m; x := x - 2$ en un entorno donde $x = 10$; pero, para $x = 10$, el procedimiento m equivale a la sentencia $nada$, por lo que obtenemos la siguiente línea donde eliminamos m sin alterar el valor de x, para pasar a la situación de la penúltima línea; la última línea dice que no queda ninguna sentencia por ejecutar (hemos acabado) con $x = 8$.

La traza anterior es un proceso *informal* pero de gran utilidad. Podemos describir el mismo cálculo con todo rigor utilizando el cálculo con transformadores de predicados. La primera observación que hacemos es que utilizando la semántica en términos de puntos fijos de la recursión, m es solución de su ecuación, lo que viene a decir que se verifica la ecuación, $\forall Z$, y *ptle*:

$$m.Z \equiv [\![\, x > 10 \to x := x - 1; m; x := x - 2 \,\Box\, x \leq 10 \to nada \,]\!] .Z$$

que conduce fácilmente a las dos siguientes ecuaciones, $\forall Z$, y *ptle*:

$$\begin{array}{lcll} x > 10 \wedge m.Z & \equiv & x > 10 \wedge x := x - 1; m; x := x - 2.Z & (*) \\ x \leq 10 \wedge m.Z & \equiv & x \leq 10 \wedge Z & (**) \end{array}$$

Estas equivalencias permiten escribir el siguiente cálculo, $\forall Z$, y *ptle*:

$$\begin{array}{ll} & x = 11 \wedge m.Z \\ = & \quad \because \text{semántica de } m \text{ en términos puntos fijos; e.d., } (*) \\ & x = 11 \wedge x := x - 1; m; x := x - 2.Z \\ = & \quad \because \text{semántica asignación} \\ & x := x - 1.(x = 10 \wedge m.x := x - 2.Z) \\ = & \quad \because \text{ecuación } (**) \\ & x := x - 1.(x = 10 \wedge x := x - 2.Z) \\ = & \quad \because \text{semántica asignación} \\ & x = 11 \wedge x := x - 1.x := x - 2.Z \\ = & \quad \because \text{lema de sustitución – Lema 4.7}(i) \\ & x = 11 \wedge x := x - 3.Z \end{array}$$

Hemos obtenido, *ptle*:

$$(x = 11 \wedge m.Z) \equiv (x = 11 \wedge x := x - 3.Z) \qquad (x11)$$

que significa: m equivale a la sentencia $x := x - 3$ en el entorno $x = 11$; en particular, $[x = 11 \;\Rightarrow\; m.x = 8]$

$=$ $\quad \because$ regla de oro
$\quad [x = 11 \wedge m.x = 8 \equiv x = 11]$
$=$ $\quad \because$ ecuación $(x11)$
$\quad [x = 11 \wedge x := x - 3.(x = 8) \equiv x = 11]$
$=$ $\quad \because$ CP
$\quad [x = 11 \wedge x = 11 \equiv x = 11]$
$=$ $\quad \because$ CP
$\quad Cierto$

y llegamos al triplete $\{x = 11\}m\{x = 8\}$ que habíamos obteniendo por el método de las trazas. Modifiquemos la tabla anotando las sentencias ejecutadas:

x	sentencia pendiente de ejecutar	sentencia ejecutada
11	m	$\square$
11	$x := x - 1; m; x := x - 2$	$\square$
10	$m; x := x - 2$	$x := x - 1$
10	$x := x - 2$	$x := x - 1; nada$
8	$\square$	$x := x - 1; nada; x := x - 2$

donde observamos que en resumen se realiza la sustitución $x := x - 3$. Tracemos ahora el programa para $x = 12$ (eliminamos la sentencia $nada$ final):

x	sentencia pendiente de ejecutar	sentencia ejecutada
12	m	$\square$
12	$x := x - 1; m; x := x - 2$	$\square$
11	$m; x := x - 2$	$x := x - 1$
8	$x := x - 2$	$x := x - 1; x := x - 3$
6	$\square$	$x := x - 1; x := x - 3; x := x - 2$

Obsérvese que la penúltima línea es posible obtenerla ya que ejecutaremos m en el entorno $x = 11$, que sabemos que termina con $x = 8$, y que además realiza la sustitución $x := x - 3$. El cálculo con transformadores para este caso sería, $\forall Z$, y *ptle*:

$\quad x = 12 \wedge m.Z$
$=$ $\quad \because (*)$
$\quad x = 12 \wedge x := x - 1; m; x := x - 2.Z$
$=$ $\quad \because$ semántica asignación
$\quad x := x - 1.(x = 11 \wedge m.x := x - 2.Z)$
$=$ $\quad \because$ ecuación $(x11)$
$\quad x := x - 1.(x = 11 \wedge x := x - 3.x := x - 2.Z)$
$=$ $\quad \because$ semántica asignación
$\quad x = 12 \wedge x := x - 1.x := x - 3.x := x - 2.Z$
$=$ $\quad \because$ lema de sustitución – Lema 4.7(i)
$\quad x = 12 \wedge x := x - 6.Z$

A la vista de lo obtenido podemos conjeturar que la tabla de valores computados por m será:

valor inicial de x	...	9	10	11	12	13	14	15	...
valor después de ejecutar m	...	9	10	8	6	4	2	0	...

Observando la tabla podemos buscar una función *casi–lineal* $f(x)$ que verifique el triplete $\{x = a\}m\{x = f(a)\}$, como la siguiente

$$f(x) = \begin{cases} x, & \text{si } x \leq 10 \\ \alpha x + \beta, & \text{si } x > 10 \end{cases}$$

Los valores de α y β pueden obtenerse sustituyendo: $\alpha 11 + \beta = 8$ y $\alpha 12 + \beta = 6$, de donde $\alpha = -2$ y $\beta = 30$. Luego:

$$f(x) = \begin{cases} x, & \text{si } x \leq 10 \\ 30 - 2x, & \text{si } x > 10 \end{cases}$$

Lo anterior es una conjetura ya que ha sido obtenido a partir de la tabla. Podemos probar con todo rigor que m satisface, $\forall Z$, y *ptle*:

$$\forall n : n \geq 11 : [\ (x = n \wedge m.Z) \equiv (x = n \wedge x := x - 3(n - 10).Z)\ ] \qquad (xn)$$

