Computer Aided Control Systems Design (CACSD)

Taipei, Taiwan. 2 September 2004

Meta-Modelling Hybrid Formalisms

Simon Lacoste-Julien [†], Hans Vangheluwe

School of Computer Science, McGill University, Montréal, Canada

Juan de Lara

E.T.S. de Informática, Universidad Autonóma de Madrid, Madrid, Spain

Pieter J. Mosterman

Simulation and Real-Time Technologies, The MathWorks, Inc., Natick, MA, USA

[†] Simon is currently a Ph.D. student at UC Berkeley

Presentation Overview

1. Domain-specific Modelling
2. HS=ES+ODE, a Hybrid Formalism
   - A simple hybrid model
   - Simulating the model
3. Meta-modelling HS
4. Model transformation
5. Conclusions

Domain-specific Modelling

- Match modellers' mental model op the problem domain
- Maximally constrain users
- Exploit domain-specific features
  for analysis, synthesis, ...

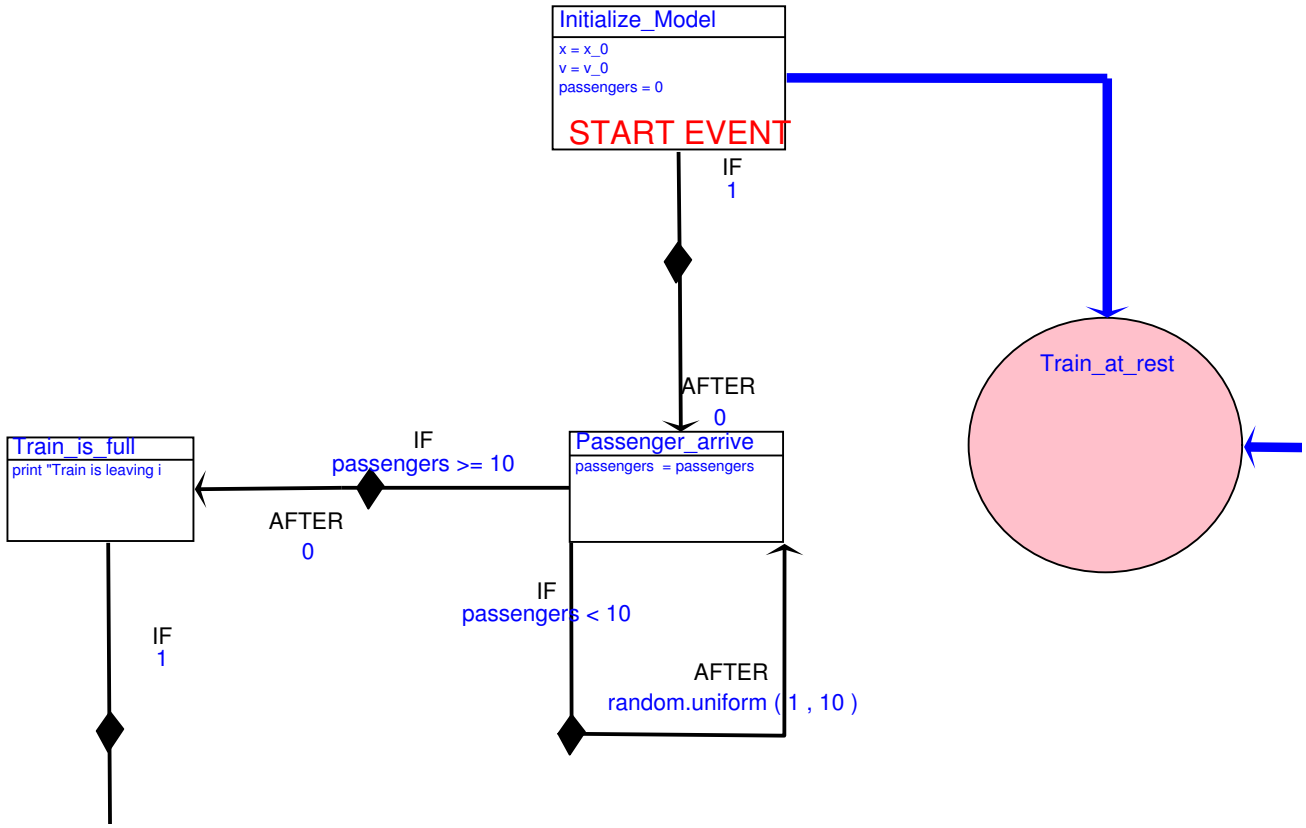
HS=ES+ODE, a Hybrid Formalism

- To demonstrate the ease with which an HS-specific modelling environment can be built
- HS combines
  - Event Scheduling (ES)
    to describe discrete-event behaviour
  - Ordinary Differential Equations (ODEs)
    to describe continuous-time behaviour
  - Constructs to link ES and ODE

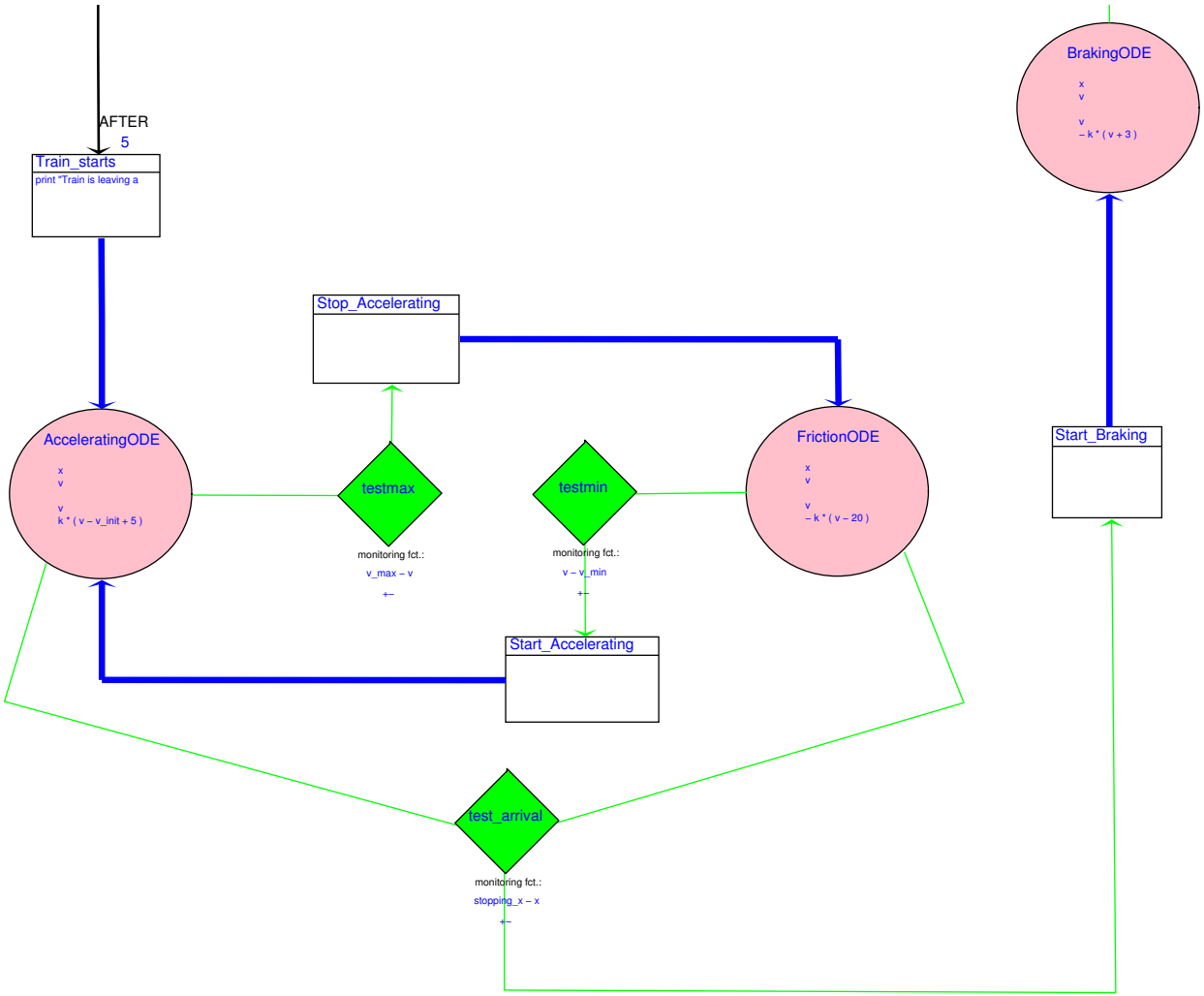
Example: Personalized Rapid Transit (PRT)

- Discrete-event behaviour: loading and unloading passengers
- Continuous-time behaviour: accelerating, decelerating
- Control: keep velocity between boundaries
  by switching between continuous “modes”

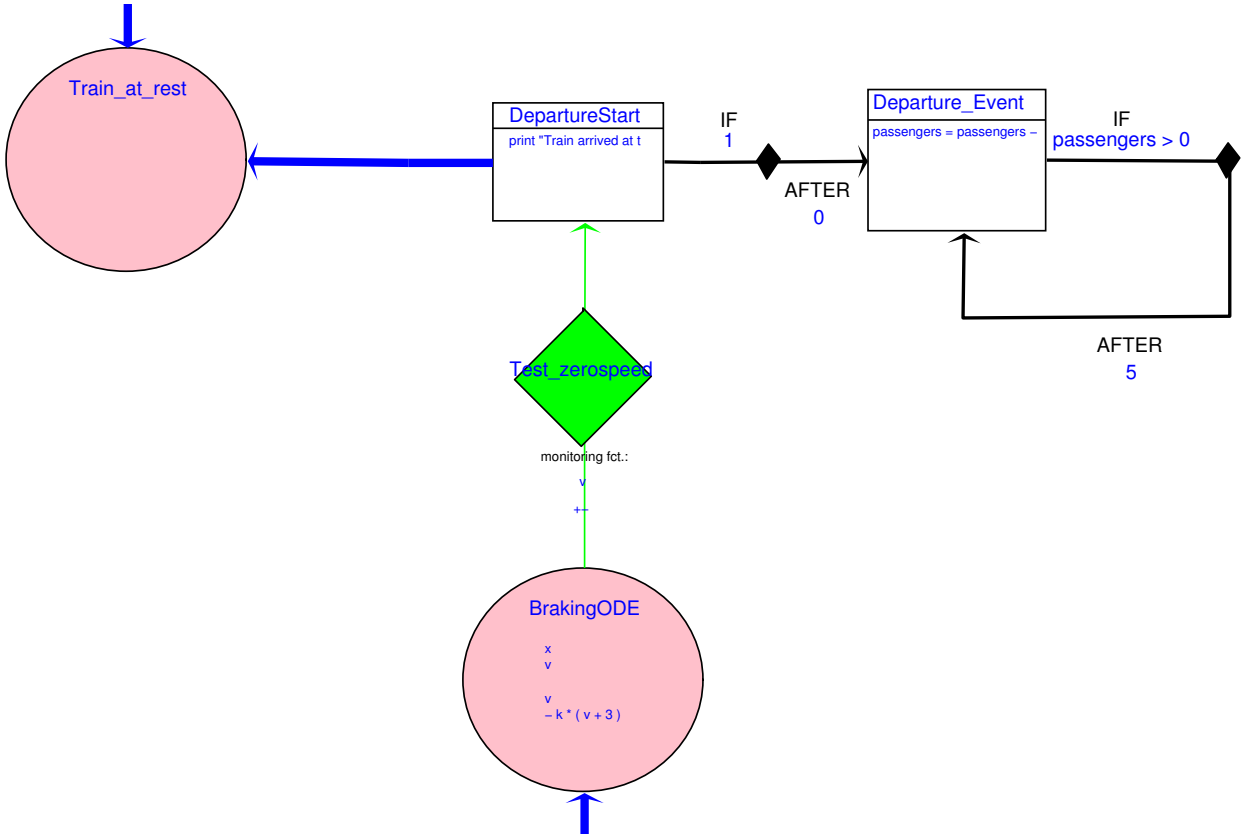
Passenger Arrival



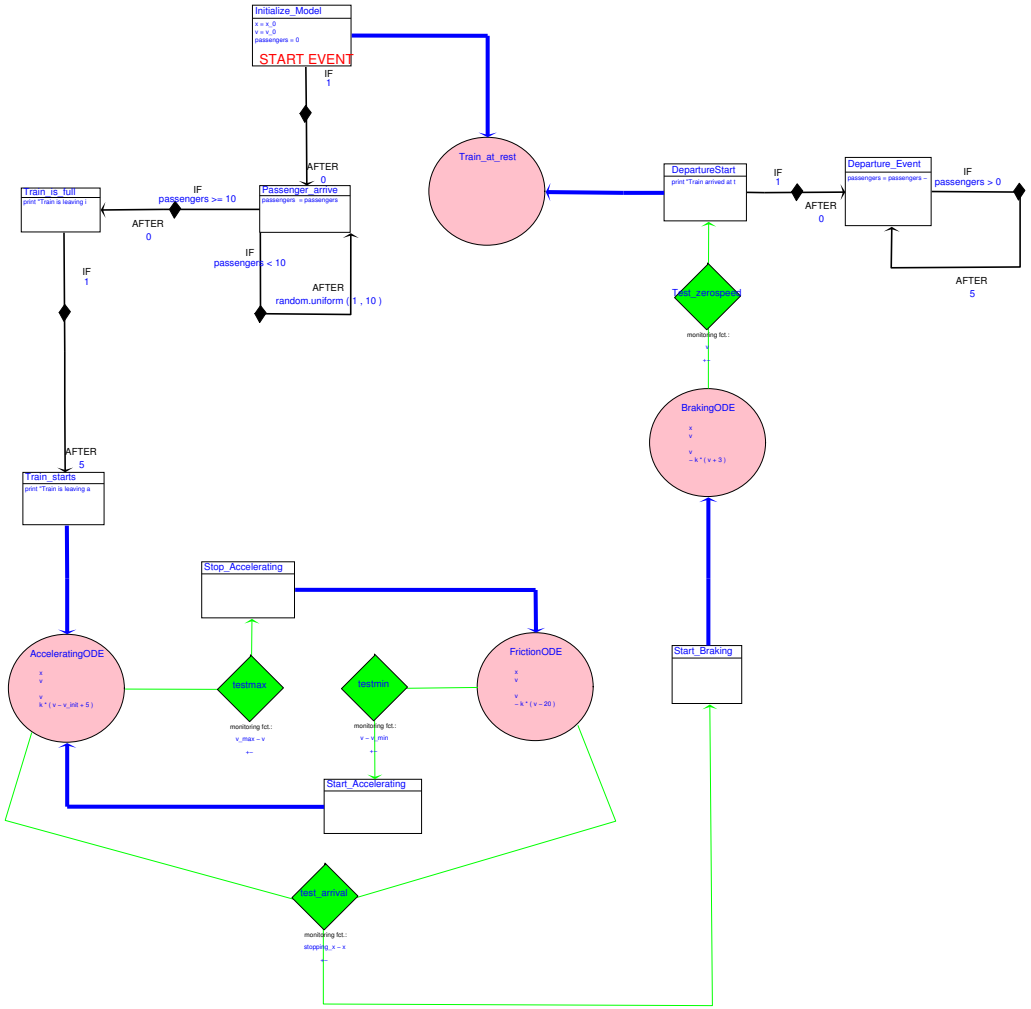
Train Motion



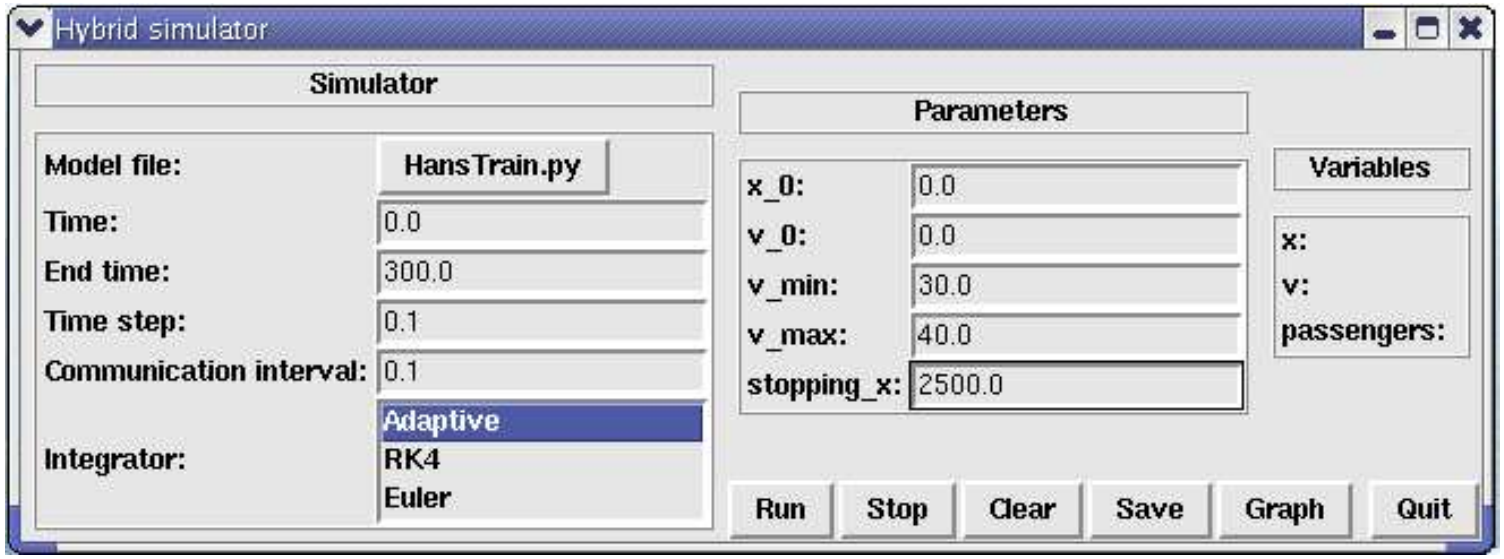
