MESM 2002

Sharjah, United Arab Emirates

29 September, 2002

A Software Architecture for Multi-paradigm Modelling

Hans Vangheluwe

McGill

School of Computer Science, McGill University, Montréal, Canada

Juan de Lara

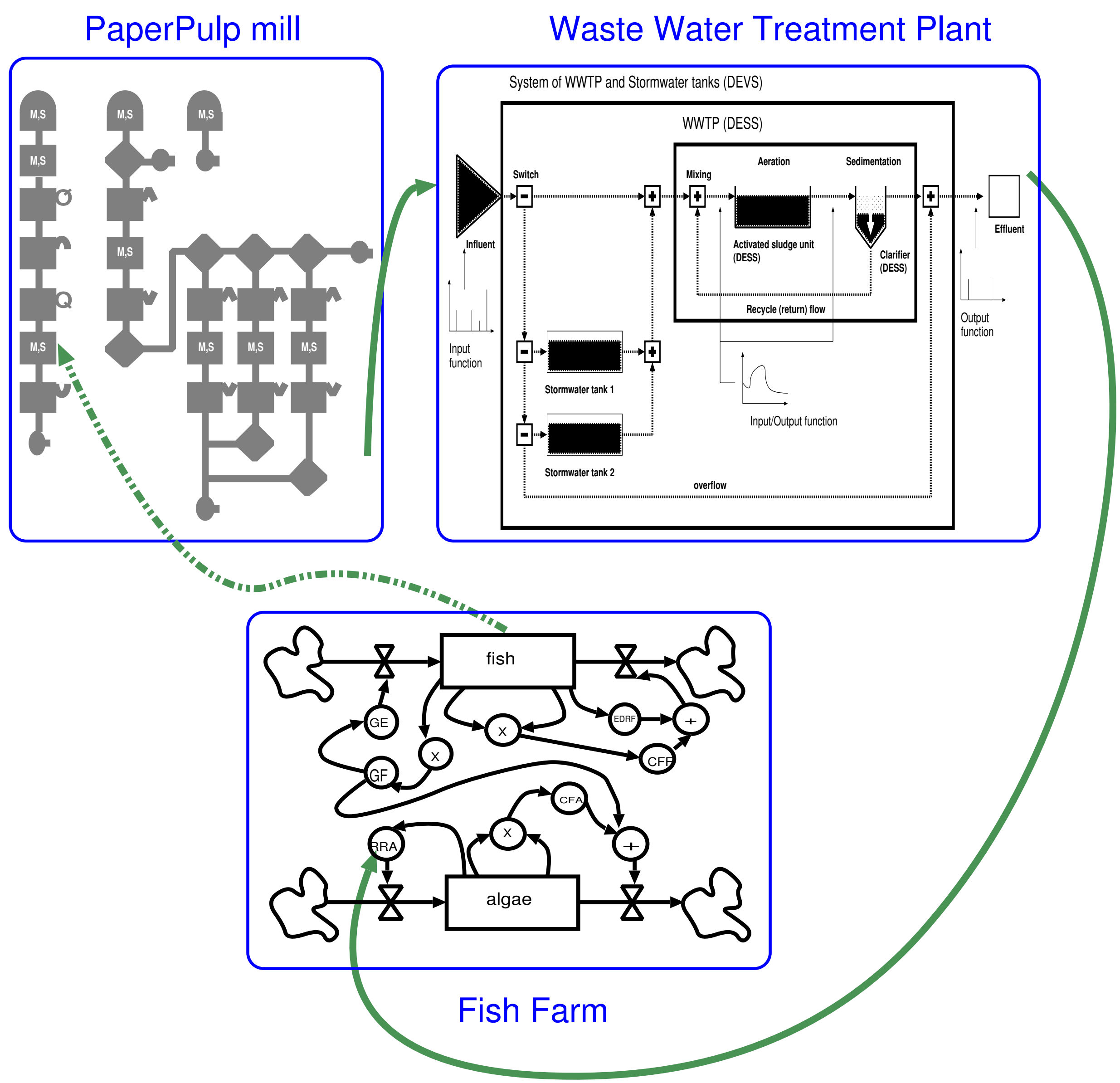
E.T.S. de Informática, Universidad Autonóma de Madrid, Madrid, Spain

Ghislain Vansteenkiste

BIOMATH department, Ghent University, Ghent, Belgium

Presentation Overview

- Complex systems
- Multi-paradigm Modelling and Simulation
- 1. Levels of abstraction
- 2. Multi-formalism Modelling and Simulation
- 3. Meta-modelling formalism syntax and semantics
- The AToM^3 environment


PaperPulp mill
Waste Water Treatment Plant
M,S
System of WWTP and Stormwater tanks (DEVS)
WWTP (DESS)
Aeration
Sedimentation
Switch
Mixing
Influent
Activated sludge unit (DESS)
Clarifier (DESS)
Effluent
Recycle (return) flow
Input function
Output function
Stormwater tank 1
Stormwater tank 2
Input/Output function
overflow
fish
GE
X
EDRF
GF
CFF
CFA
RRA
algae
Fish Farm

Complexity (system/model) due to . . .

1. Number of interacting (coupled, concurrent) components (+ feedback)
2. Variety of views at different levels of abstraction
3. Variety of components (software/hardware, continuous/discrete)
4. Uncertainty

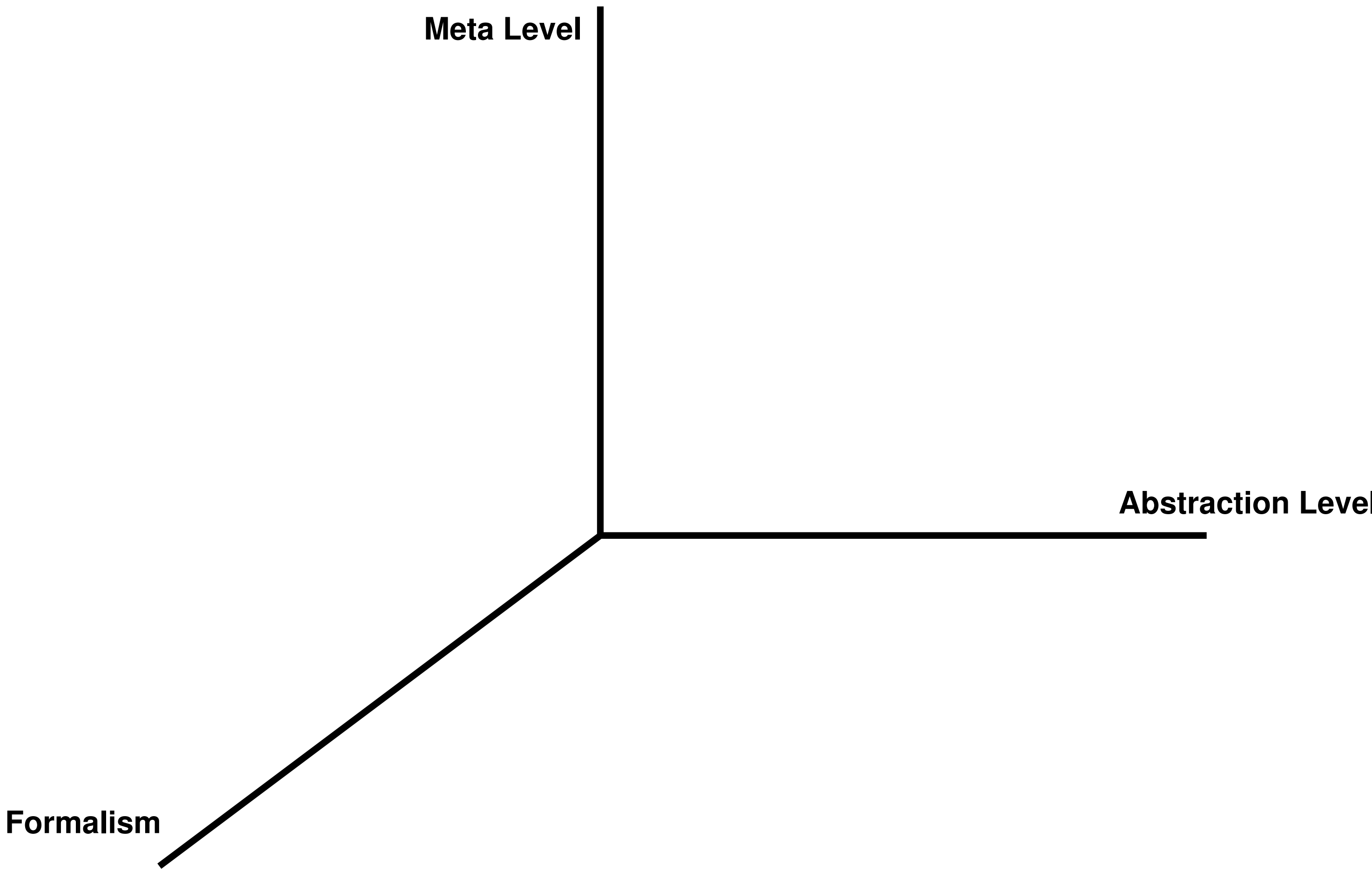
$\longrightarrow$ **focus** on 1 and 3

Proposed solution:
multi-paradigm modelling and simulation

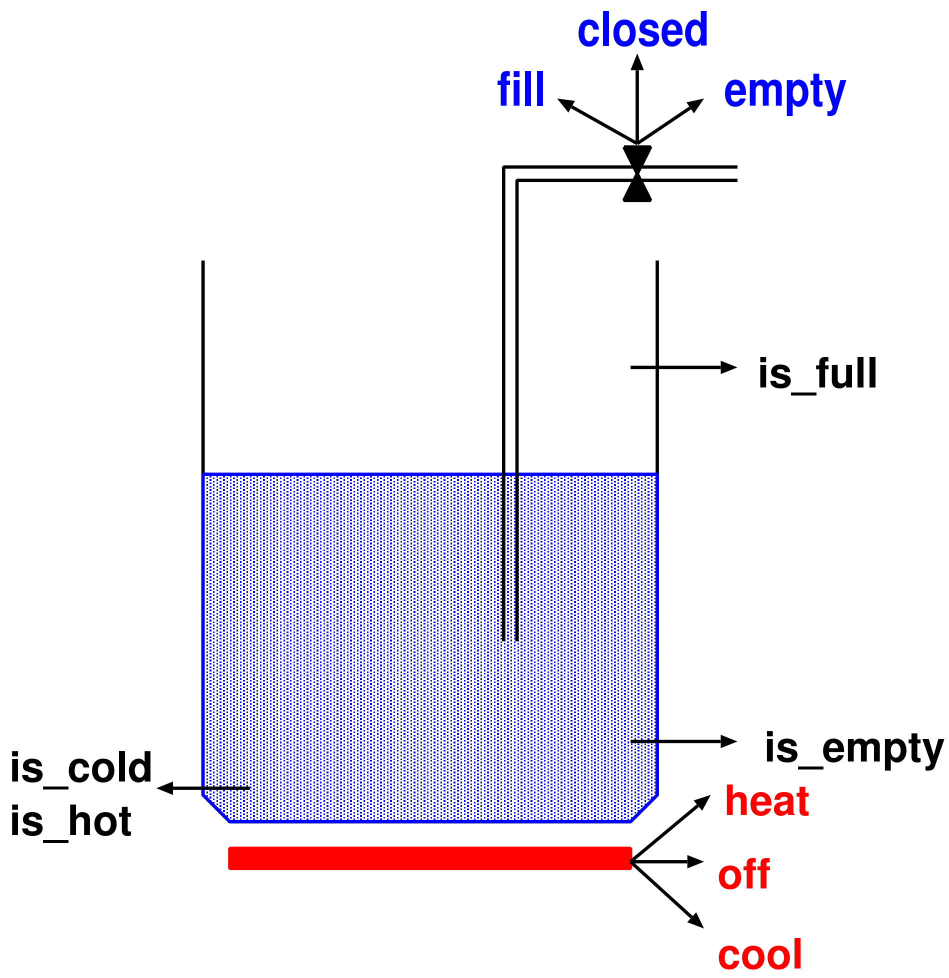
1. Different *levels of abstraction*
2. Mixing different *formalisms*
3. Modelling syntax and semantics of classes of models (formalisms): *meta-modelling*

All are closely related to *model transformation*

Multi-paradigm dimensions



System under study: T, l controlled liquid



Detailed (continuous) view, ALG + ODE formalism

Inputs (discontinuous $\rightarrow$ hybrid model):

- Emptying, filling flow rate ϕ
- Rate of adding/removing heat W

Parameters:

- Cross-section surface of vessel A
- Specific heat of liquid c
- Density of liquid ρ
- Temperature of influent T_{in}

State variables:

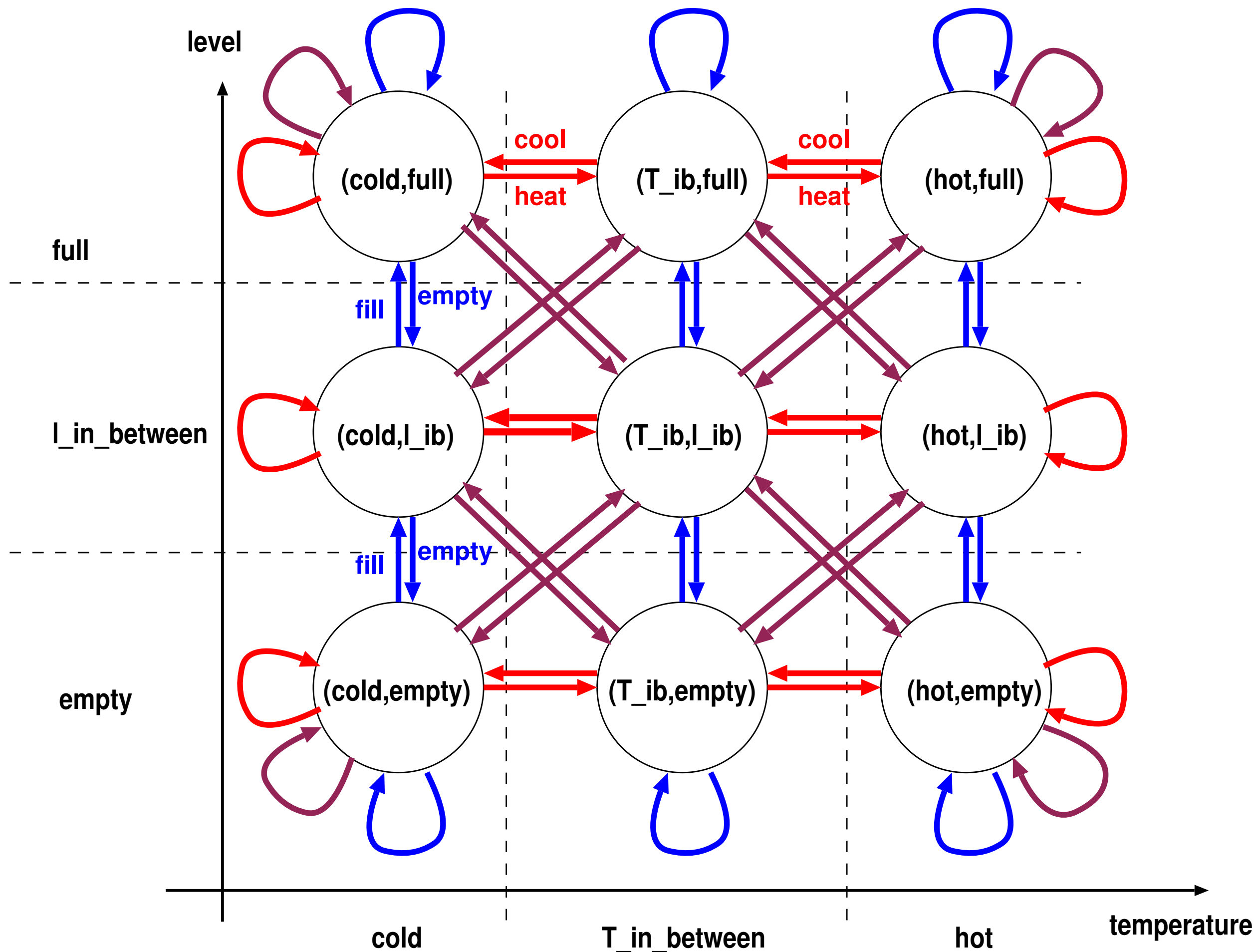
- Temperature T
- Level of liquid l

Outputs (sensors):

