Tema esame inventato
Nuovo - 1

Teoria senza Eclipse:

1. Testing di un programma

Dato una classe in Java, scrivere i casi di test per ottenere la copertura (MCDC o quella richiesta) di un metodo. A differenza degli altri anni, il metodo non sarà statico e quindi potrà essere necessario costruire l'oggetto o cambiarne lo stato in modo opportuno.

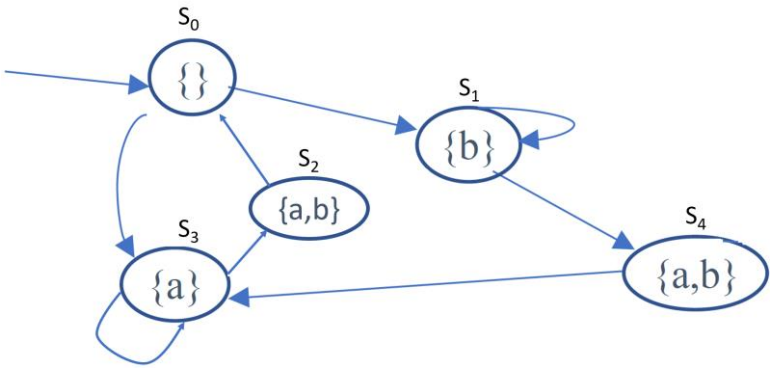
Dato il seguente codice che memorizza la quantità di pioggia negli ultimi 4 mesi:

```
public class Pioggia {

	int[] pioggia = { 0, 0, 0, 0 };

	boolean tropicale() {
		int somma = 0;
		for (int i : pioggia) {
			if (i < 10)
				return false;
			somma += i;
			if (somma > 30 && i >= 15)
				return true;
		}

		return false;
	}
}
```

Scrivere i casi di test (**simil JUNIT**) per avere la copertura delle **istruzioni**, **branch**, **MCDC** del metodo tropicale() che dice se si è verificato un monsone negli ultimi 4 mesi.

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 |= EX(a and b)
- M,s |= AF(a or b)

Nota che proprietà CTL potrebbero aver bisogno di essere trasformate per applicare l'algoritmo di model checking.

3. Combinatorial testing

Date tre variabili con i loro domini

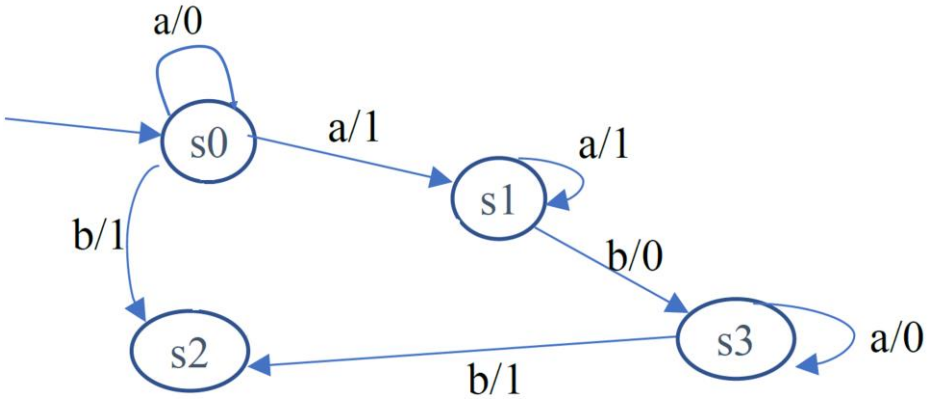
S: 1,2,3 **T**: A,B,C **V**:14,15,16

Costruisci la test suite combinatoriale pairwise usando l'algoritmo IPO. Tieni conto dei seguenti **vincoli**:

- 1 non può andare con C
- 3 non può andare con 16

4. Conformance testing

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



La macchina è corretta? (giustifica la risposta e correggi gli eventuali errori)
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 scoprire degli errori? Giustifica la risposta.

Con Eclipse:

Dato il seguente problema:
Un senso alternato ha un impianto semaforico che regola l'accesso. Ci sono due semafori alle due estremità A e B. Un semaforo può essere messo a verde solo se l'altro è rosso. Inoltre, un semaforo può essere messo a rosso solo se è giallo e a giallo solo se è verde. All'inizio i due semafori sono rossi.
L'impianto semaforico ha solo un comando che gli dice che semaforo dei due cambiare. L'impianto lo cambia solo se è possibile altrimenti lo lascia come è.

ATTENZIONE: il segnale mi dice ad ogni stato di quale semaforo devo cambiare il colore. Nessun cambiamento avviene in automatico. *Esempio*: se un semaforo è giallo e non dico di cambiare quel semaforo, il suo colore rimane a giallo.

