Especificación y Verificación de Software

Máster en Investigación en Informática (UCM)

Hoja 4 Curso 2009/2010

En los ejercicios que sea posible, ejecuta los módulos en Maude y comprueba que no se producen errores.

Ejercicio 1 Considera los siguientes módulos:

```
fmod FUNCTIONAL is
  including NAT .
  sort Counter .
  op c : Nat -> Counter .

  eq c(s(N:Nat)) = c(N:Nat) .
endfm
```

```
mod SYSTEM is
  including NAT .
  sort Counter .
  op c : Nat -> Counter .

  rl c(s(N:Nat)) => c(N:Nat) .
endm
```

¿Se diferencia en algo la ejecución del término `c(10)` en ambos módulos? A nivel semántico, ¿existe alguna diferencia?

Considera ahora la siguiente extensión de los módulos anteriores:

```
fmod FUNCTIONAL-2 is
  including FUNCTIONAL .
  op running : Counter -> Bool .
  eq running(c(N:Nat)) = N:Nat > 0 .

endfm
```

```
mod SYSTEM-2 is
  including SYSTEM .
  op running : Counter -> Bool .
  eq running(c(N:Nat)) =
     N:Nat > 0 .
endm
```

¿Cuál es el resultado de ejecutar `running(c(10))`?

Ejercicio 2 ¿Es la siguiente especificación coherente? Justifícalo, si lo es, o modifícala para que lo sea, en caso contrario.

```
mod COHERENCE is
  sort S .
  ops a b d : -> S .
  ops c f : -> S [ctor] .
  eq a = b .
  eq b = c .
  eq d = f .
  rl [l] : b => d .
endm
```

Ejercicio 3 Especifica un sistema de lectores/escritores en el que la configuración venga dada por la operación `op <_,_> : Nat Nat -> Conf`, donde el primer argumento representa el número de lectores en el sistema y, el segundo, el número de escritores. Escribe reglas que permitan entrar y salir lectores y escritores del sistema, con la condición de que no puede haber simultáneamente lectores y escritores, ni dos escritores al mismo tiempo. Utiliza el comando `search` para comprobar que, siempre que se den menos de un millón de pasos, el sistema satisface las dos condiciones.

Ejercicio 4 Resuelve el siguiente problema mediante una especificación en Maude, utilizando el comando `search`. ¿Cómo se pueden medir exactamente 9 minutos con dos relojes de arena de 4 y 7 minutos?

Ejercicio 5 Considera la siguiente especificación del problema de los filósofos, disponible en la página del curso:

```
fmod NAT/4 is
  protecting NAT .
  sort Nat/4 .
  op `[_`] : Nat -> Nat/4 .
  op _+_ : Nat/4 Nat/4 -> Nat/4 .
  op _*_ : Nat/4 Nat/4 -> Nat/4 .
  op p : Nat/4 -> Nat/4 .

  vars N M : Nat .

  ceq [N] = [N rem 4] if N >= 4 .
  eq [N] + [M] = [N + M] .
  eq [N] * [M] = [N * M] .
  ceq p([0]) = [N] if s(N) := 4 .
  ceq p([s(N)]) = [N] if N < 4 .
endfm

mod DIN-PHIL is
  protecting NAT/4 .
  sorts Oid Cid Attribute AttributeSet Configuration Object Msg .
  sorts Phil Mode .
  subsort Nat/4 < Oid .
  subsort Attribute < AttributeSet .
  subsort Object < Configuration .
  subsort Msg < Configuration .
  subsort Phil < Cid .
  op __ : Configuration Configuration -> Configuration
    [ assoc comm id: none ] .
  op _`,_ : AttributeSet AttributeSet -> AttributeSet
    [ assoc comm id: null ] .
  op null : -> AttributeSet .
  op none : -> Configuration .
  op mode`:_ : Mode -> Attribute [ gather ( & ) ] .
  op holds`:_ : Configuration -> Attribute [ gather ( & ) ] .
  op <_:_|_> : Oid Cid AttributeSet -> Object .
  op Phil : -> Phil .
  ops t h e : -> Mode .
  op chop : Nat/4 Nat/4 -> Msg [comm] .
  op init : -> Configuration .
  op make-init : Nat/4 -> Configuration .

  vars N M K : Nat .
  var C : Configuration .

  ceq init = make-init([N]) if s(N) := 4 .
  ceq make-init([s(N)]) =
      < [s(N)] : Phil | mode : t , holds : none > make-init([N])
      (chop([s(N)],[N]))
      if N < 4 .
  ceq make-init([0]) =
      < [0] : Phil | mode : t , holds : none > chop([0],[N])
      if s(N) := 4 .

  rl [t2h] : < [N] : Phil | mode : t , holds : none > =>
             < [N] : Phil | mode : h , holds : none > .
  crl [pickl] : < [N] : Phil | mode : h , holds : none > chop([N],[M])
      =>
      < [N] : Phil | mode : h , holds : chop([N],[M]) > if [M] = [s(N)] .
```

```
  rl [pickr] : < [N] : Phil | mode : h , holds : chop([N],[M]) >
     chop([N],[K])
     =>
     < [N] : Phil | mode : h , holds : chop([N],[M]) chop([N],[K]) > .
  rl [h2e] : < [N] : Phil | mode : h , holds : chop([N],[M])
     chop([N],[K]) >
     =>
     < [N] : Phil | mode : e , holds : chop([N],[M]) chop([N],[K]) > .
  rl [e2t] : < [N] : Phil | mode : e , holds : chop([N],[M])
     chop([N],[K]) >
     =>
     chop([N],[M]) chop([N],[K]) < [N] : Phil | mode : t , holds : none > .
endm
```

Hay cuatro filósofos, comiendo en una mesa circular. Inicialmente todos están pensando (`t`), pero pueden volverse hambrientos (`h`) y, después de coger los palillos izquierdo y derecho, ponerse a comer (`e`), tras lo que vuelven a pensar. Los filósofos se identifican con los naturales módulo 4; los palillos se numeran con los dos filósofos que están a su lado. Demuestra, utilizando el comando `search` a partir del estado `init`:

- nunca pueden estar comiendo simultáneamente dos filósofos *contiguos*;
- sin embargo, es posible que dos filósofos coman simultáneamente;
- es imposible que tres filósofos coman simultáneamente;
- el sistema se puede quedar bloqueado (*deadlock*).