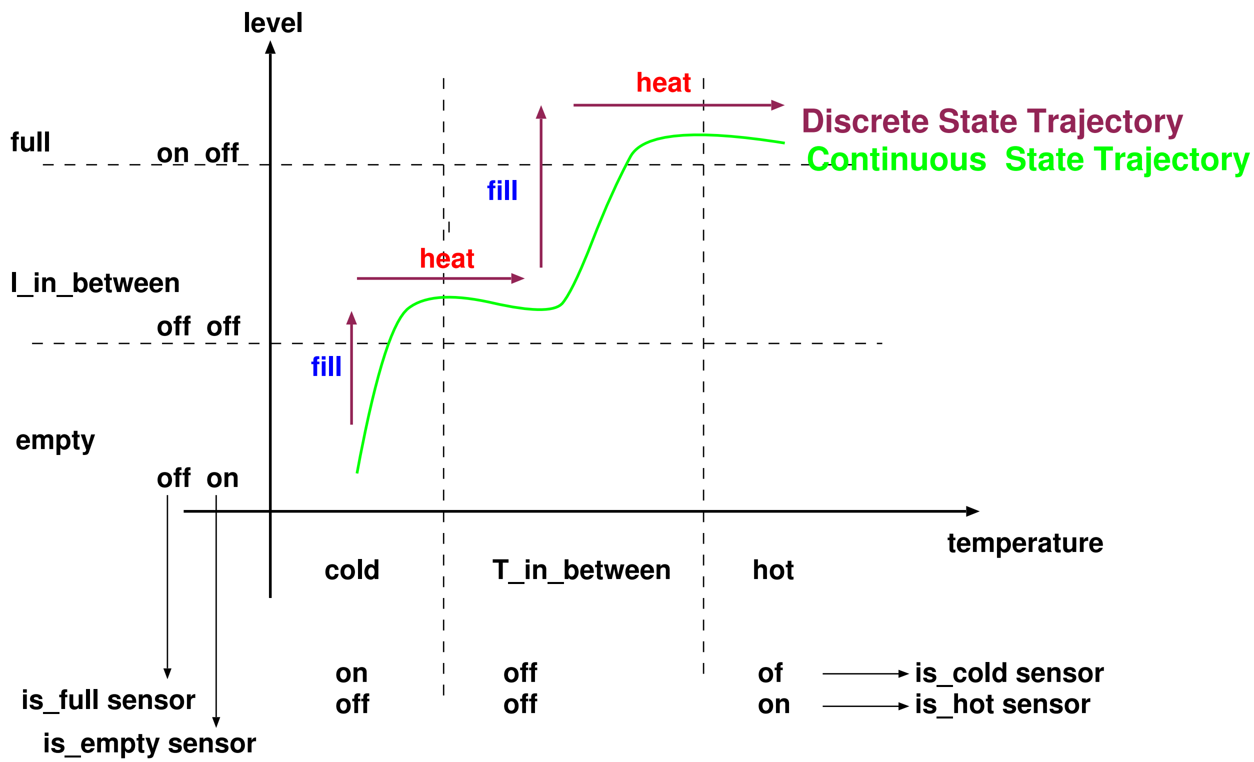
- $is\_low, is\_high, is\_cold, is\_hot$

$$
\begin{cases}
\frac{dT}{dt} & = & \frac{1}{l}\left[\frac{W}{c\rho A} - \phi(T - T_{in})\right] \\
\frac{dl}{dt} & = & \phi \\
is\_low & = & (l < l_{low}) \\
is\_high & = & (l > l_{high}) \\
is\_cold & = & (T < T_{cold}) \\
is\_hot & = & (T > T_{hot})
\end{cases}
$$

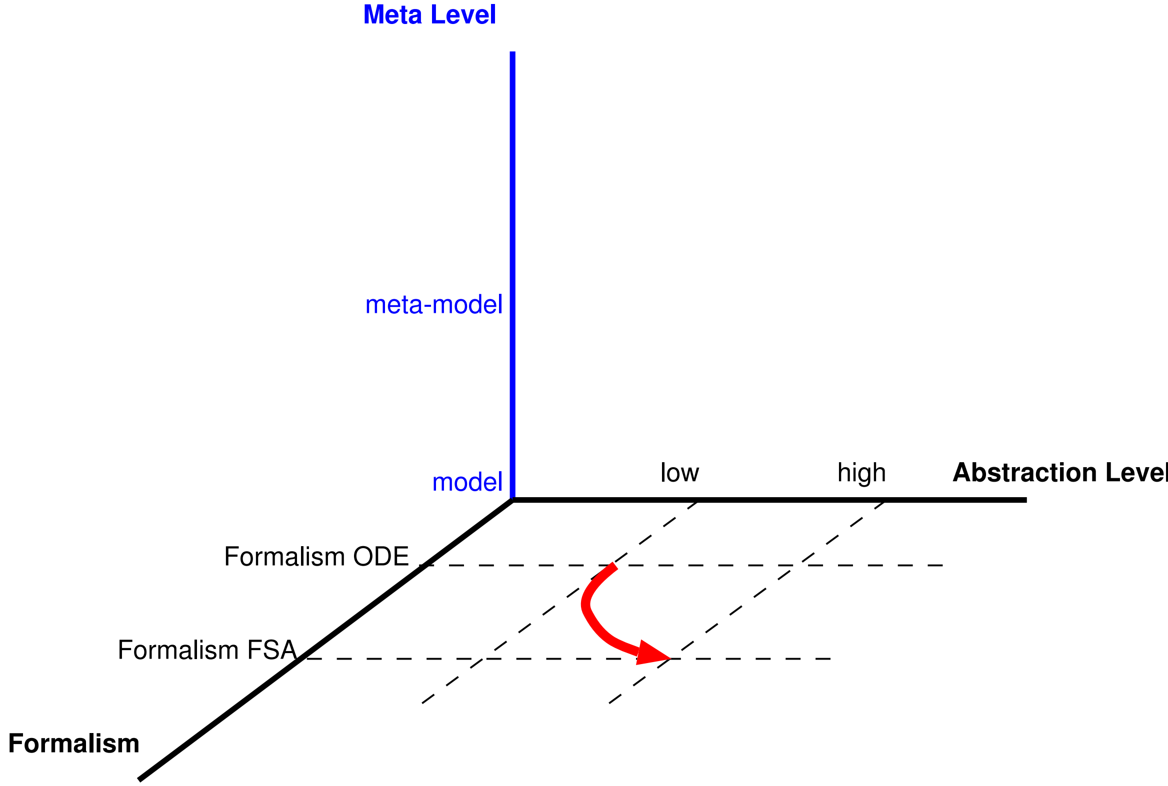
High-level (discrete) view, FSA formalism



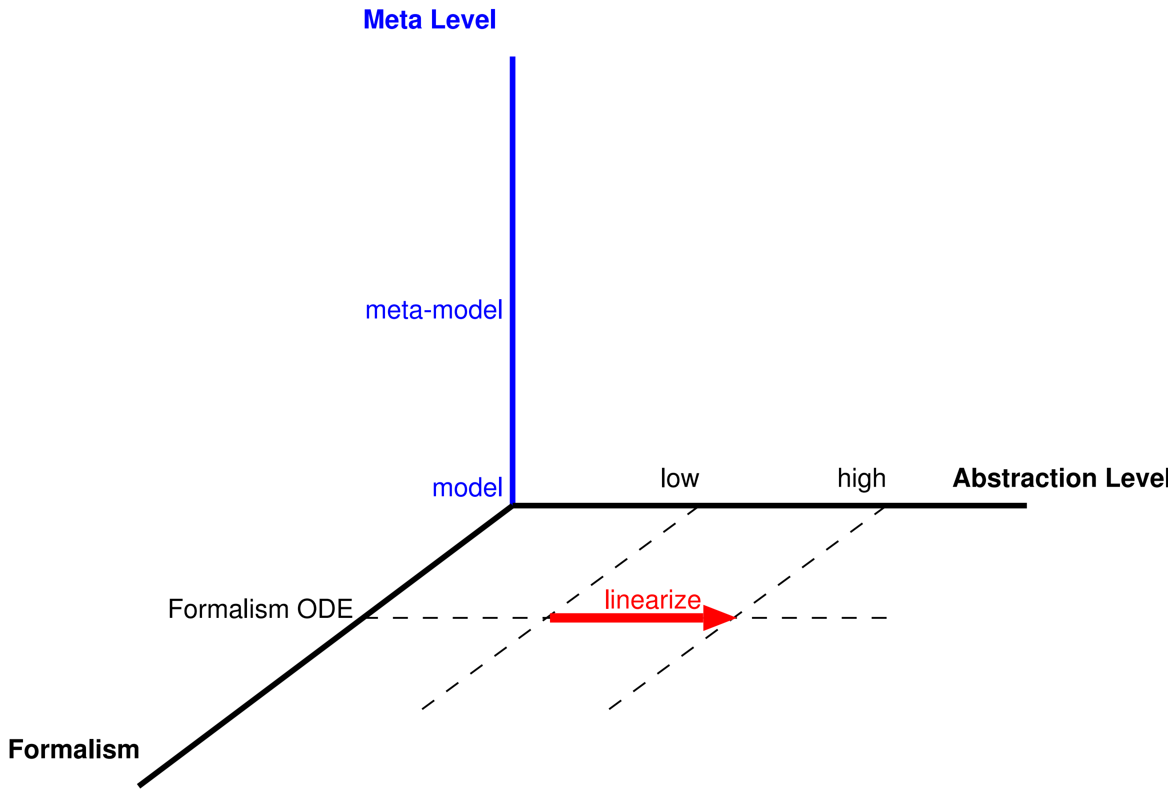
Levels of abstraction/views: trajectories



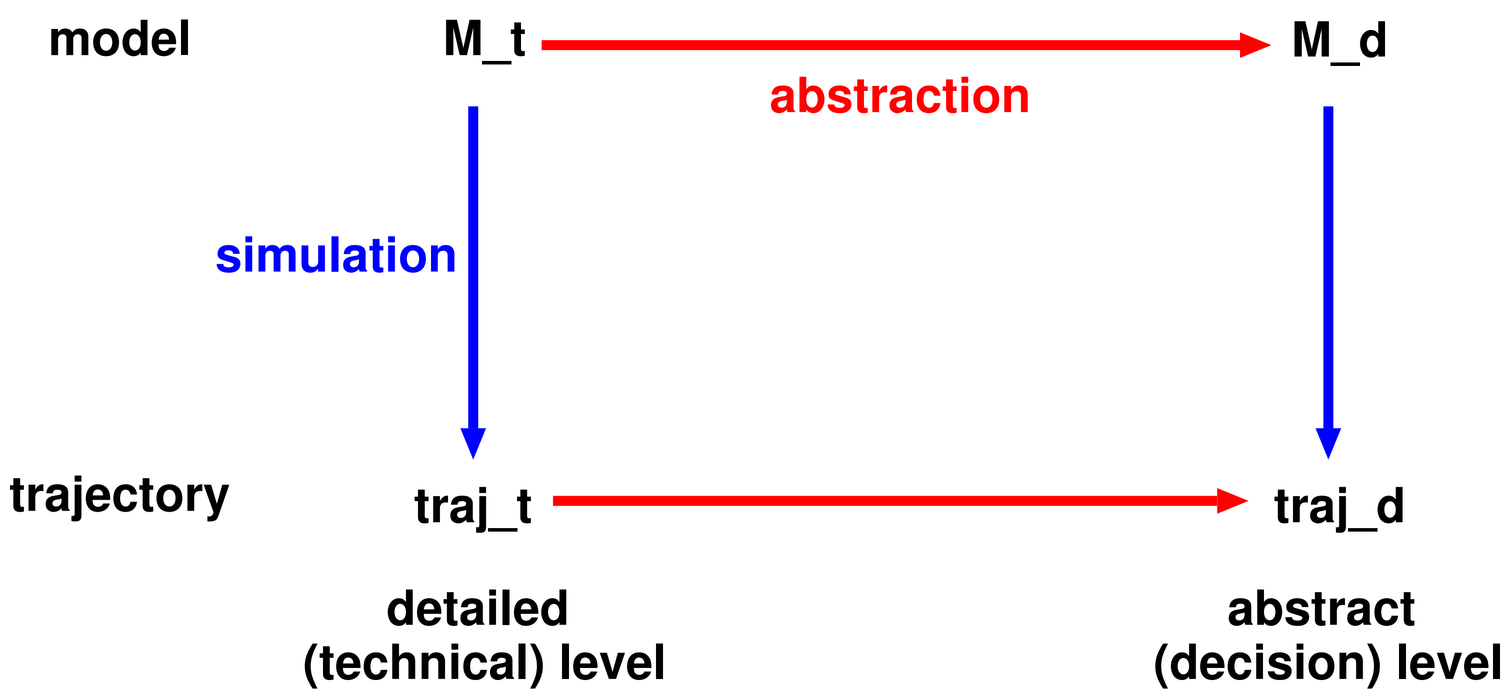
Multi-paradigm dimensions: abstraction/formalism



