Testing FSM con ModelJunit

1. predisporre il progetto

Scarica modeljunit.jar da:

http://sourceforge.net/projects/modeljunit/

Crea un progetto eclipse e metti il jar nel build path di un progetto nuovo.

Il progetto d'esempio lo trovi stto mbtxfsm.

2. scrivere il modello della FSM

Il modello FSM si scrive direttamente in Java.

Normalmente questa classe si collega alla SUT (e non è la SUT stessa) (cioè ha in se un riferimento al sistema under test).

La classe del modello FSM deve estendere l'interfaccia `FsmModel`

Con i seguenti metodi:

`Object getState()`: This method returns the current visible state of the EFSM. So this method defines an abstraction function that maps the internal state of the EFSM to the visible states of the EFSM graph. Typically, the result is a string, but it is possible to return any type of object.

`void reset(boolean)`: This method resets the EFSM to its initial state. When online testing is being used, it should also reset the SUT or create a new instance of the SUT class. The boolean parameter can be ignored for most unit testing applications.

`@Action void` *`namei`*`()`: The EFSM must define several of these action methods, each marked with an @Action annotation. These action methods define the transitions of the EFSM. They can change the current state of the EFSM, and when online testing is being used, they also send test inputs to the SUT and check the correctness of its responses.

`boolean` *`namei`*`Guard()`: Each action method can optionally have a guard, which is a boolean method with the same name as the action method but with “Guard” added to the end of the name. When the guard returns true, then the action is enabled (so may be called), and when the guard returns false, the action is disabled (so will not be called). Any action method that does not have a corresponding guard method is considered to have an implicit guard that is always true.

NOTA: il controllo di correttezza del comportamento del SUT viene fatto nei metodi @Action e/o getState. In Action dovrei controllare l'ouput, in getState che il sistema vada nello stato opportuno.

Esempio: OnOff.java

OnOff2.java ha diverse action

OnOff3.java ha anche delle condizioni

3. Ci sono due modi per testare la FSM.

A. ONLINE testing

Scrivi un caso di test Junit che testa la tua classe. Del tipo:

```
// create our model and a test generation algorithm
//Tester tester = new RandomTester(new FSM());
```

```
Tester tester = new GreedyTester(new FSM());

// build the complete FSM graph for our model, just to ensure
// that we get accurate model coverage metrics.
tester.buildGraph();

// set up our favourite coverage metric
CoverageMetric trCoverage = new TransitionCoverage();
tester.addCoverageMetric(trCoverage);

// ask to print the generated tests
tester.addListener(new VerboseListener());

// generate a small test suite of 20 steps (covers 4/5 transitions)
tester.generate(20);

tester.getModel().printMessage(trCoverage.getName() + " was " +
trCoverage.toString());
```

Puoi cambiare:

1. il tester (random ....)
2. il misuratore di coverage

B. OFFLINE testing (con l'interfaccia)

Usando il tool come interfaccia grafica possiamo esplorare il modello e generare le tracce offline.

Come aprire il progetto con il tool

1. Scrivi la classe che rappresenta la tua FSM e la tua SUT
2. metti tutto in un jar
3. esegui il tool: java -jar modeljunit-2.5-jar-with-dependencies.jar

Per far funzionare il tutto io sono riuscito solo creando un file di progetto exampleOnOff.mju

Nel quale ho modificato:

```
<className>it.unibg.tvsw.OnOff3</className>
<packageLocation>file:esercizio.jar</packageLocation>
```

Esercizi FSM testing

1. Macchinetta per il caffe

Un macchina per il caffe riceve in ingresso dei gettoni e produce quando richiesto il caffe. L'utente può inserire fino a 5 gettoni. Esistono due tipi di caffe: quello normale (1 gettone) e quello super (2 gettoni). Il credito attuale viene visualizzato sul display.

A.

Formalizza (su carta):

- gli stati della macchina

- l'input che accetta la macchina (token e produci caffe e reset?)

- la transizioni di stato

B.

prova a implementare la macchinetta in Java . Con oppurtuni output e metodi per sapere quanti crediti ci sono e così via.

Implementa la sua specifica FSM in Java

Prova ad eseguire l'online testing con il tool

Prova anche a generare tracce con il tool.