Wintersim 2002

San Diego, CA

9 December 2002

Meta-model are Models Too

Hans Vangheluwe

School of Computer Science, McGill University, Montréal, Canada

Juan de Lara

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

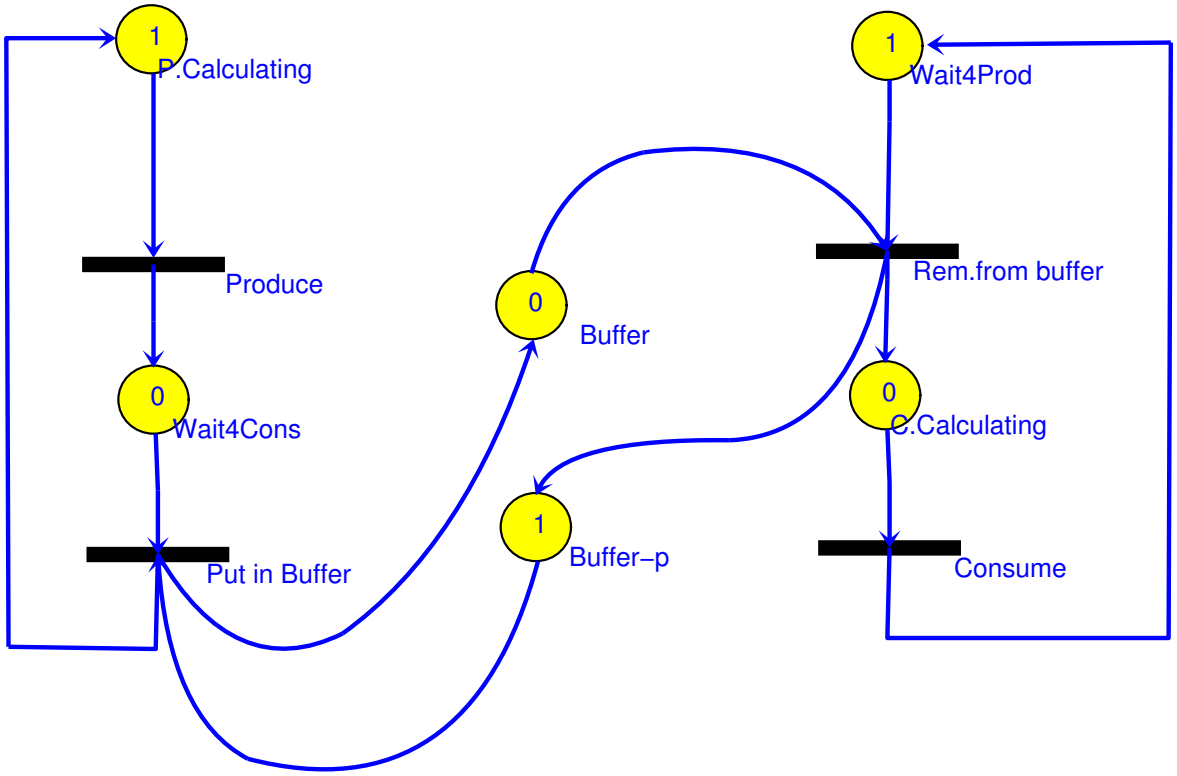
Meta-modelling and XML

- modelling and simulation
  $\rightarrow$ meta-modelling
  $\rightarrow$ meta-modelling and XML
- Experiences with
  A Tool for Multi-formalism, Meta-Modelling
  AToM^3: `http://atom3.cs.mcgill.ca`

Modelling and Simulation Wishlist

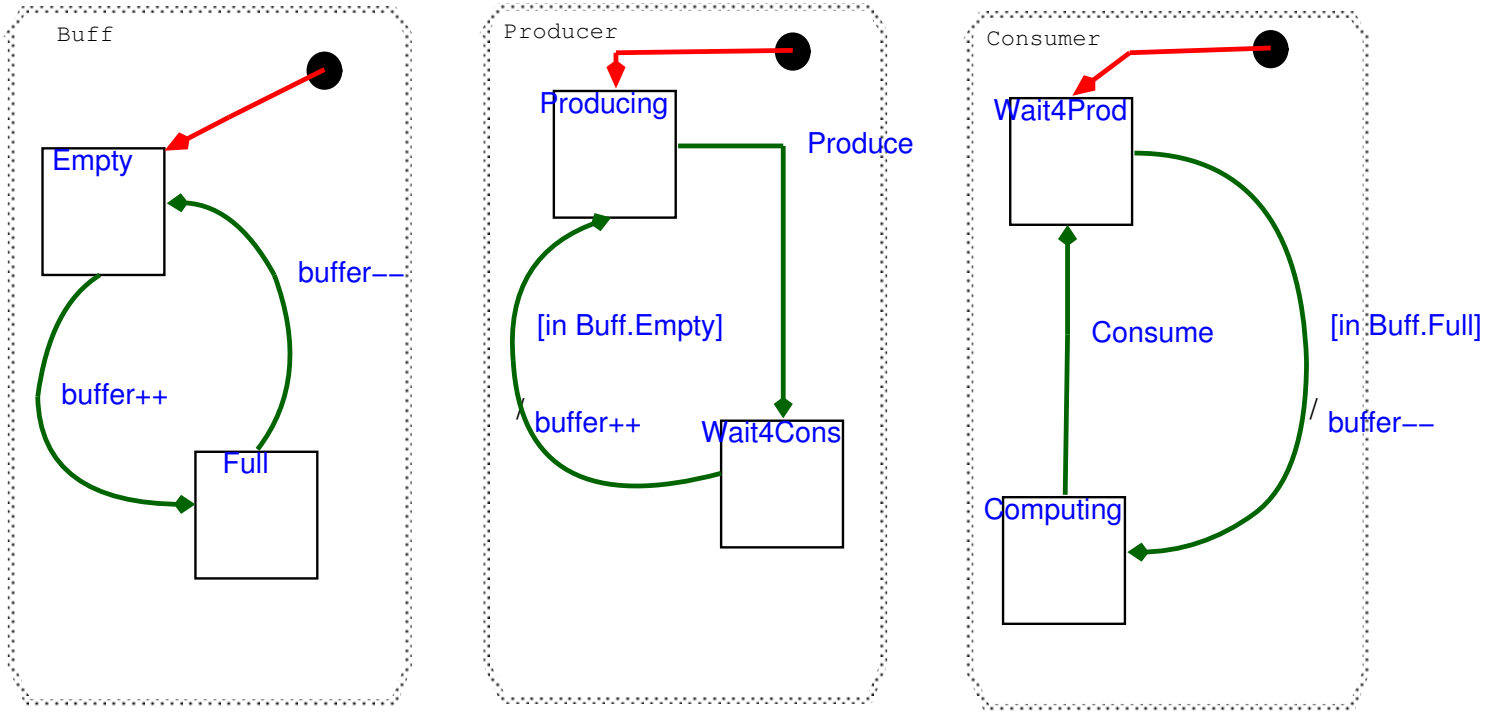
- Meaningful exchange and re-use of models
  Syntax *and* Semantics !
- Domain/problem-specific (visual) modelling & simulation environments
  Syntax *and* Semantics !
- Model transformation
  - simulation (state changes)
  - code-generation (syntax changes)
  - simplification (level of abstraction changes)
  - formalism transformation (formalism changes)
- Meaningful multi-formalism modelling

