



UNIVERSITÀ
DEGLI STUDI
DI BERGAMO

Dipartimento
di Ingegneria Gestionale,
dell'Informazione e della Produzione

AsmetaV

AVaLLa (Asmeta Validation Language)

Contatti:

Prof. Angelo Gargantini – angelo.gargantini@unibg.it

Dott. Andrea Bombarda – andrea.bombarda@unibg.it

Scenario ASM: primitive Avalla

- **Set:** comando per impostare il valore delle monitorate a un valore specifico. Simula l'ambiente
- **Check:** per verificare il valore delle funzioni dello stato corrente
- **Step:** esegue un passo dell'ASM
- **StepUntil:** esegue l'ASM finchè una condizione risulta vera
- **Invariant:** per verificare che una proprietà è sempre vera durante l'esecuzione dello scenario
- **Exec:** esegue una regola di transizione

NB: nel path NON devono esserci spazi!!!



Sintassi di Avalla

Abstract syntax	Concrete syntax
Scenario	<code>scenario name</code> <code>load spec_name</code> <code>Invariant* Command*</code> spec_name is the spec to load; invariants and commands are the script content
Invariant	<code>invariant name ':' expr ';' </code> expr is a boolean term made of function and domain symbols of the underlying ASM
Command	<code>(Set Exec Step StepUntil Check)</code>
Set	<code>set loc := value ';' </code> loc is a location term for a monitored function, and value is a term denoting a possible value for the underlying location
Exec	<code>exec rule ';' </code> rule is an ASM rule (e.g. a choose/forall rule, a conditional if, a macro call rule, ect.)
Step	<code>step ';' </code>
StepUntil	<code>step until cond ';' </code> cond is a boolean-valued term made of function and domain symbols of the ASM
Check	<code>check expr ';' </code> expr is a boolean-valued term made of function and domain symbols of the ASM

Advanced Clock



**UNIVERSITÀ
DEGLI STUDI
DI BERGAMO**

Dipartimento
di Ingegneria Gestionale,
dell'Informazione e della Produzione

Advanced Clock (1)

Modellare il funzionamento di un Clock che ad ogni passo incrementa di 1 i secondi.



Advanced Clock (2)

Scenario 1:

- Controllare che all'inizio sia mezzanotte (00:00:00);
- Fare un passo di macchina;
- Controllare che l'ora sia 00:00:01;
- Fare un passo di macchina;
- Controllare che l'ora sia 00:00:02.



Advanced Clock (3)

Scenario 2:

- eseguire la macchina fino a quando la funzione hours non vale 3;
- Controllare che l'ora sia 03:00:00;
- Fare un passo di macchina;
- Controllare che l'ora sia 03:00:01.



Advanced Clock (4)

Scenario 3:

- Impostare un invariante di scenario che afferma che l'ora è sempre minore di 3;
- Impostare una par rule che, con tre update rule, modifica l'ora in 02:01:59;
- Fare un passo di macchina;
- Controllare che l'ora sia 02:02:00;
- Impostare una par rule che, con tre update rule, modifica l'ora in 00:00:00.

Scenario 3 modificato:

- Modificare lo scenario 3 per fare in modo che l'invariante di scenario sia violato.



ATM



**UNIVERSITÀ
DEGLI STUDI
DI BERGAMO**

Dipartimento
di Ingegneria Gestionale,
dell'Informazione e della Produzione

ATM (1)

Sviluppare, e mostrare che funziona correttamente, il controllo per un ATM in cui tramite una GUI il cliente può svolgere le seguenti operazioni:

- **Op1.** Inserisce l'ID. Solo un tentativo per sessione è ammesso. Se l'inserimento è sbagliato, la tessera è espulsa.
- **Op2.** Richiesta del saldo del conto. Questa operazione è permessa solo una volta e solamente prima che si provi a prelevare dei soldi.
- **Op3.** Prelievo di soldi dal conto. Solo un tentativo è ammesso per singola sessione. Un Warning viene segnalato se l'importo richiesto è maggiore del saldo



ATM (2)

Dato il modello ASM scrivere i seguenti scenari

Scenario 1:

- Un utente inserisce la carta card1 ed il pin corretto;
- Preleva 50 euro;
- Viene disconnesso;

Dopo ogni passo controllare che lo stato atmState sia corretto.



ATM (3)

Scenario 2:

- Scrivere uno scenario in cui un utente inserisce il pin errato e gli viene negato l'accesso.
- Controllare sempre che lo stato atmState sia aggiornato in modo corretto.



