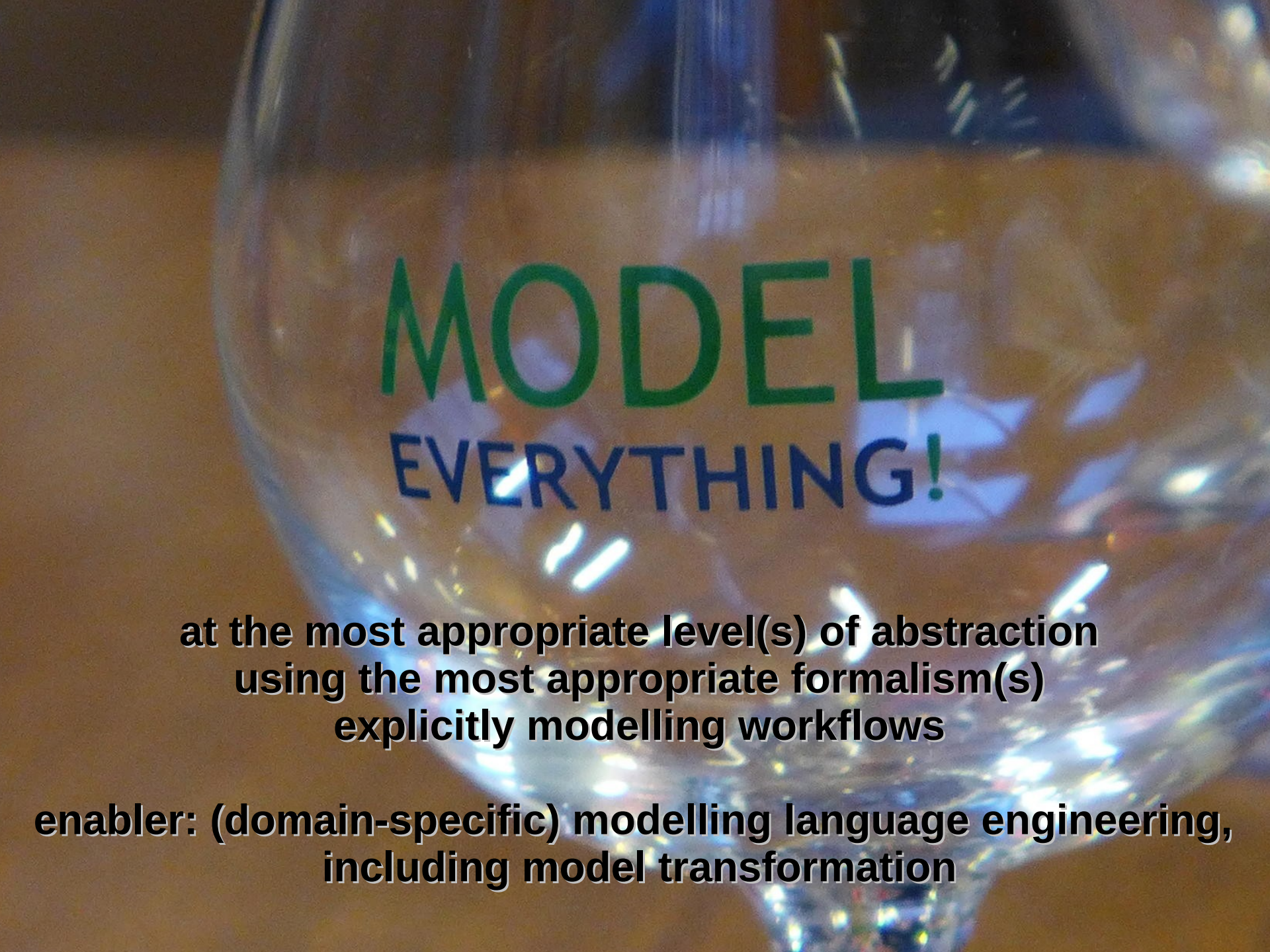


(Domain-Specific) Modelling Language Engineering

Hans Vangheluwe

<http://msdl.uantwerpen.be/>

A close-up photograph of a glass, possibly a shot glass, with the words "MODEL" and "EVERYTHING!" etched onto it. "MODEL" is in a larger, green, sans-serif font, and "EVERYTHING!" is in a smaller, blue, sans-serif font below it. The glass is partially filled with a clear liquid, and the background is blurred, showing some lights and reflections.

MODEL EVERYTHING!

**at the most appropriate level(s) of abstraction
using the most appropriate formalism(s)
explicitly modelling workflows**

**enabler: (domain-specific) modelling language engineering,
including model transformation**



MODEL
EVERYTHING!