Train Arrival and Unloading



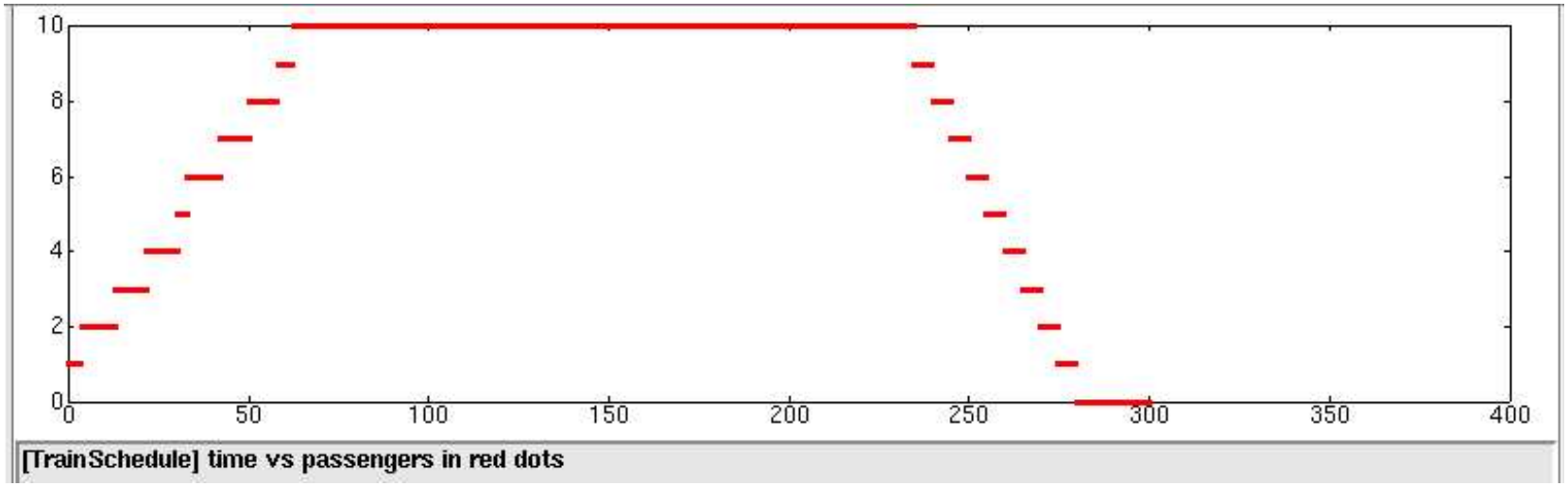
The Complete Model



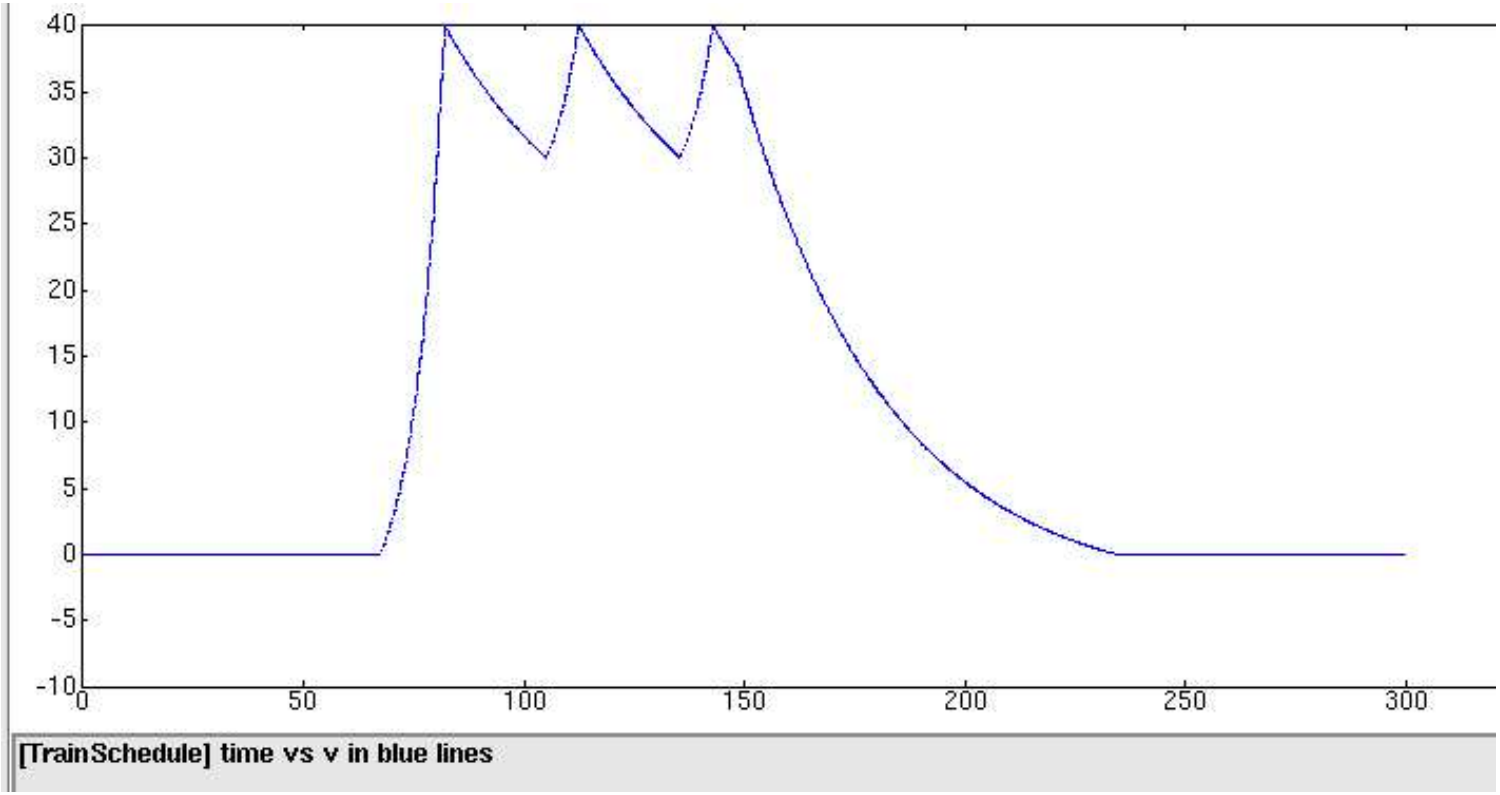
Simulating the (compiled) Model



Train Passengers



Train Velocity

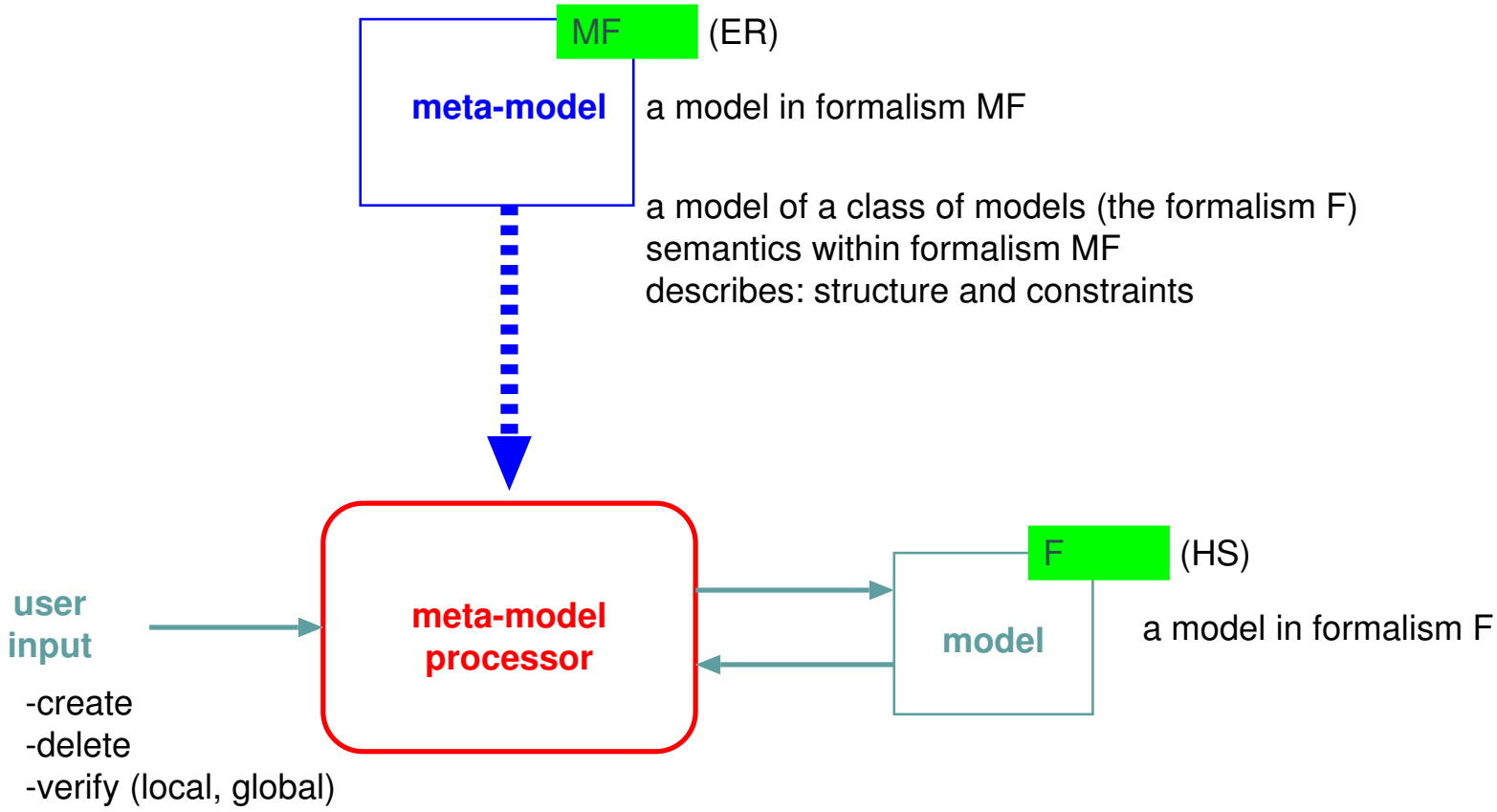


Meta-modelling (the HS formalism)

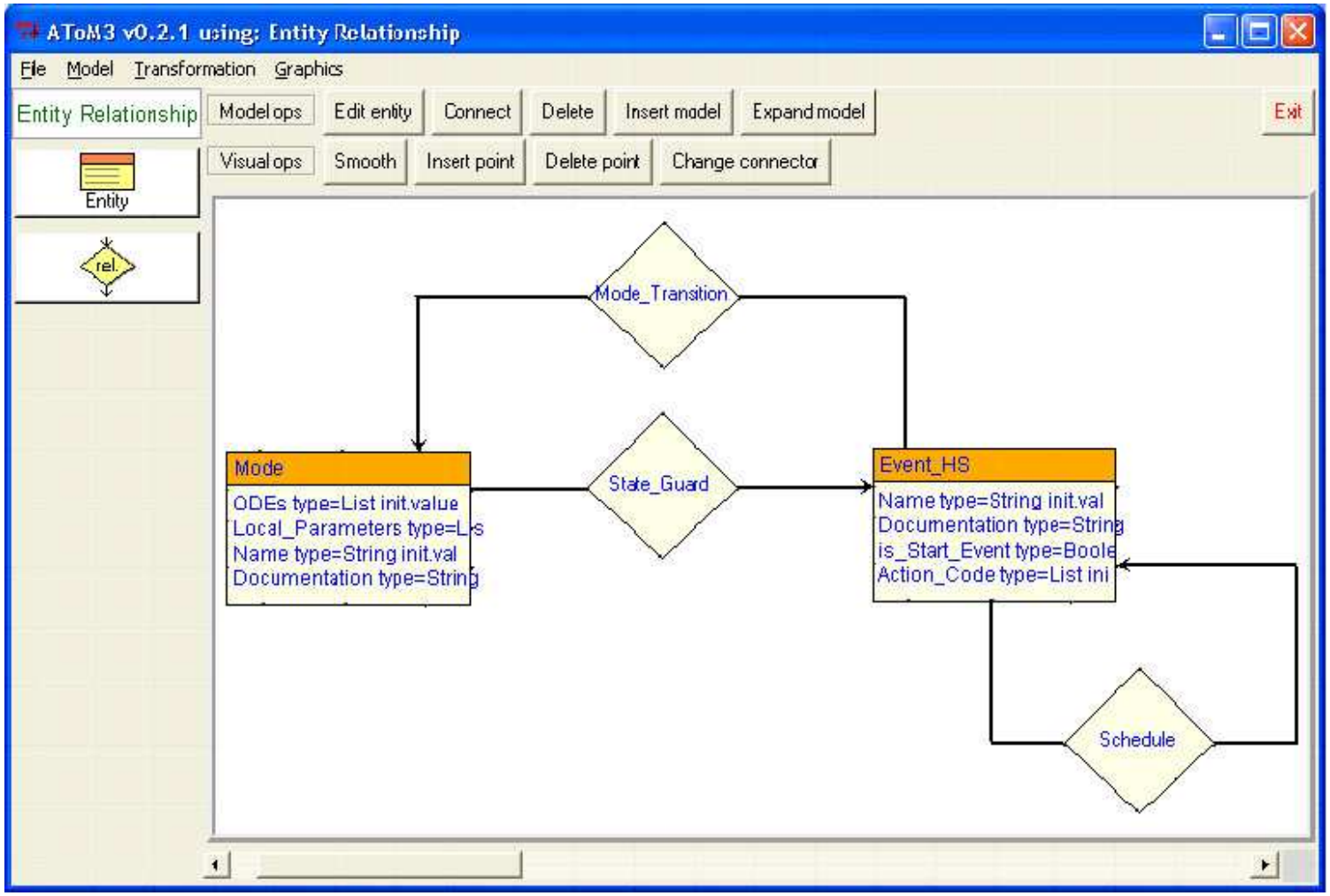
- A meta-model is **a model of** a modelling formalism
- A meta-model is itself a model. Its (textual or visual) *syntax* and *semantics* are governed by the formalism it is described in. That formalism can be modelled in a meta-meta-model.
- As a meta-model is a model, we can reason about it, manipulate it, . . . In particular, properties of (all models in) a formalism can be formally proven.
- Formalism-specific modelling and simulation tools can *automatically* be synthesized from a meta-model.

- Formalisms can be tailored to specific needs by modifying the meta-model (possibly through inheritance if specializing).
- Semantics of new formalisms can be given through extension or transformation of/between formalisms.

