

TESTING E VERIFICA DEL SOFTWARE

ESERCITAZIONE ASMETA

COFFEE VENDING MACHINE

Modeling – Simulation – Animation – Validation – Verification

Modellazione

Utilizzare il framework ASMETA per definire una specifica che modelli un distributore di caffè definito come segue:

- Una macchinetta automatica dispensa caffè, tè e latte.
- La macchinetta accetta solo monete da 50 centesimi e da 1 euro.
- Se viene inserita una moneta da 50 centesimi, la macchinetta dispensa latte (se disponibile); se viene inserita una moneta da 1 euro, invece, la macchinetta decide in modo casuale di dispensare caffè o tè (se disponibili).
- Se viene dispensata una bevanda, la sua disponibilità viene decrementata e la moneta viene conservata nella macchinetta.
- Ogni passo di ASM deve corrispondere all'inserimento di una moneta e all'eventuale erogazione di una bevanda corrispondente.
- L'utente del sistema decide, ad ogni passo di simulazione, il tipo di moneta da inserire.
- La macchina all'inizio contiene 10 unità per ogni bevanda; l'atto di erogazione di una bevanda corrisponde alla diminuzione di un'unità della disponibilità della stessa e alla conservazione della moneta (nelle monete conservate, non bisogna distinguere tra monete da 50 centesimi ed 1 euro).
- Se la bevanda non è disponibile, non viene erogata e la moneta non viene conservata.
- La macchina può contenere al massimo 25 monete. Quando la macchina è piena di monete, non accetta altre monete e, quindi, non eroga più alcuna bevanda.
- All'inizio la macchinetta non contiene alcuna moneta.

Suggerimenti:

Creare un file con estensione .asm e importare la StandardLibrary. Il nome della asm deve essere uguale a quello del file:

```
asm coffeeVendingMachine
import StandardLibrary
```

La segnatura della macchina proposta è:

signature:

```
enum domain CoinType = {HALF | ONE}
enum domain Product = {...} // caffè e altri tipi
domain QuantityDomain subsetof Integer
domain CoinDomain subsetof Integer
controlled available: ... // funzione che per ogni prodotto
                           restituisce la quantità

controlled coins: CoinDomain
monitored insertedCoin: CoinType
```

Definire, oltre alla main rule, anche una macro rule di supporto:

definitions:

```
domain QuantityDomain = {...}
domain CoinDomain = {...}
rule r_serveProduct($p in Product) = ...
main rule r_Main = ... // utilizzare la choose rule
```

Definire lo stato iniziale:

```
default init s0:
...
```

Per fare il parsing della specifica che si sta sviluppando e verificare che non ci siano errori di lessico o sintassi premere sulla spunta verde:



Se la ASM è corretta, viene stampato a console il seguente output:

```
INFO - parsing file C:\path\to\your\project\models\coffeeVendingMachine.asm
INFO - file successfully parsed for asm coffeeVendingMachine
```

Simulazione e Animazione

Per capire se il modello è corretto dal punto di vista semantico, è possibile simularlo e animarlo.