Lo anterior se prueba por inducción sobre n. El caso base corresponde a $n = 11$, y sigue de $(x11)$. El paso inductivo sería, *ptle*, y para $n > 11$,

$$\begin{array}{ll} & x = n \wedge m.Z \\ = & \quad \because (*),\ n > 11 \\ & x = n \wedge x := x - 1; m; x := x - 2.Z \\ = & \quad \because \text{semántica asignación} \\ & x := x - 1.(x = n - 1 \wedge m.x := x - 2.Z) \\ = & \quad \because \text{HI} \\ & x := x - 1.(x = n - 1 \wedge x := x - 3(n - 1 - 10).x := x - 2.Z) \\ = & \quad \because \text{semántica asignación} \\ & x = n \wedge x := x - 1.x := x - 3(n - 1 - 10).x := x - 2.Z \\ = & \quad \because \text{lema de sustitución – Lema 4.7}(i) \\ & x = n \wedge x := x - 3(n - 10).Z \end{array}$$

Ahora es fácil demostrar que se verifica:

$$\forall n : n > 10 : \{x = n\}m\{x = f(n)\}$$

En efecto, calculemos, *ptle*, $x = n \wedge m.(x = f(n))$

$$\begin{array}{ll} = & \quad \because (xn) \\ & x = n \wedge x := x - 3n + 30\,.\,(x = f(n)) \\ = & \quad \because \text{definición de } f,\ n > 10 \\ & x = n \wedge x - 3n + 30 = 30 - 2n \\ = & \quad \because \text{CP} \\ & x = n \wedge x = n \end{array}$$

De aquí, por la regla de oro, $[x = n \ \Rightarrow \ m.(x = f(n))]$, o lo que es igual, $\{x = n\}m\{x = f(n)\}$. EJEMPLO

EJEMPLO 9.11 (Enero, 95) Estudiemos ahora el procedimiento recursivo

$$m = [\![\, i > 100 \to nada \,\square\, i \leq 100 \to i := i + 1; m; m \,]\!]$$

La diferencia más importante con el procedimiento del ejemplo anterior es que aparecen dos llamadas a m en la segunda sentencia guardada; las trazas pueden realizarse de forma parecida. En primer lugar, utilizando la semántica en términos de puntos fijos encontramos las ecuaciones, $\forall Z$, y *ptle*:

$$\begin{array}{lcll} i \leq 100 \wedge m.Z & \equiv & i \leq 100 \wedge i := i + 1; m; m.Z & (*) \\ i > 100 \wedge m.Z & \equiv & i > 100 \wedge Z & (**) \end{array}$$

Comencemos viendo la traza para el primer valor no trivial, $i = 100$,

x	sentencia pendiente de ejecutar	sentencia ejecutada
100	m	$\square$
100	$i := i + 1; m; m$	$\square$
101	$m; m$	$i := i + 1$
101	m	$i := i + 1; nada$
101	$\square$	$i := i + 1; nada; nada$

donde eliminamos la sentencia $nada$ del final de la secuencia. Podemos probar la corrección de la traza anterior a través de un cálculo con transformadores,

$$\begin{array}{ll} & i = 100 \wedge m.Z \\ = & \quad \because (*) \\ & i = 100 \wedge i := i + 1.m.m.Z \\ = & \quad \because \text{semántica asignación} \\ & i := i + 1.(i = 101 \wedge m.(m.Z)) \\ = & \quad \because (**), \text{ para } Z == m.Z \\ & i := i + 1.(i = 101 \wedge m.Z) \\ = & \quad \because (**) \\ & i := i + 1.(i = 101 \wedge Z) \\ = & \quad \because \text{semántica asignación} \\ & i = 100 \wedge i := i + 1.Z \\ = & \quad \because \text{ por la regla de Leibniz: } [i = k \wedge Z(i) \equiv i = k \wedge Z(k)] \\ & i = 100 \wedge i := 101.Z \end{array}$$

de donde obtenemos, *ptle*

$$(i = 100 \wedge m.Z) \equiv (i = 100 \wedge i := 101.Z) \qquad (i100)$$

Estudiemos la traza para el valor 99:

i	sentencia pendiente de ejecutar	sentencia ejecutada
99	m	$\square$
99	$i := i + 1; m; m$	$\square$
100	$m; m$	$i := i + 1$
101	m	$i := i + 1; i := i + 1$
101	$\square$	$i := i + 1; i := i + 1$

cuya corrección sigue fácilmente:

$$\begin{array}{ll} & i = 99 \wedge m.Z \\ = & \quad \because (*) \end{array}$$

$i = 99 \wedge i := i + 1.m.m.Z$
$=$ $\quad \because$ semántica asignación
$i := i + 1.(i = 100 \wedge m.(m.Z))$
$=$ $\quad \because (i100)$, para $Z == m.Z$
$i := i + 1.(i = 100 \wedge i := i + 1.m.Z)$
$=$ $\quad \because$ semántica asignación
$i := i + 1.i := i + 1.(i = 101 \wedge m.Z)$
$=$ $\quad \because (**)$
$i := i + 1.i := i + 1.(i = 101 \wedge Z)$
$=$ $\quad \because$ semántica asignación, y lema de sustitución – Lema 4.7(i)
$i = 99 \wedge i := i + 2.Z)$

de donde concluimos, *ptle*:

$$(i = 99 \wedge m.X) \equiv (i = 99 \wedge i := i + 2.X) \qquad (i99)$$

y de aquí, $\{i = 99\}m\{i = 101\}$

$=$ $\quad \because$ regla de oro y definición de triplete
$[i = 99 \equiv (i = 99 \wedge m.i = 101)]$
$=$ $\quad \because (i99)$
$[i = 99 \equiv (i = 99 \wedge i := i + 2.(i = 101))]$
$=$ $\quad \because$ CP
$[i = 99 \equiv i = 99]$
$=$ $\quad \because$ CP
$Cierto$

Ya podemos conjeturar el comportamiento de m:

$$\forall n : n \leq 100 : [\ i = n \wedge m.Z \equiv i = n \wedge i := i + 101 - n.Z\ ] \qquad (in)$$

cuya demostración sigue fácilmente por inducción sobre n. El caso base es $n = 100$ y ya ha sido estudiado, mientras que el paso inductivo sería, para $n < 100$:

$i = n \wedge m.Z$
$=$ $\quad \because (*)$
$i = n \wedge i := i + 1.m.m.Z$
$=$ $\quad \because$ semántica asignación
$i := i + 1.(i = n + 1 \wedge m.(m.Z))$
$=$ $\quad \because$ HI
$i := i + 1.(i = n + 1 \wedge i := i + 101 - (n + 1).m.Z)$
$=$ $\quad \because$ sustitución
$i := i + 1.i := i + 100 - n.(i = 101 \wedge m.Z)$
$=$ $\quad \because (**)$
$i := i + 1.i := i + 100 - n.(i = 101 \wedge Z)$
$=$ $\quad \because$ semántica asignación, y lema de sustitución – Lema 4.7(i)
$i = n \wedge i := i + 101 - n.Z$ EJEMPLO

9.12 *[324]* **(Setiembre, 03)** *Para el procedimiento m del Ejemplo 9.11, pruebe que, $\forall k$,*

$$[i = k \wedge m.(X \wedge i > 100) \equiv i = k \wedge m.X]$$

distinguiendo para $k \leq 100$ y $k > 100$. Deduzca entonces:

$$m.(X \wedge i > 100) = m.X \qquad (mX)$$

y de aquí $m; m = m$. *Deduzca además que* m *es el bucle* $*[\![i \leq 100 \to i := i + 1]\!]$ *probando que* m *verifica la ecuación*

$$m = [\![i > 100 \to nada \,\square\, i \leq 100 \to i := i + 1; m]\!] .$$

EJEMPLO 9.13 (La forma recursiva del problema de McCarthy) En este ejemplo estudiaremos la versión recursiva del programa visto en el Ejercicio 7.36 para el cálculo de la *función 91*. Sea pues el procedimiento recursivo:

$$\begin{array}{rll} m = & [\![& i > 100 \to i := i - 10 \\ & \square & i \leq 100 \to i := i + 11; m; m]\!] \end{array}$$

Se trata de probar que tal procedimiento calcula sobre la variable i la *función 91* de McCarthy:

$$f91.z = \left\{ \begin{array}{lll} z - 10 & \text{si} & z > 100 \\ 91 & \text{si z} & z \leq 100 \end{array} \right.$$

Para ello hay que demostrar:

$$[i = z \;\Rightarrow\; m.(i = f91.z)] \qquad (*)$$

En primer lugar deducimos, según las semánticas de la condicional y la recursión (punto fijo), las ecuaciones, $\forall Z$, y *ptle*:

$$\begin{array}{lcll} x \leq 100 \wedge m.Z & \equiv & x \leq 100 \wedge i := i + 11; m; m.Z & (*) \\ x > 100 \wedge m.Z & \equiv & x > 100 \wedge i := i - 10.Z & (**) \end{array}$$

Estudiemos el comportamiento de m para el primer valor no trivial, $i = 100$,

$$\begin{array}{ll} & i = 100 \wedge m.Z \\ = & \quad \because (*) \\ & i = 100 \wedge i := i + 11.m.m.Z \\ = & \quad \because \text{semántica asignación} \\ & i := i + 11.(i = 111 \wedge m.(m.Z)) \\ = & \quad \because (**), \text{ para } Z == m.Z \\ & i := i + 11.(i = 111 \wedge i := i - 10.m.Z) \\ = & \quad \because \text{lema de sustitución} \\ & i := i + 1.(i = 101 \wedge m.Z) \\ = & \quad \because (**) \\ & i := i + 1.(i = 101 \wedge i := i - 10.Z) \\ = & \quad \because \text{lema de sustitución} \\ & i = 100 \wedge i := i - 9.Z \end{array}$$

Finalmente encontramos:

$$[i = 100 \wedge m.Z \equiv i = 100 \wedge i := i - 9.Z\,] \qquad (i100)$$

Veamos el valor anterior, $i = 99$,

$$\begin{array}{ll} & i = 99 \wedge m.Z \\ = & \quad \because (*) \\ & i = 99 \wedge i := i + 11.m.m.Z \\ = & \quad \because \text{semántica asignación} \\ & i := i + 11.(i = 110 \wedge m.(m.Z)) \end{array}$$

$=$ $\quad \because (**)$, para $Z == m.Z$
$\quad i := i + 11.(i = 110 \wedge i := i - 10.m.Z)$
$=$ $\quad \because$ lema de sustitución
$\quad i := i + 1.(i = 100 \wedge m.Z)$
$=$ $\quad \because (i100)$
$\quad i := i + 1.(i = 100 \wedge i := i - 9.Z)$
$=$ $\quad \because$ lema de sustitución
$\quad i = 99 \wedge i := i - 8.Z$

Podemos pues conjeturar el comportamiento de m:

$$\forall n : n \leq 100 : [i = n \wedge m.Z \equiv i = n \wedge i := i - n + 91.Z\,] \qquad (in)$$

que probaremos en dos fases. (Observe antes el lector que, para $i = 91$, el comportamiento de m sería igual que el de la sentencia $nada$).

— FASE I: para $90 \leq n \leq 100$, lo haremos por inducción sobre n. El caso base $n = 100$ ya ha sido probado. Veamos el paso inductivo, para $n < 100$:

$\quad i = n \wedge m.Z$
$=$ $\quad \because n < 100, (*)$
$\quad i = n \wedge i := i + 11.m.m.Z$
$=$ $\quad \because$ semántica asignación
$\quad i := i + 11.(i = n + 11 \wedge m.(m.Z))$
$=$ $\quad \because 101 \leq n + 11$, $(**)$, para $Z == m.Z$
$\quad i := i + 11.(i = n + 11 \wedge i := i - 10.m.Z)$
$=$ $\quad \because$ lema de sustitución
$\quad i := i + 1.(i = n + 1 \wedge m.Z)$
$=$ $\quad \because n + 1 \leq 100$, HI
$\quad i := i + 1.(i = n + 1 \wedge i := i - (n + 1) + 91.Z)$
$=$ $\quad \because$ lema de sustitución
$\quad i = n \wedge i := i - n + 91.Z$

— FASE II: para $n \leq 90$, lo haremos por inducción sobre n. El caso base $n = 90$ ha sido establecido en la fase anterior. Consideremos pues $n < 90$, y la siguiente prueba del paso inductivo:

$\quad i = n \wedge m.Z$
$=$ $\quad \because n < 90, (*)$
$\quad i = n \wedge i := i + 11.m.m.Z$
$=$ $\quad \because$ semántica asignación
$\quad i := i + 11.(i = n + 11 \wedge m.m.Z)$
$=$ $\quad \because$ Si $n + 11 < 90$ usamos HI; si $90 \leq n + 11 < 101$, usamos la Fase I
$\quad i := i + 11.(i = n + 11 \wedge i := i - (n + 1) + 91.m.Z)$
$=$ $\quad \because$ lema de sustitución

$$
\begin{array}{ll}
& i := i - n + 91.(i = 91 \wedge m.Z) \\
= & \quad \because \text{Fase I} \\
& i := i - n + 91.(i = 91 \wedge i := i.Z) \\
= & \quad \because \text{lema de sustitución} \\
& i = n \wedge i := i - n + 91.Z
\end{array}
$$

Finalmente calculemos, para $z \leq 100$, la expresión

$$
\begin{array}{ll}
& i = z \wedge m.(i = 91) \\
= & \quad \because (in) \\
& i = z \wedge i := i - z + 91.(i = 91) \\
= & \quad \because \text{sustitución} \\
& i = z
\end{array}
$$

de donde tendremos $\{i = z \leq 100\}m\{i = 91\}$. EJEMPLO

EJEMPLO 9.14 (Cálculo del factorial) Sea la siguiente definición para el cálculo del factorial, donde todas las variables son enteras

$$
\begin{array}{lll}
pfact = & [\![& n = 0 \rightarrow x := 1 \\
& \Box & n > 0 \rightarrow n := n - 1; pfact; n := n + 1; x := x * n \,]\!]
\end{array}
$$

Nos planteamos probar la corrección del esquema

$$\{N \geq 0\}n := N; pfact\{x = N!\} \qquad (*)$$

Una primera forma es utilizar la definición de $pfact$ como límite de los transformadores de predicados $pfact^n$ descritos en la Sección 9.1; para estos es posible probar (es engorroso pero fácil), si $N \in \mathbb{N}$,

$$
\begin{array}{lcll}
n := N; pfact^1.(x = N!) & \equiv & N = 0 & \\
n := N; pfact^2.(x = N!) & \equiv & N = 0 \vee N = 1 & \\
\ldots & & & \\
n := N; pfact^k.(x = N!) & \equiv & 0 \leq N \leq k - 1 & (1)
\end{array}
$$

de donde tendríamos

$$
\begin{array}{ll}
& n := N; pfact.(x = N!) \\
= & \quad \because n := N \text{ es continuo} \\
& \lim_k (n := N; pfact^k).(x = N!) \\
= & \quad \because \text{definición de límite en } \mathcal{T}, \text{ y } (1) \\
& \exists k : k \in \mathbb{N} : 0 \leq N \leq k \\
= & \quad \because \text{CP} \\
& N \in \mathbb{N}
\end{array}
$$

Veamos otra forma de probar la corrección del triplete $(*)$. Comencemos viendo algunos valores de los predicados $n = k \wedge pfact.Z$ para distintos valores de k:

$$
\begin{array}{ll}
& n = 0 \wedge pfact.Z \\
= & \quad \because \text{semántica puntos fijos y condicional} \qquad (0) \\
& n = 0 \wedge x := 1.Z
\end{array}
$$

$n = 1 \wedge pfact.Z$
$=$ $\quad \because$ semántica p.f. y condicional
$n = 1 \wedge n := n - 1; pfact; n := n + 1; x := x * n.Z$
$=$ $\quad \because$ semántica composición, conjuntividad
$n := n - 1.(n = 0 \wedge pfact; n := n + 1; x := x * n.Z)$
$=$ $\quad \because$ por (0)
$n := n - 1.(n = 0 \wedge x := 1.n := n + 1.x := x * n.Z)$
$=$ $\quad \because$ sustitución
$n = 1 \wedge n := n - 1.x := 1.n := n + 1.x := x * n.Z$
$=$ $\quad \because x := 1$ conmuta con $n := n - 1$ – Ejercicio 6.30 o Lema 4.7(iii)
$n = 1 \wedge x := 1.n := n - 1.n := n + 1.x := x * n.Z$
$=$ $\quad \because n := n - 1$ es inversa de $n := n + 1$ – Definición 6.24) o Lema 4.7(i)
$n = 1 \wedge x := 1.x := x * n.Z$
$=$ $\quad \because$ lema de sustitución – Lema 4.7(i)
$n = 1 \wedge x := n.Z$
$=$ $\quad \because$ semántica sustitución (*pruébese por inducción sobre* Z)
$n = 1 \wedge x := 1.Z$

Vamos a demostrar, por inducción sobre $N \in \mathbb{N}$, que

$$n = N \wedge pfact.Z \quad \equiv \quad n = N \wedge S_N.Z \tag{N}$$

donde los programas S_k se definen en forma inductiva

$$\begin{array}{lcl} S_0 & \doteq & x := 1 \\ S_N & \doteq & S_{N-1}; x := x * N \end{array}$$

Obsérvese que los programas S_k no dependen de la variable n. De la definición de S_k podemos probar, por inducción sobre N:

$$\forall N : N \in \mathbb{N} : S_N.(x = N!) \tag{1}$$

de donde obtendremos

$\{n = N\} pfact \{x = N!\}$
$=$ $\quad \because$ definición de triplete, y (N)
$\{n = N\} S_N \{x = N!\}$
$=$ $\quad \because$ definición de triplete, (1)
$Cierto$

Queda entonces probar (1) y (N). Veamos (1) por inducción:

— CASO BASE: trivial.

— PASO INDUCTIVO:

$S_{N+1}.(x = (N+1)!)$
$=$ $\quad \because$ definición de S_{N+1}
$S_N.(x := x * (N+1).(x = (N+1)!)$
$=$ $\quad \because$ semántica asignación
$S_N.(x * (N+1) = (N+1)!)$
$=$ $\quad \because N > 0$

$S_N.(x = N!)$

$=$ $\quad \because$ HI

$Cierto$

Vemos (N) por inducción, para $N \geq 1$:

— CASO BASE ($N = 1$). Estudiado anteriormente.