Petri Net model of Producer Consumer



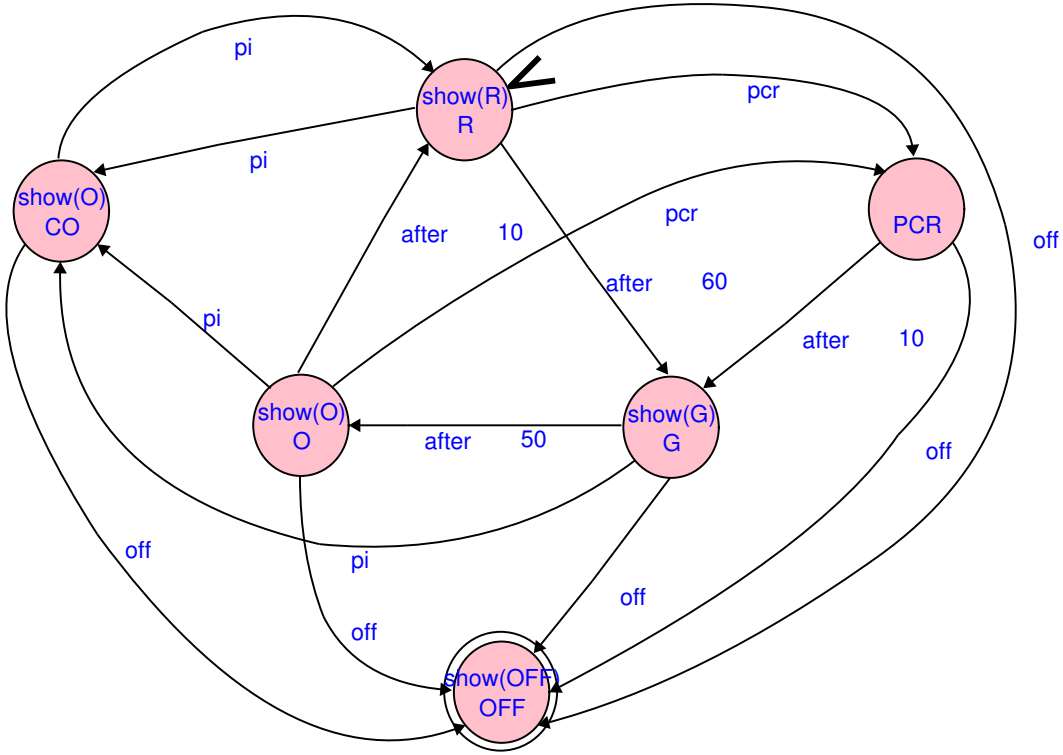
reachability analysis + simplification + simulation + code generation

Statechart model of Producer Consumer



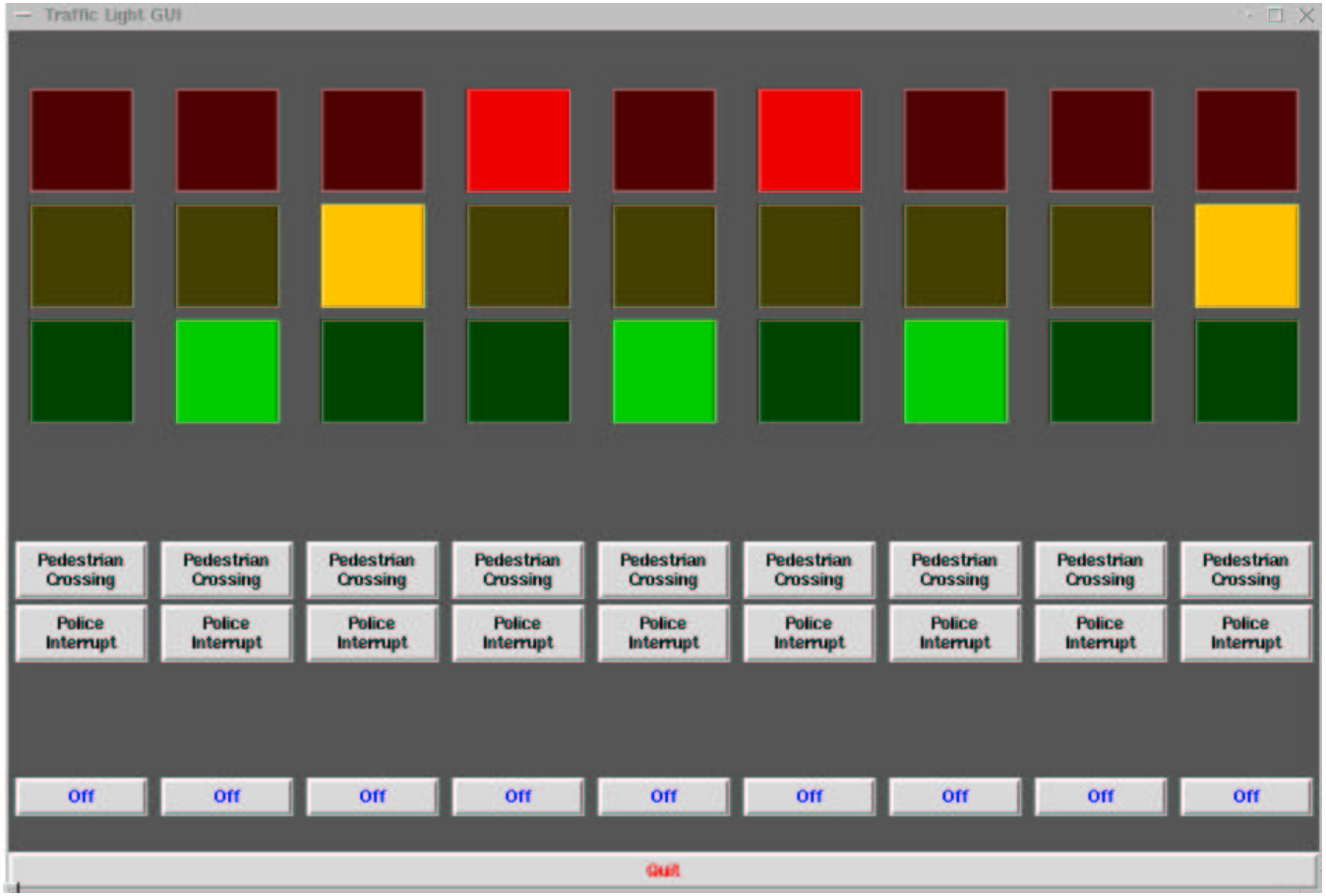
simulation + code generation + transformation to equivalent Petri Net

