Programación Concurrente en Java
Práctico 2: Verificación de Programas Concurrentes

J. Blanco, N. Wolovick

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*[1] se evalúa de izquierda a derecha.

3. El siguiente multiprograma es una versión donde sólo se marca el flujo de control de la primera componente. Demuestre la corrección de la postcondición **sin agregar ninguna otra variable auxiliar**. Luego estamos mostrando que con la mínima unidad de información que rompa la simetría del multiprograma, alcanza para probarlo en la lógica de Owicki-Gries.

Pre: $x = 0 \wedge \neg done$	
A : $x, done := x + 1, true$	B : $x := x + 1$
Post: $x = 2 \wedge done$	

   Ayuda: Una posible preaserción de $x := x + 1$ es $\{x \geq N(done)\}$ y una postaserción de esa misma sentencia es $\{x > N(done)\}$, donde $N(true) = 1$ y $N(false) = 0$.

4. Para el siguiente multiprograma,

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

   a) Demuestre la corrección del **agregando variables auxiliares** para marcar el flujo de control.

   b) 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.

[1] right-hand side, la expresión del lado derecho de la asignación.

5. El siguiente multiprograma es un ejemplo de que podemos demostrar programas que usan *message passing* dentro de la lógica de Owicki-Gries.

Pre: $x = 0 \wedge \neg done$	
$Cli : \{\neg done\}$ $x := 21$ $\{\quad\}$ $done := true$ $\{\quad\}$	$Srv : \{done \Rightarrow x = 21\}$ **if** $done \rightarrow$ **skip fi** $\{x = 21\}\{done\}$ $x := x * 2$ $\{\quad\}$
Post: $x = 42$	

6. (*) Al siguiente algoritmo se lo conoce como *Concurrent Vector Writing*. Demuéstrelo utilizando la teoría[2].

Pre: $i = 0 \wedge j = 0$	
A: **do** $i \neq N \rightarrow$ **if** $i < j \rightarrow$ **skip fi**; $a.i := 0$; $i := i + 1$ **od**	B: **do** $j \neq N \rightarrow$ $a.j := 1$; $j := j + 1$ **od**
Post: $\langle \forall k : 0 \leq k < N : a.k = 0 \rangle$	

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