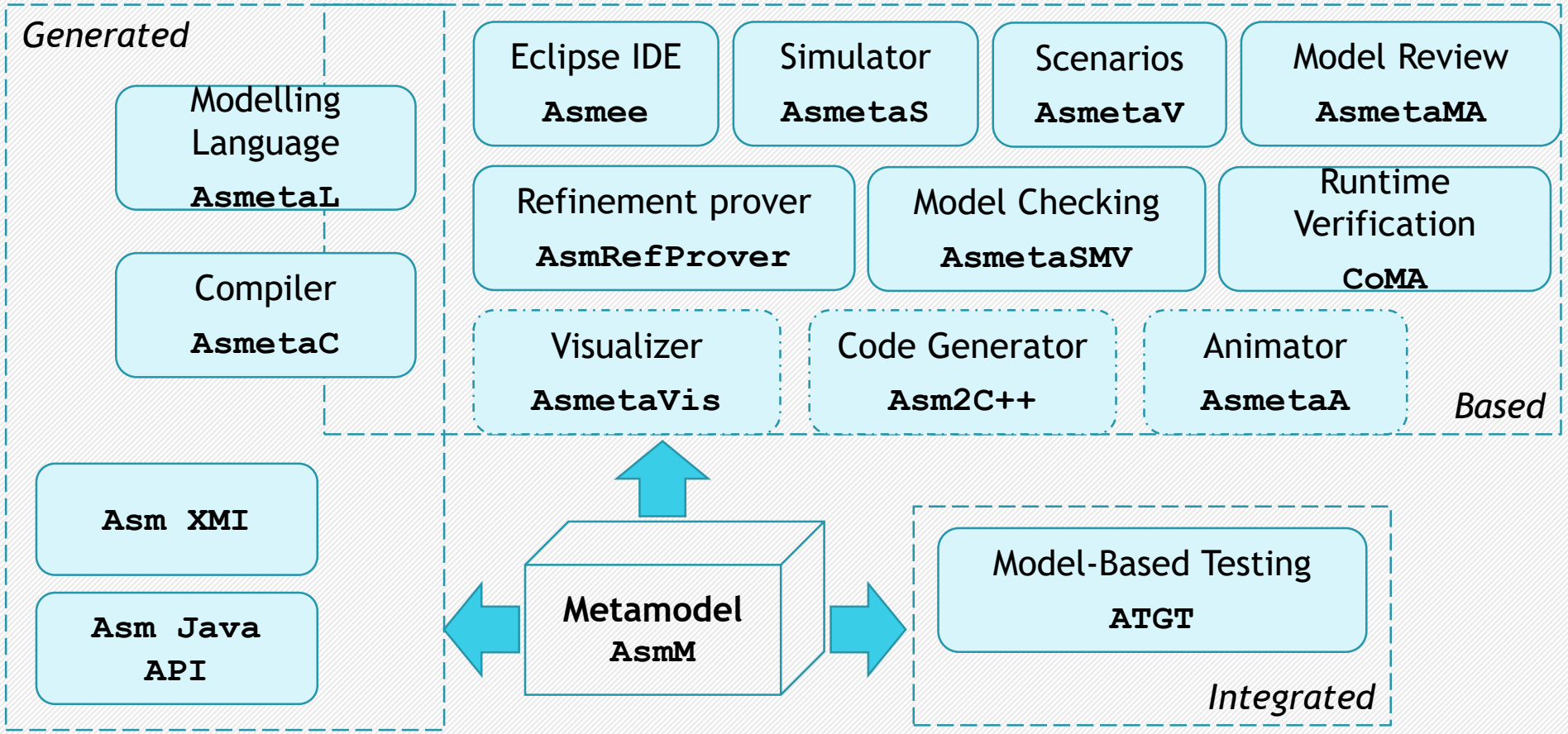
ASMETA

AsmetaLc – AsmetaS - AsmetaA – AsmetaL - AsmetaV

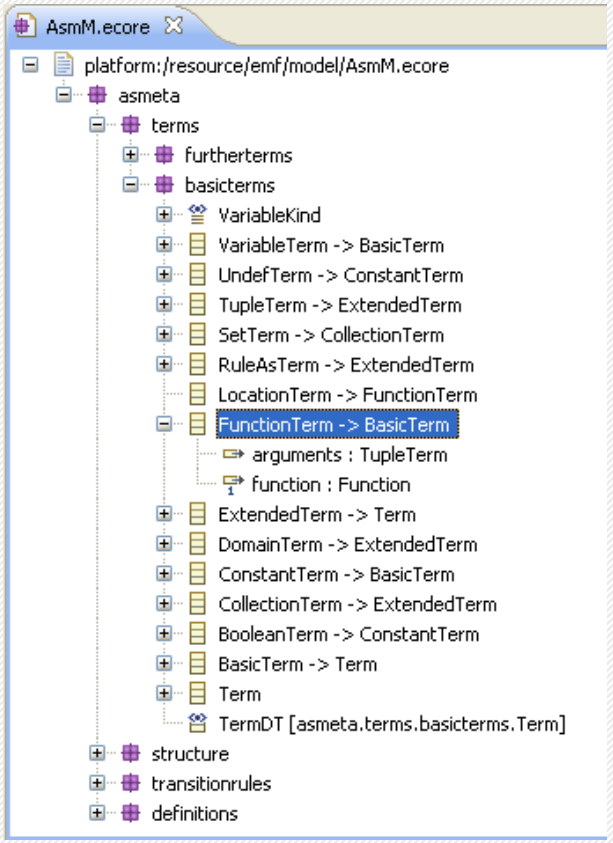
ASMETA

ASMETA


Generated
Modelling Language
AsmetaL
Compiler
AsmetaC
Asm XMI
Asm Java API
Eclipse IDE
Asmee
Simulator
AsmetaS
Scenarios
AsmetaV
Model Review
AsmetaMA
Refinement prover
AsmRefProver
Model Checking
AsmetaSMV
Runtime Verification
CoMA
Visualizer
AsmetaVis
Code Generator
Asm2C++
Animator
AsmetaA
Based
Metamodel
AsmM
Model-Based Testing
ATGT
Integrated

ASMETA

- Il toolset ASMETA è stato sviluppato partendo dalla definizione di AsmM, un metamodello per le ASMs. AsmM e stato definito utilizzando l'Eclipse Modeling Framework (EMF).



ASMETA tools

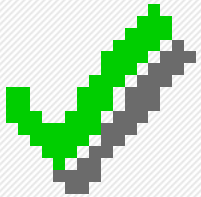
- **AsmetaL**, una sintassi concreta che permette di scrivere modelli ASMs in formato testuale;
- **AsmetaLc**, un compilatore text-to-model che permette di effettuare il parsing di modelli AsmetaL e controllare che rispettino i vincoli imposti dal metamodello AsmM (espressi come vincoli OCL);
- **AsmetaS**: un simulatore per eseguire modelli ASMs;
- **Avalla**: un linguaggio per la scrittura di scenari; gli scenari Avalla sono interpretati ed eseguiti dal tool AsmetaV;
- **ATGT**: un tool per la generazione di casi di test basato sul model checker SPIN;
- **ASMEE** (ASM Eclipse Environment): un front-end graco per la scrittura di modelli AsmetaL;
- **AsmetaSMV**: un model checker, basato su NuSMV, di modelli AsmetaL.
- **AsmetaVis**: visualizza con un grafo i modelli ASM
- **Asm2C++**: traduce i modelli ASM in linguaggio C++
- **AsmetaA**: animatore dei modelli ASM.

