Programación Concurrente
Examen para la casa

J. Blanco, N. Wolovick

4 de diciembre de 2001

1 Ejercicios

1. [5, ej.24]

   (a) Para el siguiente multiprograma de N componentes, con $2 \leq N$, demuestre que el número de componentes ejecutando los fragmentos S es a lo más $N-1$.

Pre: $true$
Comp.p: $* [\ v := p$ $;\ \textbf{if}\ v \neq p \rightarrow \textbf{skip fi}$ $;\ S.p$ $]$

   (b) Demuestre que la misma propiedad vale en el siguiente multiprograma:

Pre: $\langle \forall q :: \neg x.q \rangle$
Comp.p: $* [\ x.p := true$ $;\ \textbf{if}\ \langle \exists q : q \neq p : \neg x.q \rangle \rightarrow \textbf{skip fi}$ $;\ S.p$ $;\ x.p := false$ $]$

   (c) Explique si la situación se modifica cuando cambiamos las componentes a[1]:

Pre: $\langle \forall q :: \neg x.q \rangle$
Comp.p: $* [\ x.p := true$ $;\ v := p$ $;\ \textbf{if}\ v \neq p \vee \langle \exists q : q \neq p : \neg x.q \rangle \rightarrow \textbf{skip fi}$ $;\ S.p$ $;\ x.p := false$ $]$

[1] la atomicidad es línea a línea.

2. El siguiente multiprograma aparece como la versión 0.0 del problema de la Sincronización de Fase para 2 máquinas en [5, cap.17].

Pre: $d = 1$	
A: $*\ [\ \{d \leq 1\}$ S $;\ d := d + 1$ $;$ **if** $d \leq 1 \rightarrow$ **skip fi** $\{d \leq 1\}$ $]$	B: $*\ [\ \{1 \leq d\}$ T $;\ d := d - 1$ $;$ **if** $1 \leq d \rightarrow$ **skip fi** $\{1 \leq d\}$ $]$
Inv: $0 \leq d \wedge d \leq 2$	

Luego el programa se refina mediante una transformación al dominio booleano con los invariantes $x \equiv d \leq 1 \wedge y \equiv 1 \leq d$, resultando en la versión 0.1.

Pre: $x \wedge y$	
A: $*\ [\ \{x\}$ S $;\ x, y := \neg y, true$ $;$ **if** $x \rightarrow$ **skip fi** $\{x\}$ $]$	B: $*\ [\ \{y\}$ T $;\ y, x := \neg x, true$ $;$ **if** $y \rightarrow$ **skip fi** $\{y\}$ $]$
Inv: $0 \leq d \wedge d \leq 2$	

Complete todos los pasos intermedios para llegar a la versión 0.2 que se reproduce a continuación y demuestre que no sufre deadlock total.

Pre: $f \wedge g$	
A: $*$ [S $;\ f := false$ $;\ g := true$ $;$ **if** $f \rightarrow$ **skip fi** $]$	B: $*$ [T $;\ g := false$ $;\ f := true$ $;$ **if** $g \rightarrow$ **skip fi** $]$

3. [5, ej.32] Elimine las multiasignaciones utilizando Fortalecimiento de Guarda.

Pre: $x \wedge \neg y \wedge \neg z$		
A: $*[$**if** $x \rightarrow$ **skip fi**; $x, y := false, true$ $]$	B: $*[$**if** $y \rightarrow$ **skip fi**; $y, z := false, true$ $]$	C: $*[$**if** $z \rightarrow$ **skip fi**; $z, x := false, true$ $]$

4. [6, ej.18][7][4]Dada un colección de procesos dividida en los grupos rojos y azules, se quiere que exista exclusión mutua entre las regiones críticas de los procesos de diferente color. Procesos del mismo color pueden estar simultáneamente en sus regiones críticas.

   (a) Obtenga una especificación formal de este requerimiento

   (b) Obtenga programas que satisfagan el punto anterior utilizando la técnica del semáforo binario dividido (*SBS*).

   (c) Simplifique lo más posible los multiprogramas resultantes.

5. [8, ej.6.18] Dadas tres secuencias f, g, h no-decrecientes

$$f[0] \leq f[1] \leq \cdots \wedge g[0] \leq g[1] \leq \cdots \wedge h[0] \leq h[1] \leq \cdots$$

   derive un multiprograma de tres componentes que tenga como postcondición que las variables i, j, k son los menores enteros tal que $f[i] = g[j] = h[k]$. Cada una de las variables i, j, k sólo deberá ser modificada por uno de los procesos.

6. [8, ej.9.36][2, ej.3.2,3.3,3.4] Usando alguna de las siguientes instrucciones atómicas fetch-and-Φ[2]:

   - $\mathit{TestSet}(v, p) : v, p := \mathit{true}, v$
   - $\mathit{FetchAdd}(w, p, e) : p, w := w, w + e$
   - $\mathit{TSL}(v, p) : v, p := \mathit{false}, v$
   - $\mathit{Swap}(v, w) : v, w := w, v$
   - $\mathit{CompareSwap}(t, x, o, n) : \langle t := (o = x); \textbf{if } t \rightarrow x := n \;\Box\; \neg t \rightarrow o := x \textbf{ fi}\rangle$
   - $\mathit{FetchOr}(w, p, e) : p, w := w, w \vee e$
   - $\mathit{Decr}(v, s) : v, s := v - 1, (v - 1 = 0)$

   (a) Cree protocolos de entrada y salida para exclusión mutua de regiones críticas.

   (b) Demuestre que la solución dada en el punto anterior satisface la propiedad de seguridad de exclusión mututa para $N = 2$.

   (c) Generalize a cualquier N.

1.1 Ejercicio para PostGrado

1. [8, ej.11.34] Suponga el valor v_i almacenado en el sitio i de una red de computadoras formada por los sitios $\{0, 1, \ldots, N-1\}$. La topología de la red es la de una anillo, por lo que el sitio i sólo se puede comunicar con el sitio $(i+1)$ *mod* N. El canal C_i es el circuito virtual confiable que enlaza los sitios i y $(i+1)$ *mod* N. Dado un operador asociativo y conmutativo Ψ con identidad 0, se quiere derivar (a partir de una versión que utiliza memoria compartida) un programa distribuído que asigne un proceso por sitio. El proceso en el sitio 0 deberá hacer cierto:

$$Q : s_0 = \Psi_{0 \leq k < N} v_k$$

   (a) Derive un programa que utilize pasaje de mensajes asíncrono y haga válido Q.

   (b) Idem al punto anterior, sólo que utilizando pasaje de mensajes síncrono.

[2] también llamadas *interlock instructions*.

Referencias

[1] Gregory R. Andrews, *Concurrent Programming: principles and practice*, Benjamin Cummings, 1991

[2] Gregory Andrews, *Foundations of Multithreaded, Parallel, and Distributed Programming*, Addison Wesley, 2000

[3] K. Mani Chandy and Jayadev Misra, *Parallel Program Design, a Foundation*, Addison-Wesley, 1988

[4] Edsger W. Dijkstra, *A Tutorial on the Split Binary Semaphore*, EWD703, 1979

[5] W.H.J. Feijen and A.J.M. van Gasteren, *On a Method of MultiProgramming*, Monographs in Computer Science, Springer, 1999

[6] Rob Hoogerwoord, *Semaphores*, RH86

[7] Rob Hoogerwoord, *Two Applications of the Split Binary Semaphore*, RH221

[8] F. Schneider, *On Concurrent Programming*, Graduate Texts in Computer Science, Springer, 1997