Timed Automata model of a Traffic Light

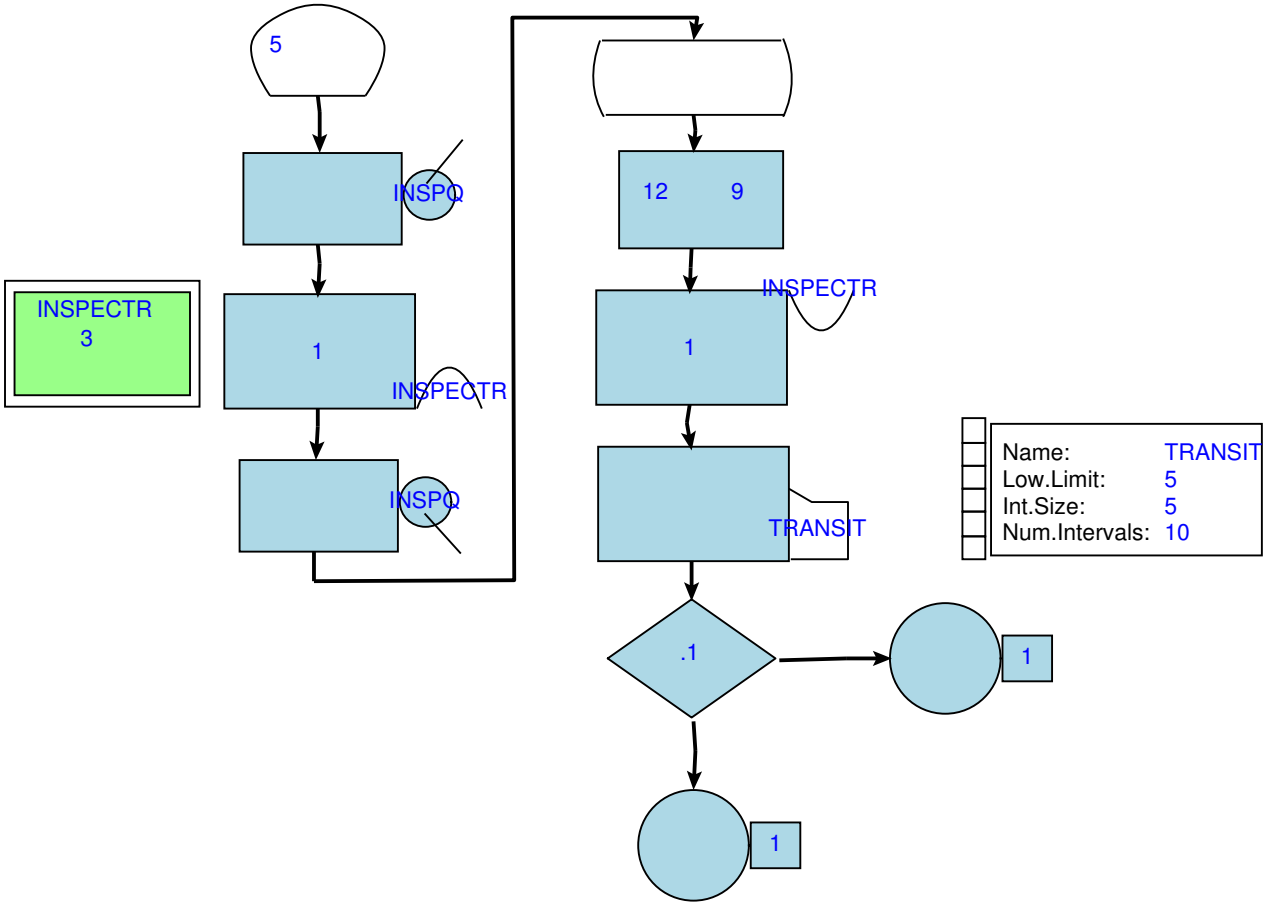


smulation + code generation

Generated Application



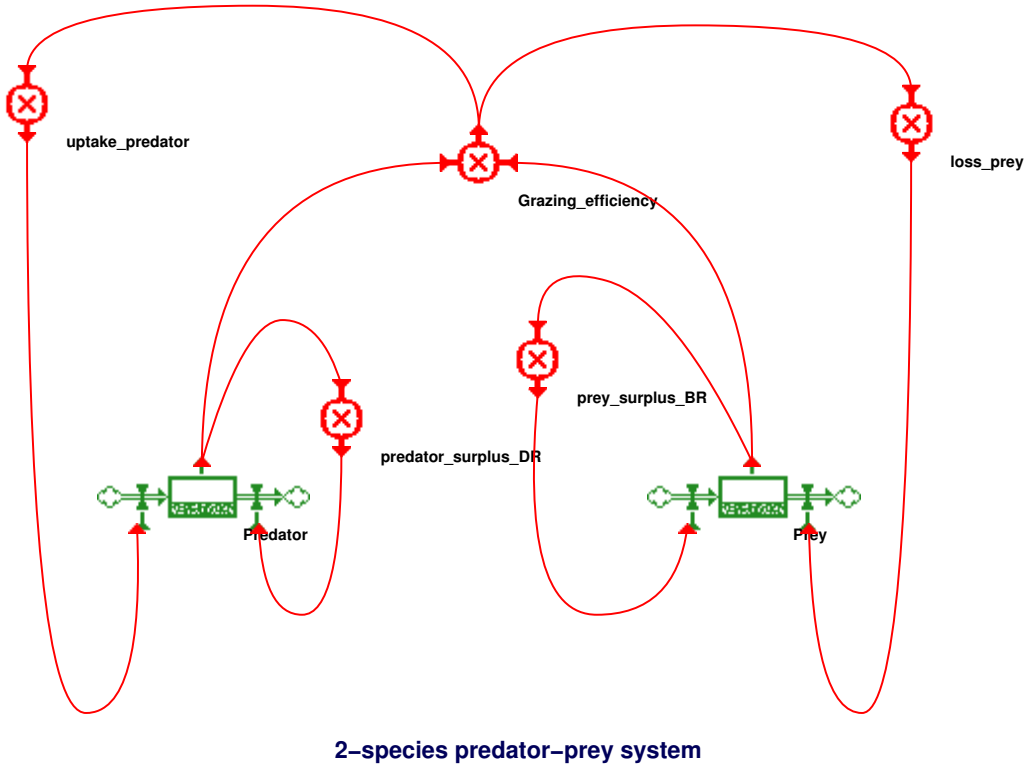
GPSS model of Manufacturing system



Generated GPSS code

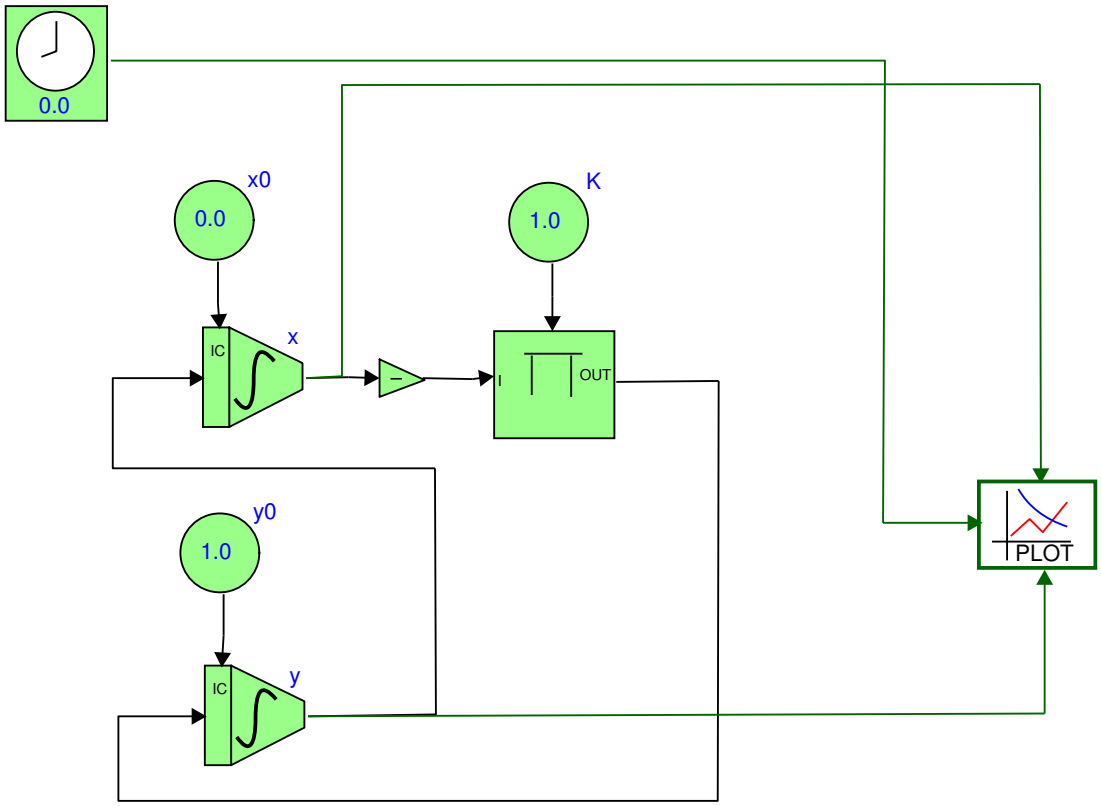
```
*                Manufacturing shop model 4
*                G. Gordon Figure 11-9/9-8
       SIMULATE
 L0    GENERATE   5               ; Create parts
 L7    QUEUE      INSPQ           ; Queue for an inspector
 L5    ENTER      INSPECTR,1      ; A single inspector becomes busy
 L8    DEPART     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



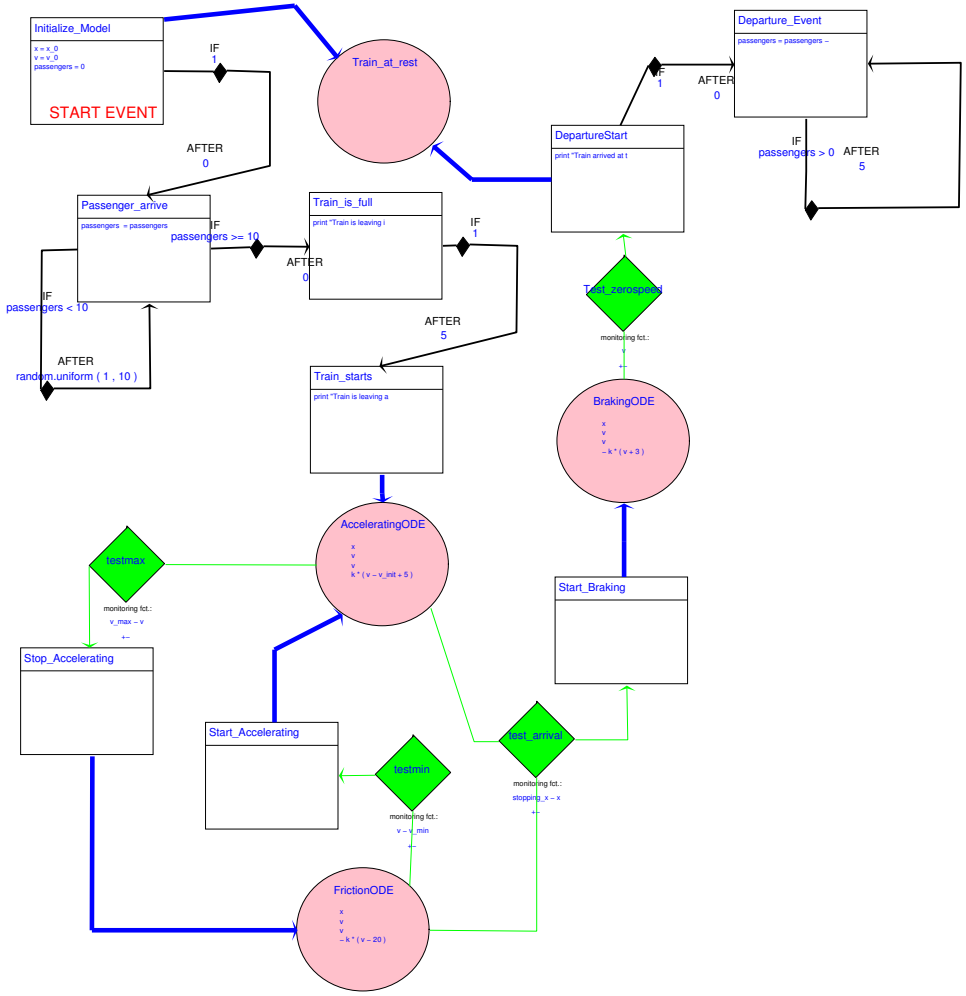
transformation to Ordinary Differential Equations + analysis

Causal Block Diagram model of Harmonic Oscillator



analysis + simplification + simulation

Event Scheduling + DAE model of a Train