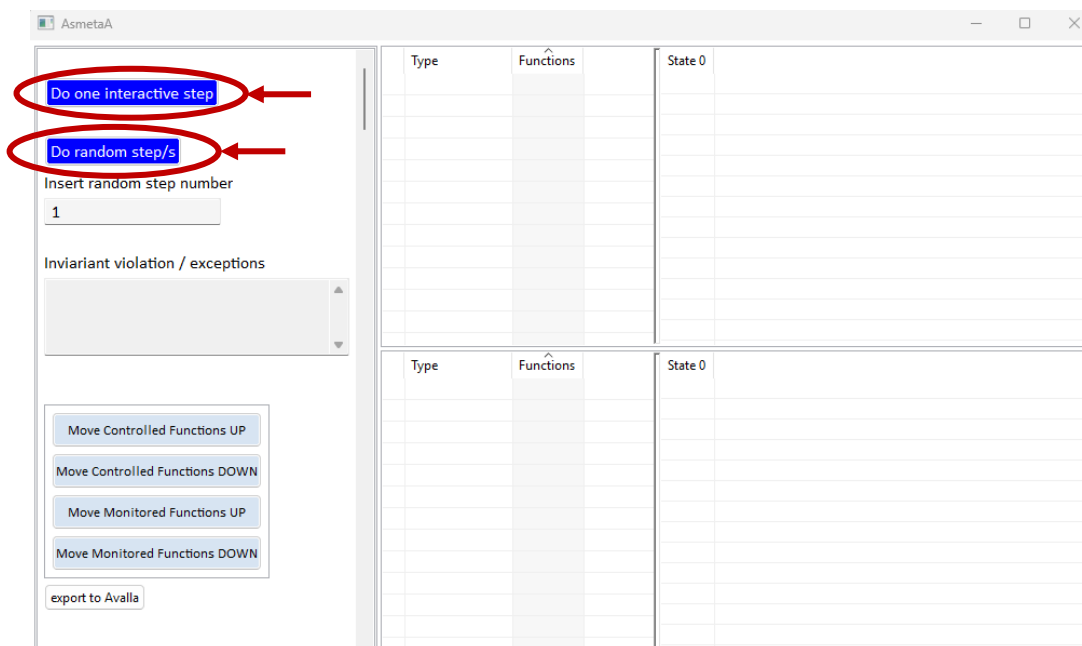
Simulare la macchina definita al punto precedente. La simulazione è testuale e può richiedere l'input (il valore delle funzioni controllate) all'utente o scegliere un valore dal dominio casualmente (simulazione random):



Animare la macchina definita al punto precedente. L'animatore permette di simulare graficamente l'esecuzione della asm:



Si può eseguire interattivamente uno step alla volta o eseguire più step casuali consecutivamente:



Validazione (stiamo realizzando il prodotto corretto?)

In ASMETA è possibile definire degli scenari nel linguaggio AVALLA e utilizzarli per validare la specifica. Potete pensare agli scenari come a dei casi di testi per codice (ad esempio metodi JUnit per codice Java). Gli scenari possono essere scritti da zero, ottenuti da una simulazione o generati automaticamente. Gli scenari devono essere definiti all'interno di un file con estensione .avalla.

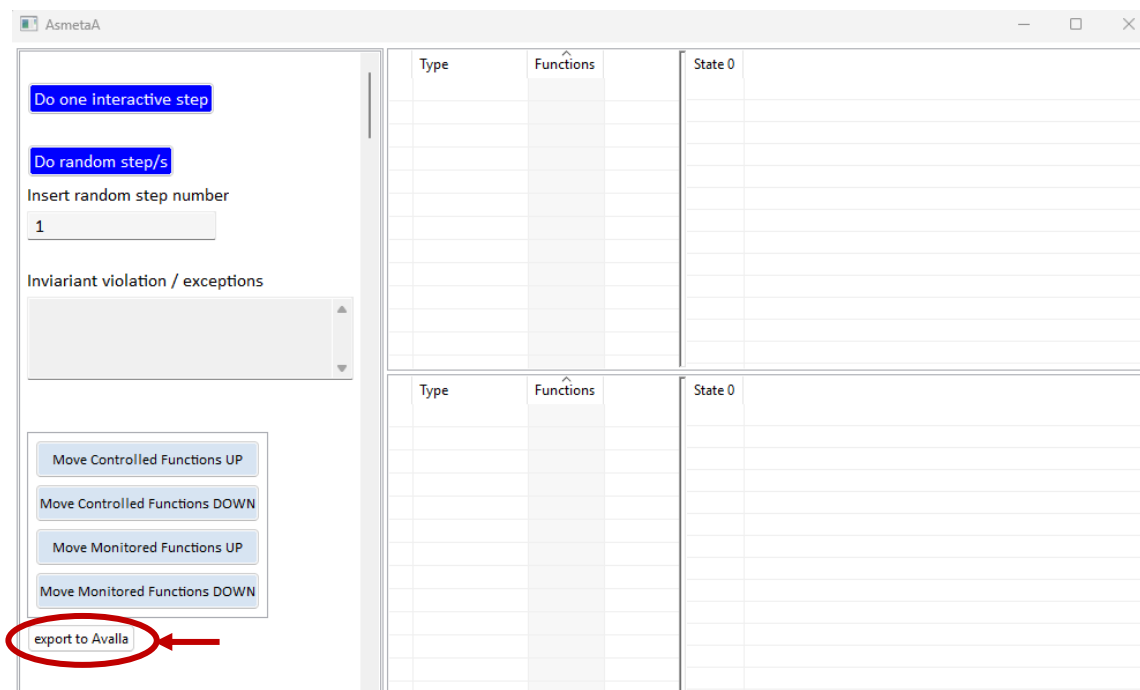
Definire da zero il seguente scenario:

- Controllare che all'inizio siano disponibili 10 unità per ogni bevanda;
- Inserire una moneta da 50 centesimi;
- Fare un passo di macchina;
- Controllare che venga erogato il latte, ossia che le unità disponibili siano 9 per il latte e 10 per le altre bevande;
- Inserire una moneta da 1 euro;
- Fare un passo di macchina;
- Controllare che venga erogato il tè **oppure** il caffè.

Definire da zero il seguente scenario:

- Controllare che all'inizio siano disponibili 10 unità per ogni bevanda;
- Inserire una moneta da 1 euro;
- Forzare l'erogazione del caffè (utilizzare il comando **pick**);
- Fare un passo di macchina;
- Controllare che venga erogato il caffè, ossia che le unità disponibili siano 9 per il caffè e 10 per le altre bevande;

Ottenere uno scenario (a vostra scelta) utilizzando l'animatore:



Una volta definiti gli scenari è possibile validarli premendo uno dei seguenti pulsanti:



Il primo pulsante esegue una semplice validazione che indica se qualche check è fallito. Il secondo pulsante mostra l'esecuzione sfruttando l'animatore, mentre il terzo calcola anche le coperture delle macro rule, delle update rule (update rule coverage) e delle conditional rule (branch coverage) ottenute dall'esecuzione dello scenario.

Che copertura è stata ottenuta dal primo scenario dell'esercizio?

Il validatore presenta anche feature più complesse, come i comandi **step until** e **exec** (vedi lezioni di teoria). È inoltre possibile utilizzare **ATGT** per generare automaticamente dei casi di test:

Tasto destro sul file .asm -> Run as... -> 1 ATGT test generator

Attenzione: con alcune specifiche (tipo quella utilizzata in questo esercizio) non funziona.

Verifica con Model Checker (*stiamo realizzando correttamente il prodotto?*)

Il framework ASMETA presenta un model checker che permette di verificare formalmente delle proprietà delle specifiche definite utilizzando la logica temporale (LTL o CTL).

Per utilizzare questa feature è necessario scaricare, installare e aggiungere alla variabile di sistema Path il model checker NuSMV (<http://nusmv.fbk.eu/>). Dopo averlo fatto, potrebbe essere necessario riavviare il computer. È inoltre necessario importare le librerie CTLLibrary.asm e LTLLibrary.asm utilizzando il comando import.

Scrivere le seguenti proprietà e verificarle:

- Una volta che un prodotto è terminato, non può più essere nuovamente disponibile in futuro.
- Nella macchinetta ci sono sempre almeno k unità di prodotto (quanto vale k?).
- Esiste uno stato in cui il latte e il tè sono terminati e ci sono ancora 9 caffè.

Scrivere una proprietà non vera e verificarla. Il model checker fornirà un controesempio che ne dimostra la falsità.

Per eseguire il model checker:

