Tema esame inventato
Nuovo - 2

Teoria senza Eclipse:

1. Testing di un programma

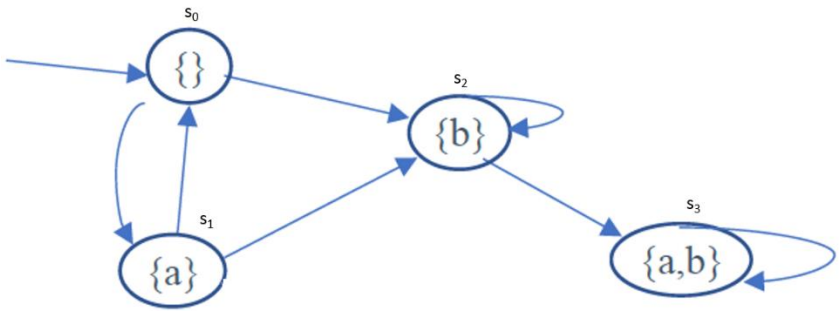
Dato il seguente codice:

```
public class conditionChecker{

	int[] a = {0,0,0,0};
	int c1 = 0;  // lower bound
	int c2 = 0; // upper bound

	boolean check() {

		if (c2<c1)
			return false;

		for (int i = 0; i < a.length; i++) {
			if (!(c1 <= a[i] && a[i] <= c2))
				return false;
		}

		return true;
	}
}
```

Scrivere i casi di test (**simil JUNIT**) per avere la copertura delle **istruzioni**, **branch**, **condizioni**, **MCDC** del metodo `check()` che controlla che tutti gli elementi contenuti nell'array a siano compresi tra c1 e c2 (con c1≤c2).

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)

- M,s |= EX(a or b)
- M,s |= AF(¬b)

- M,s |= AF(not 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

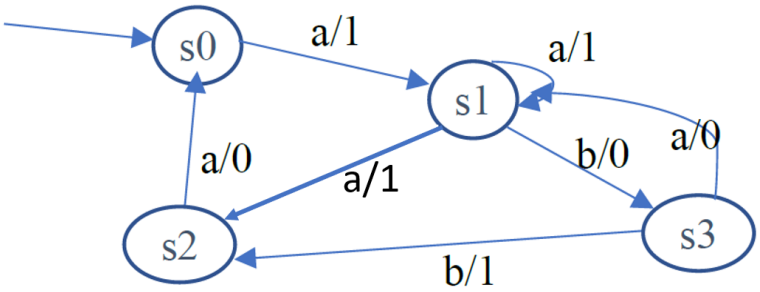
S: 1,2,3 **T**: a,b **V**:A,B,C,D

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

- 1 non può andare con C
- 2 e a non possono andare insieme con D

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:
Ci sono due giocatori (l'utente e il pc) che possono giocare un dado da 1 a 6. Ogni giocatore inizialmente ha 10€. Se un giocatore perde viene scalato 1€ e viene aggiunto all'avversario. Se tirano i dadi con lo stesso valore non viene scalato denaro. Il gioco continua finché uno dei due giocatori arriva a 0€.

In particolare:
Ad ogni passo di simulazione, il pc e l'utente scelgono il dado da giocare (entrambe **monitored** perché se scegliamo una variabile $pc tramite *choose rule* ci sono poi problemi nel settare il valore di $pc nelo scenario Avalla). Si possono verificare le seguenti condizioni:

- il dado tirato dall'utente è uguale a quello tirato dal pc: partita patta
- il dado tirato dall'utente è maggiore di quello tirato dal pc: vince l'utente
- il dado tirato dal pc è maggiore di quello tirato dall'utente: vince il pc

Inizialmente entrambi hanno un conto pari a 10€\ e viene modificato nel seguente modo:

- se l'utente vince: l'utente guadagna 1€, il pc perde 1€
- se l'utente perde: l'utente perde 1€, il pc guadagna 1€
- se è patta: nessuno guadagna/perde

Il gioco continua finché uno dei due giocatori raggiunge 0€. Quando uno dei due giocatori ha raggiunto 0€ si vuole sapere chi ha vinto il gioco (WINUSER | WINPC | PATTA) utilizzando una funzione controllata.

