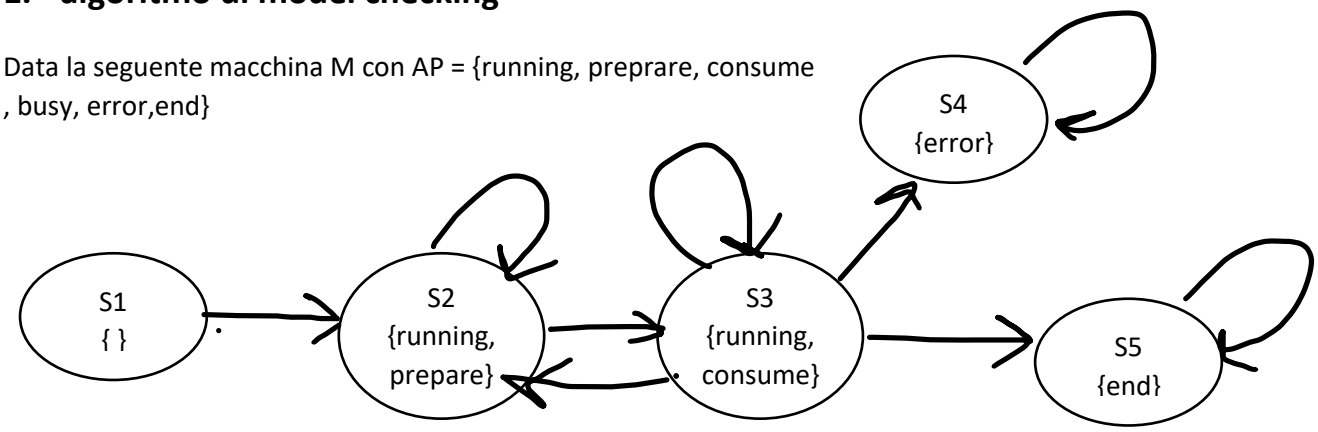
TEMA D'ESAME 13 NOVEMBRE 2023

COGNOME _____

Durata 3.30h

1. algoritmo di model checking

Data la seguente macchina M con AP = {running, preprare, consume , busy, error,end}



Mediante l'algoritmo di model checking trova in quali stati valgono queste proprietà.

1. M |= AG(running => (prepare or consume))
2. M |= error => AG(error)
3. M |= AG(EF(error))
4. M |= running => EX (not running)
5. M |= A(running U (end or error))
6. M |= EX(end or error)
7. M |= AG(error => not running)
8. M |= consume => EG(error)

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

Commenta il perché alcune proprietà valgono in alcuni stati e in altri no come ti aspetteresti. Dà anche una traduzione di ogni proprietà in linguaggio naturale

2. combinatorial testing

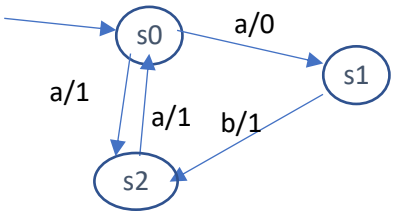
Date 3 variabili con i loro domini x: 1,2,3 y: a,b,c z: F,G

Costruisci la test suite combinatoriale pairwise usando l'algoritmo IPO (cerca di minimizzare la dimensione della test suite).

Se ci fosse il vincolo che le tuple [**x=3 e z = F]** e **[x=3 e z = G]** sono vietate, dovresti modificare la test suite? Se sì come, se no perché?

3. conformance testing

Data la seguente FSM con 3 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 modificandola come vuoi tu anche sul foglio)

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

2. Fa due esempi di errori (di quelli visti) e scrivi se possibile, per ognuno di essi, una test sequence che pur coprendo tutte le transizioni non li scopre (senza usare status message). Se aggiungi lo status, il test scopre l'errore? Se non è possibile trovare tale sequenza spiega perché.

Indica bene le test sequences come test di input e anche gli output attesi.

ESERCIZI CON ECLIPSE

Per questi esercizi prepara un file word in cui descrivi la tua soluzione.

Chiama ogni progetto COGNOME_NOME_XXX con XXX = Asmeta, JML, testing

4. Asmeta: progetto semaforo (semaforo.asm)

Scrivere un modello AsmetaL **Semafori** che modelli i semafori posti presso un attraversamento pedonale. Ci sono due semafori, uno per l'attraversamento dei pedoni e uno che regola il traffico delle macchine. Un pedone può prenotare il semaforo tramite un pulsante.

In funzionamento normale, uno dei due semafori è rosso e l'altro è verde. I semafori possono guastarsi e in tal caso lampeggiano entrambi. Per modellare il guasto, usa due modi e fai due versioni del modello (non usare mai seq):

1. i semafori si possono guastare con una probabilità del 50% (usa choose)
2. il guasto viene comunicato da un watchdog (boolean) che comunica con un segnale il guasto

Se i semafori non sono guasti:

- se l'utente segnala la volontà di attraversare tramite il pulsante, il semaforo delle macchine diventa (o rimane) rosso e quello dei pedoni verde dopo 3 secondi;
- se il pulsante non viene premuto, il semaforo delle macchine diventa dopo 3 secondi (o rimane) verde e quello dei pedoni rosso.

Ipotizza che ad ogni passo della macchina passi un secondo.

(le specifiche tralasciano alcuni dettagli, ad esempio cosa accade se uno preme quando il semaforo è già verde – fai tu delle ipotesi)-

All'inizio, il semaforo per le macchine è verde e quello per i pedoni è rosso (usa il term switch)

Prova a simulare e fai qualche screenshot dell'animatore della specifica 1.
Scrivi uno **scenario** della specifica 2 in cui il semaforo diventa lampeggiante

Proprietà CTL

Specificare tramite AsmetaSMV una proprietà temporale che verifichi che:

- esiste uno stato in cui i due semafori sono lampeggianti;
- esiste uno stato a partire dal quale inizia un cammino in cui di due semafori non sono mai lampeggianti;
- se il semaforo delle macchine è rosso quello dei pedoni deve essere verde.
- L'attesa del pedone non è mai superiore XXX secondi (quanti?)

Scrivi inoltre almeno una proprietà che è giustamente falsa e controlla che il model checker trovi il contro esempio atteso (puoi partire anche dalle proprietà sopra).

Riporta ogni proprietà nel file word e spiegale.

5. JML

Scrivi il modello sopra in JML.

La classe **Semaforo** ha

un metodo **step** che esegue un passo del semaforo,

un metodo privato **metticolore**(), che prende quale semaforo (pedone o auto) e che colore (verde, rosso, o lampeggiante) è da settare. Per questo usiamo gli interi: pedone 0, auto 1, verde 0 rosso 1 lampeggiante 2.

Scrivi i contratti con anche gli invarianti.

Scrivi un main con alcune istruzioni in modo da controllare che i contratti siano rispettati e che eventuali violazioni dei contratti vengano scoperte.

Prova i contratti con ESC.

6. Testing di programma

Elimina i contratti di JML e scrivi gli opportuni controlli al posto delle precondizioni.

Scrivi i casi di test JUNIT della classe **Semaforo** per ottenere la copertura

- Delle istruzioni
- Dei branch (solo quelli in più rispetto alle istruzioni)
- MCDC

Traduci lo scenario di Asmeta in un caso di test JUNIT.