Métodos Formales para Ingeniería de Software

Final 13 de agosto de 2023

Profesora: Dra. María Laura Cobo

La “Ciudad Escondida” es una ciudad amurallada que tiene un único puente de acceso. Si bien a simple vista este detalle no es tan excepcional en ciudades antiguas, el de esta ciudad tiene algunas particularidades. Los habitantes de la ciudad pueden ingresar sin restricciones. Por otra parte, las autoridades de la ciudad decidieron que los turistas solo pueden ingresar si en la ciudad hay más habitantes dentro de los muros que turistas. Acorde a esta restricción, los habitantes solo pueden egresar si no hay turistas o si se mantiene la misma relación de seguridad (quedan más habitantes que turistas).

El puente admite un único transeúnte por vez; de esta manera las personas ingresan y egresan de la ciudad de a una por vez.

El problema solo considera dos tipos de personas: habitantes y turistas. Una persona no puede ser habitante y turista al mismo tiempo. Para ingresar a la ciudad las personas deben registrar su intención de ingresar. Se eliminarán del registro únicamente al ingresar.

No se modelará la intención de egreso. Puede egresar cualquier persona que esté dentro de la ciudad.

A las autoridades no les interesa tener discriminado a los tipos de personas que están en la ciudad.

Es posible que la ciudad quede vacía.

Considerando el escenario descripto, resuelva:

1. Evalúe si el modelo, estático, dado a continuación es adecuado. Deje comentarios sobre los comandos utilizados y los resultados obtenidos/problemas encontrados. **No modifique el modelo en este inciso**

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


sig Persona {}

sig Habitante in Persona {}

sig Turista in Persona {}

sig Estado {

     habitantesEnCiudad: set Persona,

     poblacion: some Persona,

     esperandoIngreso: lone Persona

}
```

```
// mínima cantidad de personas

fact {#Habitante>1 and #Turista>1}


// relación turistas - habitantes

fact {all e: Estado | #(e.poblacion&Habitante) <= #(e.poblacion&Turista)
}
```

2. Elimine las relaciones que considere que considere innecesarias. Justifique sus elecciones. Corrija los errores detectados en el inciso anterior y verifique que el modelo sea correcto estáticamente.
3. Teniendo en cuenta el siguiente fact como punto de partida, complete los predicados para contar con la dinámica apropiada para el escenario dado.

```
fact traces{ inicializar[ord/first] and

     all est: Estado - ord/last | let sigEst = est.next |

     registrarse[est,sigEst] or

     ingresar[est,sigEst] or

     egresar[est,sigEst]

}
```

Las únicas acciones posibles son:

- Una persona, que no está en la ciudad, se registra y queda en espera para ingresar
- Una persona que está registrada, pasa a formar parte de la *población* de la ciudad
- Una persona que está en la ciudad (*población* de la ciudad) abandona la ciudad. Observación: no queda registro de que salió de la ciudad)

El predicado *inicializar[estado]* determina las condiciones que establezca para el estado inicial

4. Validar si es posible que un habitante esté en la ciudad y en el próximo estado ya no esté. Dejar expresado en un comentario cuál es la respuesta del analizador.
5. Validar si es posible que un turista que está en la ciudad no pueda abandonar la ciudad (hay un estado donde el turista pertenece al conjunto *población* y la pertenencia al conjunto se mantiene en todos los estados posteriores a ese).

   Dejar expresado en un comentario cuál es la respuesta del analizador.

6. Verificar la siguiente propiedad:

   En todos los estados, si no hay habitantes en la ciudad y en la cola de ingreso hay turistas y habitantes, entonces en el siguiente estado ingresará un habitante a la ciudad

   Deje expresado en un comentario cuál es la respuesta del analizador en cada caso.

7. Si se le da prioridad a la operación de ingreso por sobre las otras, ¿cambia la respuesta del analizador con respecto a la propiedad del inciso anterior? Realice la modificación y muestre el resultado dado por el analizador.