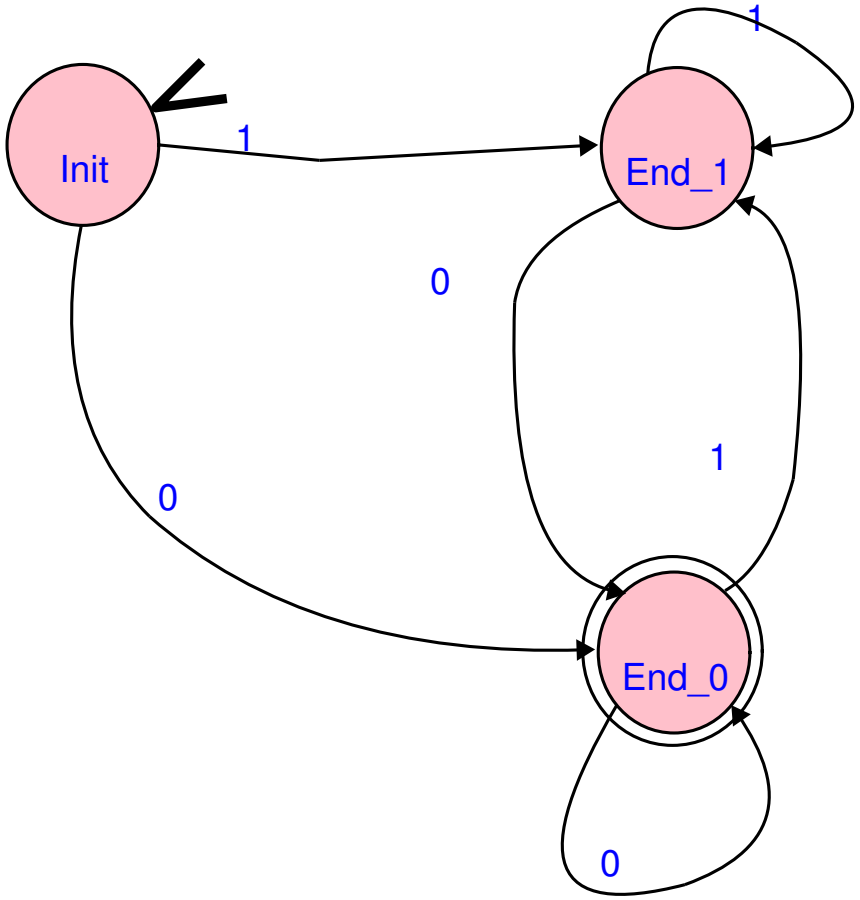
code generation

What is Meta-modelling ?

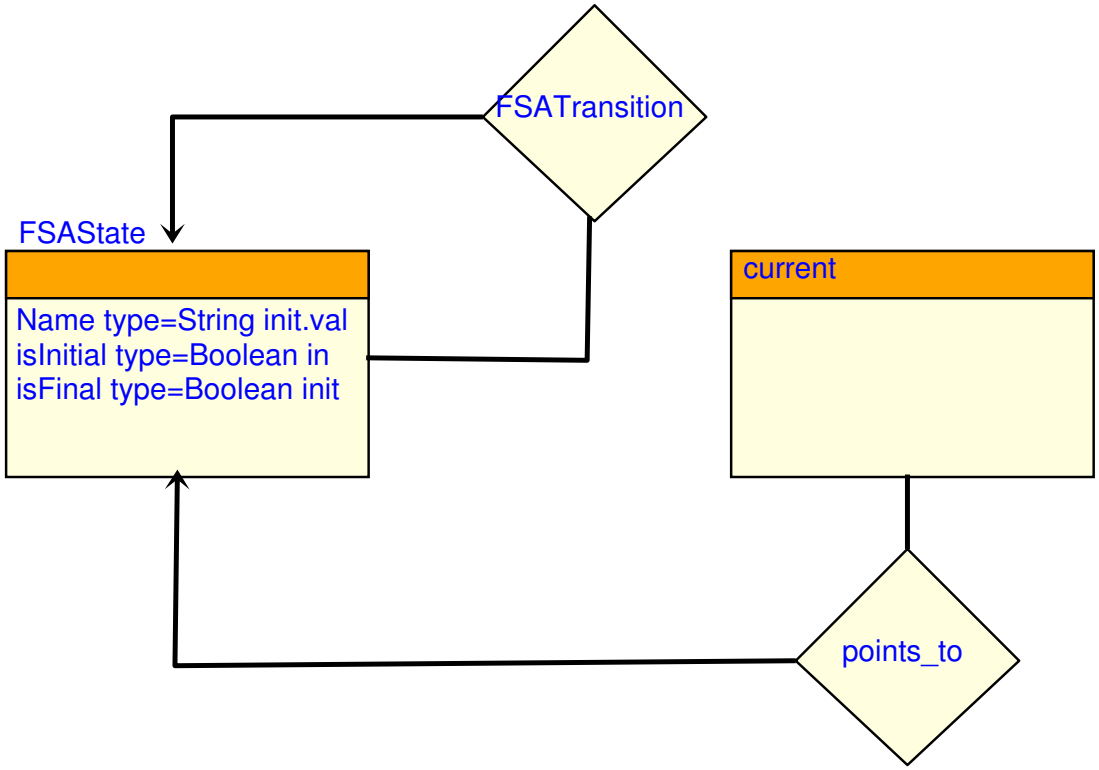
- A meta-model is **a model of** a modelling formalism
- A meta-model is itself a *model*. Its syntax and semantics are governed by the formalism it is described in. That formalism can be modelled in a meta-meta-model.
- As a meta-model is a model, we can reason about it, manipulate it, . . . In particular, 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).
  $\Rightarrow$ Building domain/application specific, possibly graphical modelling and simulation environments becomes affordable.
- Semantics of new formalisms through extension or transformation.

FSA model of Even Binary Number recognizer



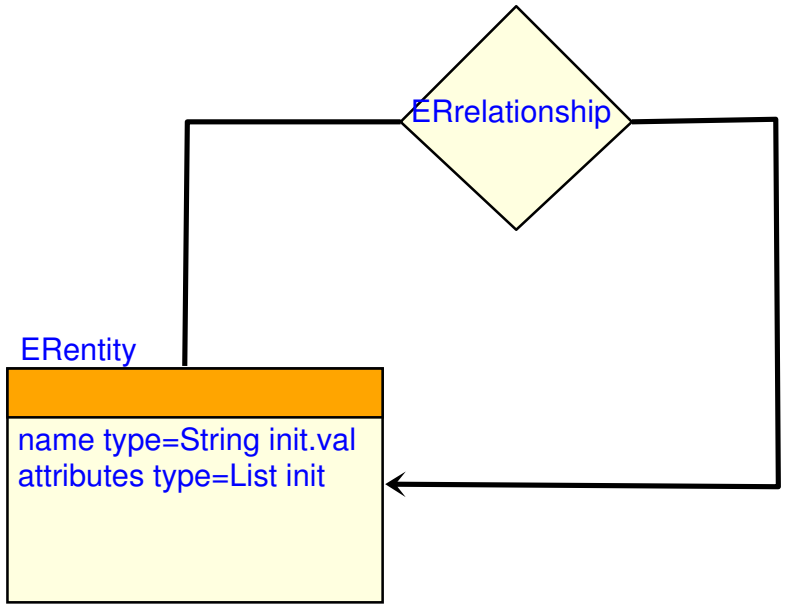
ER model of the FSA formalism syntax (meta-model)



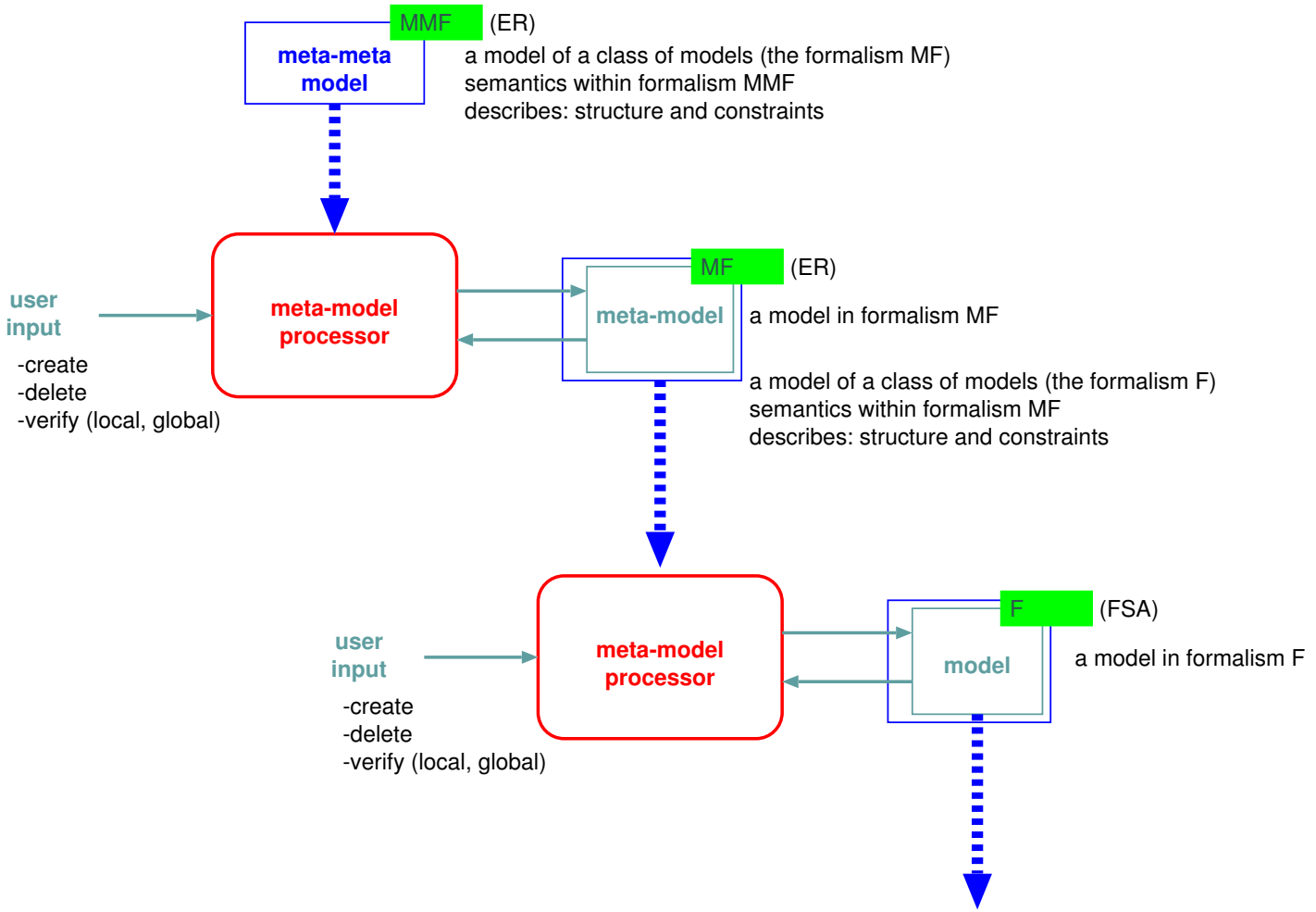
ER formalism + constraints (OCL/Python)