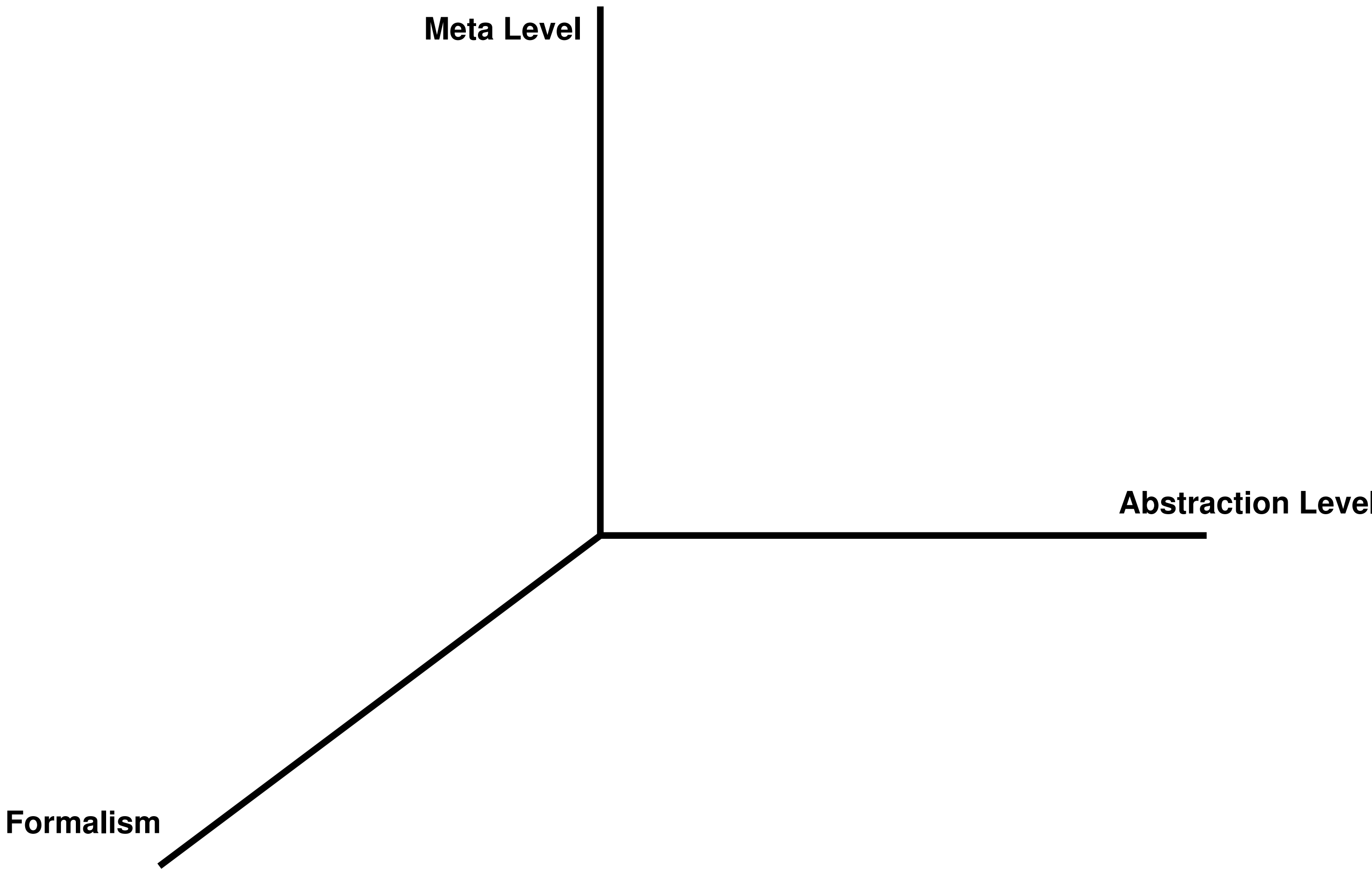
Multi-paradigm dimension: abstraction



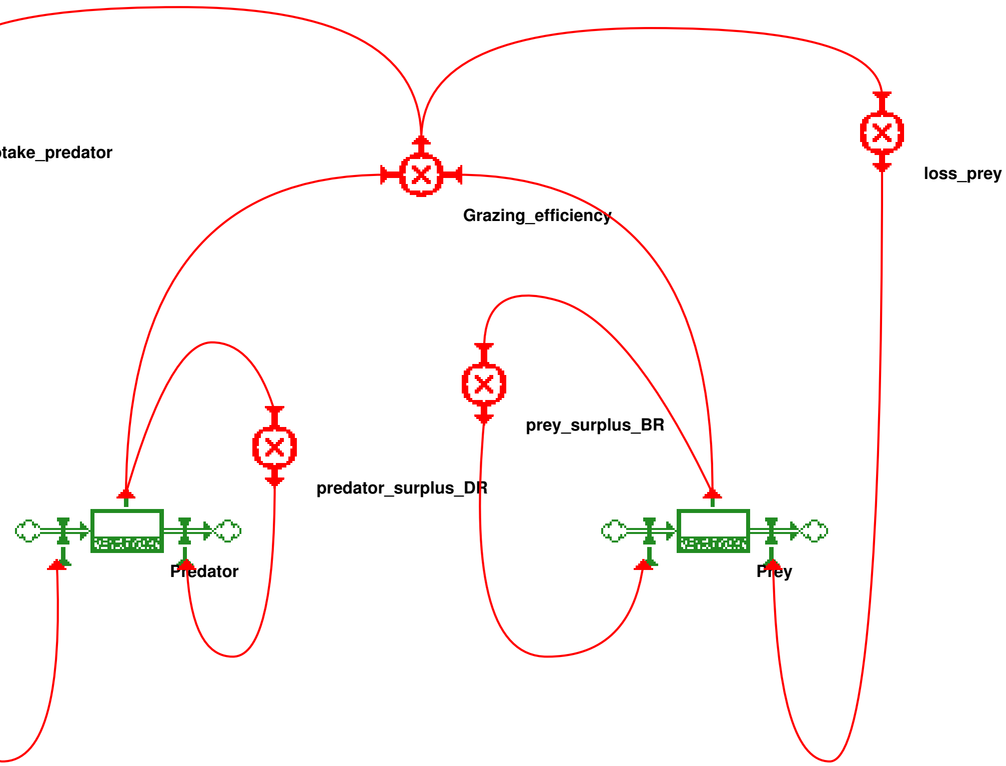
Levels of abstraction/views: morphism



Multi-paradigm dimensions: formalisms

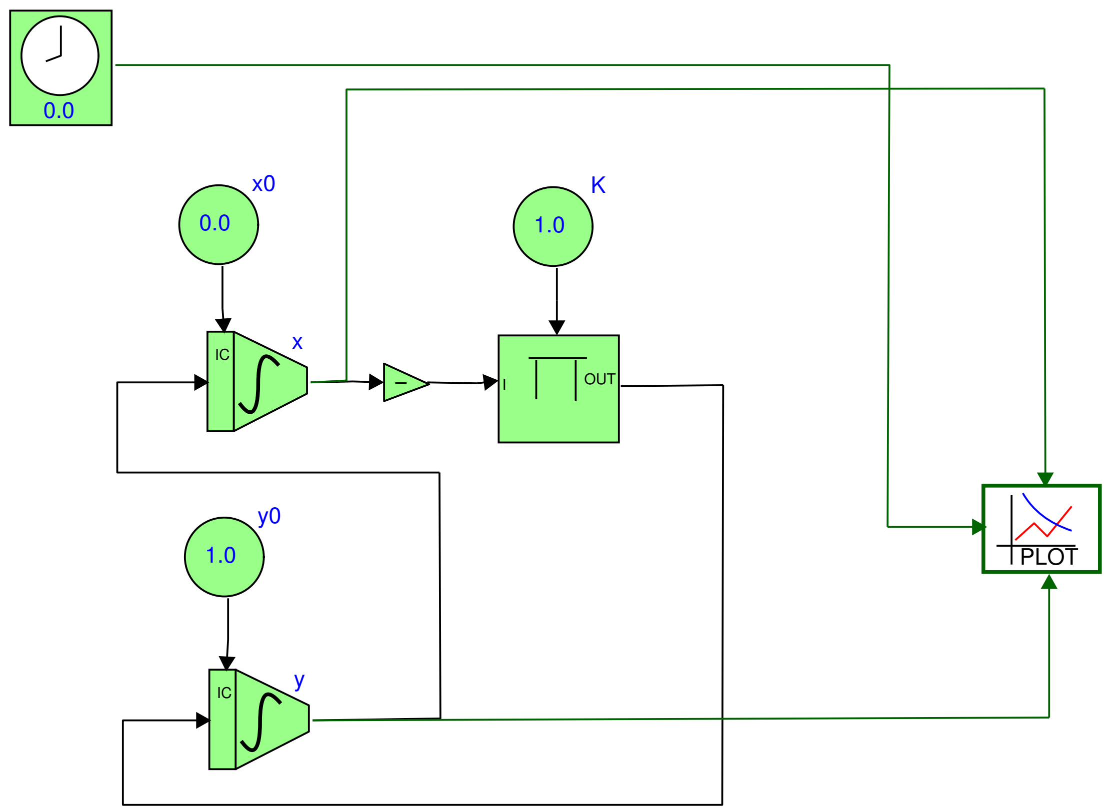


Forrester System Dynamics model of Predator-Prey

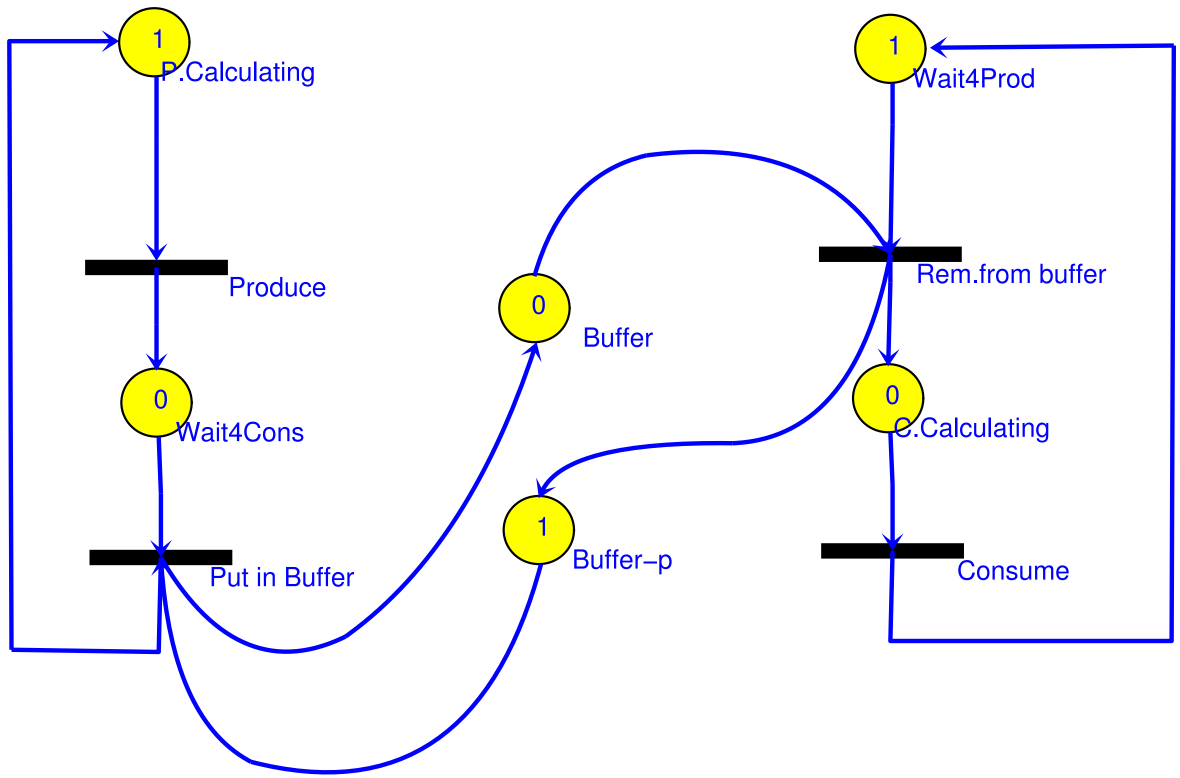


2–species predator–prey system

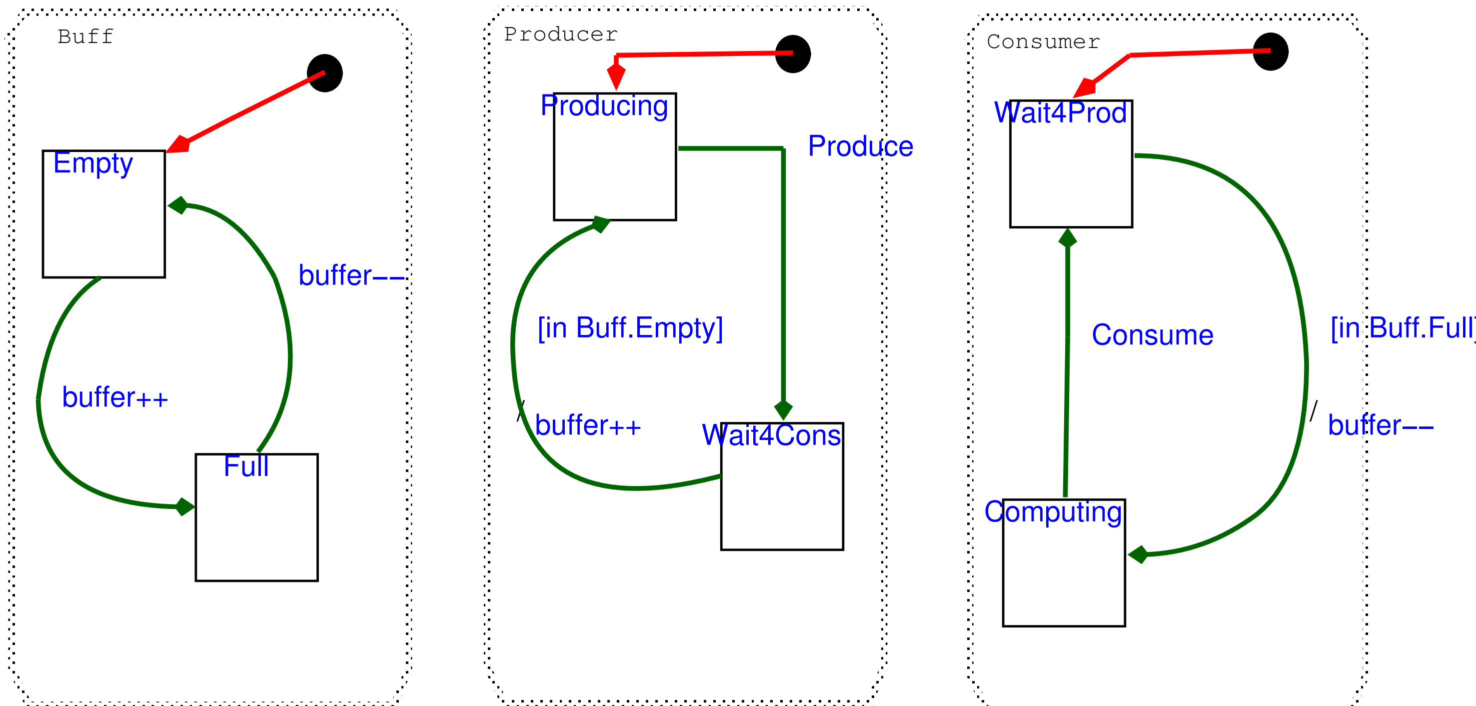
Causal Block Diagram model of Harmonic Oscillator



