Programación Concurrente
Práctico 3: Verificación (a posteriori) de Programas Concurrentes

J. Blanco, N. Wolovick

31 de octubre de 2003

1. Demuestre que el siguiente multiprograma anotado es correcto.

Pre: $x = 0$	
$A : \{x = 0 \vee x = 2\}$ $a := x$ $\{(a = 0 \vee a = 2) \wedge a \leq x\}$ $x := a + 1$ $\{x = 1 \vee x = 3\}$	B: $\{x = 0 \vee x = 1\}$ $x := x + 2$ $\{x = 1 \vee x = 2 \vee x = 3\}$
Post: $x = 1 \vee x = 3$	

2. Dado el siguiente multiprograma, muestre cuales son los posibles valores finales para x

$$\{u = 0 \wedge v = 1 \wedge w = 2\}\ x := u + v + w \parallel u := 3 \parallel v := 4 \parallel w := 5\ \{x \in S\}$$

   a) si todas las asignaciones son atómicas,

   b) si no lo son y el *rhs* se evalúa de izquierda a derecha.

3. Elija las aserciones de forma tal que el multiprograma anotado resulte correcto

Pre: $x = 0$	
A: $\{P_0\}$ $a := x$ $\{P_1\}$ $x := a + 1$ $\{P_2\}$	B: $\{Q_0\}$ $b := x$ $\{Q_1\}$ $x := b + 2$ $\{Q_2\}$
Post: $x = 1 \vee x = 2 \vee x = 3$	

4. Demuestre el siguiente multiprograma[1].

$$\{x = 0 \wedge \neg done\}\ x, done := x + 1, true \parallel x := x + 1\ \{x = 2 \wedge done\}$$

5. Los siguientes multiprogramas anotados son ejemplos que muestran la incompletitud del sistema de prueba de OG sin la Regla de las Variables Auxiliares.

[1] Notar como las variables auxiliares se utilizan para marcar el flujo de control del programa.

Demuestre, estos multiprogramas[2] [3].

$$\{x = y\}\ x := x + 1 \parallel y := y + 1\ \{x = y\}$$

$$\{true\}\ x := x + 2 \parallel x := 0\ \{x = 0 \vee x = 2\}$$

$$\{x = 0\}\ x := x + 1 \parallel x := x + 1\ \{x = 2\}$$

6. Demuestre la corrección del siguiente multiprograma:

Pre: $\neg x$	
A: $x := true$	B: **if** $x \rightarrow\ \ x := false$ **fi**
Post: $\neg x$	

También verifique que si la componente A la cambiamos por el par de instrucciones no-atómicas $A : x := true; x := true$ la postcondición $\neg x$ ya no se puede garantizar.

7. Decore el siguiente multiprograma con aserciones local y globalmente correctas, de forma tal que verifiquen la pre/post.

Pre: $\neg b$	
A: $\{P_0\}$ **do** $\neg b \rightarrow$ **skip** $\{P_1\}$ **od** $\{P_2\}$ $x := y$ $\{P_3\}$	B: $\{Q_0\}$ $y := E$ $\{Q_1\}$ $b := true$ $\{Q_2\}$
Post: $x = y$	

8. Demostrar el siguiente algoritmo paralelo *coarse-grained* que encuentra un cero de la función $f : \mathcal{Z} \rightarrow \mathcal{N}$.

Pre: $\langle \exists u : u \in \mathcal{Z} : f.u = 0 \rangle \wedge \neg found \wedge x = 0 \wedge y = 1$	
A: $\{\ Inv : p_0\}$ **do** $\neg found \rightarrow$ $\{q_0\}$ $\langle\ x := x + 1;$ **if** $f.x = 0 \rightarrow found := true$ $\Box f.x \neq 0 \rightarrow$ **skip** **fi** $\rangle$ **od** $\{p_0 \wedge found\}$	B: $\{\ Inv : p_1\}$ **do** $\neg found \rightarrow$ $\{q_1\}$ $\langle\ y := y - 1;$ **if** $f.y = 0 \rightarrow found := true$ $\Box f.y \neq 0 \rightarrow$ **skip** **fi** $\rangle$ **od** $\{p_1 \wedge found\}$
Post: $f.x = 0 \vee f.y = 0$	

Donde

$$p_0 \equiv 0 \leq x \wedge (found \Rightarrow (0 < x \wedge f.x = 0) \vee (y \leq 0 \wedge f.y = 0)) \wedge (\neg found \wedge 0 < x \Rightarrow f.x \neq 0)$$

$$p_1 \equiv y \leq 1 \wedge (found \Rightarrow (0 < x \wedge f.x = 0) \vee (y \leq 0 \wedge f.y = 0)) \wedge (\neg found \wedge y \leq 0 \Rightarrow f.y \neq 0)$$

Encontrar las aserciones q_0, q_1 y demostrar correcto el multiprograma.

[2] Notar el primer ejemplo es un programas parelelo con variables disjuntas y sin sincronización, luego la incompletitud se da con el operador $\parallel$.

[3] El tercer multiprograma ya fue demostrado en el punto 4, intente obtener una demostración alternativa, marcando completamente el flujo de programa.

9. Al siguiente algoritmo se lo conoce como *Concurrent Vector Writing*. Demuéstrelo utilizando la teoría[4].

<table>
<tr><td colspan="2">Pre: $i = 0 \wedge j = 0$</td></tr>
<tr><td>A: $\textbf{do}\ i \neq N \rightarrow$
$\quad\quad \textbf{if}\ i < j \rightarrow \textbf{skip fi};$
$\quad\quad a.i := 0;$
$\quad\quad i := i + 1$
$\quad \textbf{od}$</td><td>B: $\textbf{do}\ j \neq N \rightarrow$
$\quad\quad a.j := 1;$
$\quad\quad j := j + 1$
$\quad \textbf{od}$</td></tr>
<tr><td colspan="2">Post: $\langle \forall k : 0 \leq k < N : a.k = 0 \rangle$</td></tr>
</table>

[4]Es un ejercicio interesante comparar la verificacion *a posteriori* que se realizará, respecto a la construcción del programa como se verá en clase.