```
# 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)
```

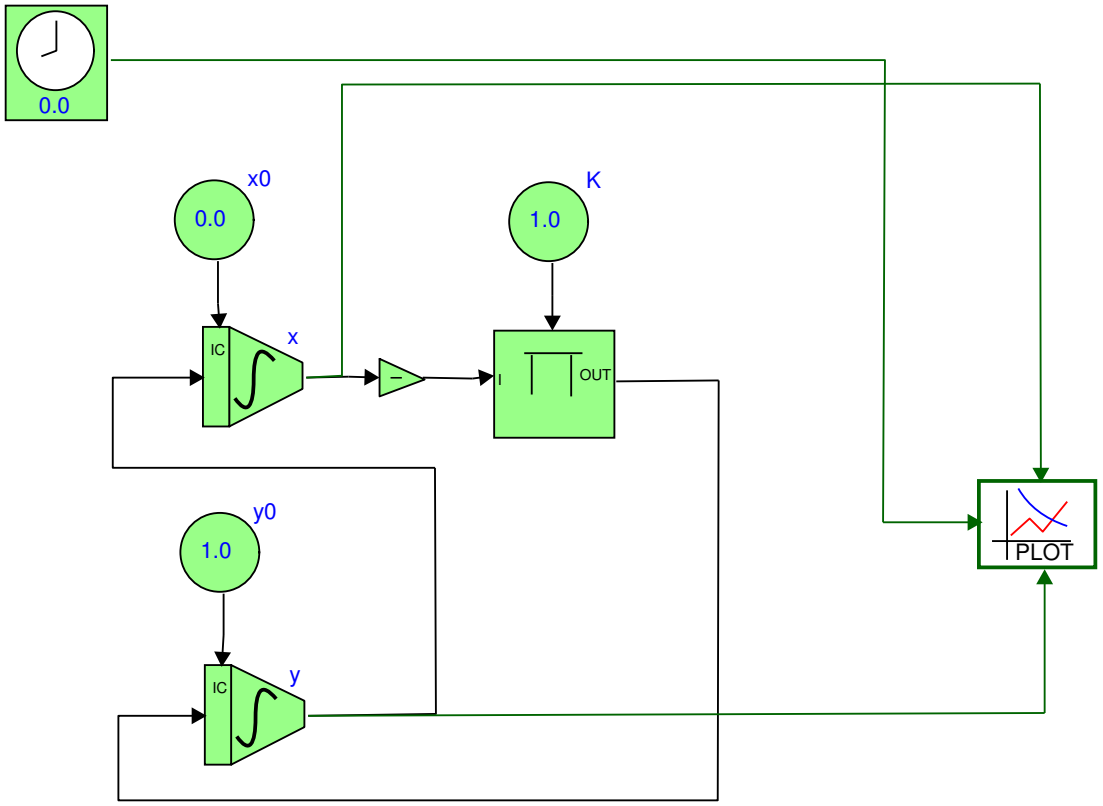
ER model of the ER formalism (meta-meta-model)



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
(FSA)
model
a model in formalism F

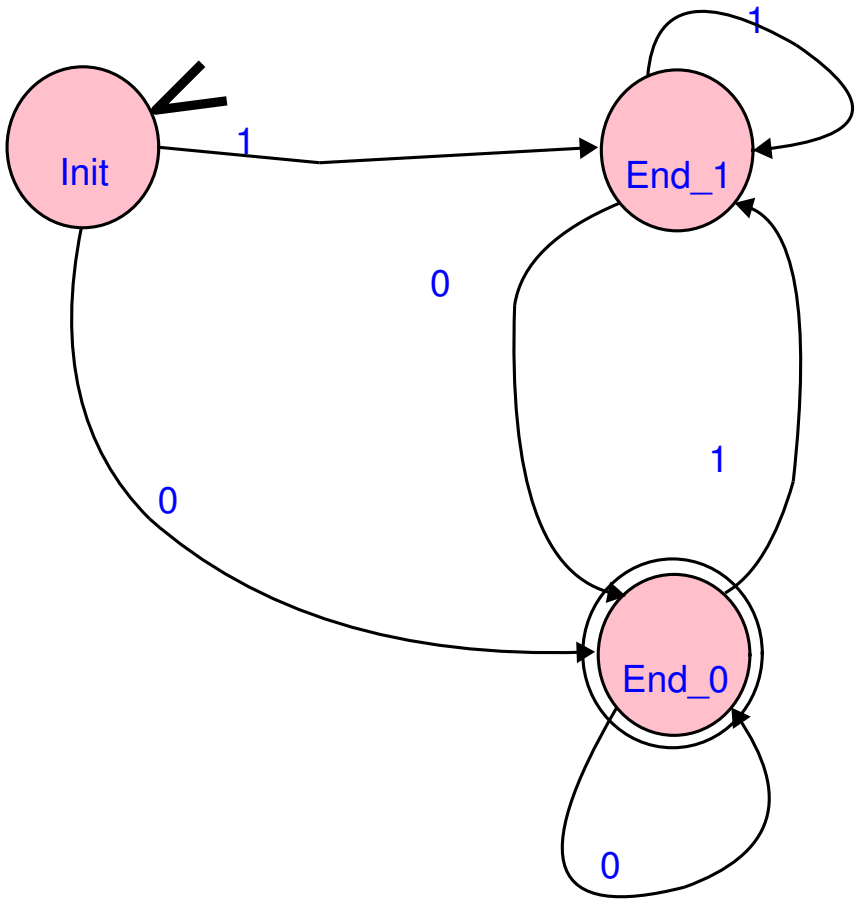
Causal Block Diagram Semantics ?



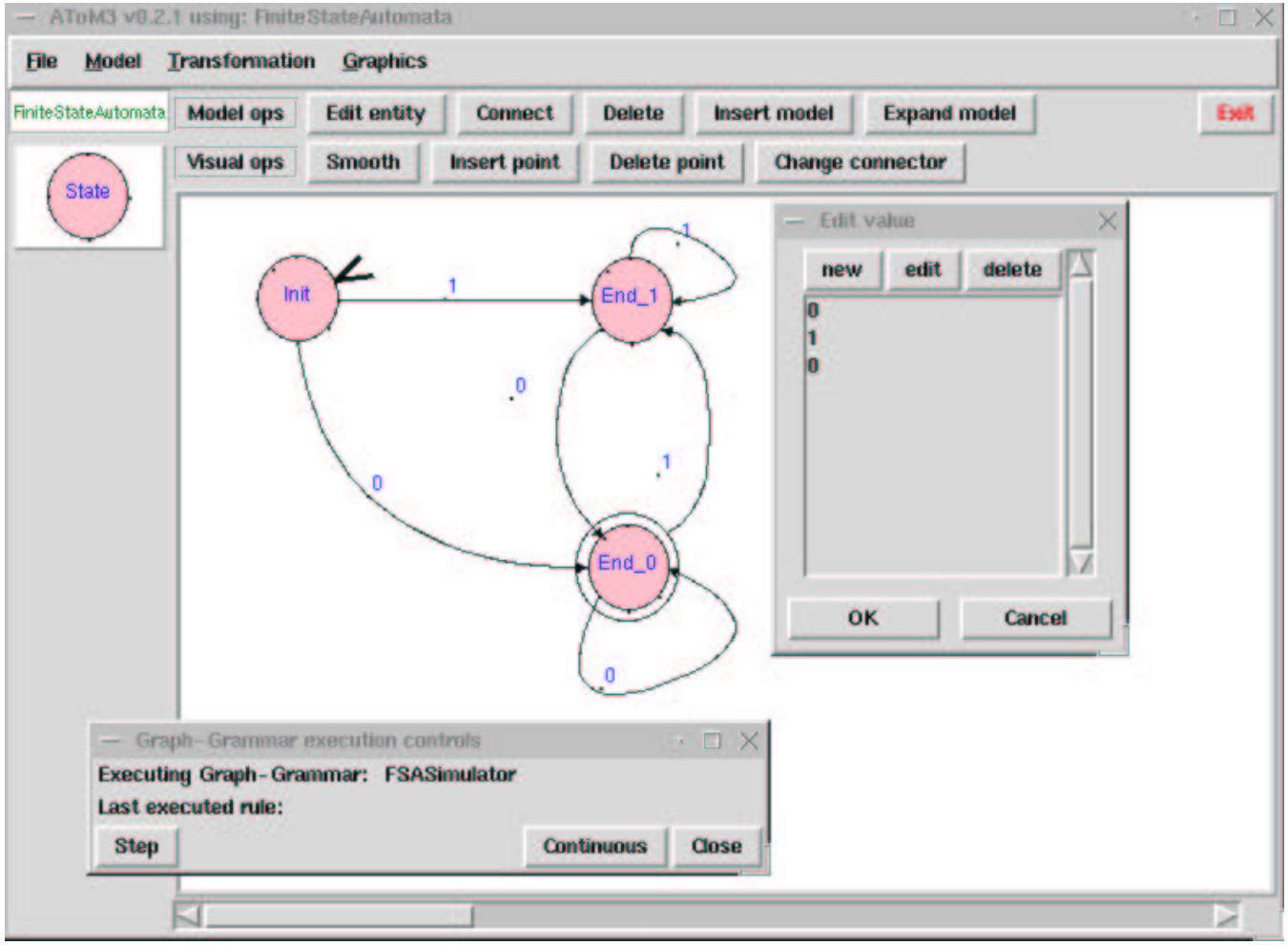
Causal Block Diagram Denotational Semantics

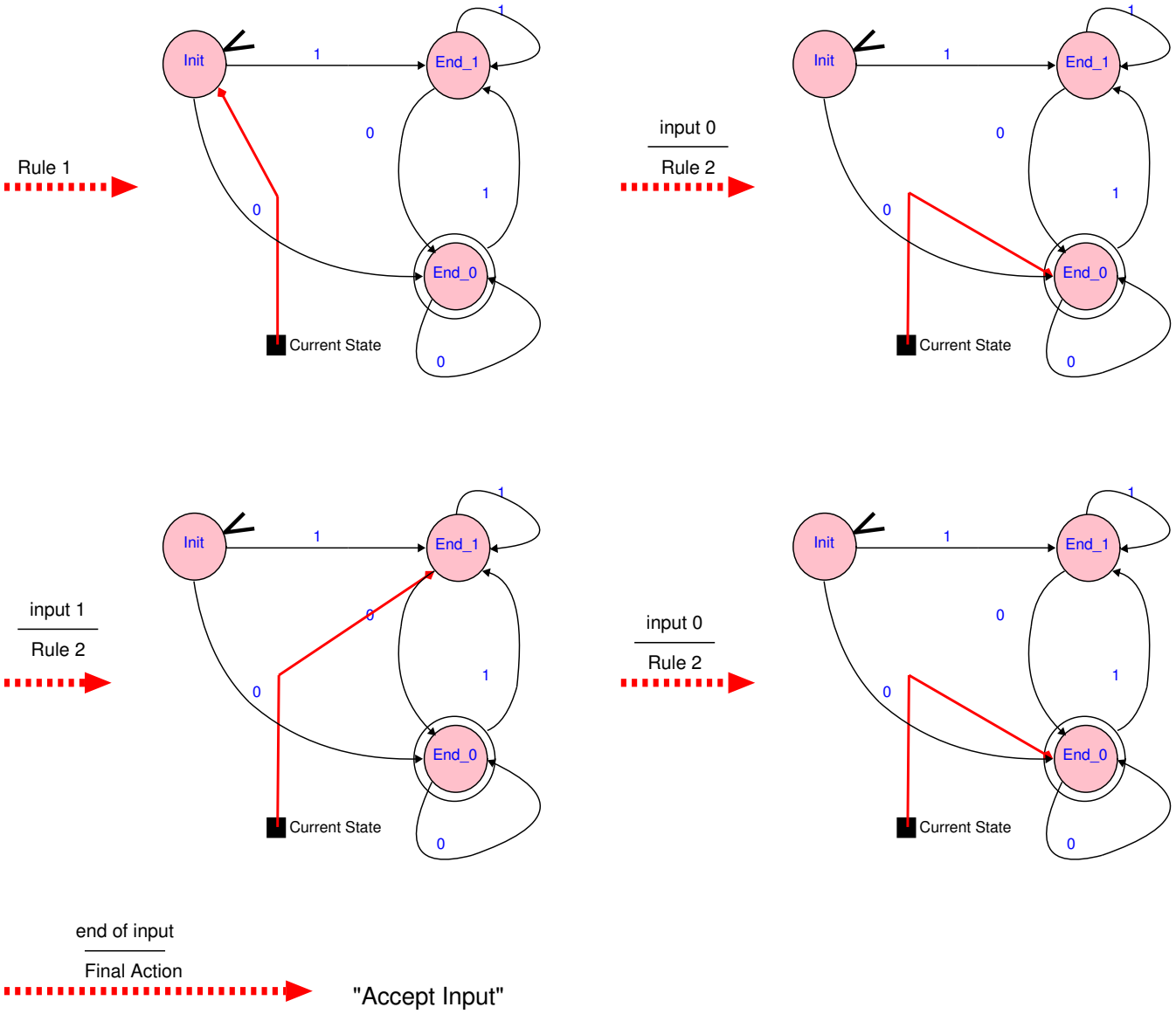
