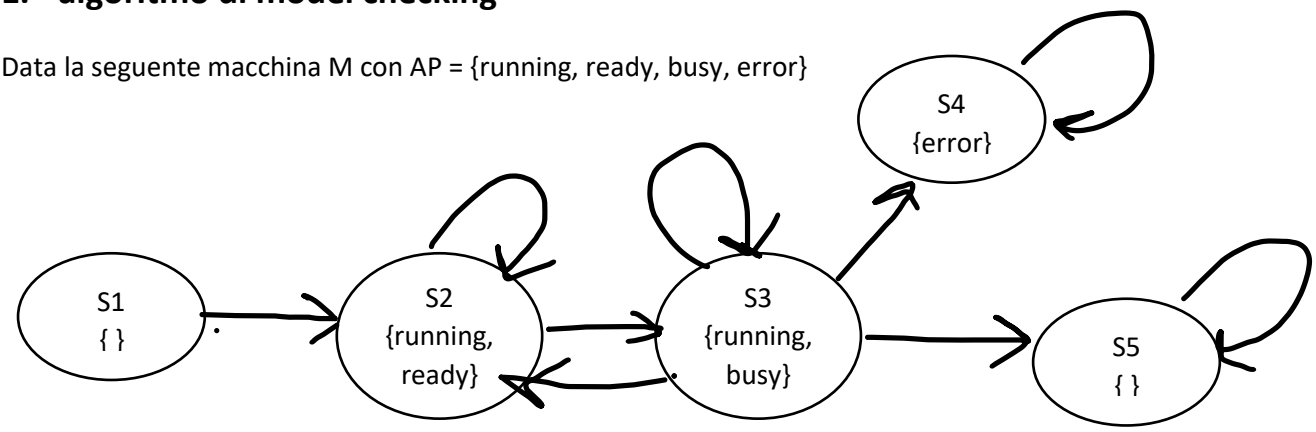
TEMA D'ESAME 14 LUGLIO 2023

COGNOME _____

Durata 3.30h

1. algoritmo di model checking

Data la seguente macchina M con AP = {running, ready, busy, error}



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

1. M |= AG(running => (ready or busy))
2. M |= AF(error)
3. M |= error => AG(error)
4. M |= running => EX (not running)
5. M |= AX EF (not running)
6. M |= E(ready or busy U not running)
7. M |= AG(error => not running)
8. M |= busy => 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. Da anche una traduzione di ogni proprietà in linguaggio naturale

2. combinatorial testing

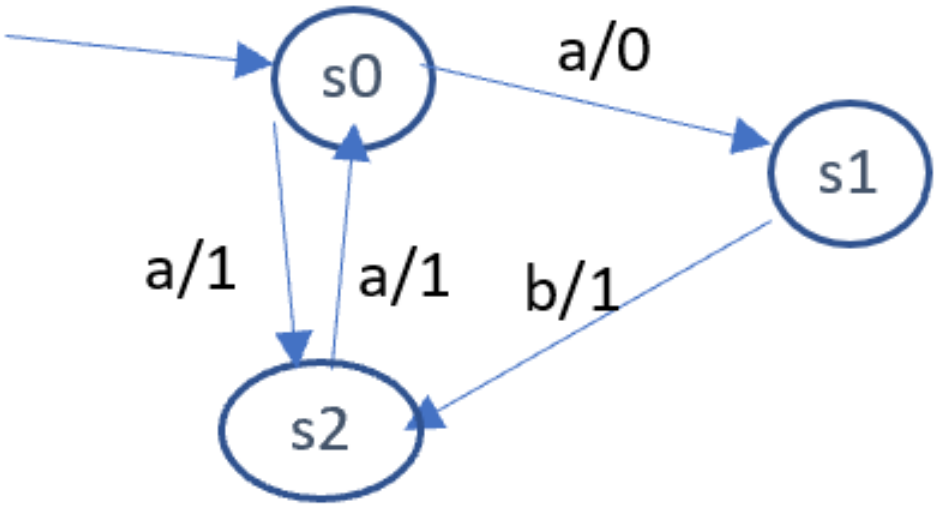
Date 3 variabili con i loro domini x: 1,2 y: a,b,c z: F,G,H

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=1 e z = F]** e **[x=1 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 freccette (darts.asm)

Scrivere un modello AsmetaL Darts che permetta a due utenti di giocare al gioco delle freccette. Consideriamo una versione modificata del gioco. Il bersaglio è diviso in due zone: RED è il centro del bersaglio e GREEN il resto del bersaglio. Identifichiamo con OUT la parete esterna al bersaglio.

Ci sono due giocatori (GIO1 e GIO2) (ma fai in modo che si possano aggiungere anche altri giocatori in futuro) che tirano una freccetta a testa (inizia GIO1).

Il giocatore sceglie una direzione (UP o DOWN) ed una velocità (FAST o SLOW) della freccetta.

- Se la velocità è SLOW, la freccetta non centra il bersaglio;
- se la velocità è FAST:
    - se la direzione è DOWN, la freccetta centra la parte GREEN del bersaglio;
    - se la direzione è UP, la freccetta può centrare la parte GREEN o la parte RED del bersaglio con la stessa probabilità.

Le vincite dei giocatori funzionano nel seguente modo. Se la freccetta colpisce la parte RED, il giocatore vince 2 euro e l'altro perde 2 euro; se colpisce la parte GREEN, il giocatore vince 1 euro ed l'altro perde 1 euro; se non colpisce il bersaglio, il giocatore perde 1 euro ed l'altro vince 1 euro.

Un giocatore non può andare mai sottozero, quindi se non ha credito sufficiente non può giocare.

Modello AsmetaL Il modello AsmetaL deve fare in modo che ogni passo di simulazione corrisponda ad un lancio della freccetta:

- l'utente deve scegliere una direzione ed una velocità;
- la zona colpita deve essere determinata secondo le regole descritte in precedenza;
    - Fai due versioni, una con il choose e una con un variabile monitorata per decidere cosa colpisce una freccetta FAST e UP
- a seconda della zona colpita, i soldi dei giocatori devono essere aggiornati secondo le regole descritte in precedenza.

All'inizio, entrambi i giocatori hanno 10 euro.

Non usare seq.

Prova a simulare e fai qualche screenshot dell'animatore.
Scrivi uno scenario in cui un giocatore vince tutto il banco (con la versione con choose o quella con la monitorata come preferisci).

Proprietà CTL:

Specificare e verificare tramite AsmetaSMV le seguenti proprietà CTL:

- esiste uno stato in futuro in cui un giocatore ha perso tutto
- esiste uno stato in cui il giocatore ha 9 euro e, dopo aver fatto il lancio, vince 2 euro;
- se il giocatore non ha più soldi, non li riguadagna più
- esiste un numero massimo (quale? X) per i soldi che un giocatore può avere.

Scrivi anche qualche altra proprietà che ti sembra significativa.

5. JML

Scrivi il modello sopra in JML generalizzando con 5 giocatori (cerca di scrivere in modo che si possa generalizzare facilmente a N giocatori).

La classe **Freccette** ha solo un metodo **gioca** che prende un intero (1 o 2) per indicare il giocatore e un intero per indicare la direzione e la velocità (1 per FAST e UP ad esempio e così via). Non usare enumerativi.

La classe memorizza i denari dei giocatori.

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 un programma

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

Scrivi i casi di test JUNIT della classe Freccette 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.