— PASO INDUCTIVO; supongamos $N > 1$

$n = N \wedge pfact.Z$

$=$ $\quad \because$ semántica puntos fijos, $n > 0$

$n = N \wedge n := n - 1; pfact; n := n + 1; x := x * n.Z$

$=$ $\quad \because$ conjuntividad, semántica composición

$n := n - 1.(n = N - 1 \wedge pfact.(n := n + 1; x := x * n.Z))$

$=$ $\quad \because$ HI, $N - 1 > 0$

$n := n - 1.(n = N - 1 \wedge S_{N-1}.(n := n + 1; x := x * n.Z))$

$=$ $\quad \because$ definición de sustitución

$n = N \wedge n := n - 1; S_{N-1}; n := n + 1; x := x * n.Z$

$=$ $\quad \because S_{N-1}$ no depende de n y conmuta con $n := n - 1$ – Ejercicio 6.30

$n = N \wedge S_{N-1}; n := n - 1; n := n + 1; x := x * n.Z$

$=$ $\quad \because n := n - 1$ es inversa de $n := n + 1$

$n = N \wedge S_{N-1}; x := x * N.Z$

$=$ $\quad \because$ definición de S_k

$n = N \wedge S_N.Z$ EJEMPLO

9.15 *[325]* **(Setiembre, 92)** *(pongamos $C \equiv Cierto$)*

(A) Siendo $SI == [\![C \to U \,\square\, C \to V]\!]$, probad $[SI.X \equiv U.X \wedge V.X]$

(B) Probad, utilizando (A),

$$[\![a \to U \,\square\, b \to V \,\square\, b \to W]\!].X = [\![a \to U \,\square\, b \to [\![C \to V \,\square\, C \to W]\!]]\!].X$$

(C) Sea m el procedimiento definido con la ecuación recursiva

$$m = [\![i > 10 \to nada \,\square\, i \leq 10 \to i := i + 2; m; m \,\square\, i \leq 10 \to i := i + 2; m]\!]$$

Probad $[m.X \equiv i > 10 \wedge X \vee i \leq 10 \wedge i := i + 2; m.(X \wedge m.X)]$.

(D) Probad $\{i = 10\}m\{X\} \equiv \{i = 10\}i := i + 2\{X\}$.

9.16 **(Febrero, 99)** *Sea el procedimiento*

$$\begin{array}{rcl} m & = & [\![\quad i > 6 \quad \to \quad nada \\ & & \square \quad i \leq 6 \quad \to \quad i := i + 1; m; i := i + 2]\!] \end{array}$$

(A) Siendo X un predicado arbitrario, probad

$$\begin{array}{l} [i = 6 \wedge m.X \;\equiv\; i = 6 \wedge i := i + 3.X] \\ [i = 5 \wedge m.X \;\equiv\; i = 5 \wedge i := i + 6.X] \end{array}$$

(B) Deducid la validez de los tripletes: $\{i = 6\}m\{i = 9\}, \{i = 5\}m\{i = 11\}$.

(C) ¿Es correcto el triplete $\{i = 2\}m\{i = 17\}$?

(D) ¿Qué expresión tiene $f(k)$ si debe verificar $\{i = k\}m\{i = f(k)\}$, para $k < 5$?

9.3. Procedimientos con parámetros. Llamadas por valor y por nombre

Un procedimiento *anónimo* se describe con la sintaxis

$$\{\textit{parámetros} \ : \ldots \to \textit{programa}\}$$

donde el programa puede contener sentencias variables, pero en ese caso dadas en un entorno; si el procedimiento es recursivo será un entorno adecuado donde aparezca nombrado tal procedimiento.

EJEMPLO 9.17 Comenzamos estudiando el procedimiento nombrado

$$\text{máx} = \{x, y :\in \mathbb{Z} \to [\![\, x \leq y \to m := y \,\square\, x \geq y \to m := x \,]\!]\}$$

Los procedimientos son útiles cuando aparecen en sentencias en forma de llamadas; una tal sentencia, bien nombra al procedimiento describiendo la sustitución a realizar (es una llamada nombrando al procedimiento y la asignación de parámetros, como por ejemplo $\text{máx}(3, 4)$), o bien, puede aparecer el procedimiento completo

$$\{x, y :\in \mathbb{Z} \to [x \leq y \to m := y \,\square\, x \geq y \to m := x](3, 4)$$

Demos en primer lugar la semántica en ausencia de recursión. Una *llamada* queda capturada por una sustitución de parámetros; así, la llamada $\text{máx}(3, 4)$ esencialmente se interpreta como la sustitución $x, y := 3, 4$ sobre el cuerpo del procedimiento, resultando la condicional (obsérvese que desaparece el símbolo ":" de la expresión de tipo $x, y :\in \mathbb{Z}$)

$$\{3, 4 \in \mathbb{Z} \to [\![\, 3 \leq 4 \to m := 4 \,\square\, 3 \geq 4 \to m := 3 \,]\!]\}$$

cuya semántica la definimos en la forma

$$3, 4 \in \mathbb{Z} \wedge [\![\, 3 \leq 4 \to m := 4 \,\square\, 3 \geq 4 \to m := 3 \,]\!].X$$

y cuyo resultado final es $m := 4.X$, que el lector podrá interpretar fácilmente: la llamada $\text{máx}(3, 4)$ tiene como efecto asignar 4 a la variable m. EJEMPLO

En general, la llamada $\{p : \ldots \to \phi\}(a)$ tiene por semántica

$$[p := a]tp(p) \wedge ([p := a]\phi).X \qquad (\textit{llamada por nombre})$$

Obsérvese que primero realizamos la sustitución de parámetros $[p := a]$ y después aplicamos sobre el predicado X; de aquí se concluye que la semántica que estamos utilizando es *llamada por nombre*; si lo hacemos al revés se obtiene la llamada por valor

$$[p := a]tp(p) \wedge [p := a](\phi.X) \qquad (\textit{llamada por valor})$$

En ambos casos aparece $[p := a]tp(p)$. Es decir, la semántica del paso de parámetros es *estricta* ya que si la evaluación del parámetro da error el resultado es $aborta$; si suprimimos tal predicado se obtienen semánticas *no estrictas*; por ejemplo la llamada *por necesidad* es la llamada no estricta y por valor

$$[p := a](\phi.X) \qquad (\textit{llamada por necesidad})$$

EJEMPLO 9.18 Sea el procedimiento

$$y\_a\_1 = \{x :\in \mathbb{Z} \to [\![Cierto \to y := 1 \,\square\, Falso \to y := x]\!]\}$$

y evaluemos de varias formas la llamada $y\_a\_1(1/0)$:

- *por necesidad*

  $$\begin{array}{ll} & [x := 1/0]([\![Cierto \to y := 1 \,\square\, Falso \to y := x]\!].X) \\ = & \\ & [x := 1/0](y := 1.X) \\ = & \quad \because \text{si } x \text{ no aparece en } X \\ & y := 1.X \end{array}$$

  y de aquí, $[\, y\_a\_1[1/0].(y = 1) \equiv Cierto \,]$.

