Especificación y Verificación de Software

Máster en Investigación en Informática (UCM)

Hoja 5 Curso 2009/2010

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

Ejercicio 1 Obtén la definición de la semántica de los operadores $\Box$, $\Diamond$ y $\mathcal{W}$ a partir de su definición en términos de $\bigcirc$ y $\mathcal{U}$ y de la semántica de estos.

Ejercicio 2 Una especificación en Maude describe el comportamiento de un sistema concurrente que es representado mediante el siguiente constructor:

```
ops none req snok sok sdnk rfst rnok rinc rok : -> Label [ctor] .
op <_,_,_> : Sender Receiver Label -> State [ctor] .
```

El tercer componente del estado es una etiqueta que representa el nombre de la transición utilizada para alcanzar el estado actual (o `none`, si el nombre es irrelevante). El sistema debería satisfacer las siguientes propiedades:

1. Una petición `req` debe ir seguida de una confirmación `sok`, `snok` o `sdnk` antes de la siguiente petición.
2. Una indicación `rfst` debe ir seguida de una de las indicaciones `rok` o `rnok` antes del comienzo de una nueva transmisión (nueva petición del emisor).
3. La confirmación `sok` debe ir precedida de una señal `rok`.
4. La señal `rnok` debe ir precedida de una confirmación `snok` o `sdnk`.

Especifica estas propiedades de manera que puedan ser comprobadas con el comprobador de modelos de Maude. Puede suponerse que ninguna transición se repite entre dos transiciones `req`.

Si el sistema se representara mediante tres objetos:

```
< S : Sender | ... > < R : Receiver | ... > < L : Label >
```

¿qué cambios habría que hacer para representar las mismas propiedades?

Ejercicio 3 Utiliza el comprobador de modelos y la especificación del problema *crossing the river*, modificando quizás las proposiciones atómicas, para encontrar una secuencia de movimientos en la que el lobo se coma al cordero, aunque no sin que antes este se haya comido la col.

Ejercicio 4 Construye un módulo que importe `META-LEVEL` y que incluya operaciones

```
ops isReducible? isRewritable? : Module Term -> Bool .
```

que devuelvan `true` si el segundo argumento representa un término que se puede simplificar en el módulo metarrepresentado por medio de ecuaciones o reglas, respectivamente.

1. Implementa `isReducible?` de manera que devuelva `true` si el término se puede simplificar por medio de ecuaciones, aunque no se le pueda aplicar ninguna regla.
2. Implementa `isRewritable?` de manera que devuelva `true` solo si existe una regla de reescritura que se pueda aplicar directamente sobre el término, sin simplificarlo antes.

Ejercicio 5 Define una función al metanivel que tome como argumento la metarrepresentación de un módulo y devuelva el módulo resultante de aplicar la encapsulación y refinamiento de reglas vistas en el ejemplo *crossing the river*. Suponemos que:

1. El estado se construye con una operación

   ```
   op __ : System System -> System [ctor assoc comm] .
   ```

2. Todas las reglas se dan al nivel del tipo `System`.
3. Existe una operación `enabled : System -> Bool` que es la que debe utilizarse para refinar las reglas.