ASMETA

- Tutto il codice e rilasciato sotto licenza GPL.
- È ospitato su Sourceforge e può essere scaricato con qualsiasi client svn da: http://svn.code.sf.net/p/asmeta/code/
- È possibile navigare nel repository a partire da questo indirizzo: http://sourceforge.net/p/asmeta/code/HEAD/tree/
- La StandardLibrary può essere scaricata da: http://tinyurl.com/StandardLibrary
- Potete scaricare il tutto dal marketplace di Eclipse (cerca Asmeta)

AsmetaLc

Parser

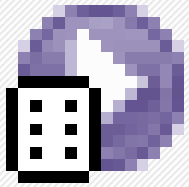


- Il parser AsmetaLc (AsmetaL Compiler) permette di processare specifiche AsmetaL e controllarne la consistenza rispetto ai vincoli OCL del metamodello.

AsmetaS

Opzioni di esecuzione del simulatore

- Run ASM interactively: le variabili monitorate sono inserite nella console



- Run ASM randomly: le variabili monitorate sono scelte casualmente dal simulatore

Il simulatore invoca automaticamente il parser prima di simulare l'ASM

AsmetaA

Opzioni di esecuzione dell'animatore

- Run ASM interactively: le variabili monitorate sono inserite nelle box

- Run ASM randomly: le variabili monitorate sono scelte casualmente

L'animatore si basa sul simulatore

Esercizi

Algoritmo di Euclide

Morra cinese

Macchina del caffè

Algoritmo di Euclide

```
Function MCD (a,b)
        while a!=b
                if a>b
                        a:=a-b
                else
                        b:=b-a
        return a
```

Algoritmo di Euclide

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

Coffee vending machine

- Una macchinetta automatica dispensa caffe, te 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.

Coffee vending machine

- 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 unita per ogni bevanda; l'atto di erogazione di una bevanda corrisponde alla diminuzione di un'unita della disponibilita 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 e disponibile, non viene erogata e la moneta non viene conservata.
- La macchina puo contenere al massimo 25 monete; quando la macchina e piena di monete, non accetta altre monete e, quindi, non eroga piu alcuna bevanda.
- All'inizio la macchinetta non contiene alcuna moneta.

Morra cinese

Lo scopo del gioco e quello di sconfiggere l'avversario scegliendo un segno (arma) in grado di battere quello dell'altro, secondo le seguenti regole:

1. Il sasso spezza le forbici (vince il sasso)
2. Le forbici tagliano la carta (vincono le forbici)
3. La carta avvolge il sasso (vince la carta)

- Se i due giocatori scelgono la stessa arma, il gioco è pari e si gioca di nuovo.

Morra cinese

Creare un modello ASM che permetta di giocare a Morra cinese contro il computer. Il modello deve avere le seguenti caratteristiche:

- ogni step della macchina corrisponde ad una singola giocata;
- l'utente deve avere la possibilità di scegliere uno dei tre simboli;
- il computer deve scegliere in modo casuale uno dei tre simboli;
- la macchina deve valutare la giocata e segnalare il vincitore.

Morra cinese

Tre possibili raffinamenti successivi del modello Asm possono essere:

1. Memorizzare il numero di partite vinte dall'utente e dal computer;
2. Impostare il numero massimo di partite che devono essere giocate;
3. Se, dopo aver giocato il numero massimo di partite, il computer e l'utente sono in parità, permettere che continuino a giocare fino a quando uno dei due non vince.

AsmetaV

Scenario ASM: primitive Avalla

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

Esercizi

Advanced Clock

ATM

Advanced clock

Dato il modello ASM scrivere i seguenti scenari:

- 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

- 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

- 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, modica 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, modica 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

Dato il modello ASM scrivere i seguenti scenari:

- Scenario 1:
  - un utente inserisce la carta card1 ed il pin corretto;
  - preleva 50 euro;
  - viene disconesso;
  - dopo ogni passo controllare che lo stato atmState sia corretto.

ATM

- 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

- 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

- Scenario 4:
  - Impostare un invariante di scenario che afferma che la somma disponibile nel bancomat non e mai inferiore a 900 euro;
  - eseguire una serie di prelievi, facendo in modo che l'invariante non sia mai violato.

Esercizi riepilogativi

Scrivi il modello e poi simula gli scenari

Sluice gate

- A small sluice, with a rising and falling gate, is used in a simple irrigation system. A computer system is needed to control the sluice gate: the requirement is that the gate should be held in the fully open position for ten minutes in every three hours and otherwise kept in the fully closed position. The gate is opened and closed by rotating vertical screws. The screws are driven by a small motor, which can be controlled by clockwise, anticlockwise, on and off pulses.

Sluice gate

- Scenario 1
  - Scrivere uno scenario in cui si mostri il ciclo di vita completo della chiusa: FULLYCLOSED - OPENING - FULLYOPENED - CLOSING - FULLYCLOSED.
- 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 e condizionato al valore di una determinata locazione monitorata (es. si passa da FULLYCLOSED a OPENING se passed(closedPeriod) e true): se la monitorata vale false la chiusa non cambia stato.

Ferryman

- Un ferryman deve portare sull'altra sponda di un fiume un wolf, una goat ed un cabbage. Puo' trasportarne solo uno alla volta. Ci sono due situazioni di pericolo: il wolf mangia la goat se il ferryman non e' presente a controllare; la goat mangia il cabbage se il ferryman non e' 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

- Scenario 1
  - Scrivere uno scenario in cui si mostrino tutti i passi necessari (settaggio della monitorata carry con il comando set ed esecuzionedel 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).
- Scenario 2
  - Posizionare il ferryman, la goat ed il wolf sulla sponda destra; I eseguire tutti i passi che servono per trasportare tutti gli attori sulla sponda destra.