$$\begin{cases} \frac{dx}{dt} = y & x(0) = 0 \\ \frac{dy}{dt} = -Kx & y(0) = 1 \\ K = 1 \end{cases}$$

FSA model Operational Semantics ?



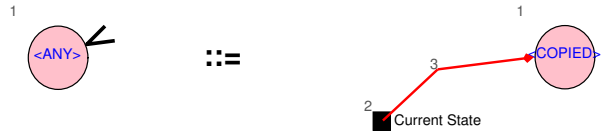
Simulation steps



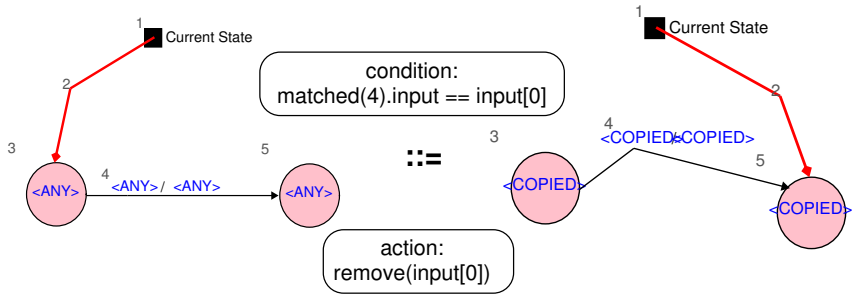

Rule 1
Init
End_1
End_0
1
0
Current State
input 0
Rule 2
input 1
Rule 2
input 0
Rule 2
end of input
Final Action
"Accept Input"

Graph Grammar *model* of FSA OpSem

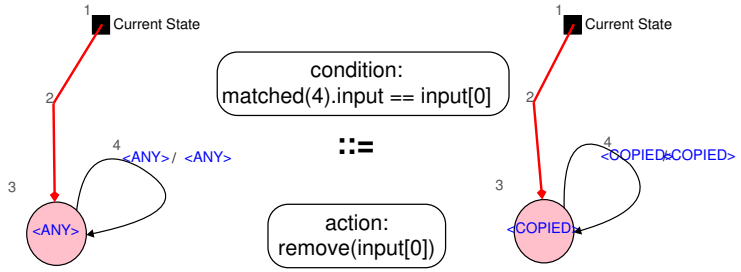
Rule 1 (priority 3) **Locate Initial Current State**



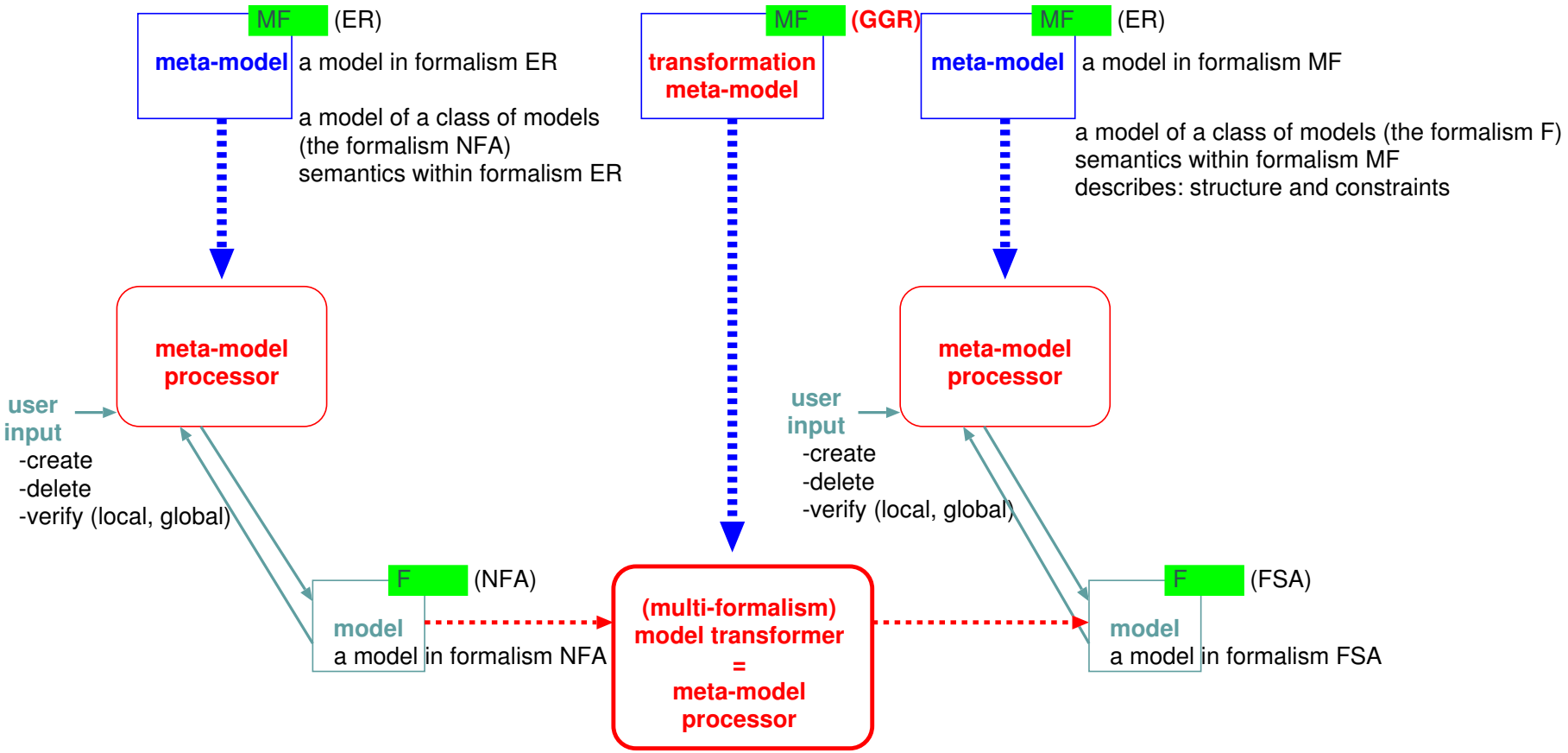
