Tema esame inventato
Nuovo - 4

Teoria senza Eclipse:

1. Testing di un programma

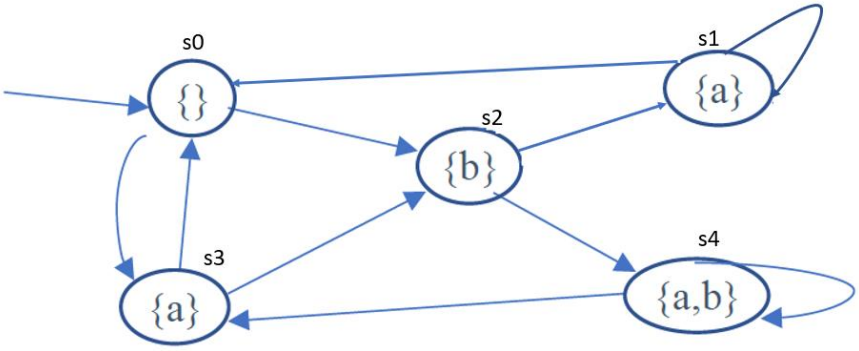
Dato il seguente codice:

```
public class Budget{
	int spese[] = {0,0,0,0};

	// non modificare questo valore:
	int soglia = 300;

	// budget restituisce:
	// - 1 se ho speso almeno 100€ in meno rispetto alla soglia
	// - 2 se ho speso esattamente uguale alla soglia
	// - 3 se ho speso più della soglia o non ho speso almeno 100€ in meno
	public int budget() {
		int sum=0;
		int res=3;

		for(int x : spese)
			sum+=x;

		int diff=soglia-sum;

		if (sum < soglia && diff>100)
			res=1;
		else if(sum==soglia || diff==0)
			res=2;

		return res;
	}
}
```

Scrivere i casi di test (**simil JUNIT**) per avere la copertura delle **istruzioni**, **branch**, **MCDC** del metodo check() che restituisco quanto scritto nel commento.

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(b)
- M,s |= EX(a and b)
- M,s |= AF(a or b)
- M,s |= E(T U ¬a)

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

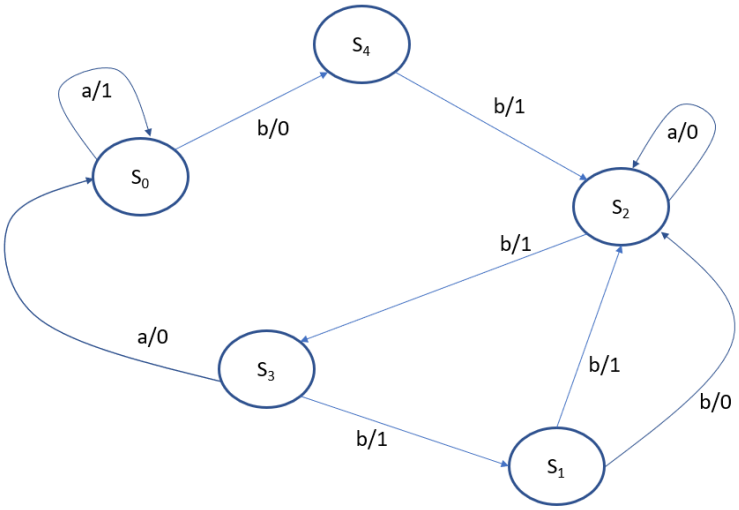
D1: A,B,C **D2**: 6,8 **D3**:L,M,N

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

- A non può andare con L e M
- B non può andare con 8

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:
Uno scommettitore ogni domenica esegue 5 scommesse sulle 5 partite di calcio di un campionato (assumi che il campionato ha solo 10 squadre). La vincita/perdita è regolata dalle seguenti regole:

1. se indovina il risultato (UNO, DUE o X), vince un euro;
2. se non indovina il risultato e c'è stato un pareggio (X), perde un euro;
3. se non indovina il risultato e una squadra ha vinto (UNO o DUE), perde due euro.

Lo scommettitore si è imposto delle regole per le sue scommesse:

- per ogni partita ha un budget separato, quindi la vincita/perdita di ogni scommessa incrementa/decrementa il budget riservato alla singola partita
- se arriva ad almeno 50 euro in totale allora si ferma
- se il budget di almeno una partita non permette di ripagare la perdita massima per quella partita, smette di scommettere
- Naturalmente se dovesse perdere tutto, si ferma.