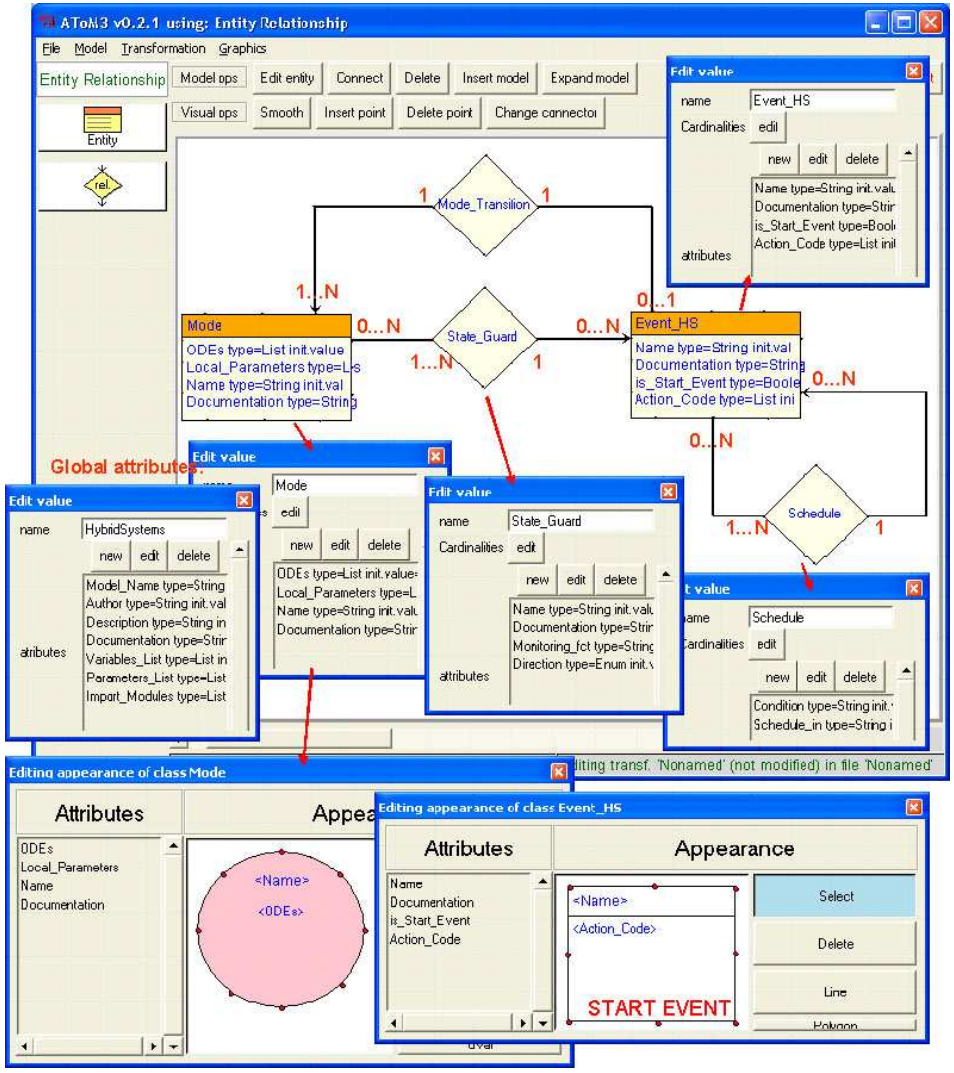
Meta-modelling



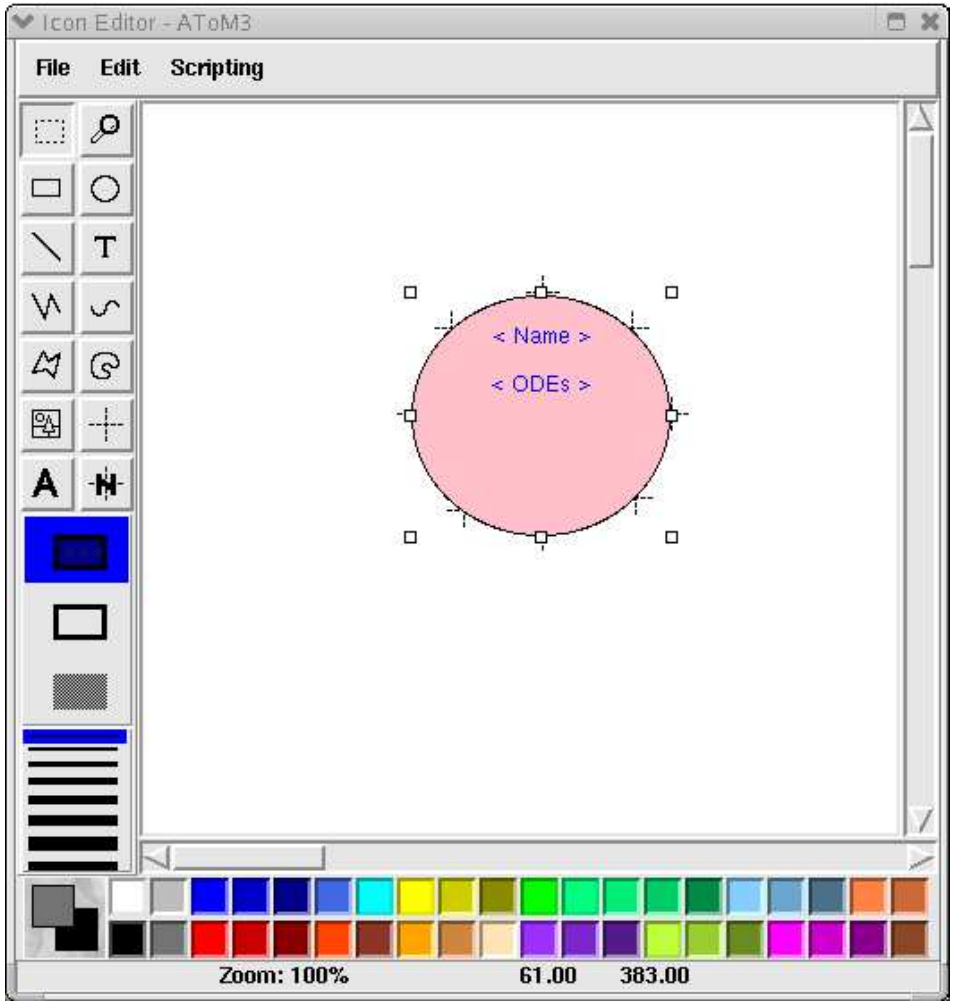
HS Meta-model



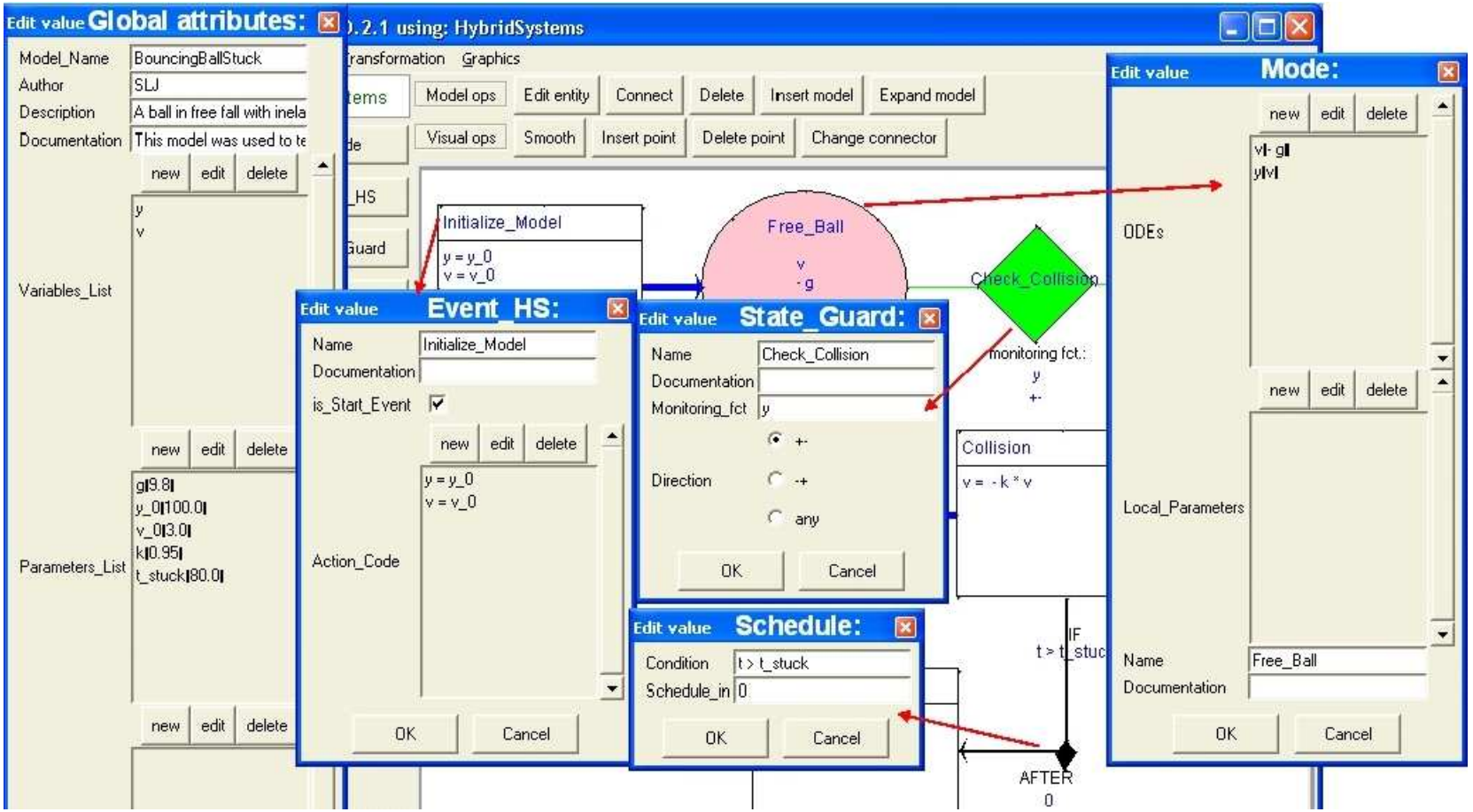
HS Meta-model Details



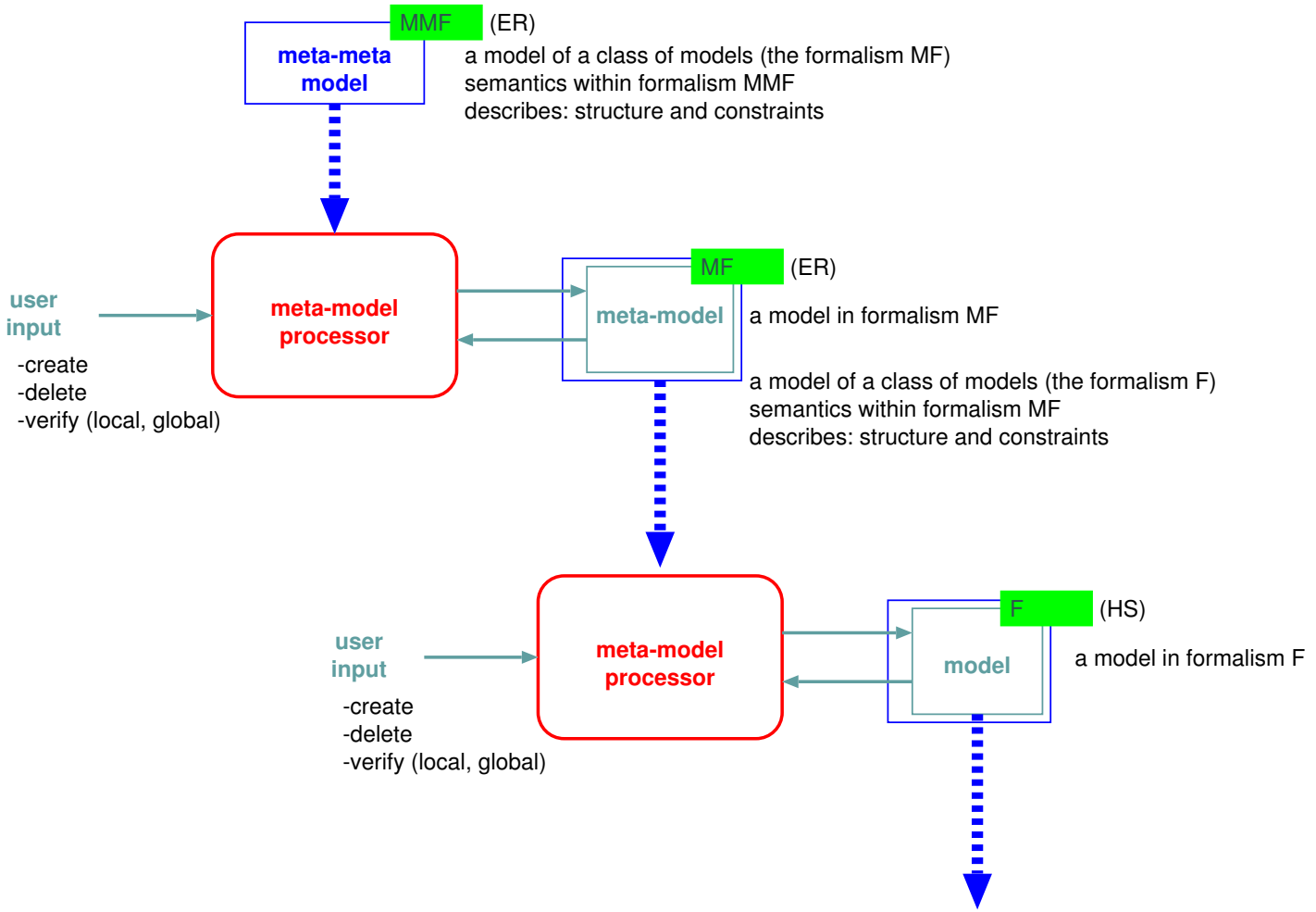
Editing Entity Visual Representation



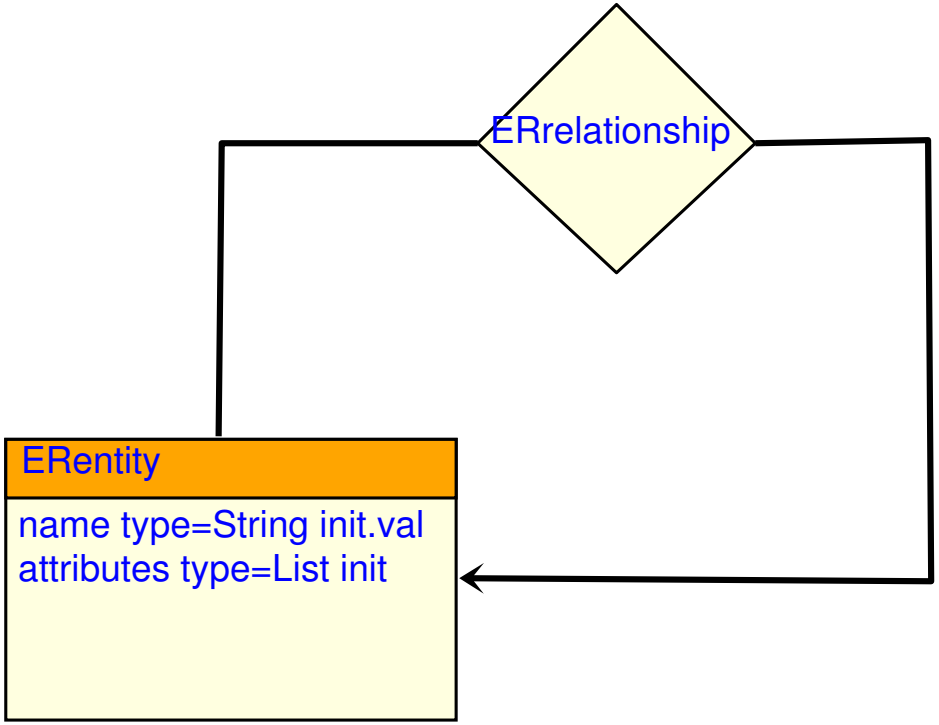
The Synthesized HS Modelling Environment



Meta-meta-. . .


MMF
(ER)
meta-meta model
a model of a class of models (the formalism MF)
semantics within formalism MMF
describes: structure and constraints
user input