Rule 2 (priority 1) **State Transition**



Rule 3 (priority 2) **Local State Transition**



Model Transformation meta-specification



Model Transformation Uses (1)

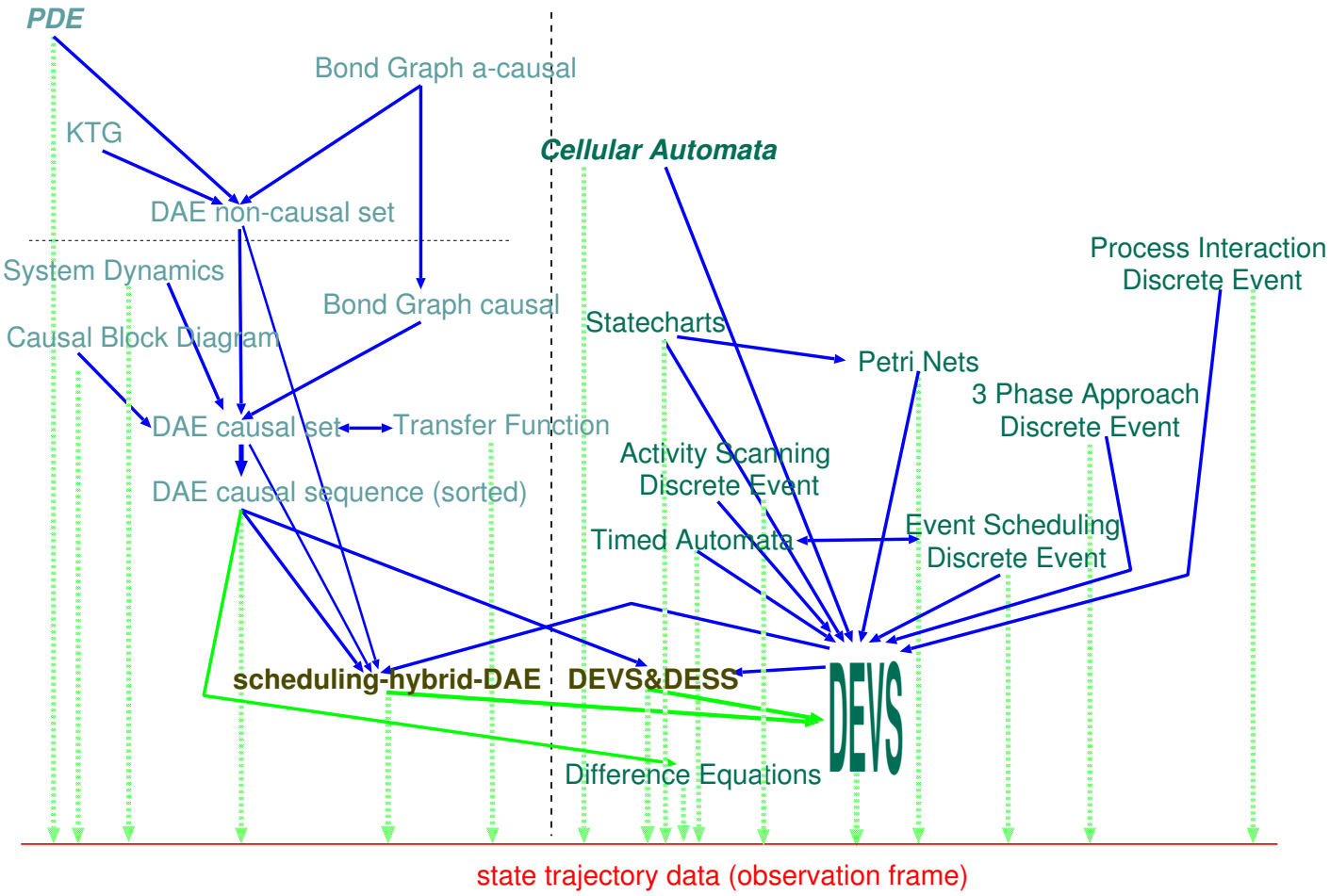
- Code generation
- Operational Semantics (reference simulator)
- Denotational Semantics

May model transformation as Graph Grammar

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.

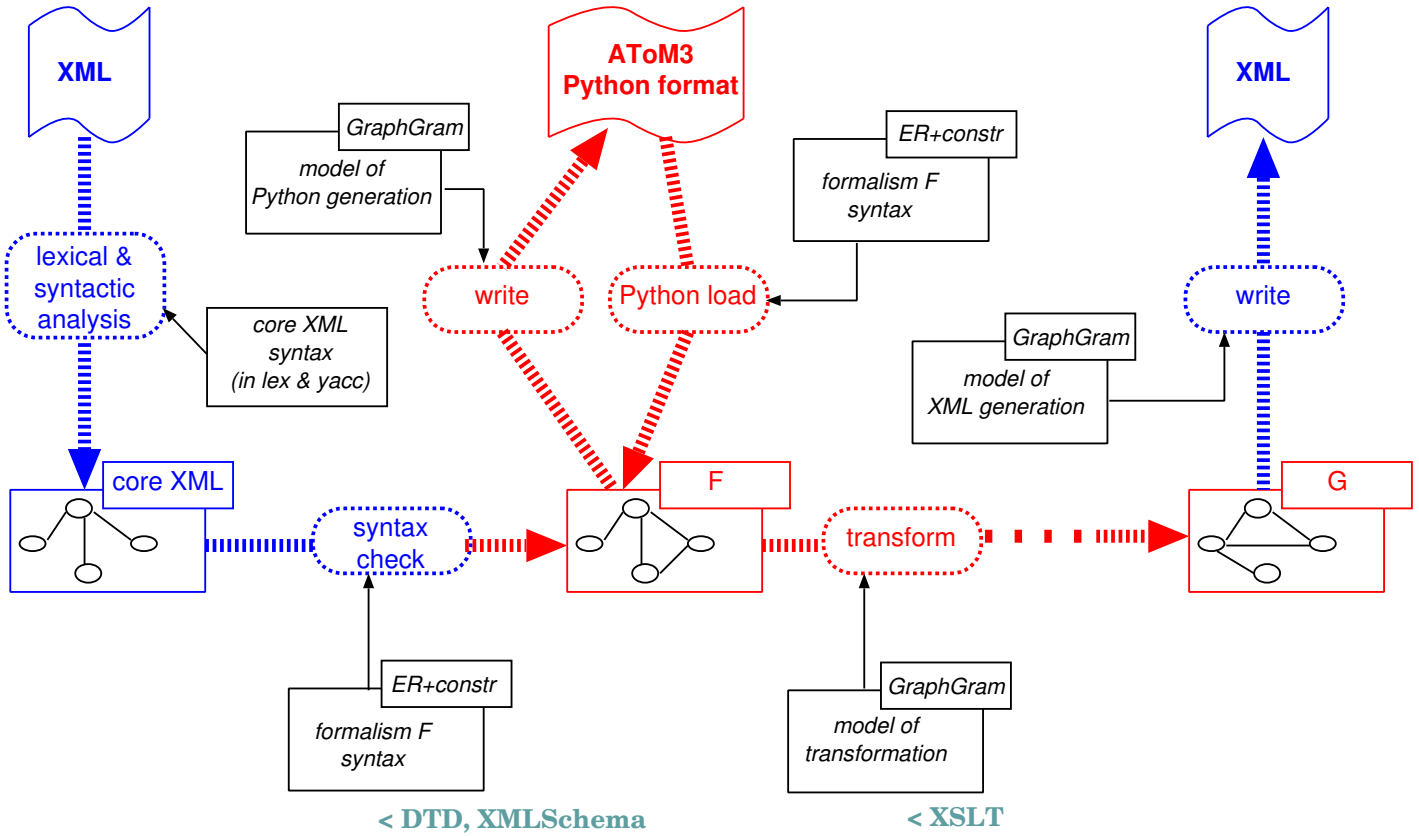
Formalism Transformation Graph



Saving (meta-)models: core-XML

