Programación Concurrente

Javier Blanco   Nicolás Wolovick
Fa.M.A.F., Universidad Nacional de Córdoba.

28 de abril de 2003

Capítulo 1

Semáforos

1.1. Exclusión mutua y Semáforos

Los algroitmos para asegurar exclusión mutua para más de dos componentes usando solamente sentencias que usan a lo sumo una variable compartida a lo sumo una vez son relativamente complejos. Esta es una de las razones para introducir ciertas primitivas de sicronización con un grado de atomicidad más grueso cuya atomicidad sea asegurada por la implementación de éstas en un sistema de cómputo dado. En este capítulo se introducirá una de las formas de sincronización que se ha establecido como universal: los *semáforos*.

Consideremos brevemente el problema de exclusión mutua para N procesos. Dados N componentes los cuales ejecutan repetidamente $*[ncs.i; cs.i]$ se requiere sincronizarlos de manera tal que en cualquier punto del multiprograma haya a lo sumo uno de ellos ejecutando su correspondiente sección crítica $cs.i$. Para especificar formalmente este problema usamos una variable auxiliar r la cual cuenta cuantos procesos se encuentran ejecutando su sección crítica. El requisito de exclusión mutua se formaliza usando un invariante global

$$
\begin{array}{|ll|}
\hline
\text{Pre: } r = 0 & \\
\hline
\text{Comp.}i\text{: } *[& ncs.i \\
\quad ; & r := r + 1 \\
\quad ; & cs.i \\
\quad ; & r := r - 1 \\
 &] \\
\hline
\text{Inv: } ?r \leq 1 & \\
\hline
\end{array}
$$

La sentencia $r := r - 1$ no causa problemas (widening), pero la sentencia $r := r + 1$ necesita como precondición $r \leq 0$. Una manera obvia de asegurarlo (a cambio de introducir sentencias con un grado grueso de atomicidad) es realizar la asignación dentro de una sentencia condicional:

Pre: $r = 0$
Comp.i: $*[\ ncs.i$ $;\ \textbf{if}\ r \leq 0 \rightarrow r := r + 1\ \textbf{fi}$ $;\ cs.i$ $;\ r := r - 1$ $]$
Inv: $r \leq 1$

Puede verse que no hay riesgo de deadlock total, aunque sí de starvation. Dada la simplicidad de la solución, varios sistemas proveen sentencias atómicas que permiten implementar este tipo de solución. Usualmente, hace falta realizar un cambio de coordenadas para pasar al dominio de los naturales, cambiando la variable r por s a través del siguiente invariante: $s = 1 - r$, obteniéndose la siguiente solución:

Pre: $s = 1$
Comp.i: $*[\ ncs.i$ $;\ \textbf{if}\ 1 \leq s \rightarrow s := s - 1\ \textbf{fi}$ $;\ cs.i$ $;\ s := s + 1$ $]$
Inv: $0 \leq s$

Las sentencias atómicas que manipulan s reciben los nombres propios

$P.s\ :\ \textbf{if}\ 1 \leq s \rightarrow s := s - 1\ \textbf{fi}$
$V.s\ :\ s := s + 1$

Por lo cual, la solución del problema de exclusión mutua para N procesos puede expresarse como sigue:

Pre: $s = 1$
Comp.i: $*[\ ncs.i$ $;\ P.s$ $;\ cs.i$ $;\ V.s$ $]$
Inv: $0 \leq s$

1.2. Productor y consumidor

En algunos casos pueden implementarse acciones atómicas de granularidad gruesa de manera eficiente usando semáforos, en general a partir de algún cambio de coordenadas. Presentaremos un ejemplo clásico y luego haremos algunos comentarios acerca del método.

