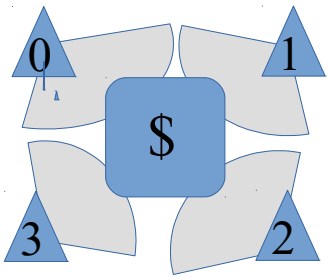
Testing e verifica del software – 6 Giugno 2016

Sensori Allarme



Una cassaforte è protetta sui 4 spigoli da 4 sensori di allarme (numerati da 0 a 3 come in figura). Ogni sensore sorveglia due lati. L'operatore può accederli e spegnerli, però un sensore può essere spento solo se i suoi due vicini sono accesi (altrimenti si creerebbe un lato non protetto).
All'inizio i sensori sono tutti accesi.

Compito

In un file word annota tutto quello che fai. Aggiungi anche gli screenshot per i punti principali (ad esempio copertura ...). Per effettuare gli screenshots usa lo strumento di cattura di windows. Anche la documentazione sarà valutata per ogni singolo progetto (circa del 15%).

Sviluppa un progetto in eclipse per ognuna delle domande. Ogni progetto dovrà avere come nome il tuo COGNOME_**ES** (con ES il tipo dell'esercizio – vedi titoli degli esercizi). Puoi anche andare in parallelo (ad esempio modello NUSMV e codice Java). Metti i progetti e il documento word in una directory con nome COGNOME_NOME (che puoi usare come workspace). Alla fine crea un file zip e consegna quello.

Commenta per bene tutto il codice che scrivi.

Java

In Java scrivi una classe **Allarme** che implementa lo stato dei 4 sensori con un array di 4 boolean (vero = acceso, falso = spento). Ha tre metodi:

```
public boolean setSensore(int sensore, boolean acceso)
```

che accende (se acceso è true) o spegne il sensore passato con il suo numero identificativo. Restituisce true solo se cambia lo stato del sensore, altrimenti restituisce false;

```
public boolean acceso(int sensore)
```

Restituisce true se e solo se il sensore con numero "sensore" esiste ed è acceso.

```
public boolean pericolo()
```

che restituisce true se due sensori adiacenti sono accesi.

1. ES = TESTING

Scrivere l'implementazione e i casi di test con Junit. Scrivi una classe di test AllarmeTest con i seguenti test:

1. un caso di test che mostri che è possibile spegnere due sensori non adiacenti.
2. un caso di test in cui provi a spegnere tutti e quattro ma controlli che non è possibile

Esegui il controllo della copertura (istruzioni, branch, decisioni, MCDC). Se il caso sopra non è sufficiente ad avere una copertura soddisfacente, aggiungi i casi di test necessari (in una classe separata per ogni copertura). Se ritieni che qualche decisione non sia copribile giustificalo.

Metti la directory test con i test separata (oltre src).

Prova con Randoop a generare casi di test (con pochi casi di test, tipo 10) e prova a valutarne la copertura.

2. ES = JML

Scrivi i contratti JML. Cerca di scrivere sia le precondizioni, che le postcondizioni dei metodi. Cerca di spostare nelle precondizioni alcuni controlli che facevi all'inizio dei metodi nel precedente esercizio.

Cerca di scrivere anche invarianti.

Prova i contratti JML con una classe main in cui chiami i diversi metodi.

Prova anche a modificare il codice e controlla che i contratti siano violati. Documenta bene le violazioni e le loro cause in commenti e nel file di documento.

3. ES = KEY Verifica dei programmi

CONSIGLIO. Inizia da una classe con contratti **vuoti** (true) e **aggiungi** mano a mano i contratti però sempre dimostrandone la loro correttezza – cioè le prove devono essere sempre chiuse **vere**. La valutazione dipenderà dal massimo contratto che riuscite a dimostrare vero (contratti non provati non verranno valutati).

Se la tua classe contiene dei metodi delle librerie (tipo system.out.println) eliminali. Anche Random va eliminata se l'hai usata.

4. ES = NUMSV

Specificare il sistema di allarme utilizzando NuSMV e le sue proprietà usando LTL e/o CTL. Puoi modellare la scelta di un sensore da modificare oppure puoi anche non avere alcun input e scegliere non deterministicamente il valore dei sensori (in modo corretto). Puoi rappresentare i sensori con un array, con 4 booleani o con 4 moduli (se viene più compatta la specifica).

Tra le proprietà desiderate (alcune potrebbero essere false) c'è:

- P1: non posso mai avere una situazione di pericolo
- P2: non posso mai avere i sensori 0 e 2 contemporaneamente spenti.
- P3: una volta spento un sensore può essere sempre riacceso.
- P4: un sensore acceso può essere sempre spento

- P5: se il sensore 0 è acceso e quello 1 è spento, allora 0 rimarrà acceso almeno fino a quando 1 non si accenderà (se si accenderà).

Provare se sono vere oppure no. Se sono false presentare un contro esempio (se viene prodotto).

Aggiungi anche altre proprietà che ti sembrano importanti e commenta.

Per ogni proprietà aggiungi una descrizione in Italiano di quello che vuol dire.

5. ES = FSM (MBT con le FSM)

Scrivi una FSM usando ModelJunit (non connetterla con la sut, usa ModelJUnit solo per costruire la EFDM). Prova sia a fare online testing che offline testing.

Salva il disegno della macchina come immagine.

Prendi le prime 5 transizioni di un caso di test qualsiasi e prova a tradurle in caso di test Junit dell'allarme (basta anche che descrivi come faresti).

6. ES = COMB (Combinatorial testing)

Rappresenta in combinatorial testing il sistema con lo stato dei 4 sensori, pericolo e con il fatto che non devi avere mai pericolo. Introduci anche gli opportuni vincoli.

Applica il combinatorial testing.

Prova a generare la copertura pairwise.