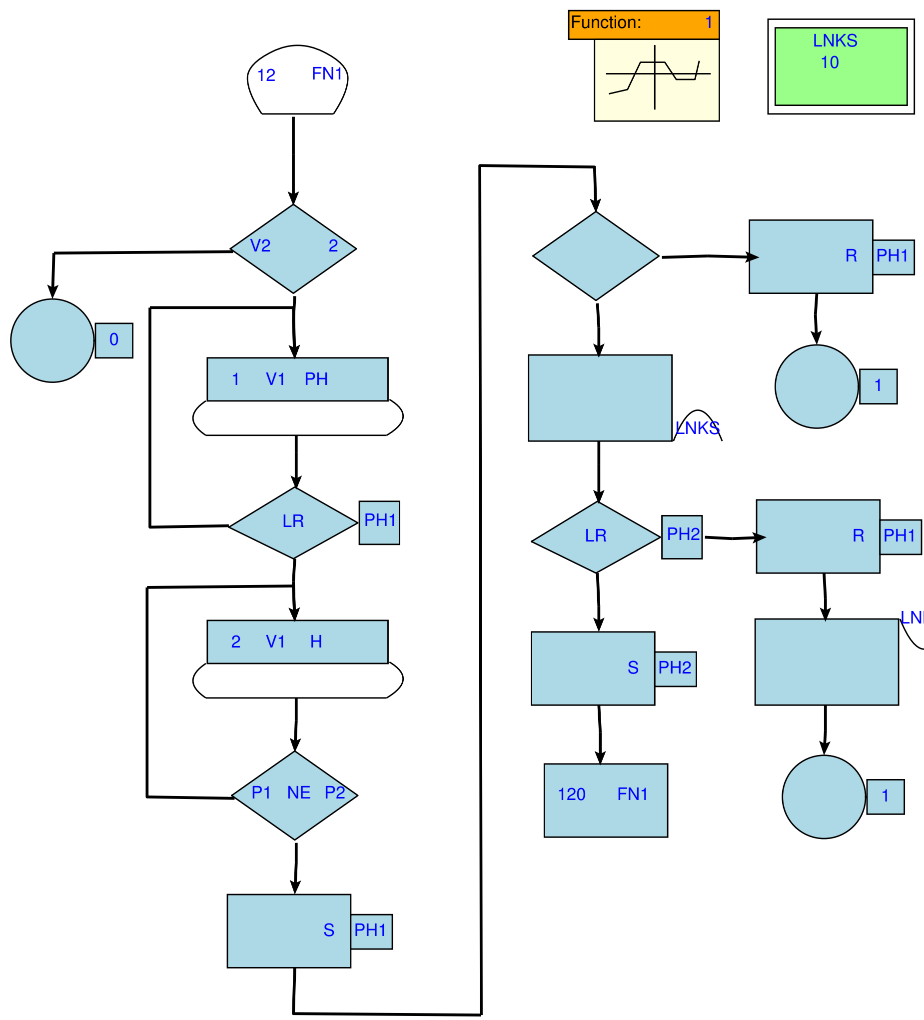
Petri Net model of Producer Consumer



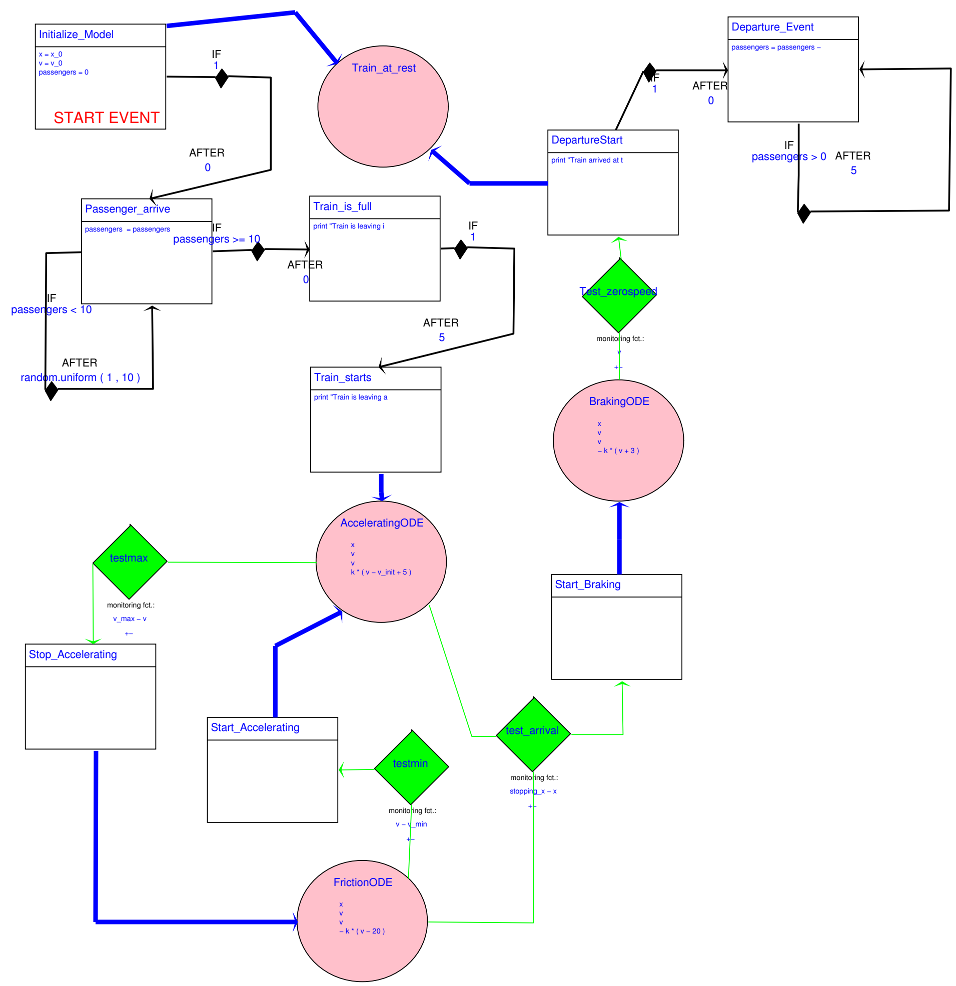
Statechart model of Producer Consumer



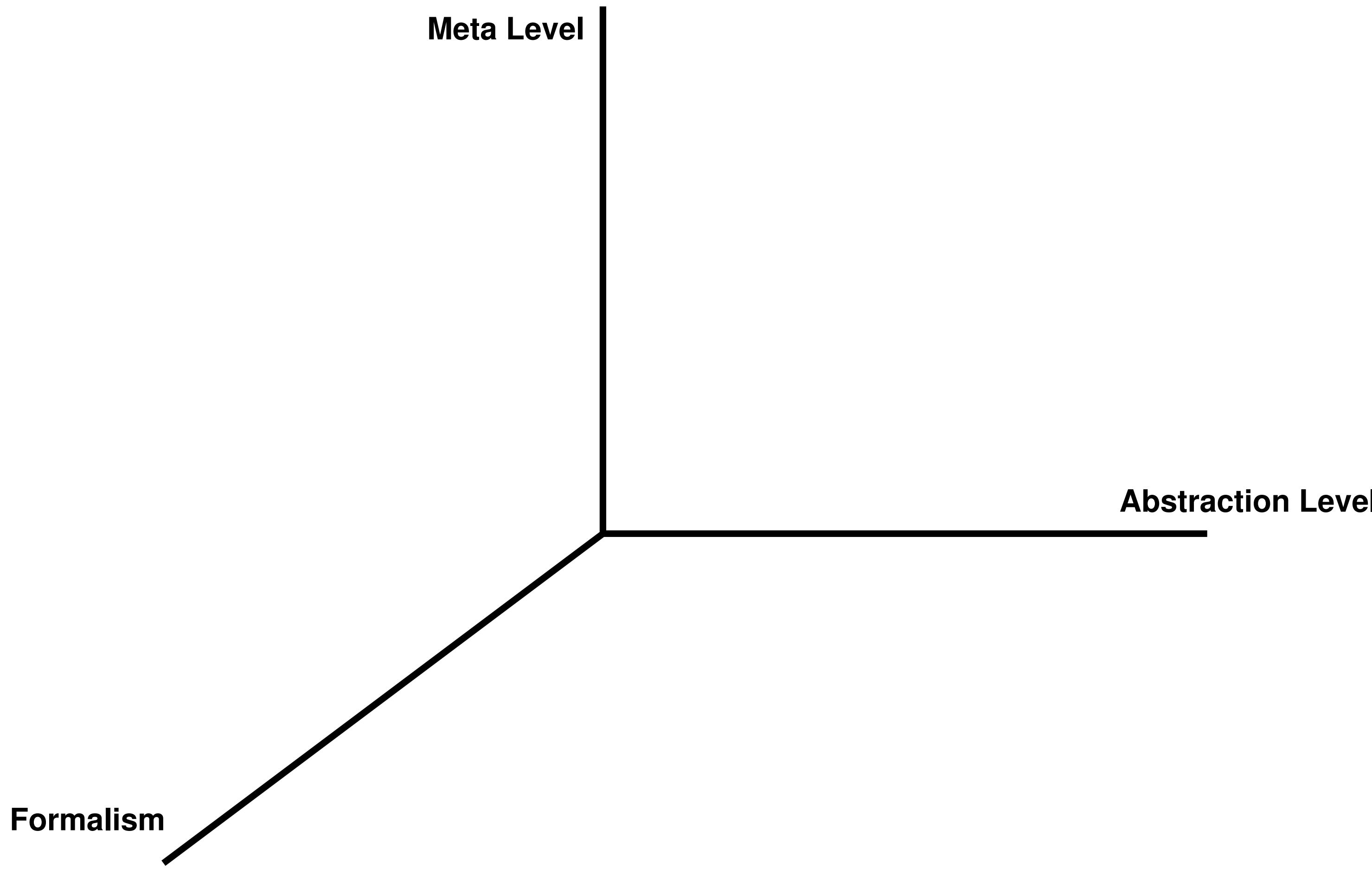
GPSS model of Telephone Exchange



Event Scheduling + DAE model of a Train



Multi-paradigm dimensions: meta

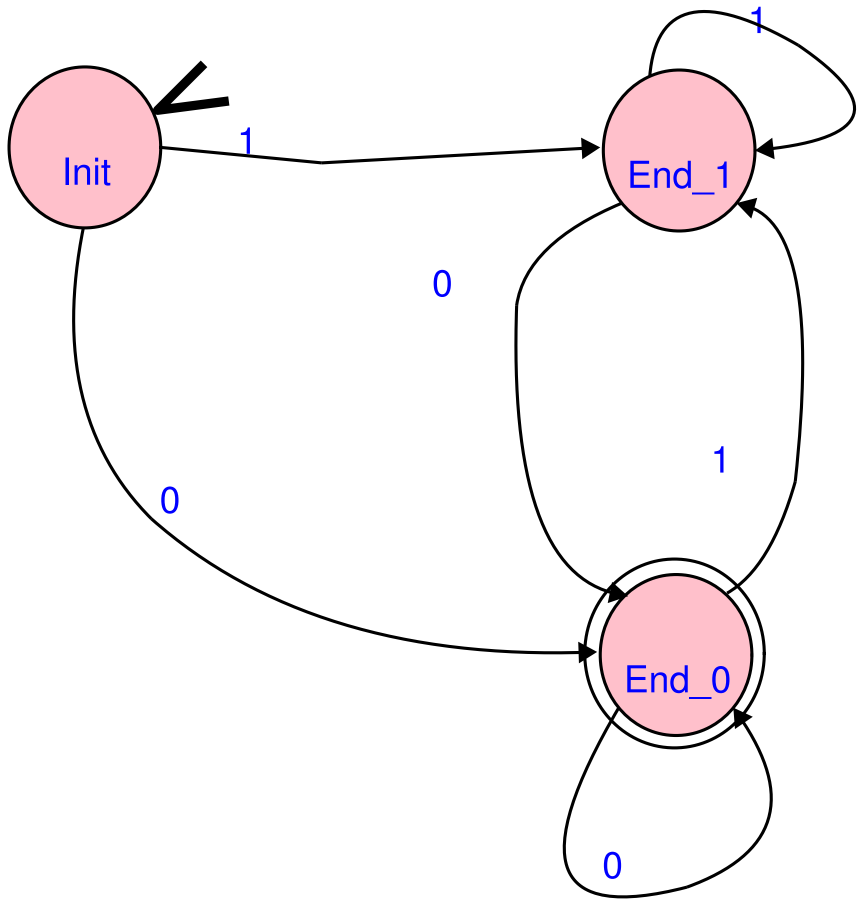


What is Meta-modelling ?

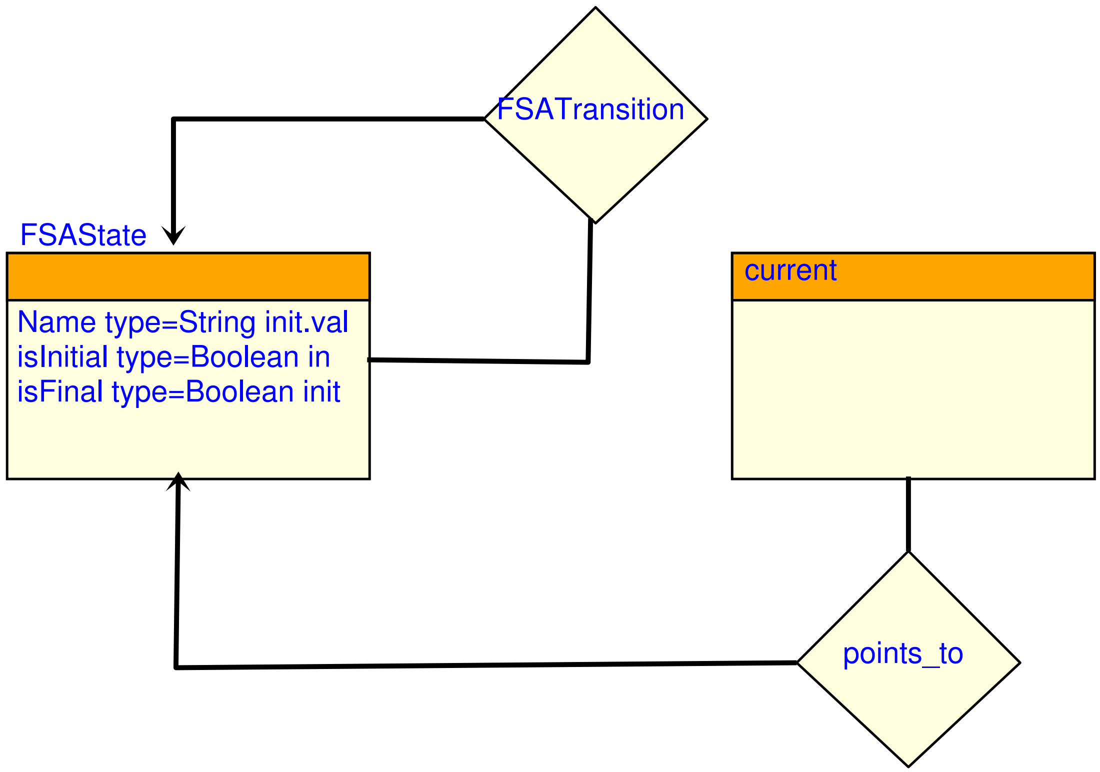
- A meta-model is **a model of** a modelling formalism
- A meta-model is itself a model. Its syntax and semantics are governed by the formalism it is described in. That formalism can be modelled in a meta-meta-model.
- As a meta-model is a model, we can reason about it, manipulate it, ... In particular,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

FSA model of Even Binary Number recognizer



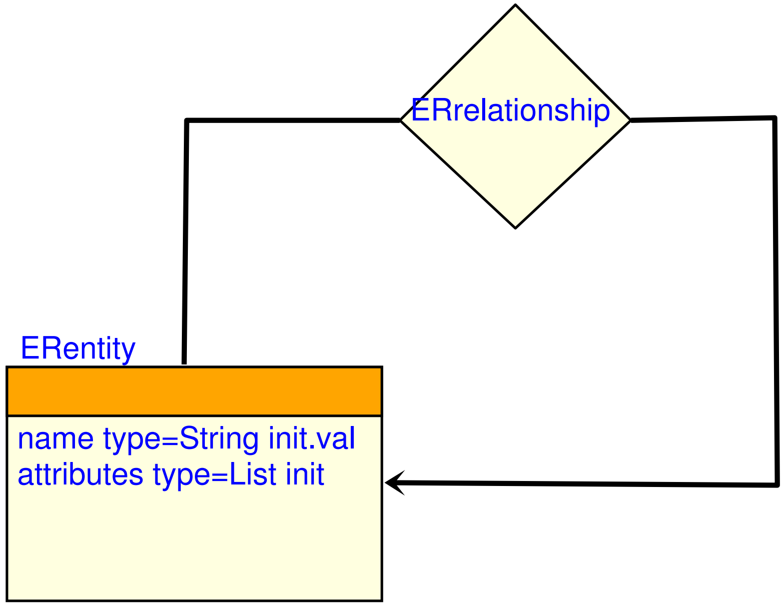
ER model of the FSA formalism syntax (meta-model)