- *por valor o por nombre*

  $$\begin{array}{ll} & [x := 1/0]tp(x) \wedge \ldots \\ = & \\ & 1/0 \in \mathbb{Z} \wedge \ldots \\ = & \\ & Falso \end{array}$$

de donde $y\_a\_1(1/0) = aborta$. EJEMPLO

9.19 *Probad que si la sentencia ϕ no asigna ninguna variable de la expresión a entonces:*

$$[p := a]\phi.X = [p := a](\phi.X)$$

y la semántica por valor y por nombre coinciden. Este no es el caso general, como muestra el siguiente ejemplo.

EJEMPLO 9.20 Dado el procedimiento $(x, y :\in \mathbb{Z})$

$$curioso \; = \{i :\in \mathbb{Z} \to x := i; y := i\}$$

parece obvio el predicado $curioso(a).(x = y)$. Pero esto no es cierto en todos los casos; en particular si la expresión a contiene alguna variable asignada por el cuerpo del procedimiento; como por ejemplo en

$$\begin{array}{ll} & curioso(x+1).(x = y) \\ = & \quad \because \text{semántica por nombre} \\ & [i := x+1]'i :\in \mathbb{Z}' \wedge [i := x+1](x := i; y := i).(x = y) \\ = & \\ & x+1 \in \mathbb{Z} \wedge x := x+1; y := x+1.(x = y) \\ = & \\ & x \in \mathbb{Z} \wedge x := x+1.(x = x+1) \\ = & \\ & Falso \end{array}$$

mientras que si evaluamos por valor

$$\begin{array}{ll} & curioso(x+1).(x = y) \\ = & \quad \because \text{semántica por valor} \\ & [i := x+1]'i :\in \mathbb{Z}' \wedge [i := x+1](x := i; y := \mathsf{i}.(x = y)) \\ = & \\ & x+1 \in \mathbb{Z} \wedge i := x+1.Cierto \\ = & \\ & x \in \mathbb{Z} \end{array}$$

EJEMPLO

Si p es una variable, entonces el programa $p := a; \phi$ tiene por semántica

$$
\begin{array}{ll}
= & p := a; \phi.X \\
 & p := a.(\phi.X) \\
= & \quad \because p \text{ es una variable, semántica estricta de la asignación (página 55)} \\
 & [p := a]tp(p) \wedge [p := a](\phi.X)
\end{array}
$$

Es decir, la misma semántica de la llamada por valor $\{p : \ldots \to \phi\}(a)$; por tanto, y operacionalmente, ello significa *primero evaluar el valor a, asignarlo* a p y después *ejecutar* ϕ; es decir, la información relativa al parámetro se pasa por valor ¿Cuál es en ese caso la diferencia? En el cuerpo del procedimiento

$$\{p : \ldots \to \phi\}$$

p es una variable local, mientras que en

$$p := a; \phi$$

p *debe ser* una variable global en el contexto del programa. ¿Cuál de las dos semánticas tiene más sentido? Es evidente que las dos, y de hecho cada lenguaje tiene su propio mecanismo de paso de parámetros; nosotros consideraremos siempre la semántica en el paso de parámetros por nombre, por lo que el predicado

$$\{i :\in \mathbb{Z} \to x := i; y := i\}(x+1).(x = y)$$

es falso; obsérvese que realmente el problema es que el procedimiento provoca efectos laterales sobre variables *globales* ($x, y :\in \mathbb{Z}$) y sabemos que los efectos laterales no son bien *recibidos*.

EJEMPLO 9.21 Sean las definiciones

$$
\begin{array}{l}
A, B, C, D \in \mathbb{Z}; \qquad \text{— } \textit{constantes enteras} \\
m :\in \mathbb{Z}; \\
\text{máx} = \{x, y :\in \mathbb{Z} \to [\![x \leq y \to m := y \,\square\, x \geq y \to m := x]\!]\}
\end{array}
$$

Vamos a demostrar la corrección del esquema

$$
\begin{array}{l}
m := A; \\
\text{máx}(m, B); \text{máx}(m, C); \text{máx}(m, D) \\
\{m = \text{máximo}(A, B, C, D)\}
\end{array}
$$

Para ello basta calcular (con semántica por nombre)

$$
\begin{array}{ll}
 & m := A; \text{máx}(m, B).(m = \text{máximo}(A, B)) \\
= & \quad \because \text{semántica por nombre} \\
 & \wedge \begin{array}{l} m := A.(m, B \in \mathbb{Z} \\ [\![m \leq B \to m := B \,\square\, m \geq B \to m := m]\!].(m = \text{máximo}(A, B)) \ ) \end{array} \\
= & \quad \because \text{semántica selectiva} \\
 & \wedge \begin{array}{l} m := A.(m, B \in \mathbb{Z} \wedge (m \leq B \ \Rightarrow \ m := B.m = \text{máximo}(A, B)) \\ (m \geq B \ \Rightarrow \ m := m.m = \text{máximo}(A, B)) \ ) \end{array} \\
= & \quad \because \text{semántica asignación} \\
 & \wedge \begin{array}{l} m := A.(m, B \in \mathbb{Z} \wedge (m \leq B \ \Rightarrow \ B = \text{máximo}(A, B)) \\ (m \geq B \ \Rightarrow \ m = \text{máximo}(A, B)) \ ) \end{array}
\end{array}
$$

$$
\begin{array}{ll}
= & \quad \because \text{semántica asignación} \\
 & A, B \in \mathbb{Z} \\
 & \wedge (A \leq B \;\Rightarrow\; B = \text{máximo}(A, B)) \wedge (A \geq B \;\Rightarrow\; A = \text{máximo}(A, B)) \\
= & \quad \because \text{CP} \\
 & A, B \in \mathbb{Z}
\end{array}
$$

EJEMPLO

El siguiente ejemplo modela el paso de resultados.

