

Tema esame inventato - 2

1. Testing di un programma

Dato il seguente programma in Java che restituisce true se la somma dei valori negativi e dispari contenuti nell'array *arr* è minore o uguale a *val* (con *val*=numero strettamente negativo):

```
// arr: può contenere sia valori positivi che negativi
// val: è un valore negativo (strettamente)
public boolean oddNegSumLessX(int[] arr, int val) {
    if(val >= 0)
        return false;

    int sum = 0;

    for(int x : arr)
        if(x % 2 != 0 && x < 0)
            sum += x;

    if(sum <= val)
        return true;
    else
        return false;
}
```

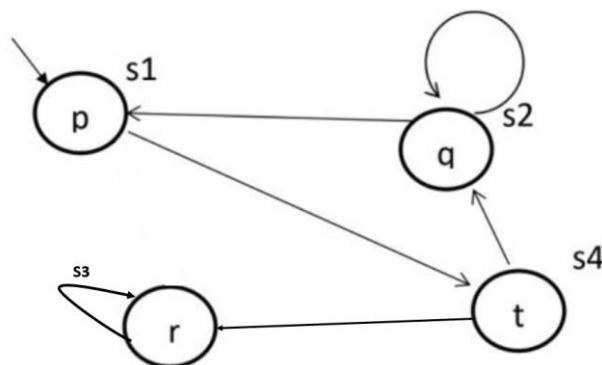
Scrivere i casi di test per:

1. Copertura delle istruzioni
2. Copertura dei branch
3. Copertura delle decisioni
4. Copertura MCDC

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).

2. Algoritmo di model checking

Data la seguente macchina M



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

- $M, s \models \text{AF}(p \text{ or } q)$
- $M, s \models \text{AG}(r)$
- $M, s \models \text{EX}(q \text{ or } t)$

3. Combinatorial testing

Date quattro variabili con i loro domini

D1: A,B

D2: 6,7,8

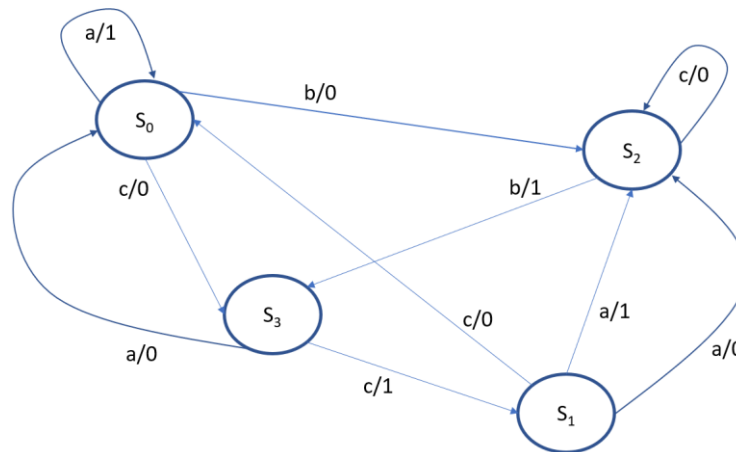
D3: L,M

D4: 9,2,4

Costruisci la test suite combinatoriale pairwise usando l'algoritmo IPO.

4. Conformance testing

Data la seguente FSM, tre input {a,b,c} e due output {1,0}.



La macchina è correttamente definita? (giustifica la risposta)

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

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:

`(\forallall <dominio>;<range_valori>;<condizione>)`

`(\existsexists <dominio>;<range_valori>;<condizione>)`

6. Logica temporale con Asmeta

Un edificio ha 2 piani e un ascensore permette di visitarli mediante l'algoritmo *sabbath*. L'algoritmo prevede che l'ascensore si muova automaticamente dal primo al secondo piano e viceversa (1->2->1->2->...). Quando il bottone *signalPorta* viene premuto (valore *true*), l'ascensore apre le porte e rimane allo stesso piano fino a quando *signalPorta* torna al valore *false*. Tornato al valore *false*, l'ascensore chiude le porte e continua con l'algoritmo *sabbath*.

Prova le seguenti proprietà con LTL o CTL a tua scelta:

1. Se il bottone viene premuto, nel prossimo stato la porta sarà aperta.
2. Prima o poi la porta si aprirà.
3. Quando la porta è aperta, rimane aperta fino a quando *signalPorta* torna a *false* (se torna a *false*).
4. Il piano nello stato successivo rimane lo stesso se *signalPorta* è *true* (scrivi una proprietà che valga sia per il piano 1 che per il piano 2)

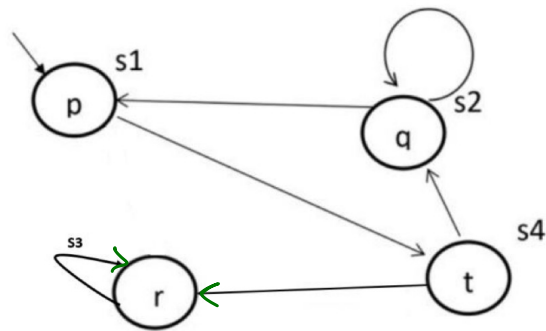
C'è qualche proprietà che invece è giustamente falsa e il cui controesempio ti aiuta a capire come funziona l'ascensore?