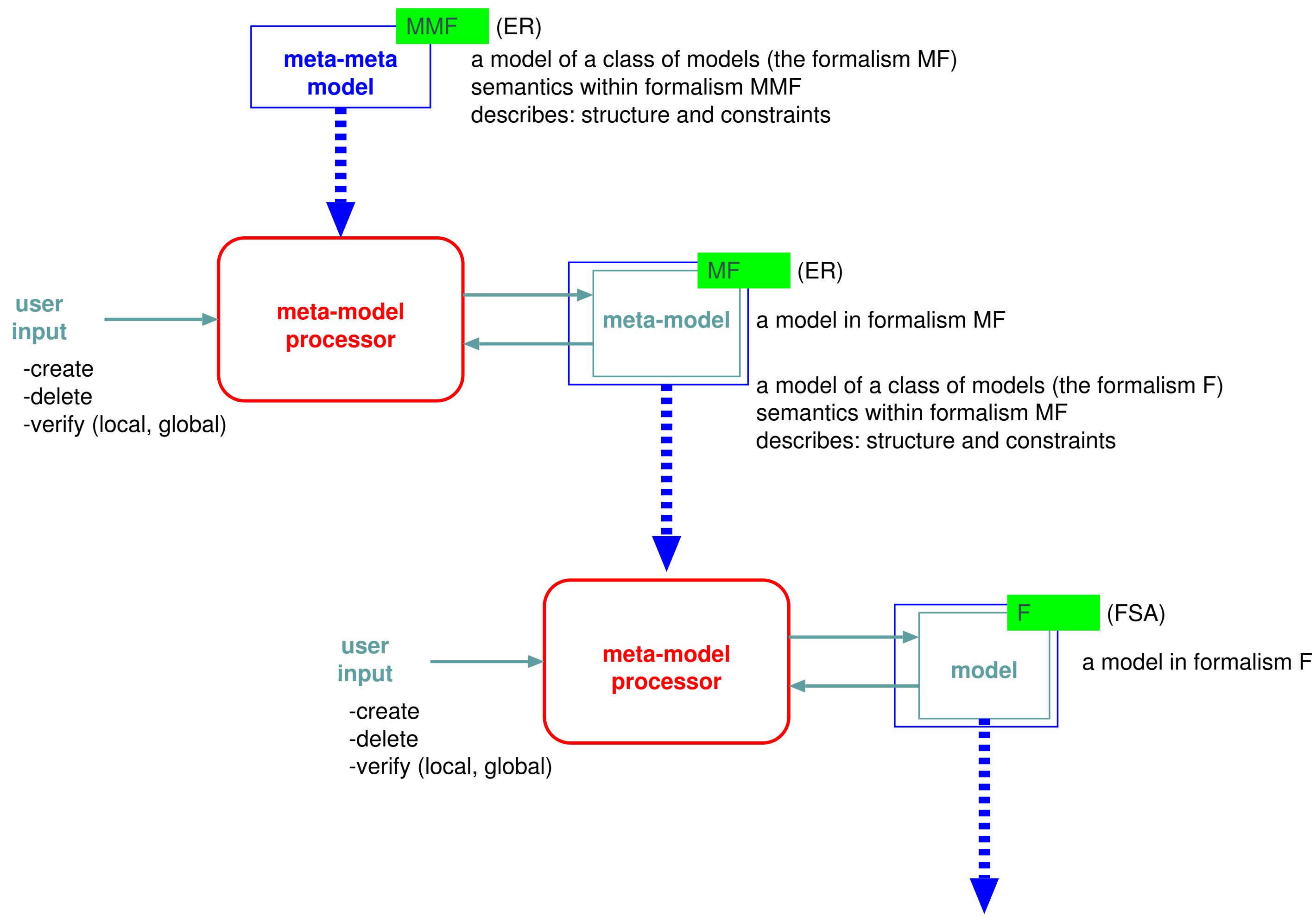
ER formalism + constraints (OCL/Python)

```
# 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)
```

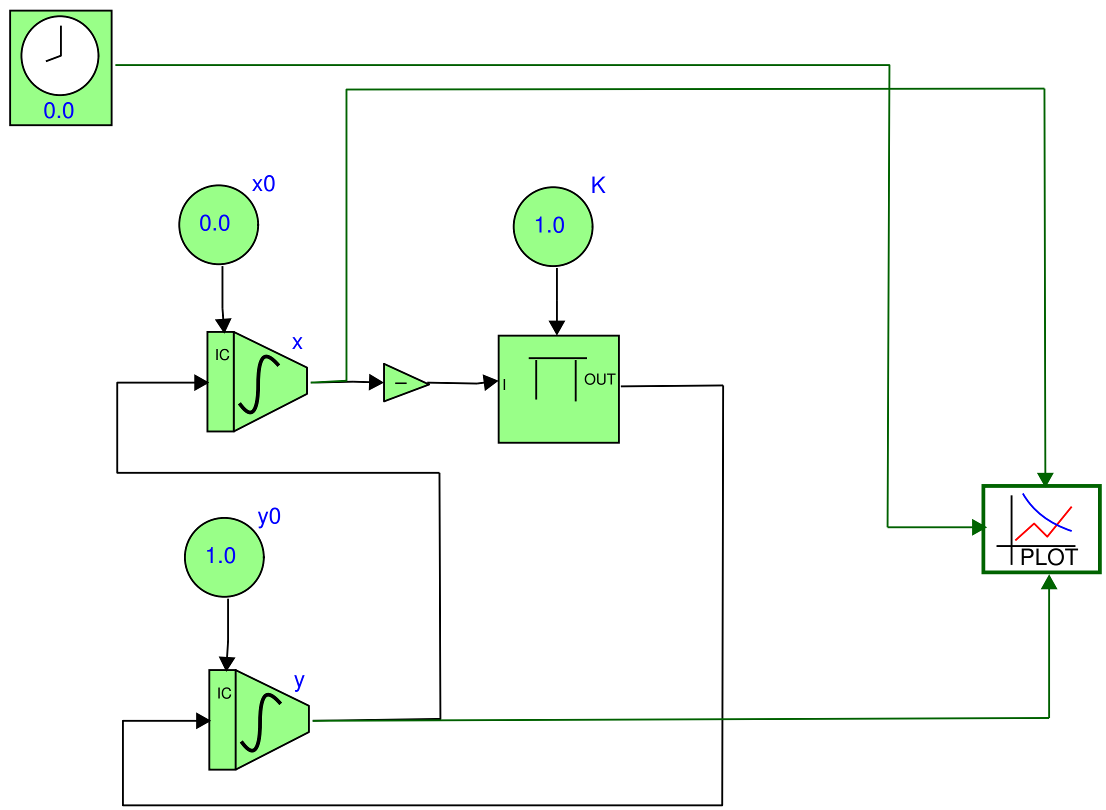
ER model of the ER formalism (meta-meta-model)



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

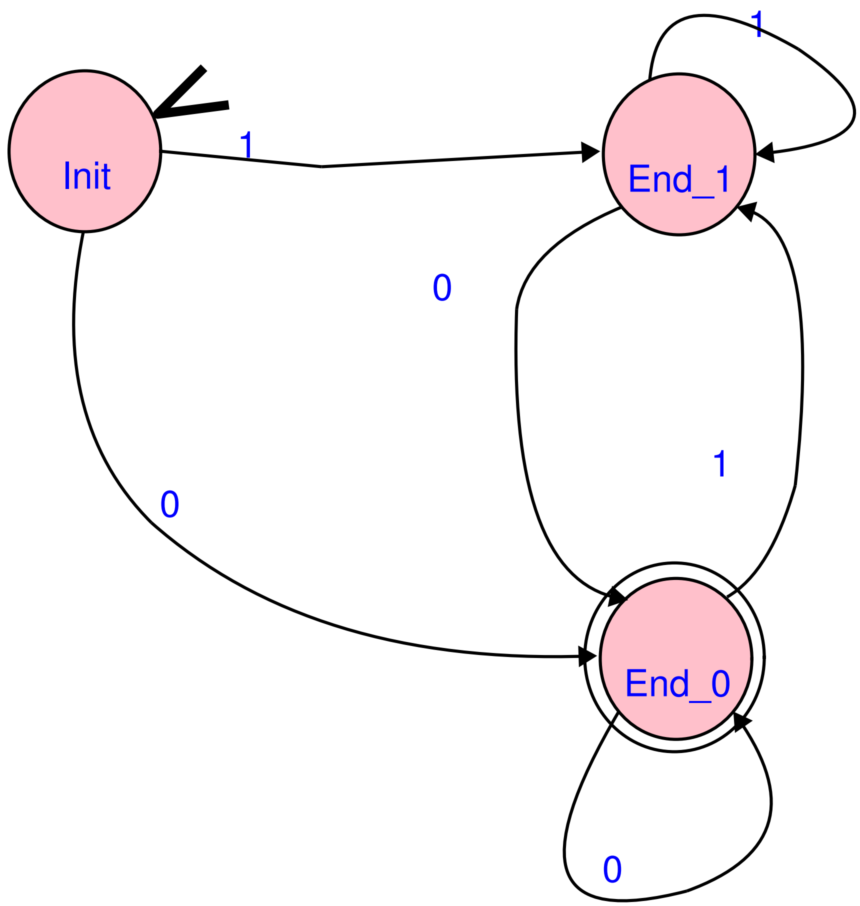
Causal Block Diagram Semantics ?



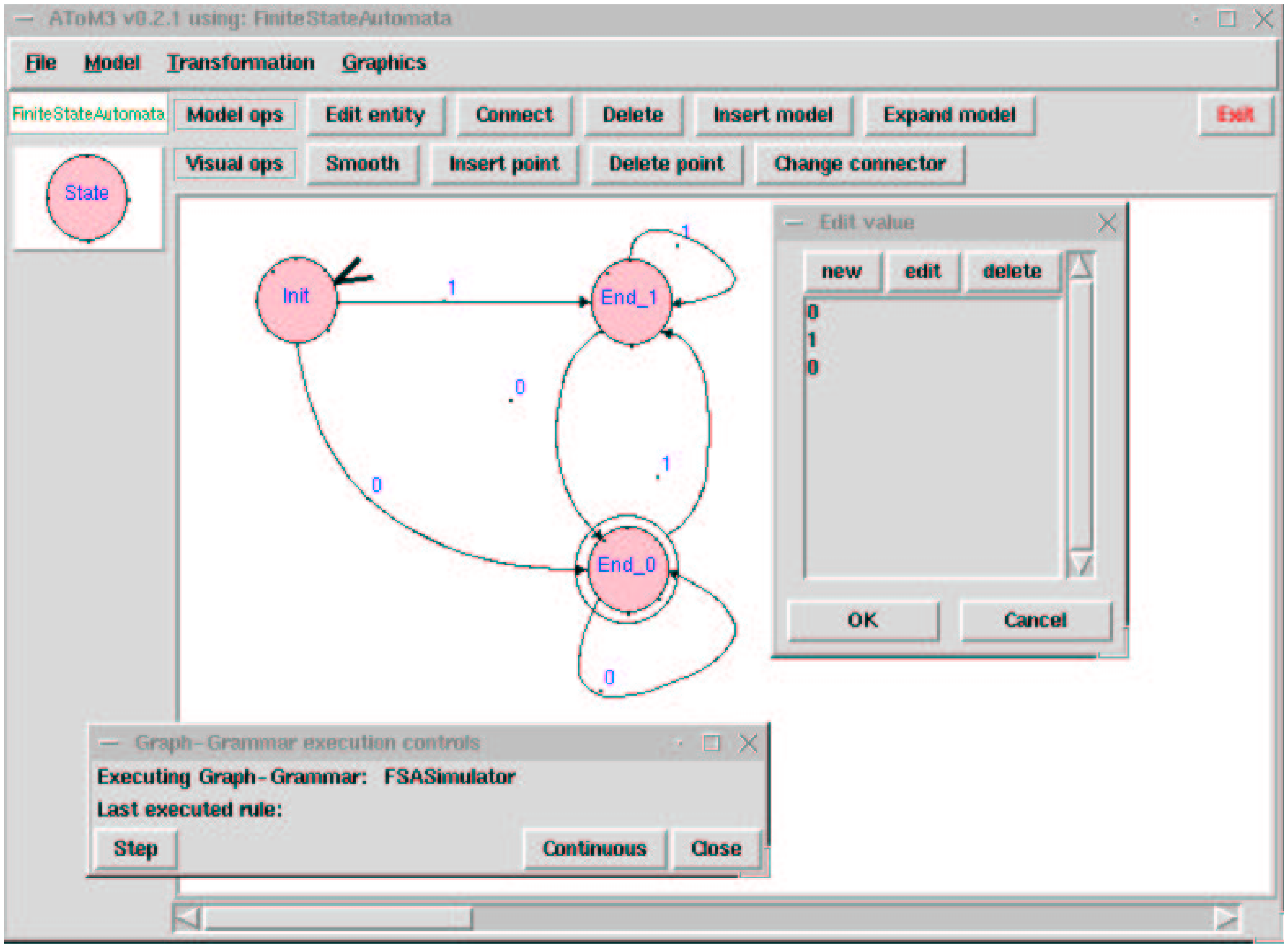
Causal Block Diagram Denotational Semantics