EJEMPLO 9.22 Considérense las definiciones

$$
\begin{array}{l}
A, B \in \mathbb{Z}; \qquad \text{— } \textit{constantes enteras} \\
z :\in \mathbb{Z}; \\
\max 2 \;= \{x, y, m :\in \mathbb{Z} \to [\![\, x \leq y \to m := y \,\square\, x \geq y \to m := x \,]\!]\,\}
\end{array}
$$

donde hemos sustituido la variable m del ejemplo anterior por un parámetro. En este contexto tiene sentido una llamada tal como $\max 2\,(A, B, z)$ y el significado será *recoger* en la variable z el valor del máximo; en efecto:

$$
\begin{array}{ll}
 & \max 2\,(A, B, z).(z = \text{máximo}(A, B)) \\
= & \quad \because \text{semántica por nombre} \\
 & [x, y, m := A, B, z](x, y, m :\in \mathbb{Z}) \wedge \\
 & [x, y, m := A, B, z][\![\, x \leq y \to m := y \\
 & \qquad\qquad \square\, x \geq y \to m := x \,]\!]\,.(z = \text{máximo}(A, B)) \\
= & \quad \because \text{sustitución, } z :\in \mathbb{Z} \\
 & A, B \in \mathbb{Z} \wedge [\![\, A \leq B \to z := B \,\square\, A \geq B \to z := A \,]\!]\,.(z = \text{máximo}(A, B)) \\
= & \quad \because \text{semántica selectiva y asignación} \\
 & A, B \in \mathbb{Z} \\
 & \wedge (A \leq B \;\Rightarrow\; B = \text{máx}(A, B)) \wedge (A \geq B \;\Rightarrow\; A = \text{máximo}(A, B)) \\
= & \quad \because \text{CP} \\
 & A, B \in \mathbb{Z}
\end{array}
$$

EJEMPLO

9.4. Semántica para llamadas recursivas

Como en el caso de procedimientos o sentencias nombradas (sin parámetros), podemos incluir un entorno o contexto donde *declarar* cada programa variable, sea ésta del procedimiento con parámetros o sin ellos; si el cuerpo ϕ de un procedimiento

$$\{p : \ldots \to \phi\}$$

no tiene programas variables, entonces está perfectamente definida la semántica (por nombre) de las llamadas. Consideremos ahora ϕ un programa con una variable S, y sea la declaración

$$S = \{p : \ldots \to \phi\}$$

donde ϕ puede contener llamadas a S con parámetros; en ese caso, para cada expresión a y para cada transformador $t \in \mathcal{T}$, tenemos un transformador

$$[S := t][p := a]\{p : \ldots \to \phi\} \;\doteq\; [p := a]tp(p) \wedge [S := t][p := a]\phi$$

y la aplicación

$$t \longmapsto [p := a]tp(p) \wedge [S := t][p := a]\phi$$

es continua y tiene un mínimo punto fijo, que es la semántica de $S(a)$.

EJEMPLO 9.23 Sea la definición del factorial

$$x :\in \mathbb{N}$$
$$xfact = \{n :\in \mathbb{N} \to [\![\, n = 0 \to x := 1 \,\square\, n > 0 \to xfact(n-1); x := x * n \,]\!]\,\}$$

vamos a demostrar, por inducción, $\forall N : N \in \mathbb{N} : [xfact(N).(x = N!)]$.

— CASO BASE: para $N = 0$ tenemos, *ptle*

$xfact(0).(x = 0!)$
$=$ $\quad \because$ semántica por nombre y punto fijo
$0 \in \mathbb{N} \wedge [\![\, 0 = 0 \to x := 1 \,\square\, 0 > 0 \to xfact(-1); x := x * 0 \,]\!]\,.(x = 0!)$
$=$ $\quad \because$ semántica condicional
$x := 1.(x = 0!)$
$=$
$Cierto$

— PASO INDUCTIVO; supongamos $[xfact(N).(x = N!)]$, entonces, *ptle*

$xfact(N+1).(x = (N+1)!)$
$=$ $\quad \because$ semántica por nombre, punto fijo y condicional
$xfact(N); x = x * (N+1).(x = (N+1)!)$
$=$ $\quad \because$ semántica composición y asignación
$xfact(N).(x * (N+1) = (N+1)!)$
$=$ $\quad \because$ definición de $p!$
$xfact(N).(x = N!)$
$=$ $\quad \because$ HI
$Cierto$ EJEMPLO

Ejercicios

9.24 *[326] Sean las definiciones*

$$u :\in \mathbb{N}$$
$$xfact = \{x, n :\in \mathbb{N} \to [\![\, n = 0 \to x := 1 \,\square\, n > 0 \to xfact(n-1, x); x := x * n \,]\!]\,\}$$

Demostrad que se cumple $\forall N : N \in \mathbb{N} : xfact(N, u).(u = N!)$.

9.25 *Sean las definiciones*

$$u :\in \mathbb{N}$$
$$eucl = \{x, y, z :\in \mathbb{N} \to [\![\, y = 0 \to z := x \,\square\, y > 0 \to eucl(y, x \bmod y, z) \,]\!]\,\}$$

Demostrad por inducción

$$\forall a, b : a, b \in \mathbb{N} : a > b \;\Rightarrow\; [\; eucl(a, b, u).(u = MCD(a, b)) \;]$$

AYUDA.- El conjunto $\mathcal{C} \equiv \mathbb{N} \times \mathbb{N}$ *es un conjunto bien construido para la relación de orden lexicográfica.*

9.26 *[326] Sea T el procedimiento definido con la ecuación recursiva*

$$T = S; [\![b \to T \,\square\, \neg b \to nada]\!]$$

donde S es una sentencia sana y continua.

(A) Dad la semántica de T en forma inductiva.