Ejemplo 1.2.1

Uno de los problemas que más se ha estudiado en programación concurrente es el del *productor y consumidor*. Este problema consta en su versión más simple con dos componentes, una de las cuales - el productor - produce de alguna manera datos y los envía a la otra componente - el consumidor - a través de un buffer acotado. Dado el tamaño del buffer, el problema es sincronizar los procesos de manera que se asegure la consistencia de los datos transmitidos. Para ello es necesario evitar que se lean datos de un buffer vacío o que se escriban datos sobre un buffer lleno. Podemos plantear la siguiente especificación del problema:

Pre: $p = 0 \wedge c = 0$	
Prod: $*$[producir.n ; $\{?\ p - c < N\}$; $buf.(p\ mod\ N) := n$; $p := p + 1$]	Cons: $*$[$\{?\ 0 < p - c\}$; $m := buf.(c\ mod\ N)$; $c := c + 1$; consumir.m]

Puede encontrarse directamente una solución a este problema precediendo cada una de las aserciones en consideración por una sincronización condicional de la siguiente manera:

Pre: $p = 0 \wedge c = 0$	
Prod: $*$[producir.n ; **if** $p - c < N \rightarrow$ **skip fi** ; $\{p - c < N\}$ $buf.(p\ mod\ N) := n$; $p := p + 1$]	Cons: $*$[**if** $0 < p - c \rightarrow$ **skip fi** ; $\{0 < p - c\}$ $m := buf.(c\ mod\ N)$; $c := c + 1$; consumir.m]

La corrección local queda inmediatamente garantizada, mientras que la global de cada una de las aserciones se obtiene por widening.

La idea entonces es "implementar" el programa usando semáforos, para lo cual es necesario transformar las sentencias condicionales en operaciones P sobre semáforos manteniendo la corrección de la anotación. Comencemos con el productor, introduciendo un semáforo s. Se postula un predicado Q que pueda ser una precondición correcta para la operación $P.s$. Esto ocurre, dada la semántica de P, si $Q \wedge 0 < s \Rightarrow p - c < N$.

Calculamos

$$
\begin{array}{ll}
& Q \wedge 0 < s \Rightarrow p - c < N \\
\equiv & \{ \text{ cálculo proposicional } \} \\
& Q \Rightarrow (0 < s \Rightarrow p - c < N) \\
\equiv & \{ \text{ álgebra } \} \\
& Q \Rightarrow (0 < s \Rightarrow 0 < N - p + c) \\
\Leftarrow & \{ \text{ transitividad } \} \\
& Q \Rightarrow (s \leq N - p + c)
\end{array}
$$

Luego, el Q más débil que puede tomarse es justamente $s \leq N - p + c$, por lo cual obtenemos

Pre: $p = 0 \wedge c = 0$	
Prod: $*[$ producir.n ; $\{?\ s \leq N - p + c\}$ **if** $p - c < N \rightarrow$ **skip fi** ; $\{p - c < N\}$ $buf.(p\ mod\ N) := n$; $p := p + 1$]	Cons: $*[$ **if** $0 < p - c \rightarrow$ **skip fi** ; $\{0 < p - c\}$ $m := buf.(c\ mod\ N)$; $c := c + 1$; consumir.m]

Ahora puede reemplazarse el **if** por $P.s$, aplicando fortalecimiento de la guarda, dado que s es una variable nueva y por lo tanto la asignación a ella no invalida la anotación.

Faltaría ver que la aserción es correcta. Globalmente no hay problema (widening) mientras que para asegurar su corrección local propondremos que sea invariante del ciclo. Para que valga inicialmente se propone como precondición que $s = N$ y la invariancia surge de la corrección (obvia) de la siguiente anotación:

Pre: $p = 0 \wedge c = 0 \wedge s = N$	
Prod: $*[$ producir.n ; $\{s \leq N - p + c\}$ **if** $0 < s \rightarrow s := s - 1$ **fi** ; $\{s \leq N - p + c - 1\}$ $buf.(p\ mod\ N) := n$; $p := p + 1$ $\{s \leq N - p + c\}$]	Cons: $*[$ **if** $0 < p - c \rightarrow$ **skip fi** ; $\{0 < p - c\}$ $m := buf.(c\ mod\ N)$; $c := c + 1$; consumir.m]

Si bien el progrma es parcialmente correcto, adolece del peligro de deadlock, dado que el semáforo nunca es incrementado. Para satisfacer la regla básica de progreso (dada una sentencia condicional en una componente, el resto del sistema debe tener el potencial de volver verdadera su guarda), es necesario que el consumidor pueda incrementar el semáforo. Este incremento puede comprometer la corrección global de las aserciones nuevas en el productor. El único punto del programa donde puede incrementarse el semáforo manteniendo la corrección de las aserciones es simultáneamente con $c := c + 1$. El programa queda entonces como sigue:

Pre: $p = 0 \wedge c = 0 \wedge s = N$	
Prod: $*[$ producir.n ; $\{s \leq N - p + c\}$ **if** $0 < s \rightarrow s := s - 1$ **fi** ; $\{s \leq N - p + c - 1\}$ $buf.(p\ mod\ N) := n$; $p := p + 1$ $\{s \leq N - p + c\}$]	Cons: $*[$ **if** $0 < p - c \rightarrow$ **skip fi** ; $\{0 < p - c\}$ $m := buf.(c\ mod\ N)$; $c, s := c + 1, s + 1$; consumir.m]

De manera análoga, puede transformarse el consumidor. Presentamos el programa con todas sus anotaciones, dejando al lector el ejercicio de reconstruir todos los pasos de demostración.

Pre: $p = 0 \wedge c = 0 \wedge s = N \wedge t = 0$	
Prod: $*[$ producir.n ; $\{s \leq N - p + c\}$ **if** $0 < s \rightarrow s := s - 1$ **fi** ; $\{s \leq N - p + c - 1\}$ $buf.(p\ mod\ N) := n$; $p, t := p + 1, t + 1$ $\{s \leq N - p + c\}$]	Cons:$*[$ $\{t \leq p - c\}$ **if** $0 < t \rightarrow t := t - 1$ **fi** ; $\{t \leq p - c - 1\}$ $m := buf.(c\ mod\ N)$; $c, s := c + 1, s + 1$; $\{t \leq p - c\}$ consumir.m]

Para obtener una solución satisfactoria resta sólo eliminar al asignación míltiple. Presentamos dos soluciones diferentes. En la primera se fortalecerá la anotación introduciendo nuevas variables s' y t' las cuales van a reemplazar a s y t respectivamente. Para poder aplicar fortalecimiento de la guarda, será necesatrio mantener invariante

$$(0 \leq s' \Rightarrow 0 \leq s) \wedge (0 \leq t' \Rightarrow 0 \leq t)$$

La forma más simple de asegurar este invariante, es requerir el siguiente el cual obviamente lo implica:

$$s' \leq s \wedge t' \leq t$$

la forma de mantenerlo es la usual obteniéndose el siguiente programa. La corrección de la anotación es inmediata.

Pre: $p = 0 \wedge c = 0 \wedge s = N \wedge s' = N \wedge t = 0 \wedge t' = 0$	
Prod: $*[$ producir.n	Cons:$*[$ $\{t \leq p - c\}$
; $\{s \leq N - p + c\}$	**if** $0 < t \rightarrow t, t' := t - 1, t' - 1$ **fi**
if $0 < s \rightarrow s, s' := s - 1, s' - 1$ **fi**	; $\{t \leq p - c - 1\}$
; $\{s \leq N - p + c - 1\}$	$m := buf.(c \ mod \ N)$
$buf.(p \ mod \ N) := n$	; $c, s := c + 1, s + 1$
; $p, t := p + 1, t + 1$	$\{s' < s\}$
$\{t' < t\}$	$s' := s' + 1$
$t' := t' + 1$	; $\{t \leq p - c\}$
$\{s \leq N - p + c\}$	consumir.m
]	]
Inv: $s' \leq s \wedge t' \leq t$	

Reemplazando las guardas y eliminando las variables auxiliares queda la siguiente solución

Pre: $p = 0 \wedge c = 0 \wedge s' = N \wedge t' = 0$	
Prod: $*[$ producir.n	Cons:$*[$ $\{t \leq p - c\}$
if $0 < s' \rightarrow s' := s' - 1$ **fi**	**if** $0 < t' \rightarrow t' := t' - 1$ **fi**
$buf.(p \ mod \ N) := n$	$m := buf.(c \ mod \ N)$
; $p := p + 1$	; $c := c + 1$
$t' := t' + 1$	$s' := s' + 1$
]	consumir.m
	]

Por último, se reestablecen los nombres y se reemplazan por las operaciones sobre semáforos.