```
document         ::= prolog element misc*
prolog           ::= VERSION? ENCODING? misc*
misc             ::= COMMENT | attribute_decl
attribute_decl   ::= ATTDEF NAME attribute+ ENDDEF
element          ::= START attribute* empty_or_content
empty_or_content ::= SLASH CLOSE | CLOSE content END NAME? CLOSE
content          ::= (DATA | misc | element)*
attribute        ::= NAME (EQ VALUE)?
```

XML and Meta-modelling



Alternatives

1. Generate DTDs from meta-model to describe model syntax
2. Models are attributed typed directed graphs
   $\rightarrow$ Graph Exchange Language (GXL)
   `http://www.grupo.de/GXL`

GXL Example

```
<gxl>
 <node id="Customers" type="GENERATE">
  <attr name="A" value="50"/>
  <attr name="B" value="10"/>
 </node>
 <node id="End" type="TERMINATE">
  <attr name="A" value="1"/>
 </node>
 <edge begin="Customers" end="End" type="ConnectBlock">
  <attr colour="BLUE">
 </edge>
</gxl>
```

Conclusions

- Meta-model formalism syntax
- Graph Grammars *models* for all model Transformations
- Variations (flavours) of formalisms (syntax and semantics)
- Simulator (reference implementation)
- Model exchange with XML, GXL
- Meta-modelling Environment (ATOM^3)