(B) Sea el bucle $\mathcal{R} \doteq *[\![b \to S]\!]$; probad $[\, T.X \;\Rightarrow\; S.\mathcal{R}.X \,]$.

(C) Probad, utilizando la semántica inductiva de $\mathcal{R}$, $[\, S.\mathcal{R}.X \;\Rightarrow\; T.X \,]$.

(D) Deducid de (B) y (C) que $T = S; \mathcal{R}$ ¿Qué interpretación tiene?

9.27 *Escribid los primeros términos de la sucesión de transformadores que aproximan al transformador de A para el sistema formal de definiciones recursivas*

$$A = [\![b \to B]\!] \qquad\qquad B = [\![b \to A]\!]$$

Probad que $\forall X :: \neg(\{b\}A\{X\})$

9.28 *[327] Siendo x una variable entera, sea T el procedimiento recursivo*

$$\begin{array}{rlll} T = & [\![& x > 0 \to x := x - 1; T \\ & \square & x \leq 0 \to T]\!] \end{array}$$

Demostrad vía puntos fijos que T: $T = aborta$.

9.29 *[328] Siendo x e y variables enteras, sea el procedimiento*

$$asig = \{ i :\in \mathbb{Z} \to [\![x \neq i \to x := y \,\square\, x = i \to y := i]\!] \}$$

(A) ¿Es cierto el triplete $\{Cierto\}asig(x+1)\{x = y\}$ según que consideremos semántica por valor o por nombre?

(B) Escribid una llamada de la forma $asig(E)$ que tenga semántica por valor y semántica por necesidad distintas.

9.30 *[328] Sea el procedimiento*

$$\begin{array}{rlll} euc = \{x, y, z :\in \mathbb{N} \to & [\![& x > y & \to \quad euc(x - y, y, z) \\ & \square & x < y & \to \quad euc(x, y - x, z) \\ & \square & x = y & \to \quad z := x \;\;]\!] \} \end{array}$$

Demostrad: $\forall a, b \in \mathbb{N} : a, b > 0 : euc(a, b, u).(u = MCD(a, b))$.
AYUDA.- Si $\mathbb{P} \equiv \mathbb{N} - \{0\}$, el conjunto $\mathcal{C} \equiv \mathbb{P} \times \mathbb{P}$ es un conjunto bien construido para la relación de orden lexicográfica.

9.31 *[329]* **(Diciembre, 92)** *Sea el procedimiento recursivo (véase también el Ejemplo 9.11 de página 195):*

$$m = [\![i > 100 \to nada \,\square\, i \leq 100 \to i := i + 1; m; m]\!]$$

Completad los comentarios de la siguiente demostración del triplete

$$\begin{array}{ll} & \{i = 100\}m\{i = 101\} \\ = & \quad \because \underline{\qquad\qquad\qquad\qquad} \\ & [i = 100 \;\Rightarrow\; m.i = 101] \\ = & \quad \because \underline{\qquad\qquad\qquad\qquad} \\ & [(i = 100 \wedge m.i = 101) \equiv i = 100] \end{array}$$

Pero, ptle

$$
\begin{array}{ll}
 & i = 100 \wedge m.i = 101 \\
= & \quad \because \_\_\_\_\_ \\
 & i = 100 \wedge i := i + 1.m.m.i = 101 \\
= & \quad \because \_\_\_\_\_ \\
 & i := i + 1.(i = 101 \wedge m.m.i = 101) \\
= & \quad \because \_\_\_\_\_ \\
 & i := i + 1.(i = 101 \wedge m.i = 101) \\
= & \quad \because \_\_\_\_\_ \\
 & i := i + 1.(i = 101 \wedge i = 101) \\
= & \quad \because \_\_\_\_\_ \\
 & i = 100
\end{array}
$$

9.32 *[329]* **(Junio, 99)** *Sea el procedimiento recursivo* $(x :\in \mathbb{N})$

$$
f = \{y, a, u :\in \mathbb{N} \rightarrow \quad [\!] \quad y > 0 \quad \rightarrow \quad f(y-1, x*a, u) \\
\qquad\qquad\qquad\qquad\quad \square \quad y \leq 0 \quad \rightarrow \quad u := a \,]\!] \, \}
$$

(A) Probad $\forall a, x, y : a, x, y \in \mathbb{N} : [f(y, a, u).(u = ax^y)]$.

(B) Probad el triplete $\{Cierto\} a := 1; f(p, a, u) \{u = x^p\}$*, siendo* $p, a, u :\in \mathbb{N}$.

9.33 **(Junio, 00)** *Sea el procedimiento (*$z :\in \mathbb{Z}$ *es una declaración global):*

$$
mul = \{x, y :\in \mathbb{Z} \rightarrow [\!] \quad x = 0 \rightarrow z := 0 \\
\qquad\qquad\qquad\qquad\quad \square \quad x \neq 0 \rightarrow mul(x-1, y); z := z + y \,]\!]
$$

Probad $\quad \forall a, b : a \in \mathbb{N} \wedge b \in \mathbb{Z} : [\, mul(a, b).(z = ab) \,]$.

9.34 *[329]* **(Setiembre, 01)** *Probad el triplete* $\{i = 100\} m \{i = -101\}$*, siendo*

$$
m = \quad [\!] \quad i < 0 \rightarrow i := i + 1 \\
\qquad\quad \square \quad i \geq 0 \rightarrow i := i - 1; m; i := i - 1 \,]\!]
$$

9.35 *[330] Sea* $Azar_{x012} \doteq [\!] \, x > 0 \rightarrow x := 0 \, \square \, x > 1 \rightarrow x := 1 \, \square \, x > 2 \rightarrow x := 2 \,]\!]$.

(A) Estudiad los tripletes:

$$
\{x = a > 0\} Azar_{x012} \{0 \leq x < a\} \qquad\qquad \{x > 1\} Azar_{x012} \{x = q\}
$$

(B) Estudiad el bucle $*[\!] \, x > 0 \rightarrow Azar_{x012} \,]\!]$

(C) Estudiad el procedimiento recursivo definido como:

$$
m = \quad [\!] \quad x \leq 3 \rightarrow Azar_{x012} \\
\qquad\quad \square \quad x > 3 \rightarrow x := x - 2; m; x := x + 2 \,]\!]
$$

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.