5. Asmeta

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

1. il saldo dell'utente può assumere un qualsiasi valore nell'intervallo[0, 20]€
2. nel sistema ci sono sempre 20€
3. esiste un cammino in cui il saldo dell'utente è sempre maggiore di 1€
4. in ogni caso prima o poi la partita finirà sempre

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 testi tutti e tre i possibili esiti del gioco).

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.

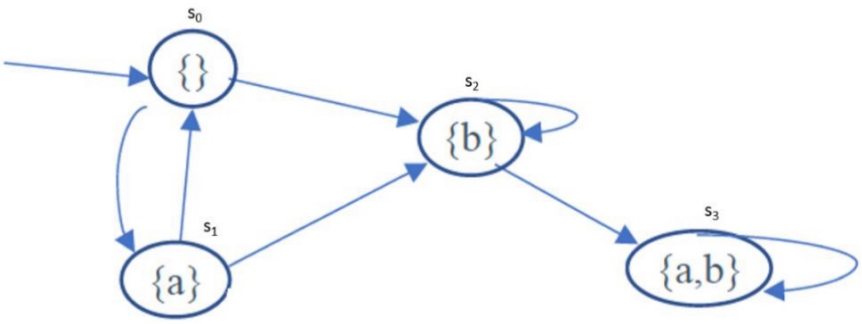
Dimostra con KeY che i contratti siano rispettati (NON USARE ENUMERATIVI).

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)

- M,s |= EX(a or b)
- $M, s_2 \models AF(\neg b)$

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

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

2) $EX(a \wedge b)$

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

3) $AF(a \vee b)$

	S_0	S_1	S_2	S_3
a		x		x
b			x	x
$a \vee b$		x	x	x
$AF(a \vee b)$	x	x	x	x

NB

4) $EX(a \vee b)$

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

NB

5) $AF(\neg b)$

	S_0	S_1	S_2	S_3
b			x	x
$\neg b$	x	x		
$AF(\neg b)$	x	x		

3. Combinatorial testing

Date tre variabili con i loro domini

S: 1,2,3 **T**: a,b **V**:A,B,C,D

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

- 1 non può andare con C
- 2 e a non possono andare insieme con D

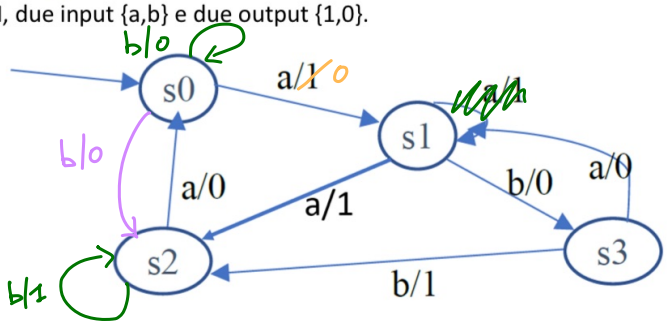
S	T	V
1	a	A
2	a	B
3	a	C
1	b	B
2	b	C
3	b	A
2	a	D
3	b	D
1		C
1	b	D
2	a	A
3	a	B

S-T ✓
T-V ✓
S-V ✓

non ho coppie non testate per via dei vincoli

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 : abb = TS1
  => output atteso : 101
- Copertura transizioni : abababbbaaabab = TS2
  => output atteso : 10001 0111100
- output error : S_0 -> a/0 -> S_1
  => trovo sia con TS1 che con TS2,
  infatti output : cop. stati = 001 ≠ TS1
  cop. transizioni = 0... ≠ TS2
- Transfer error : S_0 -> b/0 -> S_2
  => non trovo né con TS1 né con TS2, tutto stesso