Esame di testing e verifica del software – 04/02/2022 3.15h

Senza appunti.

Senza eclipse

1. Testing di un programma

Dato il seguente programma in Java, che restituisce il numero pari positivo più grande in un array. Se non esiste restituisce -1

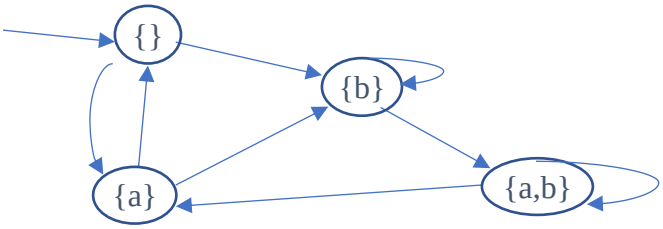
```
int maxp(int[] a) {
	int max = -1;
	if (! (a == null || a.length == 0)) {
		for (int x : a) {
			if (x > 0 && x % 2 == 0 && x > max)
				max = x;
		}
	}
	return max;
}
```

Scrivi i casi di test per: la copertura delle istruzioni (1), di branch (2), delle condizioni (3) e MCDC (4). Cerca di **minimizzare** i casi di test necessari per avere queste coperture. Non è necessario che tu scriva i test come junit (però deve essere possibile scrivere i test junit dai tuoi test).

Potrei fare un caso di test che copre le istruzioni senza coprire tutti i branch?

2. algoritmo di model checking

Data la seguente macchina M



Mediante l'algoritmo di model checking, dire in quali stati s valgono le proprietà:

- M,s |= AG(a and b)
- M,s |= E(a U b)

3. combinatorial testing

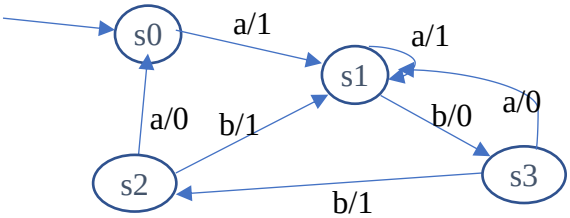
Date tre variabili con i loro domini S: 1,2 , T: 3,4 V:5,6,

Costruisci la test suite combinatoriale pairwise usando l'algoritmo IPO

Riesci a costruire una test suite compatta con 4 casi di test e una allungata con almeno 5 casi di test (ma non ripetuti)?

4. conformance testing

Data la seguente FSM con 4 stati, due input a e b e due output 1 e 0



La macchina è correttamente definita? (giustifica la risposta – se non lo è, correggi la macchina a tuo piacimento)

Scrivi due test sequences, una per la copertura degli stati e una per la copertura delle transizioni.

Fa due esempi di errori (di quelli visti) e controlla se riesci a scoprirli con i test che hai trovato tu (può succedere che tu non riesca a scoprirli?) Giustifica la risposta.

Con eclipse

5. JML e KEY

Riscrivi il programma dell'esercizio 1 in cui metti gli opportuni contratti e provi la correttezza. Puoi semplificare il codice se i contratti che metti rendono superflui alcuni controlli. Puoi riscrivere il codice a tuo piacimento per portare a termine la dimostrazione. Ricordati la sintassi dei quantificatori:

(\forall <dominio>;<range_valori>;<condizione>)

(\exists <dominio>;<range_valori>;<condizione>)

6. logica temporale con asmeta

Scrivi un modello ASM per un semaforo che però diventa verde solo su richiesta (usa ad esempio con una monitorata boolean). Il semaforo fa il ciclo regolare: ROSSO -> GIALLO -> VERDE -> ROSSO.

Prova ad animare il sistema e fai uno screenshot del comportamento.

Prova le seguenti proprietà sia come LTL che come CTL:

1. Non può mai passare a VERDE direttamente da ROSSO
2. Quando è ROSSO rimarrà sempre ROSSO a meno che ci sia una richiesta
3. Se c'è una richiesta allora prima o poi diventa VERDE.
4. In qualsiasi istante, prima o poi potrebbe diventare VERDE

Se qualcuna è falsa spiega perché.

Scrivi e spiega anche una tua proprietà vera e una falsa il cui controesempio ti aiuta a capire come funziona il semaforo.