Auditoría, Calidad y Fiabilidad Informáticas

Universidad Complutense de Madrid Curso 2020/2021

Práctica 4: Arquitectura en estrella

Especifica y analiza en Maude una arquitectura de red en estrella. Para ello completa el fichero `p4_individual.maude`; los tipos `Sistema` y `Localizacion` ya están definidos, por lo que nuestro sistema tendrá la forma `{1 | ...}...{n | ...}`, donde cada `{i | ...}` es una `Localizacion` de identificador `i` y su contenido serán nodos y mensajes (de tipos `Nodo` y `Msj`, respectivamente). Nodos y mensajes se definen y comportan como se explica a continuación:

1. Definición

Ejercicio 1 Los `Contenidos` de una `Localizacion` son un conjunto de nodos y mensajes. Define los `subsort` y los operadores necesarios para definir esta estructura de datos. En el resto de la práctica trabajaremos asumiendo que existe un único nodo en cada `Localizacion`.

Ejercicio 2 Los nodos tienen los siguientes constructores:

- Un constructor para *extremos*, que almacenan su propio identificador, el identificador de la localización en la que se encuentra el centro de la estrella, un estado (puede ser `inactivo`, `esperando` y `activo`), una lista de identificadores de extremos (los "amigos" del extremo) y un `String` con los mensajes recibidos.
- Un constructor para el *centro*, que almacena su propio identificador, una tabla hash para asociar los identificadores de los extremos con el identificador de su `Localizacion` y un estado (`inactivo` o `activo`).
  - No uses números naturales para identificar los nodos, para poder distinguir entre identificadores de nodos e identificadores de localizaciones.
  - Es posible que te resulte más fácil definir la tabla hash en un módulo funcional separado que sea importado por `RED`.

Ejercicio 3 Define un mensaje `info`, que tiene como argumentos:

- El identificador de la `Localizacion` en la que se encuentra el centro (es decir, un natural).
- El identificador de la `Localizacion` en la que se encuentra el extremo que manda el mensaje (es decir, otro natural).
- El identificador del extremo que manda el mensaje.

Ejercicio 4 Define un mensaje `respuesta-info` que tiene como argumento el identificador del nodo al que va dirigido.

Ejercicio 5 Define un mensaje `to_:_` que tiene como argumentos:

- El identificador del nodo al que va dirigido.
- Un `String` con un mensaje.

Ejercicio 6 Define una función `numNodos` que cuenta el número de nodos en un sistema.

2. Comportamiento

Ejercicio 7 Cuando en una misma localización tenemos un mensaje y el nodo al que va dirigido el mensaje se procesa.

Ejercicio 8 Cuando un mensaje va dirigido a un nodo en otra localización tenemos las siguientes opciones:

- Los extremos mandan su mensaje al centro.
- El centro usa su tabla hash para enviar el mensaje a la localización correcta.

Ejercicio 9 El comportamiento del mensaje `info` es como sigue:

- El mensaje `info` lo envían los extremos en estado `inactivo` para indicar su dirección y su nombre. Al enviarlo pasa al estado `esperando`.
- Este mensaje es recibido por el centro y se utiliza para actualizar la tabla.
- El centro pasa de `inactivo` a `activo` en cuanto recibe uno de estos mensajes.
- En la misma regla el centro envía `respuesta-info` al extremo como respuesta.

Ejercicio 10 Cuando un extremo recibe el mensaje `respuesta-info` actualiza su estado y pasa a `activo`.

Ejercicio 11 Los nodos con amigos mandan un mensaje de la forma `to_:_` a dichos amigos diciéndoles `"hola"`. Asegúrate de que solo manden uno de estos mensajes a cada amigo (es válido borrar amigos de la lista).

Ejercicio 12 Cuando un mensaje `to_:_` llega a un nodo el mensaje se concatena a lo que ya habíamos recibido.

Ejercicio 13 Define, en un módulo `EJEMPLO`, un sistema inicial con un centro y tres extremos, todos ellos inicialmente inactivos. Cada extremo es amigo de los otros dos extremos e inicialmente ha recibido `""`. Utiliza el comando `rew` para ejecutarlo.

Ejercicio 14 Utiliza el comando `search` para comprobar que el número de nodos permanece invariable durante toda la ejecución.

3. Análisis

Ejercicio 15 Crea un módulo `PROPS` para definir propiedades de *model checking* y define el estado sobre el que demostrarás las propiedades.

Ejercicio 16 Define propiedades para:

- Comprobar si un cierto nodo existe, dado su identificador.
- Comprobar si algún nodo tiene como amigo a un cierto nodo (dados los identificadores de ambos).
- Comprobar si existe un mensaje de la forma `to_:_` para un cierto nodo, dado su identificador.
- Comprobar si la cantidad de nodos es una cierta cantidad, dada como argumento.
- Comprobar si la cantidad de extremos es una cierta cantidad, dada como argumento.

Ejercicio 17 Comprueba las siguientes propiedades con el término inicial de la sección anterior:

- La cantidad de nodos no varía.
- Si un nodo existe y otro lo tiene como amigo, le acaba mandando un mensaje.
- Cualquier mensaje acaba desapareciendo.