Computer Automated Multi-Paradigm Modelling: Meta-Modelling and Graph Transformation

Hans Vangheluwe

School of Computer Science, McGill University, Montréal, Canada

Juan de Lara

E.T.S. de Informática, Universidad Autónoma de Madrid, Madrid, Spain
Modelling and Simulation Wishlist

- Meaningful exchange and re-use of models
  Syntax \textit{and} Semantics!

- Domain/problem-specific (visual) modelling & simulation environments
  Syntax \textit{and} Semantics!

- Model transformation
  - simulation (state changes)
  - code-generation (syntax changes)
  - simplification (level of abstraction changes)
  - formalism transformation (formalism changes)

- Meaningful multi-formalism modelling
CAMPaM
Computer Automated Multi-Paradigm Modelling

1. Different *levels of abstraction*

2. Mixing *different formalisms* (coupling, transformation)

3. Modelling classes of models (formalisms) by *meta-modelling*

4. Modelling *transformations* explicitly
   (in the Graph Grammar formalism) for
   - Operational Semantics (reference simulator)
   - Code generation
   - Denotational Semantics

TOMACS 12(4) 2002: Special Issue on CAMPaM.
Mosterman and Vangheluwe.
Meta-Model For Timed Automata

Generates

A Timed Automata Model of a Traffic Light

Transform to Code

Simulation
What is Meta-modelling?

- A meta-model is **a model of** a modelling formalism.

- A meta-model is itself a *model*. Its syntax and semantics are governed by the formalism it is described in. That formalism can be modelled in a meta-meta-model.

- As a meta-model is a model, we can reason about it, manipulate it, ... In particular, properties of (all models in) a formalism can be formally proven.

- Formalism-specific modelling and simulation tools can *automatically* be generated from a meta-model (*e.g.*, in AToM$^3$ A Tool for Multi-formalism Meta-Modelling).
• Formalisms can be *tailored* to specific needs by modifying the meta-model (possibly through inheritance if specializing).

  ⇒ Building domain/application specific, possibly graphical modelling and simulation environments becomes affordable.

• Semantics of new formalisms through extension or transformation.
Model of Timed Automata Abstract Syntax (in ER)
Model of Timed Automata Concrete Visual Syntax
ER formalism + constraints (OCL/Python)

```python
# check for unique input labels (FSA)
for transition1 in state.out_connections:
    for transition2 in state.out_connections:
        if transition1 != transition2:
            if transition1.in == transition2.in:
                return("Non-determinism: input " + transition1.in)
```
Meta-Model For Timed Automata

Generates

A Timed Automata Model of a Traffic Light

Simulation

Transformation

Code

Wintersim 2003, 8 December, New Orleans
hv@cs.mcgill.ca
CAMPaM: Meta-Modelling and Graph Transformation
A valid model in the Timed Automata formalism
ER model of the ER formalism (meta-meta-model)
Meta-meta...
Meta-Model For Timed Automata
Generates
A Timed Automata Model of a Traffic Light
Model Transformation meta-specification

- **Meta-model**: a model in formalism ER
- **Transformation meta-model**: a model of a class of models (the formalism NFA) semantics within formalism ER
- **Meta-model**: a model in formalism MF
- **Transformation meta-model**: a model of a class of models (the formalism F) semantics within formalism MF describes: structure and constraints

- **Model transformer**
- **Meta-model processor**
  - User input: create, delete, verify (local, global)
  - A model in formalism NFA
  - A model in formalism FSA
  - A model in formalism MF

Wintersim 2003, 8 December, New Orleans
hv@cs.mcgill.ca
CAMPaM: Meta-Modelling and Graph Transformation
Model Transformation Uses (1)

- Operational Semantics (reference simulator)
- Code generation
- Denotational Semantics

May model transformation in the Graph Grammar formalism
Formalism transformation uses (2)

- Add new formalisms without much effort (only Δ).
- Re-use lower level modelling/simulation environment.
- Answer questions at “optimal” level.
- Optimization possible at every level.
- Semantics of coupled multi-formalism models.
Operational Semantics of TA: Rewrite Rule 1

CONDITION: node(6).ArrivalTime > node(4).Time + node(3).Time

TIME += node(3).Time
Last Trans = TIME
Operational Semantics of TA: Rewrite Rule 2

\[
\text{TA\_MoveTimeTransSelf}
\]

\[
\text{Priority 2}
\]

CONDITION: \(\text{node(5).ArrivalTime} > \text{node(3).Time} + \text{node(2).Time}\)

\[
\text{TIME} += \text{node(2).Time}
\]

\[
\text{Last Trans} = \text{TIME}
\]
Operational Semantics of TA: Rewrite Rule 3

\[ \text{CONDITION: } \text{node(6).Value == node(3).Value} \]

\[ \text{TIME: } \text{node(6).ArrivalTime} \]
\[ \text{Last Trans: } \text{TIME} \]

\[ \text{TIME = node(6).ArrivalTime} \]
\[ \text{Last Trans = TIME} \]
Operational Semantics of TA: Rewrite Rule 4

\[ \text{TA\_MoveTransSelf} \]

\[ \text{Priority 4} \]