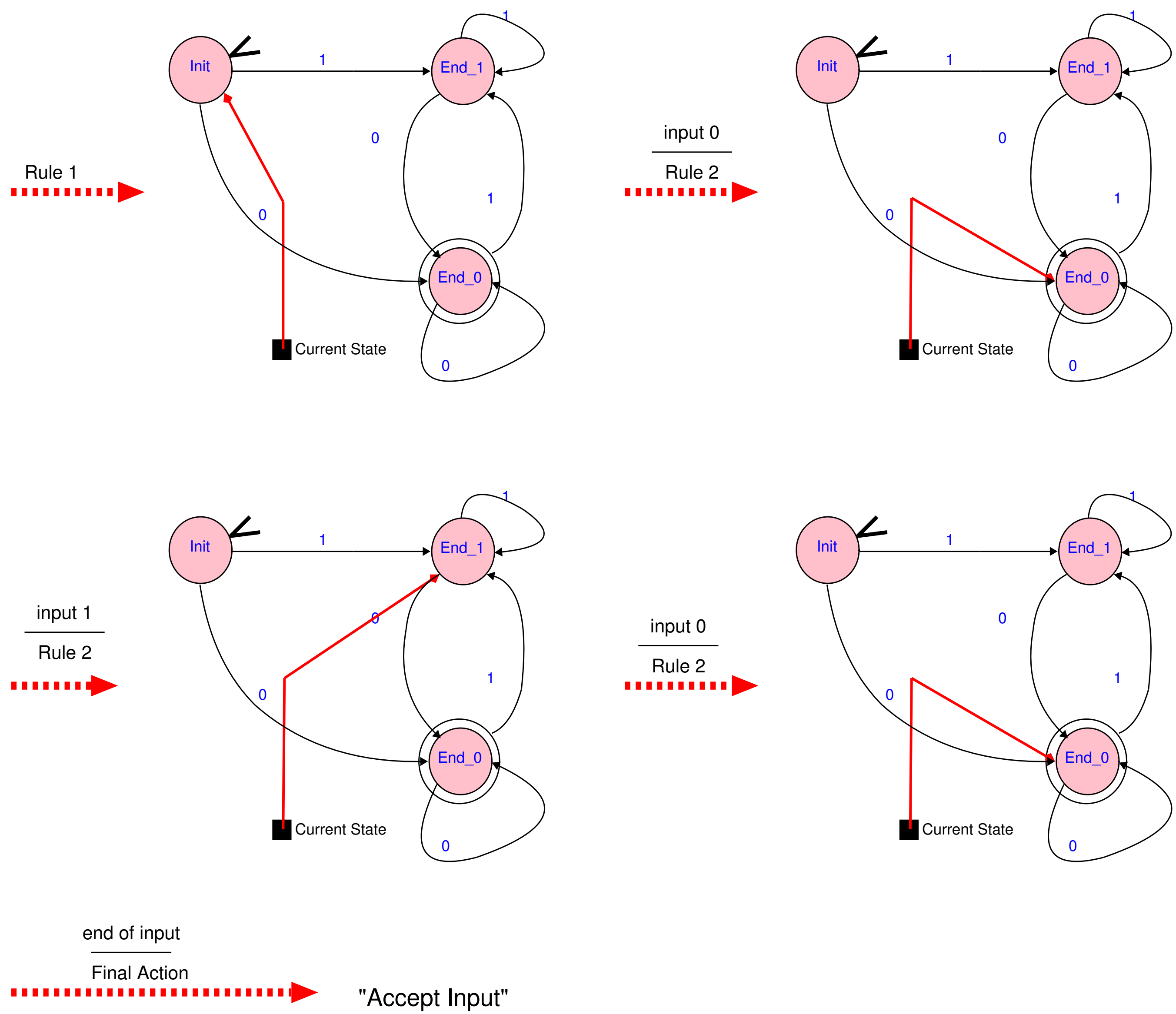
$$
\begin{cases}
\frac{dx}{dt} = y & x(0) = 0 \\
\frac{dy}{dt} = -Kx & y(0) = 1 \\
K = 1
\end{cases}
$$

FSA model Operational Semantics ?

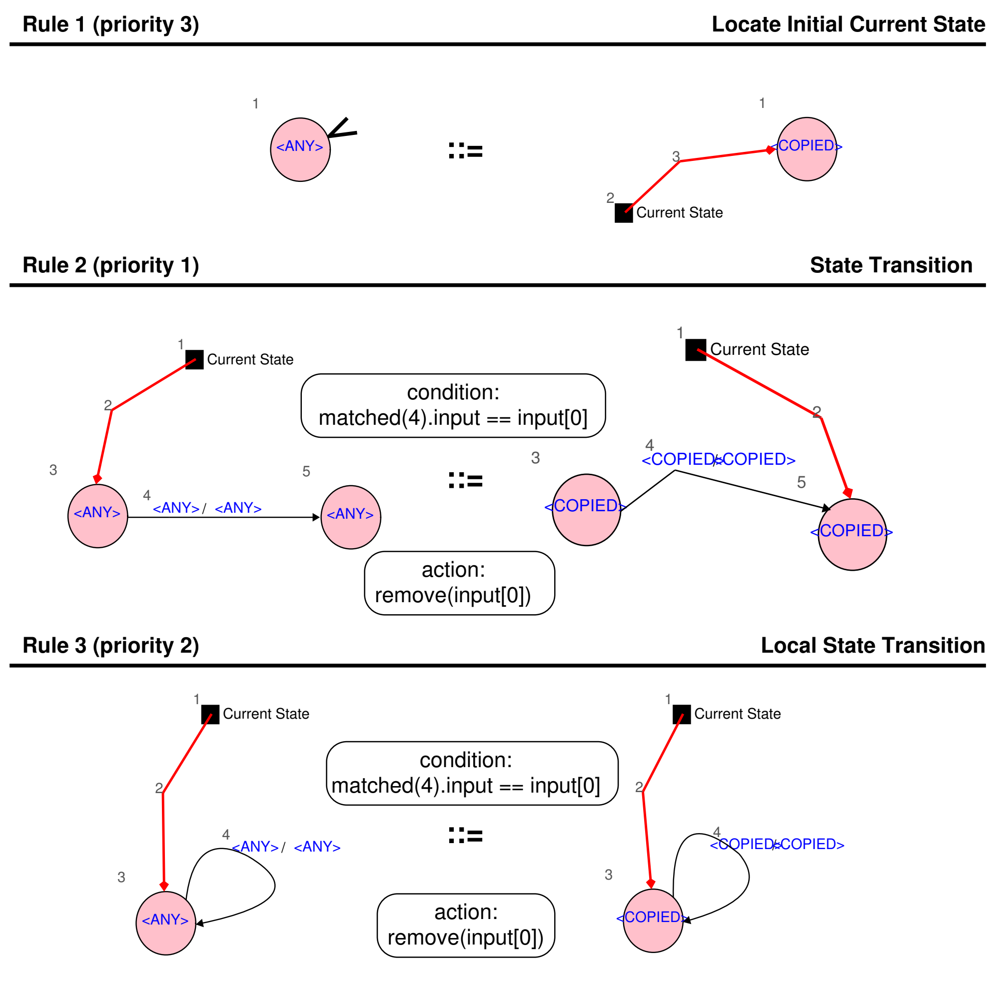


Simulation steps

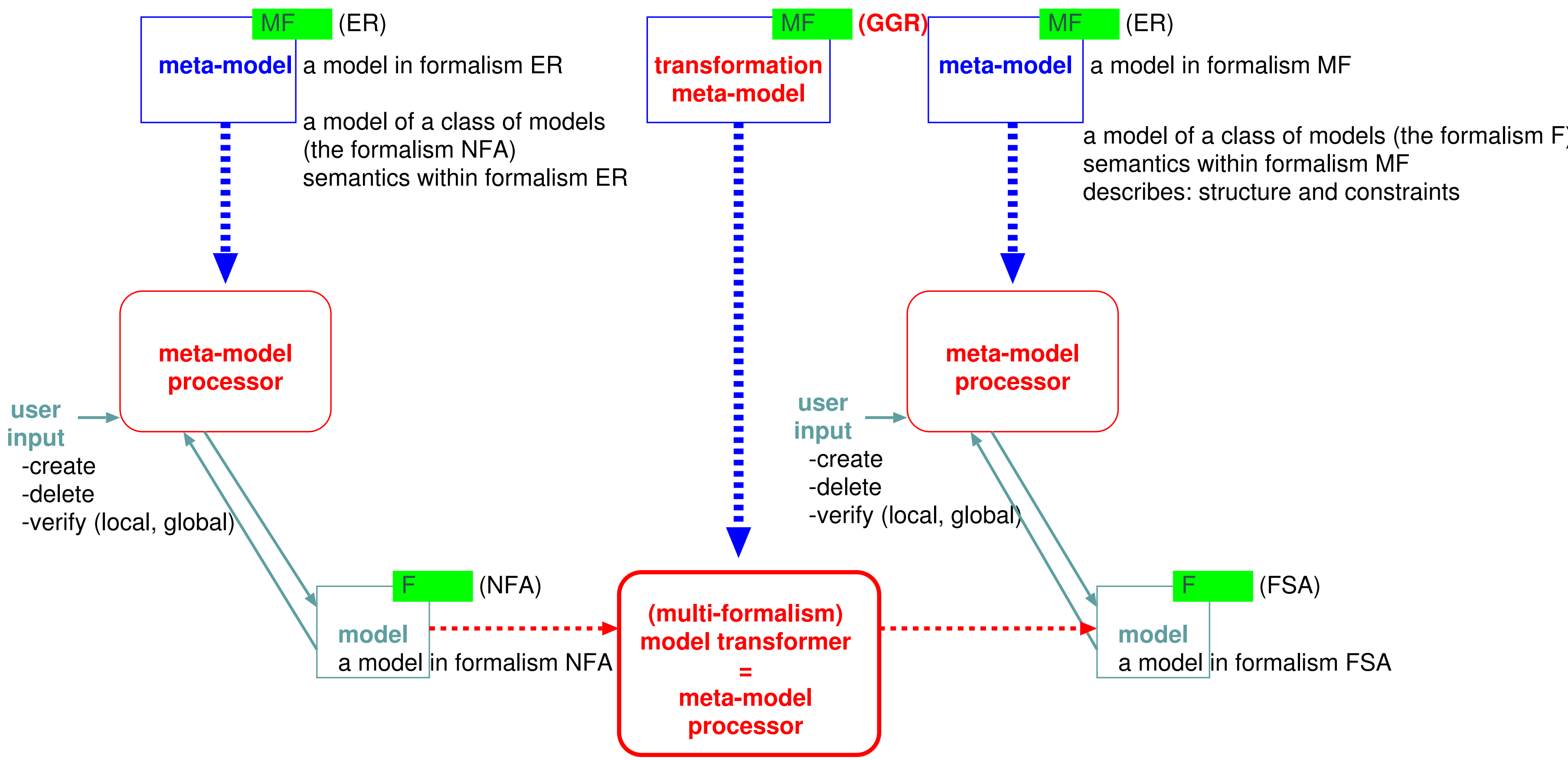



Rule 1
Init
1
End_1
1
0
1
0
End_0
0
Current State
input 0
Rule 2
Init
1
End_1
1
0
1
0
End_0
0
Current State
input 1
Rule 2
Init
1
End_1
1
0
1
0
End_0
0
Current State
input 0
Rule 2
Init
1
End_1
1
0
1
0
End_0
0
Current State
end of input
Final Action
"Accept Input"

