Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

Métodos Formales para Ingeniería de Software

Examen de Promoción - Modelado de Dinámica con Relaciones

24 de Noviembre de 2023

Considere el clásico problema de sincronización **Lectores y Escritores**. En este problema se cuenta con un área de memoria compartida (sección crítica) y una cantidad de procesos o hilos (diferenciados en lectores y escritores). La idea es proteger los datos compartidos a través de exclusión mutua (dos hilos de ejecución o procesos no pueden acceder al mismo dato al mismo tiempo). De esta manera, un proceso puede ingresar a la sección crítica cuando no hay nadie en ella, y egresa luego de los ciclos que necesite para realizar la operación deseada. Inicialmente no hay ningún proceso utilizando la sección crítica.

Ejercicio 1: Considere el modelo brindado en el **Anexo** del examen. El mismo intenta modelar el escenario en el que se cuenta con una sección crítica, hay al menos un lector y un escritor, y como máximo hay 3 procesos de cada tipo.

Considere que los cambios de estado se producen porque:

- un único lector o escritor ingresa en la sección crítica; o
- un único lector o escritor realiza una operación que utiliza la sección crítica; o
- un único lector o escritor egresa de la sección crítica.

A partir del modelo brindado en el anexo, resuelva los siguientes incisos:

a) Validar el modelo utilizando el analizador. Indicar explícitamente en un comentario qué irregularidades se encontraron en la definición dada.

b) Realizar una copia del archivo *.als* elaborado hasta el momento, y trabajar sobre el nuevo archivo. Modificar el modelo para subsanar las irregularidades detectadas y modelar el problema correctamente, sin utilizar relaciones ternarias y considerando dos secciones críticas en lugar de una.

c) Validar el modelo elaborado en el inciso anterior utilizando el analizador.

Ejercicio 2: Verificar si se cumple la siguiente propiedad utilizando el analizador:

"*Todo lector que ingresa en una sección crítica eventualmente la abandona*".

Expresar su respuesta en un comentario, justificando la misma a partir de los resultados obtenidos por el analizador.

Ejercicio 3: Realizar una copia del archivo *.als* elaborado hasta el momento, y trabajar sobre el nuevo archivo. Efectúe las modificaciones necesarias para priorizar la salida de un proceso de la sección crítica por sobre el resto de las operaciones de cambio, y revalide la propiedad indicada en el ejercicio anterior.

Ejercicio 4: Podemos notar que la solución planteada para el problema resulta sub-óptima, porque es posible que un lector bloquee el acceso a la sección crítica cuando otro lector la quiere usar. Los lectores no tienen conflicto ya que no modifican datos, por lo que podría permitirse que múltiples lectores accedan en simultáneo a una misma sección crítica.

Realizar una copia del archivo *.als* elaborado hasta el momento, y trabajar sobre el nuevo archivo. Añadir una restricción que asegure que: "*Los lectores no están obligados a quedar en espera si la sección crítica a la que quieren acceder está siendo utilizada por un lector*".

Efectuar las modificaciones necesarias para que el modelo permita operar con esta nueva restricción. Validar el modelo resultante utilizando el analizador.

Anexo:

```
open util/ordering[Estado] as ord

abstract sig Proceso {}
sig Lector, Escritor extends Proceso {}
one sig MemoriaCompartida {}

sig Estado{
              seccionCritica: Proceso -> lone MemoriaCompartida
}

fact cantidadProcesos { #Lector!=none and #Lector<=3 and #Escritor!=none }

fact init { no ((ord/first).seccionCritica) }

fact traces { all e: Estado-ord/last | let eSig = e.next | ingreso[e, eSig] or
                                                           egreso[e, eSig] or
                                                           realizarOperacion[e, eSig]
}

pred ingreso[e1, e2: Estado]{
    one p: Proceso |
                    --Pre
                    no (p.(e1.seccionCritica)) and
                    --Post
                    (e2.seccionCritica) = (e1.seccionCritica) + (p->MemoriaCompartida)
}


pred realizarOperacion[e1, e2: Estado]{
    one p: Proceso|
                   --Pre
                   ((p->MemoriaCompartida) in (e1.seccionCritica)) and
                   --Post
                   ((p->MemoriaCompartida) in (e2.seccionCritica))
}

pred egreso[e1, e2: Estado]{
    one p: Proceso |
                    --Pre
                    ((p->MemoriaCompartida) in (e1.seccionCritica)) and
                    --Pos
                    (e2.seccionCritica) = (e1.seccionCritica) - (p->MemoriaCompartida)
}
```