Pre: $p = 0 \wedge c = 0 \wedge s = N \wedge t = 0$	
Prod: $*[$ producir.n	Cons:$*[$ $P.t$
; $P.s$	; $m := buf.(c \ mod \ N)$
; $buf.(p \ mod \ N) := n$	; $c := c + 1$
; $p := p + 1$	; $V.s$
; $V.t$	; consumir.m
]	]

Un alternativa es introducir variables para tener en cuenta el espacio de tiempo entre que se incrementan los contadores de cuanto se ha producido y consumido y la habilitación del respectivo semaáforo. Usamos para ello variables ap y ac que pueden tomar sólo valores 0 o 1.

Pre: $p = 0 \wedge c = 0 \wedge s = N \wedge t = 0 \wedge ap = 0 \wedge ac = 0$	
Prod: $*[$ producir.n ; $\{s + ac \leq N - p + c\}$ **if** $0 < s \rightarrow s := s - 1$ **fi** ; $\{s + ac \leq N - p + c - 1\}$ $buf.(p\ mod\ N) := n$; $p, ap := p + 1, 1$; $t, ap := t + 1, 0$ $\{s + ac \leq N - p + c\}$ $]$	Cons:$*[$ $\{t + ap \leq p - c\}$ **if** $0 < t \rightarrow t := t - 1$ **fi** ; $\{t + ap \leq p - c - 1\}$ $m := buf.(c\ mod\ N)$; $c, ac := c + 1, 1$; $s, ap := s + 1, 0$; $\{t + ap \leq p - c\}$ consumir.m $]$

Si ahora se reemplazan las operaciones sobre s y t y se eliminan las variables auxiliares, se obtiene la misma solución.

Algunos comentarios y comparaciones son pertinentes.

1. Este ejemplo fue presentado por Schneider en el libro [?], pero la derivación difiere sutilmente en algunos puntos. La principal diferencia radica en que en ese trabajo se elijen aserciones más fuertes que las consideradas aquí. En particular, se pide que el valor de cada semáforo sea igual a alguna diferencia entre cantidad de elementos producidos, cantidad de elementos consumidos y el tamaño del bufer, en lugar de menor o igual como en el desarollo presentado aquí. Se recomienda al lector realizar la derivación de la solución con semáforos usando esos predicados más fuertes.

2. La decisión de tomar los predicados más débiles tiene algunas consecuencias para la derivación del programa. Una es que las operaciones V sobre el semáforo son introducidas para asegurar progreso, mientras que la la corrección parcial queda demostrada previamente. Esto permite una buena separación de incumbencias.

3. Quizá la diferencia más notoria aparece en el úlmimo refinamiento del grado de atomicidad de la solución. En el ejemplo presentado en este trabajo, para eliminar la asignación múltiple se realiza solamente un *fortalecimiento* de la anotación, mientras que usando el predicado más fuerte es necesario diseñar una nueva anotación argumentando que preserva el significado original del problema a resolver, rompiendo con esto con la separación de incumbencias que debería existir entre las sucesivas versiones del multiprograma.

1.3. Cambio de variables

EL método aplicado en el ejemplo anterior es una modificación del método presentado en el libro [?] el cual es una adaptación de otro propuesto por Andrews en [?, ?]. Este método sirve para transformar algunos programas que usan sentencias atómicas con guardas cualesquiera en programas cuyo único medio de sincronización es a través de semáforos. La idea es reescribir las sentecias atómicas condicionales como operaciones P manteniendo la corrección de la anotación y sin introducir peligro de deadlock. Para ello se sugiere realizar los siguientes pasos para reemplazar una sentencia de la forma

$$\textbf{if } B \to T \textbf{ fi } \{R\}$$

1. Determinar un predicado Q (el cual puede incluir una variable nueva s de tipo semaáforo) tal que

   $$\{Q \wedge 0 < s\} \ P.s \ \{R\}$$

   o bien, dado que s no puede aparecer en R,

   $$[Q \wedge 0 < s \Rightarrow R]$$

2. Reemplazar la sentencia atómica condicional por $P.s$ y postular una precondición para ella la cual implique Q.

3. Asegurar que las nuevas aserciones son correctas y que hay progreso. Para esto puede ser necesario agregar sentencias $V.s$ junto a las sentencias que modifican las variables de Q.