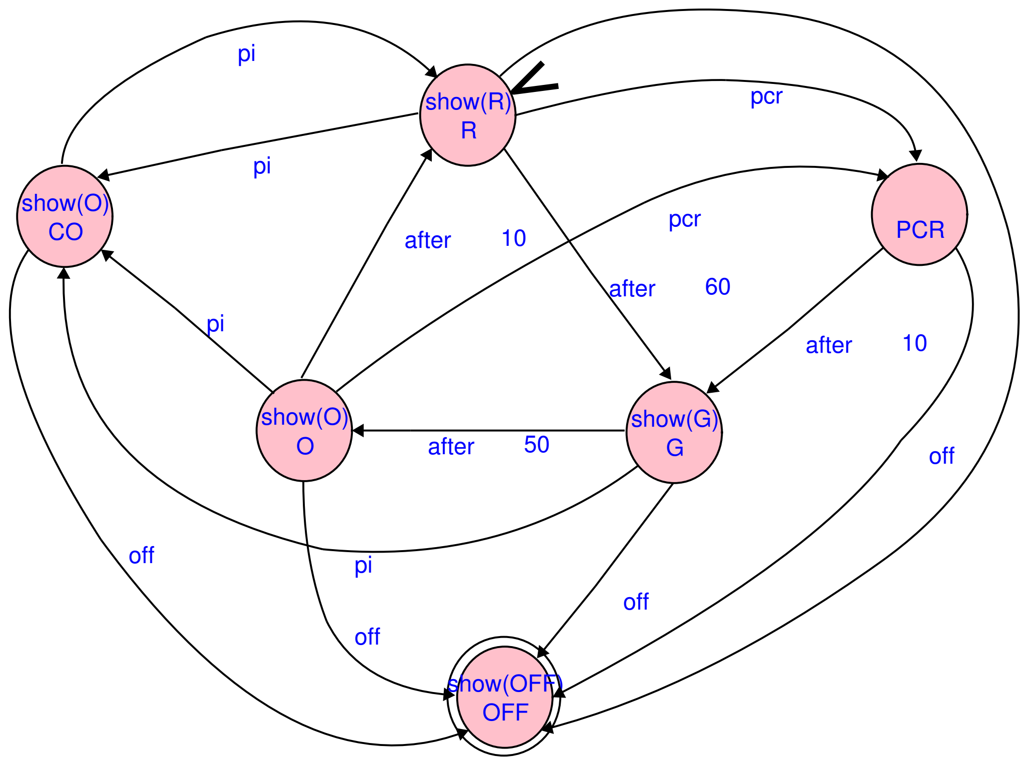
Graph Grammar model of FSA OpSem



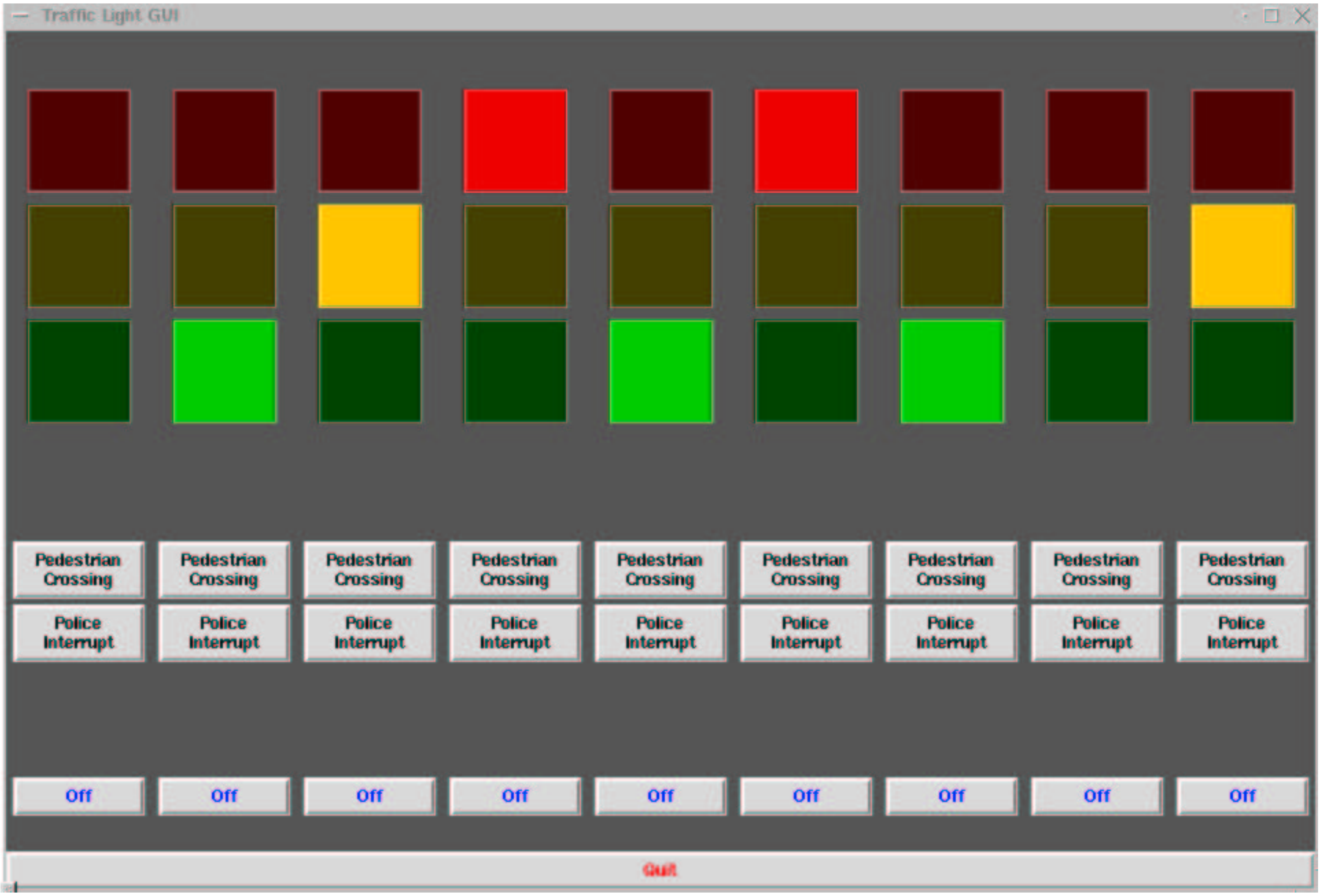
Model Transformation meta-specification



Timed Automata model of a Traffic Light + codegen



Generated Application

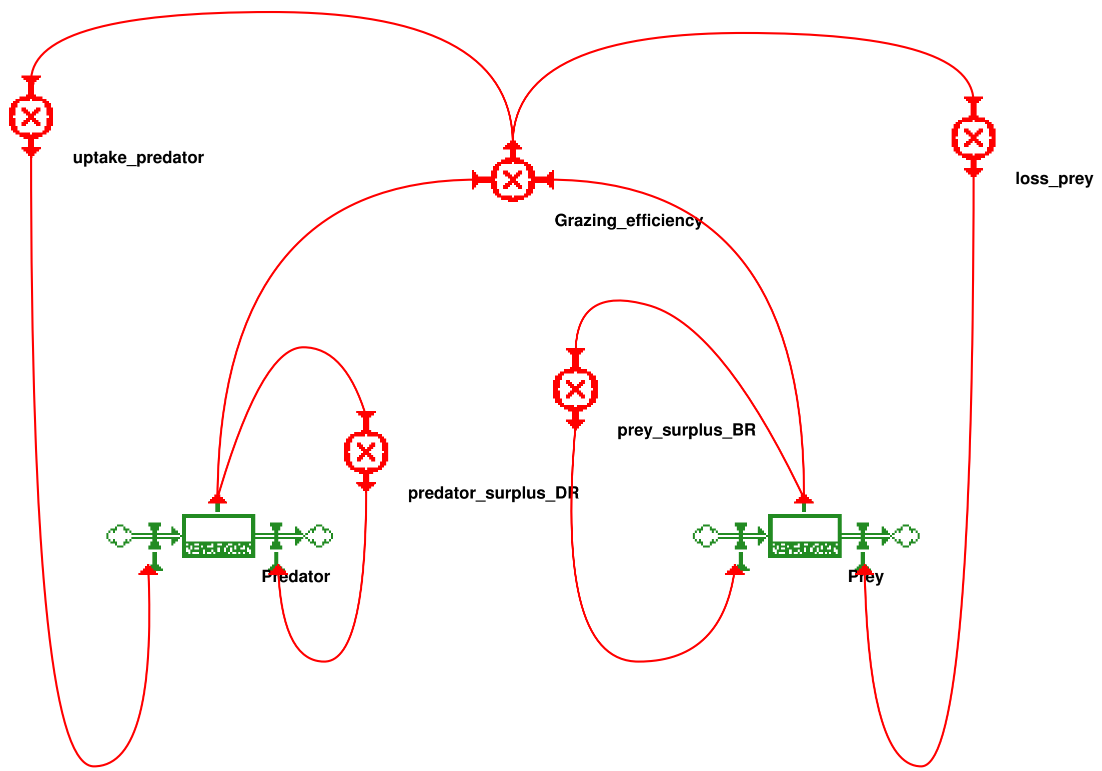


Model Transformation Uses (1)

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

May model transformation as Graph Grammar

FSD Denotational Semantics ?



2–species predator–prey system

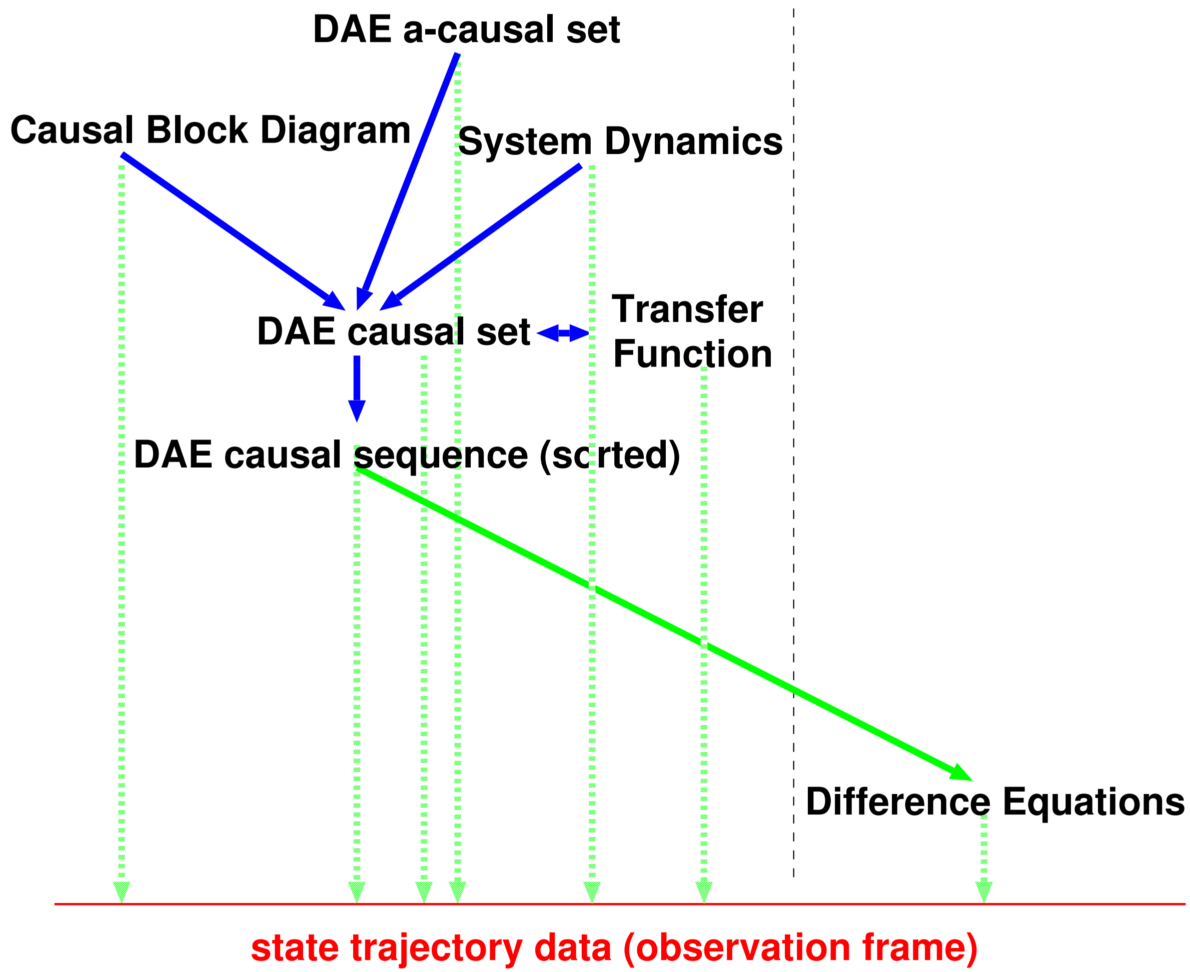
FSD denotational semantics in terms of DAE

- Semantics of “level” block:

$$\frac{d\ level}{dt} = BR - DR.$$

- Semantics of “algebraic” block: algebraic relationship between block’s I/O signals
- Semantics of the full model: set of components’ semantics equations.

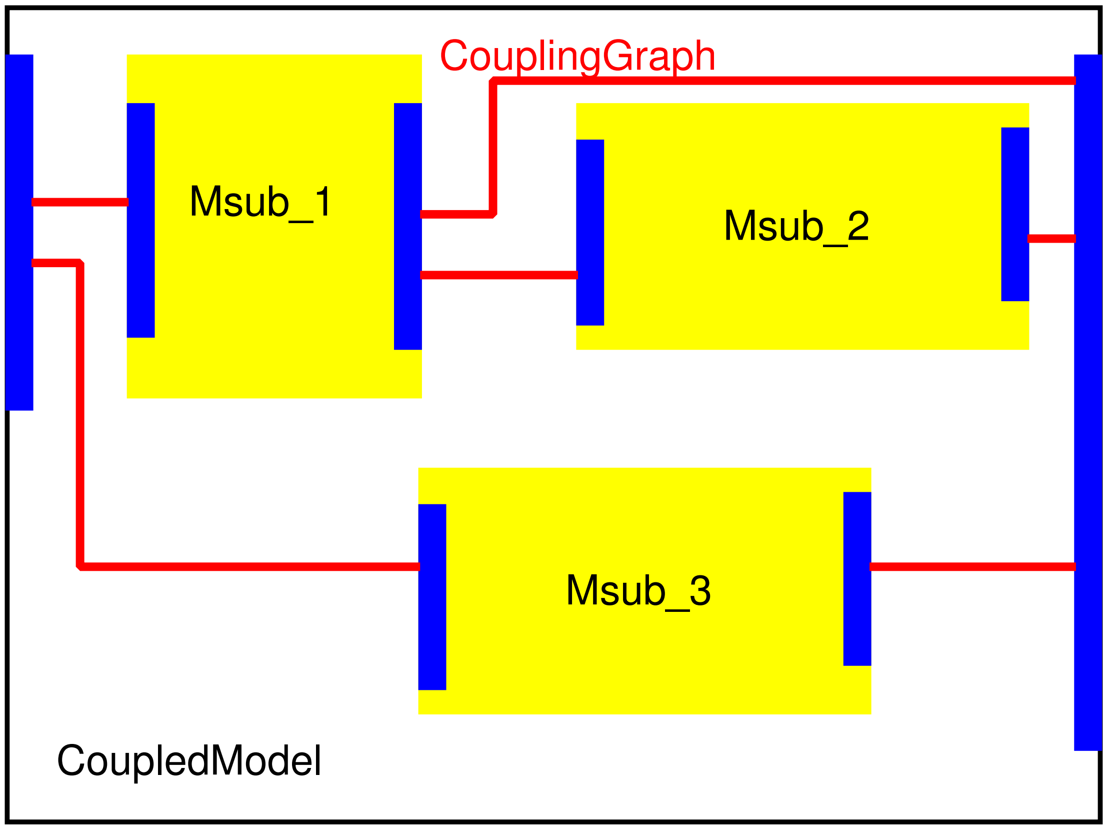
Formalism Transformation



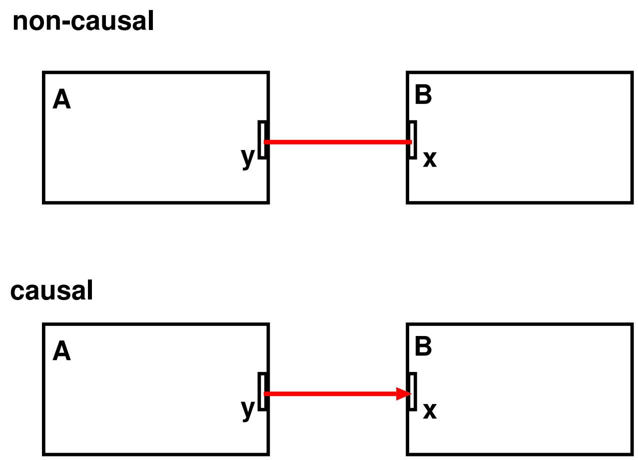
Formalism transformation uses (2)

- Add new formalisms without much effort (only Δ).
- Re-use lower level modelling/simulation environment.
- Answer questions at "optimal" level.
  1. System Dynamics: influences, domain-knowledge.
  2. DAE: algebraic dependency cycles.
  3. ALG + ODE: linear ?
  4. Trajectory, given initial conditions.
- Optimization possible at every level.
- Semantics of coupled multi-formalism models.

Compositional Modelling: Coupled Model (network)



Closure under Coupling/Composition: Block Diagram



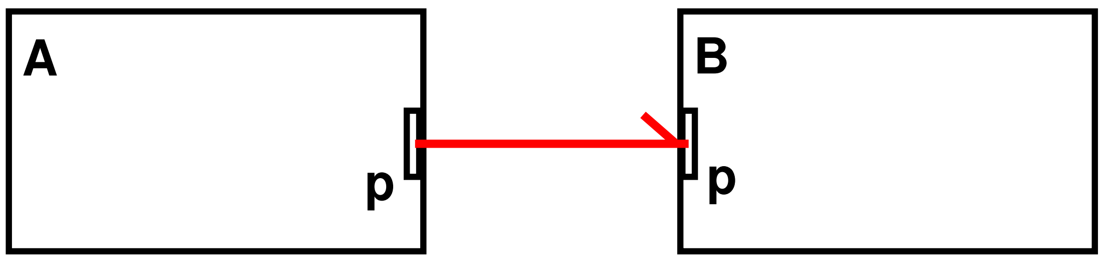
Non-Causal:

$$A.y = B.x$$

Causal:

$$B.x := A.y$$

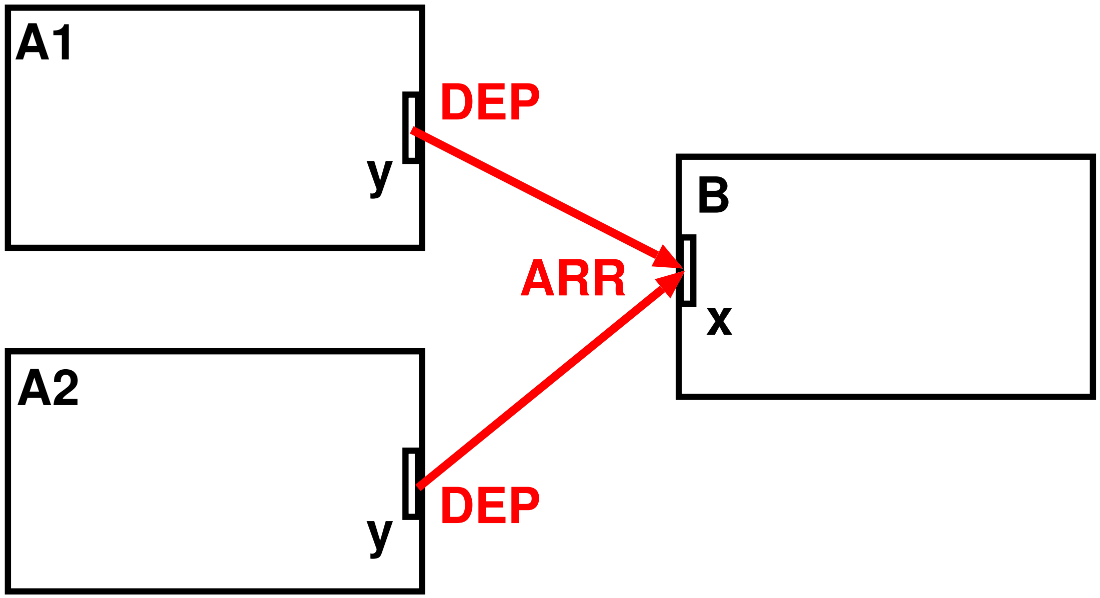
Closure under Coupling/Composition: non-causal Bond Graph