CONDITION: node(5).Value == node(2).Value

\[ \text{TIME} = \text{node}(5).\text{ArrivalTime} \]

\[ \text{Last Trans} = \text{TIME} \]
Meta-Model For Timed Automata

Generates

A Timed Automata Model of a Traffic Light

Simulation

Transformation/Code
Simulation Trace, initial state

Wintersim 2003, 8 December, New Orleans
hv@cs.mcgill.ca
CAMPaM: Meta-Modelling and Graph Transformation
Simulation Trace, step 1
Simulation Trace, step 2

TIME: 10.0
Last Trans.: 10.0

End
Simulation Trace, step 3
Simulation Trace, step 4
Simulation Trace, step 5
Simulation Trace, step 6
Simulation Trace, step 7
Simulation Trace, step 8

TIME: 100.0
Last Trans.: 100.0

End

TIME: 100.0
Last Trans.: 100.0
Simulation Trace, step 9
Meta-Model For Timed Automata

Generates

A Timed Automata Model of a Traffic Light
Equivalent Timed Transition Petri Net model
Meta-Model For Timed Automata

Generates

A Timed Automata Model of a Traffic Light

Simulation

Transformation

Code

Wintersim 2003, 8 December, New Orleans
hv@cs.mcgill.ca
CAMPaM: Meta-Modelling and Graph Transformation
Generated Application
Traffic Light model in the Statechart Formalism
Conclusion: EVERYTHING IS A MODEL!

- **Meta-model** a formalism’s abstract and concrete syntax
- Graph Grammars *models* for all model Transformations
- Variations (flavours) of formalisms (syntax and semantics)
- Simulator (reference implementation)
- A Meta-modelling Environment (ATOM³)

Petri Net model of Producer Consumer

reachability analysis + simplification + simulation + code generation
Reachability Graph for Producer Consumer

- **Produce**
  - [P.Calculating, Wait4Prod, Buffer→p]
  - [Wait4Cons, Wait4Prod, Buffer→p]
  - [Buffer, P.Calculating, Wait4Prod]
  - [Wait4Cons, Buffer, Wait4Prod]
  - [C.Calculating, Wait4Cons, Buffer→p]
  - [C.Calculating, P.Calculating, Buffer→p]

- **Put in Buffer**
  - [Buffer, P.Calculating, Wait4Prod]
  - [Wait4Cons, Buffer, Wait4Prod]
  - [C.Calculating, P.Calculating, Buffer→p]

- **Rem from buffer**
  - [C.Calculating, Wait4Cons, Buffer→p]
  - [C.Calculating, P.Calculating, Buffer→p]

- **Consume**
  - [C.Calculating, Wait4Cons, Buffer]
  - [C.Calculating, P.Calculating]
  - [C.Calculating]
GPSS model of a Manufacturing system
Generated GPSS code

* Manufacturing shop model 4
* G. Gordon Figure 11-9/9-8

SIMULATE
L0 GENERATE 5 ; Create parts
L7 QUEUE INSPQ ; Queue for an inspector
L5 ENTER INSPECTR,1 ; A single inspector becomes busy
L8 DEPART INSPQ ; Leave the inspector queue
L9 MARK ; Start counting transit time
L1 ADVANCE 12,9 ; Inspect
L6 LEAVE INSPECTR,1 ; Make the inspector idle again
L10 TABULATE TRANSIT ; Tabulate parts’ transit time
L2 TRANSFER .1,ACC,REJ ; Randomly determine defective parts
ACC TERMINATE 1 ; Accepted parts
REJ TERMINATE 1 ; Rejected parts

TRANSIT TABLE M1,5,5,10
INSPECTR STORAGE 3

START 1000
END
Forrester System Dynamics model of Predator-Prey

2–species predator–prey system

transformation to Ordinary Differential Equations + analysis
Causal Block Diagram model of Harmonic Oscillator

analysis + simplification + simulation
Event Scheduling + DAE model of a Train

code generation
Formalism Transformation Graph

- **PDE**
- **ODE**
- **DAE non-causal set**
- **DAE causal set**
- **DAE causal sequence (sorted)**
- **Causal Block Diagram**
- **System Dynamics**
- **Bond Graph a-causal**
- **Bond Graph causal**
- **Cellular Automata**
- **Statecharts**
- **Petri Nets**
- **Transfer Function**
- **Activity Scanning**
- **Discrete Event**
- **Timed Automata**
- **Event Scheduling**
- **Discrete Event**
- **3 Phase Approach**
- **Discrete Event**
- **DEVS & DESS**
- **scheduling-hybrid-DAE**
- **Difference Equations**
- state trajectory data (observation frame)