-create
-delete
-verify (local, global)
meta-model processor
MF
(ER)
meta-model
a model in formalism MF
a model of a class of models (the formalism F)
semantics within formalism MF
describes: structure and constraints
user input
-create
-delete
-verify (local, global)
meta-model processor
F
(HS)
model
a model in formalism F

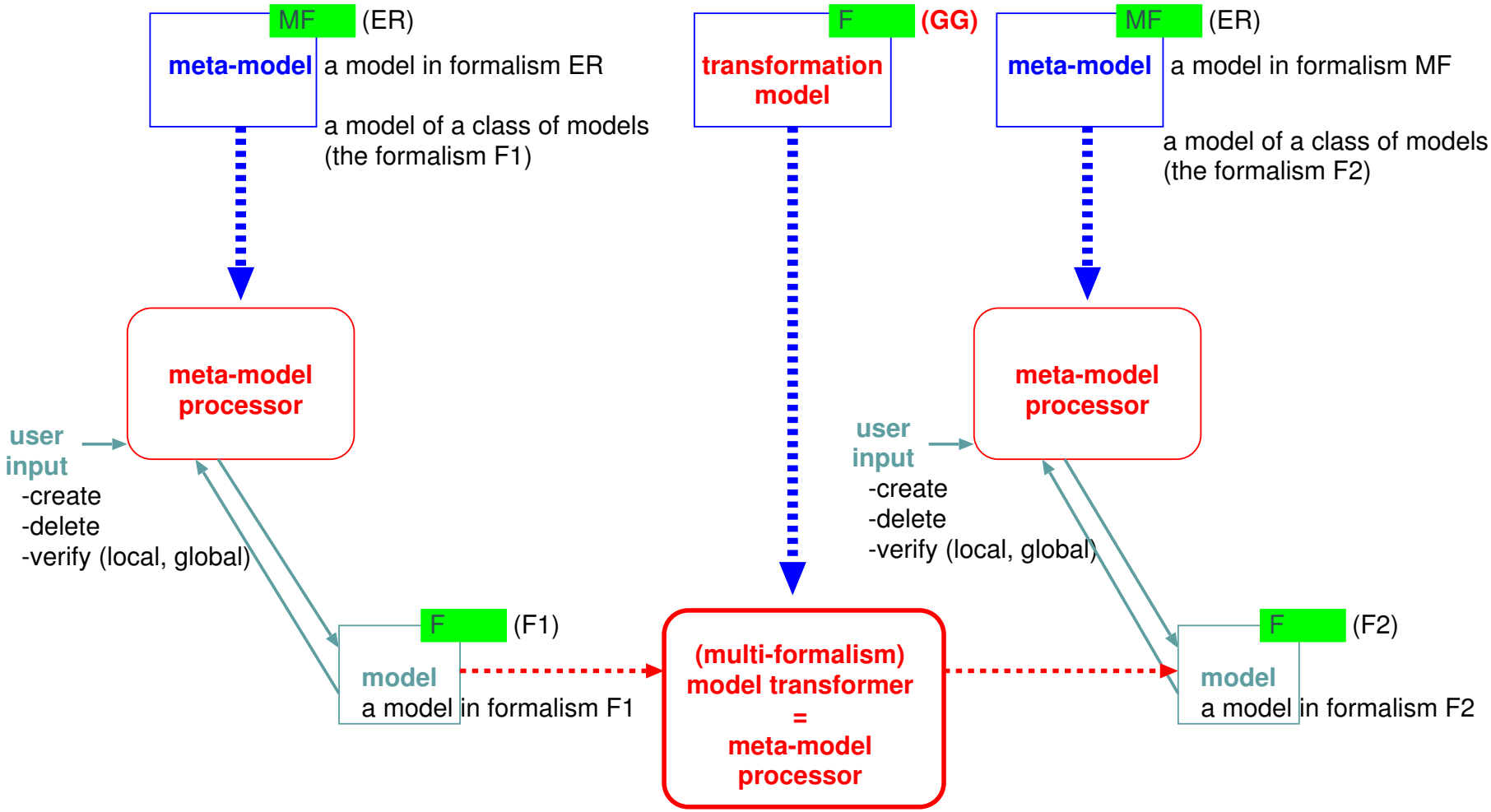
HS' meta-meta model



Transformation of Models

- Specify “operational” semantics (reference simulator)
- Specify “denotational” semantics (map onto known formalism)
- Transform between abstraction levels
- Optimize models
- . . .
- *Model* transformation in the Graph Grammar formalism

Model Transformation Specification



Conclusions

- Meta-modelling, with appropriate tool support,
  allows building formalism/domain-specific (visual) modelling environments in record time
- $\Rightarrow$ no reason to NOT use “tailored” formalisms
- AToM^3 is A Tool for Multi-formalism and Meta-Modelling
  `atom3.cs.mcgill.ca`
- Graph Grammars to explicitly model *transformations*