$$A.p.effort = B.p.effort$$

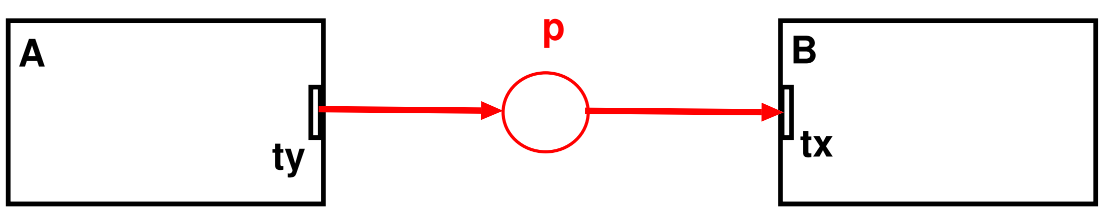
$$A.p.flow = B.p.flow$$

Closure under Coupling/Composition: Discrete Event



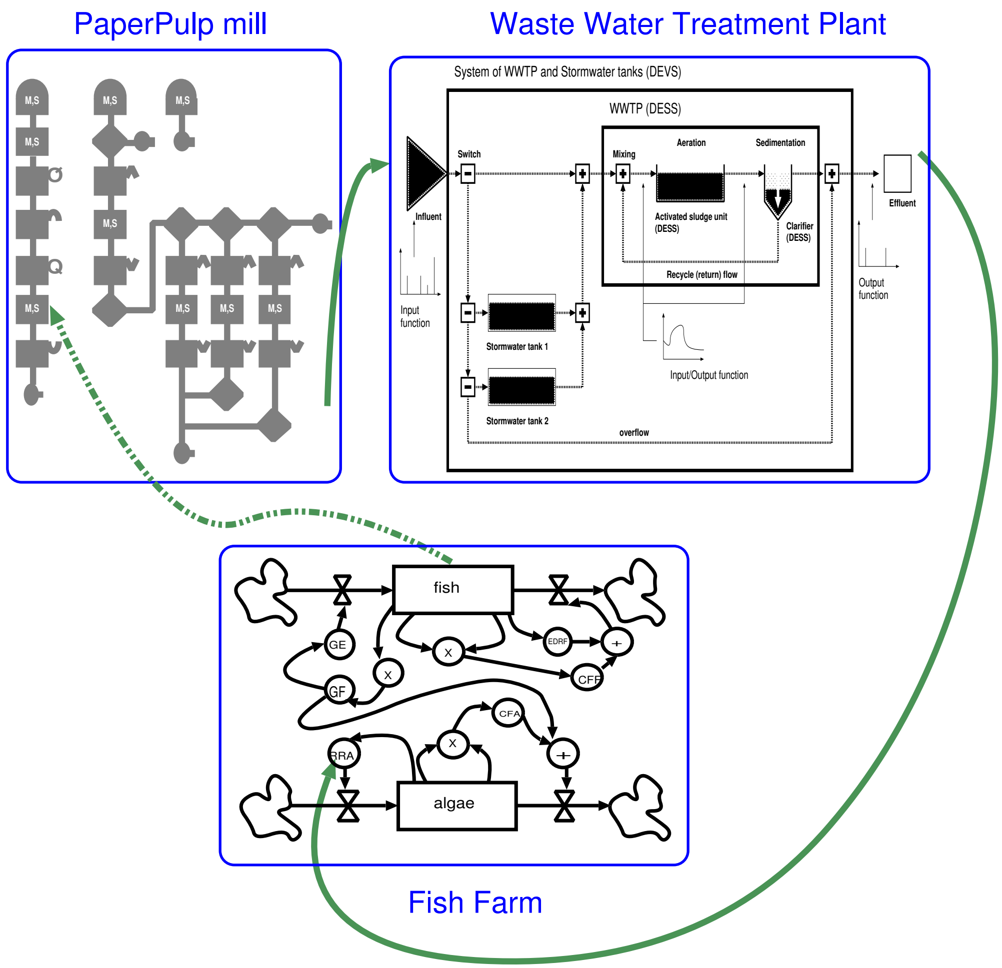
- *schedule* ARRivals
- *resolve* collisions

Closure under Coupling/Composition: Petri Net



- Transitions $A.ty, B.tx$ are used as ports
- Coupling between ports by means of a place p

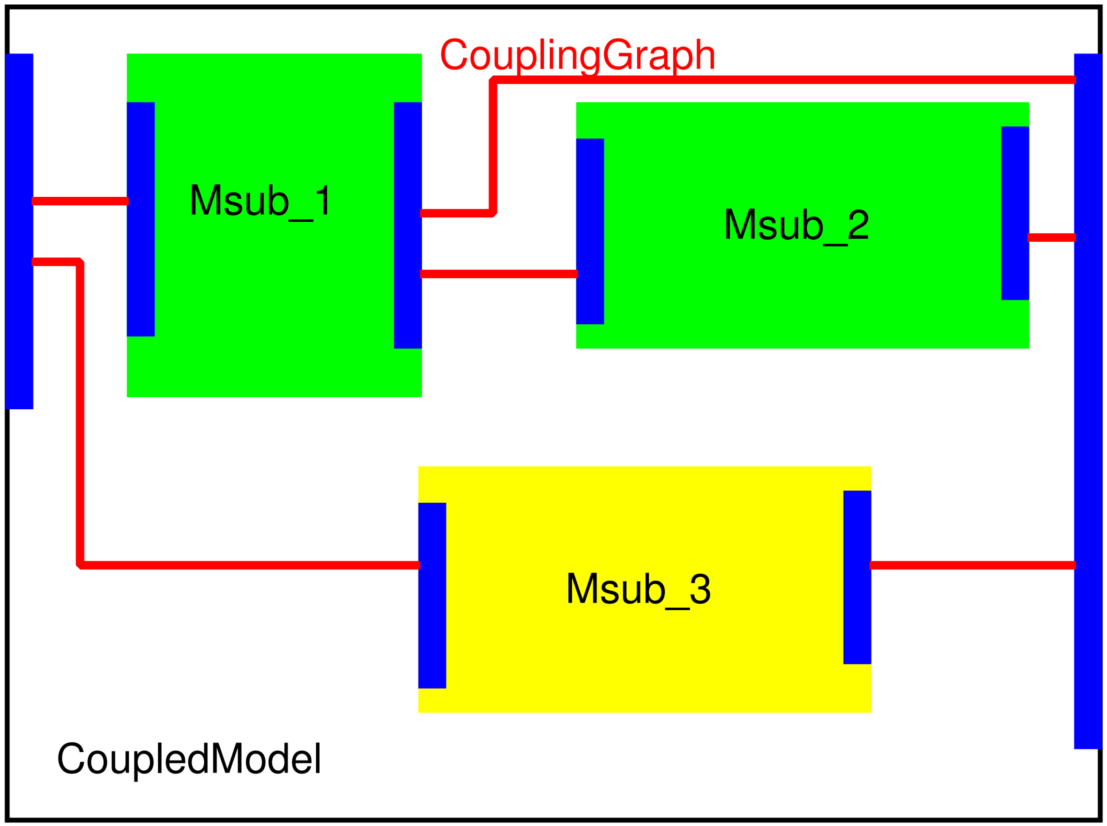
Complex System: Coupling Different Formalisms



Semantics of Coupled Models

1. Super-formalism subsumes all formalisms
2. Co-simulation (coupling resolved at trajectory level)
3. Transform to common formalism

Multi-formalism coupled model: co-simulation



Co-simulation of multi-formalism coupled models

- Sub-models simulated with formalism-specific simulators.
- Interaction due to coupling is resolved at trajectory level.

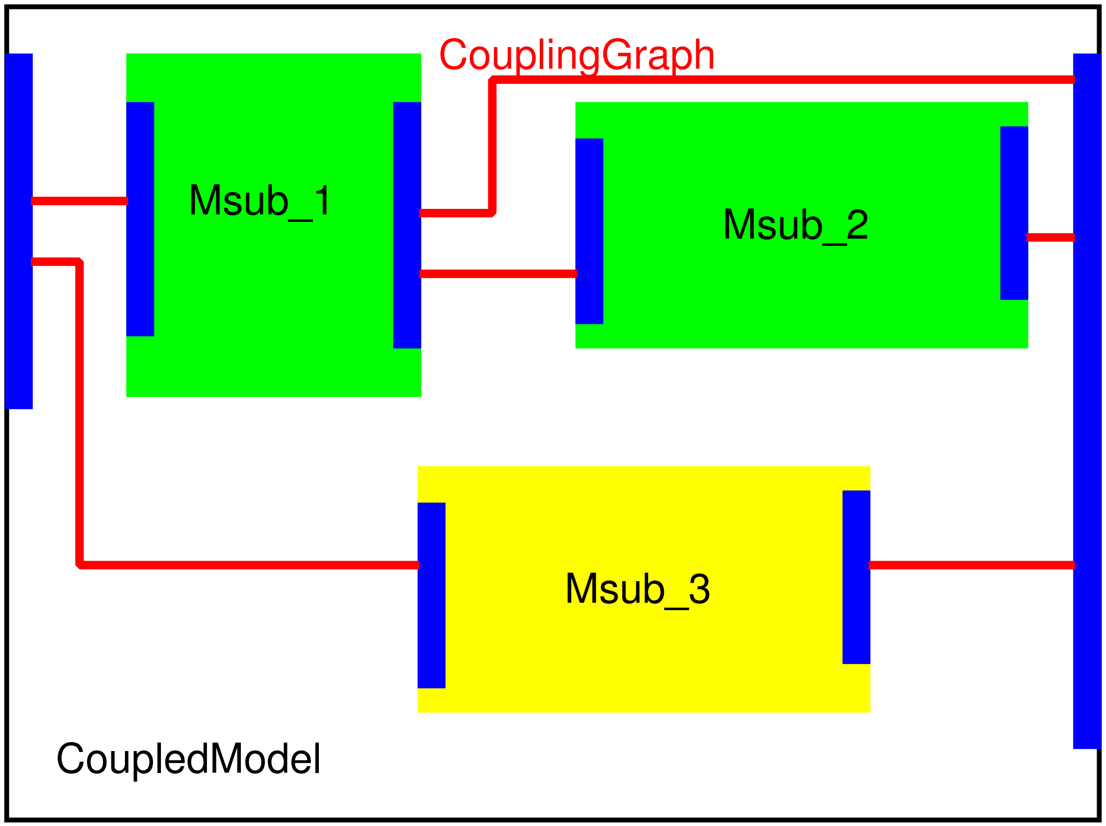
$\rightarrow$ Loss of information.

$\rightarrow$ Questions can *only* be answered at trajectory level.

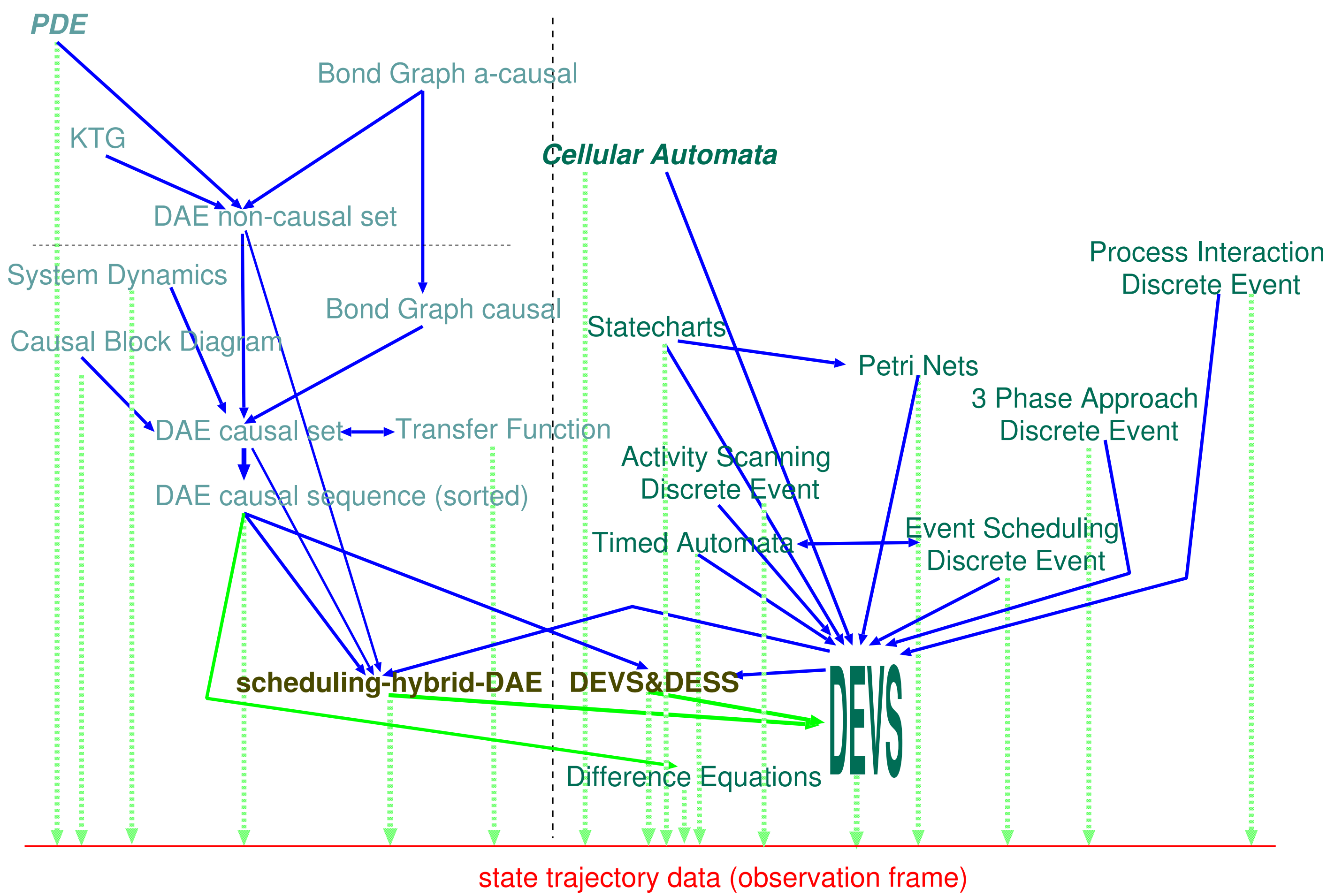
$\rightarrow$ Speed and numerical accuracy problems
for continuous formalisms.

$\rightarrow$ Meaningful for discrete-event formalisms (but beware of legitimacy !).
Basis of the DoD High Level Architecture (HLA)
for simulator interoperability.

Multi-formalism coupled model: multi-formalism modelling



Formalism Transformation Graph



Multi-formalism modelling $\neq$ co-simulation

1. Start from a coupled multi-formalism model. Check consistency of this model (*e.g.,* whether causalites and types of connected ports match).
2. Cluster all formalisms described in the same formalism.
3. For each cluster, implement closure under coupling.
4. Look for the best common formalism in the Formalism Transformation Graph all the remaining different formalisms can be transformed to. Worst case: trajectory level (fallback to co-simulation).
5. Transform all the sub-models to the common formalism.
6. Implement closure under coupling of the common formalism.

The Future . . .

- Formalism Transformation (FTG)
- Graph Grammars *models* for all Transformations
- Simulator Meta-specification (reference implementation)
- Model exchange (DTD from meta-model, XML from model)
- Variations (flavours) of formalisms (syntax and semantics)
- Automatic equivalence proofs (bi-simulation)
- Meta-modelling Environment (ATOM^3)