Scrivi poi i seguenti scenari, controllando che il valore delle funzioni sia quello atteso:

- L'ascensore parte dal piano 1 con la porta chiusa. Nessun segnale viene attivato e l'ascensore raggiunge il piano 2 mantenendo le porte chiuse. L'ascensore raggiunge il piano 1: il segnale viene attivato e la porta si apre.
- L'ascensore parte dal piano 1 e raggiunge il piano 2. Raggiunto il piano 2, viene forzata l'apertura della porta mantenendo *signalPorta* a false. Nel prossimo stato l'ascensore chiude comunque le porte e va al piano 1.

2. Algoritmo di model checking

Data la seguente macchina M



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

- $M, s \models AF(p \text{ or } q)$
- $M, s \models AG(r)$
- $M, s \models EX(q \text{ or } t)$

1) $AF(p \vee q)$

	s ₁	s ₂	s ₃	s ₄
p	x			
q		x		
$p \vee q$	x	x		
$AF(p \vee q)$	x	x		

2) $AG(r) = \neg E(\text{true} \cup \neg r)$

	s ₁	s ₂	s ₃	s ₄
r			x	
$\neg r$	x	x		x
$E(\text{true} \cup \neg r)$	x	x		x
$\neg E(\dots)$			x	

3) $EX(qvt)$

	s_1	s_2	s_3	s_4
q		x		
t				x
qvt		x		x
$EX(qvt)$	x	x		x

3. Combinatorial testing

Date quattro variabili con i loro domini

D1: A,B

D2: 6,7,8

D3: L,M

D4: 9,2,4

Costruisci la test suite combinatoriale pairwise usando l'algoritmo IPO.

D1	D2	D3	D4
A	6	L	9
A	7	M	2
A	8	L	4
B	6	M	2
B	7	L	9
B	8	M	4
A	6	M	4
A	7	M	4
A	7	L	2
A	7	M	9

1) D1-D2 ✓

2) D1-D3 ✓

D2-D3 ✓

3) D1-D4 ✓

D2-D4 ✓

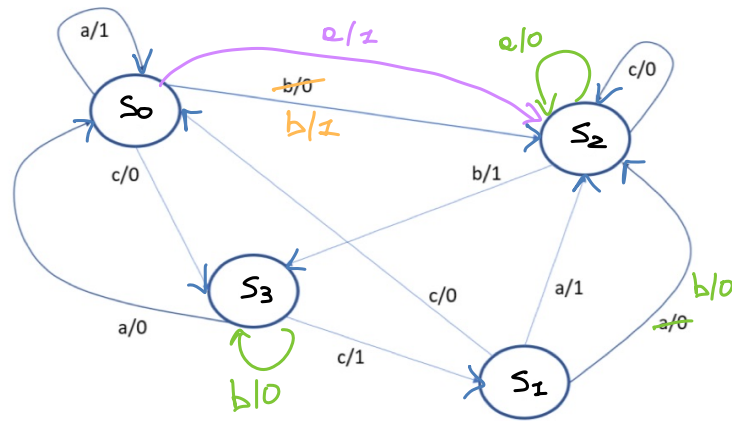
D3-D4 ✓

4. Conformance testing

Data la seguente FSM, tre input {a,b,c} e due output {1,0}.

$$I = \{a, b, c\}$$

$$O = \{0, 1\}$$



La macchina è correttamente definita? (giustifica la risposta)

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

È completamente definita + deterministica

- Cop. stati: $hbcc = TS1$

\Rightarrow output atteso = 0110

- Cop. transizioni: $bcahbcbcbccca = TS2$

\Rightarrow output atteso = 000101011110001

$S_0: \not\phi \not\phi \not\phi$ $S_1: \not\phi \not\phi \not\phi$ $S_2: \not\phi \not\phi \not\phi$ $S_3: \not\phi \not\phi \not\phi$

- Output error: $S_0 \rightarrow b/1 \rightarrow S_2$

\Rightarrow trova sia con TS1 che con TS2, infatti:

output: cop. stati = 1110 \neq TS1

cop. transizioni = 11. \neq TS2

- Transfer error: $S_0 \rightarrow a/1 \rightarrow S_2$

\Rightarrow non lo trova né con TS1 né con TS2, serve status