5. Asmeta

Scrivi la specifica Asmeta del problema usando più costrutti possibili (ad esempio derivate, macro rule etc.).
Verificare le seguenti proprietà:

1. non accade mai che i semafori siano entrambi verdi
2. il semaforo 2 può diventare sempre verde (non solo allo stato iniziale)
3. se un semaforo è verde allora l'altro è rosso
4. se entrambi i semafori sono rossi e viene scelto il semaforo 1, nello stato successivo il semaforo 1 è verde
5. il semaforo 1 non può mai essere verde

Se c’è qualche proprietà che invece è giustamente falsa e il cui controesempio ti aiuta a capire come funziona, spiegalo. Scrivi almeno una proprietà che è giustamente falsa e controlla che il model checker trovi il contro esempio atteso.
Scrivi uno scenario Avalla di un caso significativo (fai un esempio in cui un semaforo fa un ciclo completo).

6. JML e KeY

Scrivi il codice in Java del problema implementato prima in Asmeta con in contratti opportuni. Cerca di scrivere sia le precondizioni, che le postcondizioni dei metodi. 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.

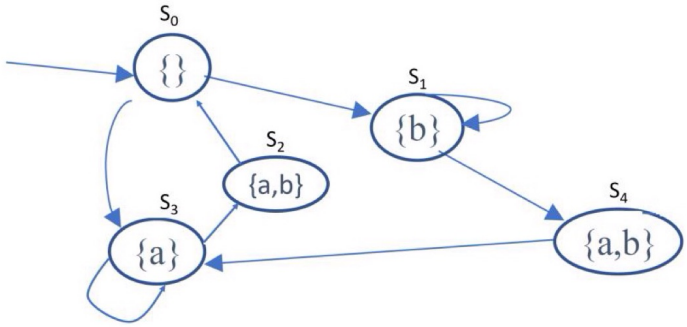
Dimostra con key che i contratti siano rispettati.

7. Model based testing

Traduci lo scenario avalla nel caso di test JUNIT per il codice java che hai scritto al punto 6.

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 |= EX(a and b)
- M,s |= AF(a or b)

1) $AG(a \wedge b) = \neg E[true \; U \; \neg(a \wedge b)]$

	S_0	S_1	S_2	S_3	S_4
a			x	x	x
b		x	x		x
$a \wedge b$			x		x
$\neg(a \wedge b)$	x	x		x	
$E[true \; U \; \neg(a \wedge b)]$	x	x	x	x	x
$\neg E[..]$					

2) $EX(a \wedge b)$

	S_0	S_1	S_2	S_3	S_4
a			x	x	x
b		x	x		x
$a \wedge b$			x		x
$EX(a \wedge b)$		x		x	

3) AF(a V b)

	S_0	S_1	S_2	S_3	S_4
a			x	x	x
b		x	x		x
a V b		x	x	x	x
AF(a V b)	x	x	x	x	x

3. Combinatorial testing

Date tre variabili con i loro domini

S: 1,2,3 **T**: A,B,C **V**:14,15,16

Costruisci la test suite combinatoriale pairwise usando l'algoritmo IPO. Tieni conto dei seguenti **vincoli**:

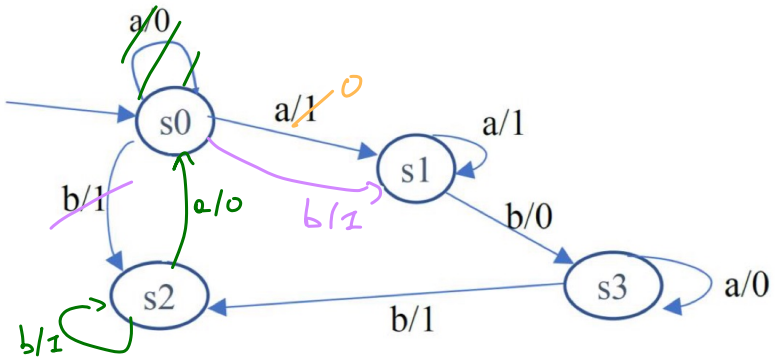
- 1 non può andare con C
- 3 non può andare con 16

S	T	V
1	A	14
1	B	15
~~1~~	~~C~~	~~16~~
2	A	15
2	B	16
2	C	14
~~3~~	~~A~~	~~16~~
3	B	14
3	C	15
1	B	16
3	B	16

ho dei vincoli ma devo comunque poter testare le coppie non incluse in quei casi ∅

4. Conformance testing

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



La macchina è corretta? (giustifica la risposta e correggi gli eventuali errori)

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 scoprire degli errori? Giustifica la risposta.

- Copertura stati: abba = ts1
  => output atteso: 1010
- Copertura transizioni: aabababb
  => output atteso: 11001101
- Output error: S_0 -> a/0 -> S_1
  => trovo sia con ts_1 che ts_2
- Transfer error: S_0 -> b/1 -> S_1
  => non trovo né con ts_1 né con ts_2 => serve considerare lo stato