ATM (4)

Scenario 3:

- impostare il saldo di card1 a 10 euro;
- autenticare card1;
- provare a prelevare 50 euro;
- successivamente provare a prelevare 10 euro

Dopo ogni passo controllare che lo stato atmState sia corretto; controllare anche che il saldo di card1 ed i soldi disponibili nel bancomat (moneyLeft) siano aggiornati in modo corretto.



ATM (5)

Scenario 4:

- Impostare un invariante di scenario che afferma che la somma disponibile nel bancomat non è mai inferiore a 900.
- Eseguire una serie di prelievi, facendo in modo che l'invariante non sia mai violato.



Sluice Gate



**UNIVERSITÀ
DEGLI STUDI
DI BERGAMO**

Dipartimento
di Ingegneria Gestionale,
dell'Informazione e della Produzione

Sluice Gate (1)

Una piccola chiusa, con un cancello in grado di aprirsi e chiudersi, è utilizzata in un semplice sistema di irrigazione. Un sistema informatico viene utilizzato per controllarla:

- E' richiesto che il cancello rimanga in posizione FULLY OPEN per 10 minuti ogni 3 ore, e tenuto FULLY CLOSED per la restante parte del tempo.
- Il cancello viene aperto e chiuso ruotando delle viti verticali. Le viti sono mosse da un piccolo motore che può essere controllato con segnali ON, OFF, CLOCKWISE, ANTICLOCKWISE



Sluice Gate (2)

Scenario 1:

- Scrivere uno scenario in cui si mostri il ciclo di vita completo della chiusa: FULLYCLOSED – OPENING – FULLYOPENED – CLOSING – FULLYCLOSED.



Sluice Gate (3)

Scenario 2:

- Modificare lo scenario 1 per fare in modo che il passaggio da uno stato a quello successivo della chiusa avvenga in due passi di macchina. Ogni passaggio di stato della chiusa è condizionato al valore di una determinata locazione monitorata (ad esempio si passa da FULLYCLOSED a OPENING se `passed(closedPeriod)` è true). Se la monitorata è false, invece, la chiusa non cambia stato.



Algoritmo di Euclide



**UNIVERSITÀ
DEGLI STUDI
DI BERGAMO**

Dipartimento
di Ingegneria Gestionale,
dell'Informazione e della Produzione

Algoritmo di Euclide (1)

Creare un'ASM in AsmetaL che implementi l'algoritmo di Euclide nel modo seguente:

- Ogni step della macchina deve corrispondere ad un'iterazione del ciclo While;
- I valori di cui bisogna calcolare l'MCD devono essere impostati nello stato iniziale ($\text{numA} = 6409$ e $\text{numB} = 3289$);

Modificare il modello visto in precedenza per fare in modo che, in simulazione, venga richiesto all'utente il valore dei due numeri su cui eseguire l'MCD.



Algoritmo di Euclide (2)

Scenario 1:

- Eseguire la macchina fino a quando $\text{numA} = \text{numB}$;
- Controllare che il valore di numA sia 13



Algoritmo di Euclide (3)

Scenario 2:

- Scrivere un invariante che afferma che numA è sempre maggiore di numB;
- Modificare i valori di numA e numB, rispettivamente, a 26 e 24;
- Eseguire la macchina fino a quando numA = numB;
- Controllare che il valore di numA sia 2.



Ferryman problem



Ferryman problem (1)

Un ferryman deve portare sull'altra sponda di un fiume un wold, una goat ed un cabbage, e può trasportarne solo uno per volta.

Ci sono due situazioni di pericolo:

- Il wolf mangia la goat se il ferryman non è presente a controllare;
- La goat mangia il cabbage se il ferryman non è presente a controllare;

All'inizio sono tutti sulla sponda LEFT. In simulazione, ad ogni passo, bisogna decidere chi deve essere trasportato sull'altra sponda dal FERRYMAN: la GOAT, il CABBAGE, il WOLF oppure fare attraversare il fiume con nessuno a bordo (NONE)



Ferryman problem (2)

Scenario 1:

- Scrivere uno scenario in cui si mostrino tutti i passi necessari (settaggio della monitorata «carry» con il comando set ed esecuzione del passo con il comando step) a trasportare tutti gli attori sulla sponda destra.

Dopo ogni passo controllare che le posizioni degli attori siano quelle attese (comando check).



Ferryman problem (3)

Scenario 2:

- Posizionare il ferryman, la goat ed il wolf sulla sponda destra;
- Eseguire tutti i passi che servono per trasportare tutti gli attori sulla sponda destra.

Dopo ogni passo controllare che le posizioni degli attori siano quelle attese (comando check).