All'inizio lo scommettitore ha riservato 5 euro per ogni partita.

5. Asmeta

Se si utilizza un *enumerativo*, mettere PARI al posto di X in Asmeta

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

Verificare le seguenti proprietà:

1. esiste un cammino in cui il giocatore arriva a 50
2. è possibile che sia azzerato il budget
3. non puoi mai perdere tutto

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 il gioco 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.

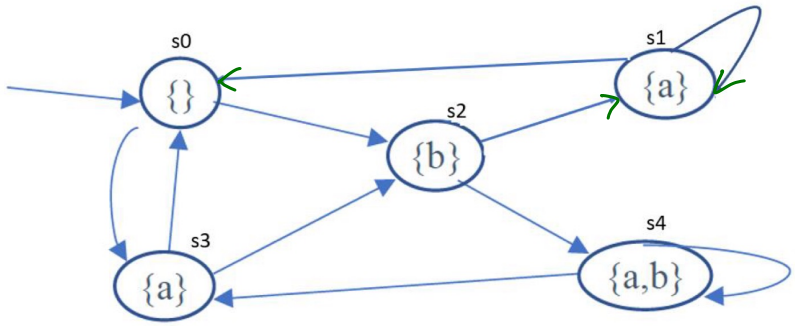
Dimostra con KeY che i contratti siano rispettati (NON USARE ENUMERATIVI).

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



1) $AG(b) = \neg E(true \cup \neg b)$

	S_0	S_1	S_2	S_3	S_4
b			x		x
$\neg b$	x	x		x	
$E(true \cup \neg b)$	x	x	x	x	x
$\neg E(\ldots)$					

2) $EX(a \wedge b)$

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

NB

3) AF(a ∨ b)

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

4) E(true U ¬a)

	S_0	S_1	S_2	S_3	S_4
a		x		x	x
¬a	x		x		
E(true U ¬a)	x	x	x	x	x

3. Combinatorial testing

Date tre variabili con i loro domini

D1: A,B,C **D2**: 6,8 **D3**:L,M,N

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

- A non può andare con L e M
- B non può andare con 8

D1	D2	D3
A	6	N
B	6	L
C	6	M
A	B	N

1) D1 - D2 ✓

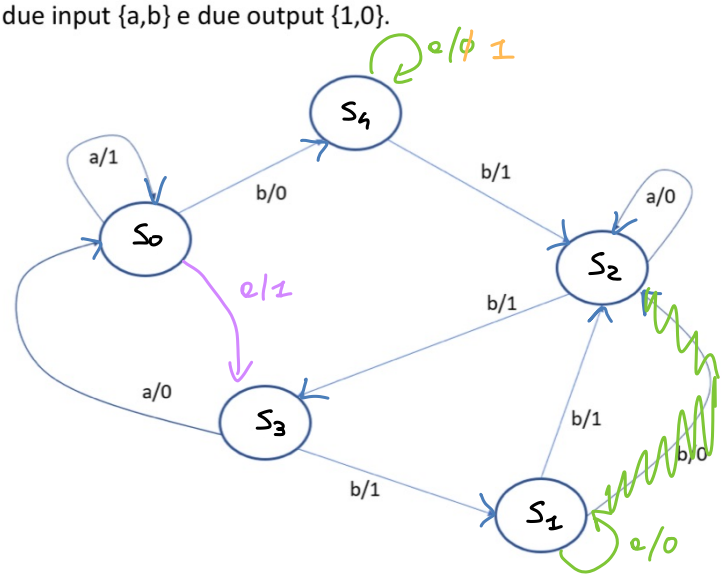
2) D1 - D3 ✓

D2 - D3 ✓

B	6	M	AL
C	8	L	AM
B	6	N	BB
C	6	N	
B	8	M	

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.

completamente definito + deterministica

- Copertura stati : bbbb = TS1
  => output atteso = 0111
- Cop. transizioni : babbbabbbaa = TS2
  => output atteso = 00101101101
- Output error : S_4 -> a/1 -> S_4
  => trovo con TS2, ma non con TS1, infatti
  output = 01... ≠ TS2
- Transfer error : S_0 -> a/1 -> S_3
  => non trovo né con TS1, né con T2, stesso stato