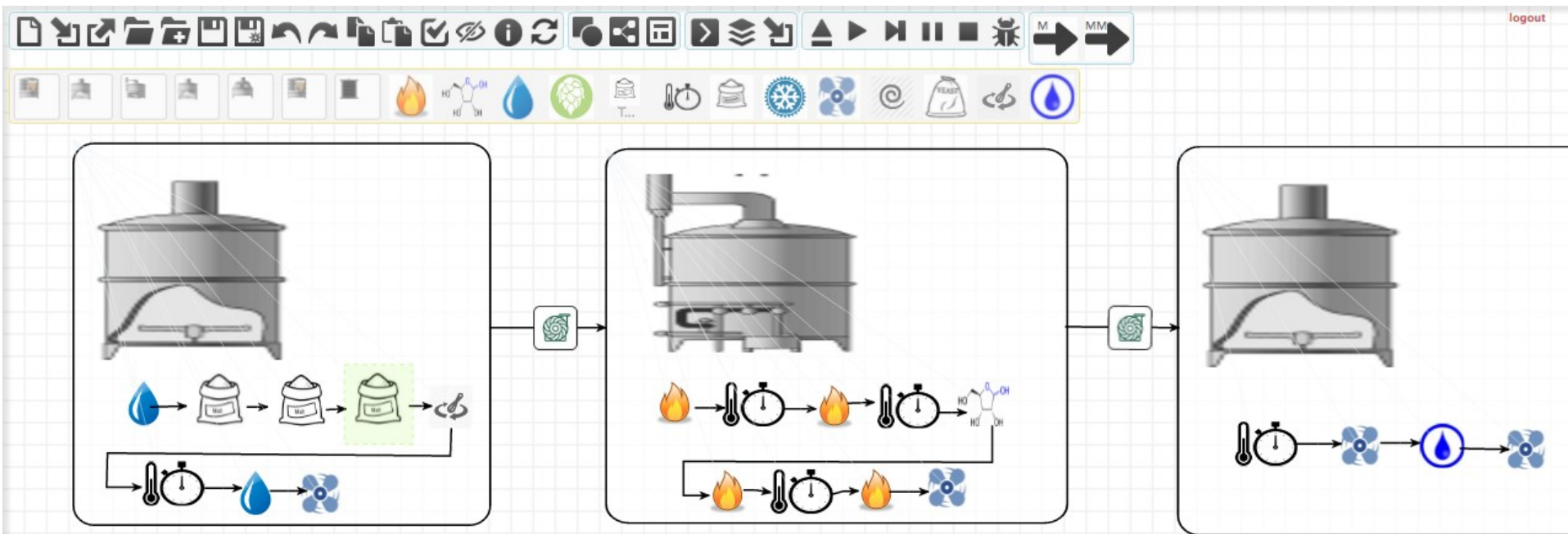


DSM TP 2014
Theory and Practice

5th International Summer School
on Domain Specific Modeling

Antwerp, Belgium
25 - 29 August

Thomas Kühne

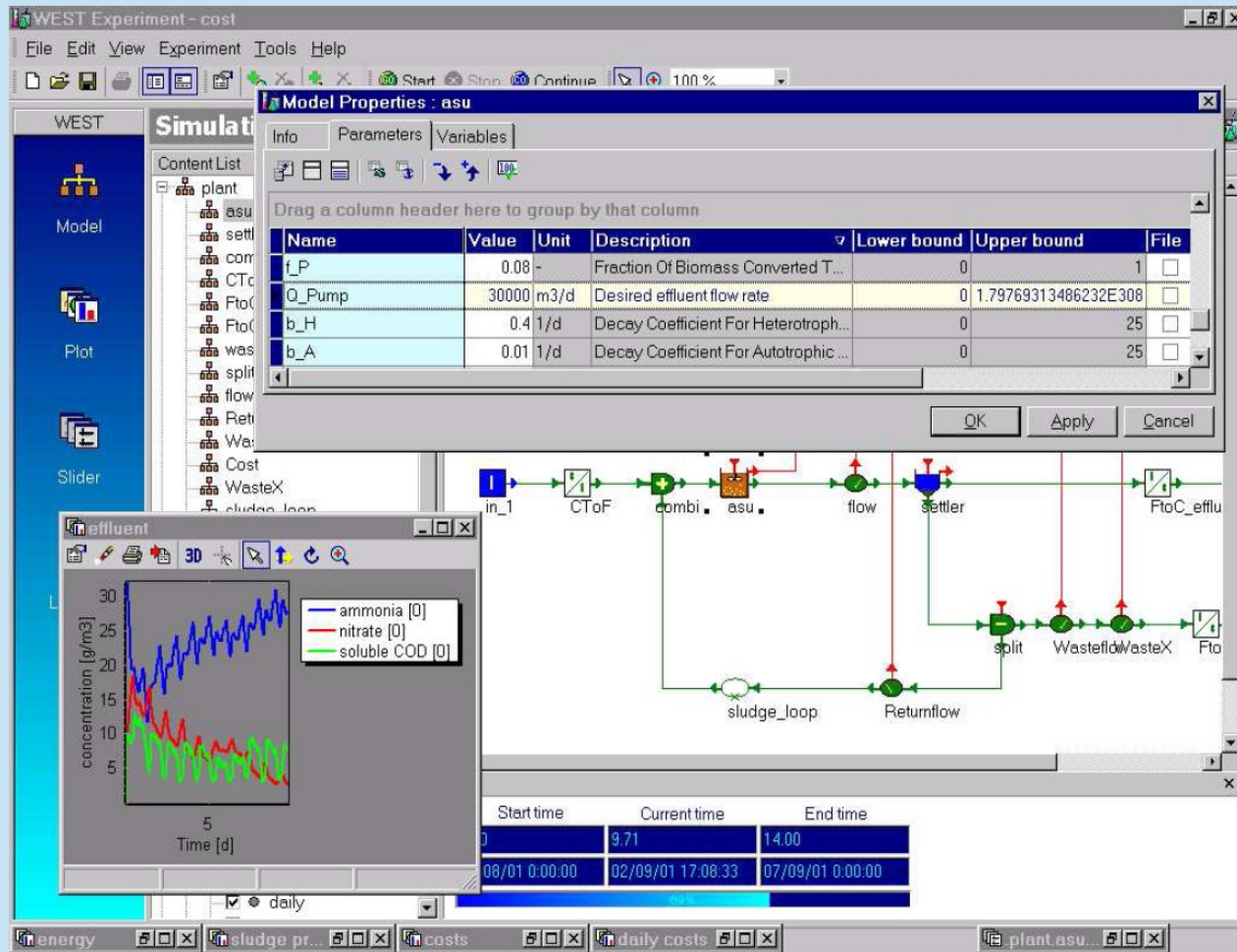


Joachim Denil

Show Chat
send screenshare invitation
send modelshare invitation



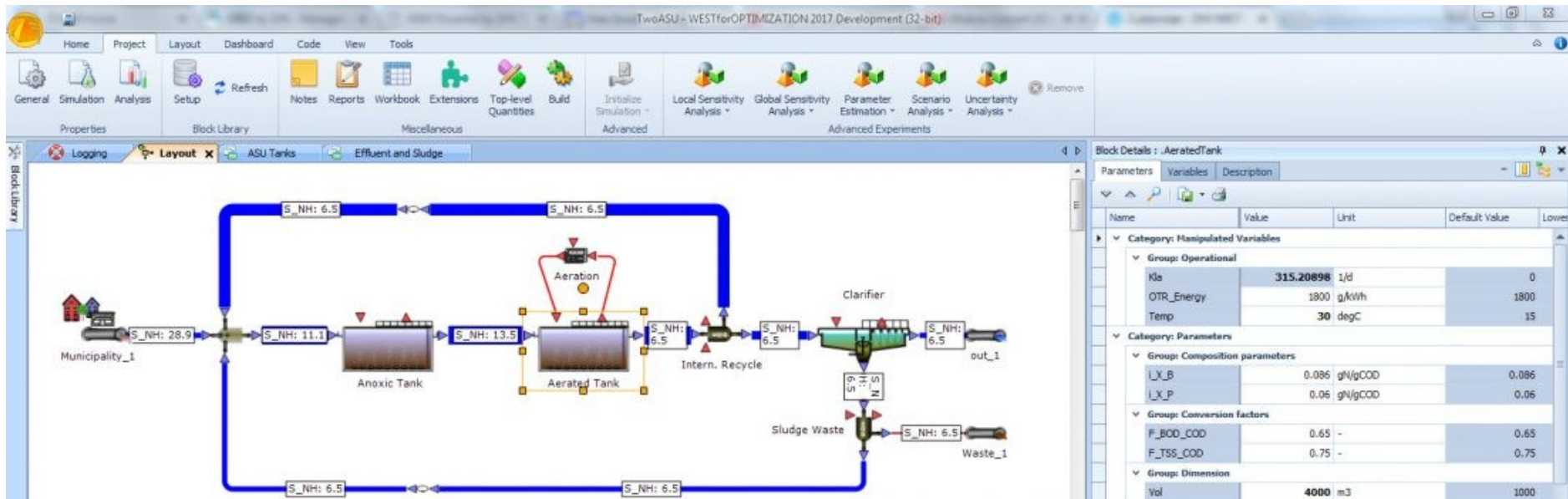
DS(V)M Environment



[WEST: modelling biological wastewater treatment.](#)

Henk Vanhooren, Jurgen Meirlaen, Youri Amerlinck, Filip Claeys, Hans Vangheluwe and Peter A. Vanrolleghem.

Journal of Hydroinformatics 5 (2003) 27-50



<http://www.mikebydhi.com/products/west>

Why DS(V)M ?

(as opposed to General Purpose modelling)

- **match the user's mental model** of the problem domain
- **maximally constrain** the user (to the problem at hand)
 - ⇒ easier to learn
 - ⇒ avoid errors
- **separate** domain-expert's work from analysis/transformation expert's work

Anecdotal evidence of 5 to 10 times speedup

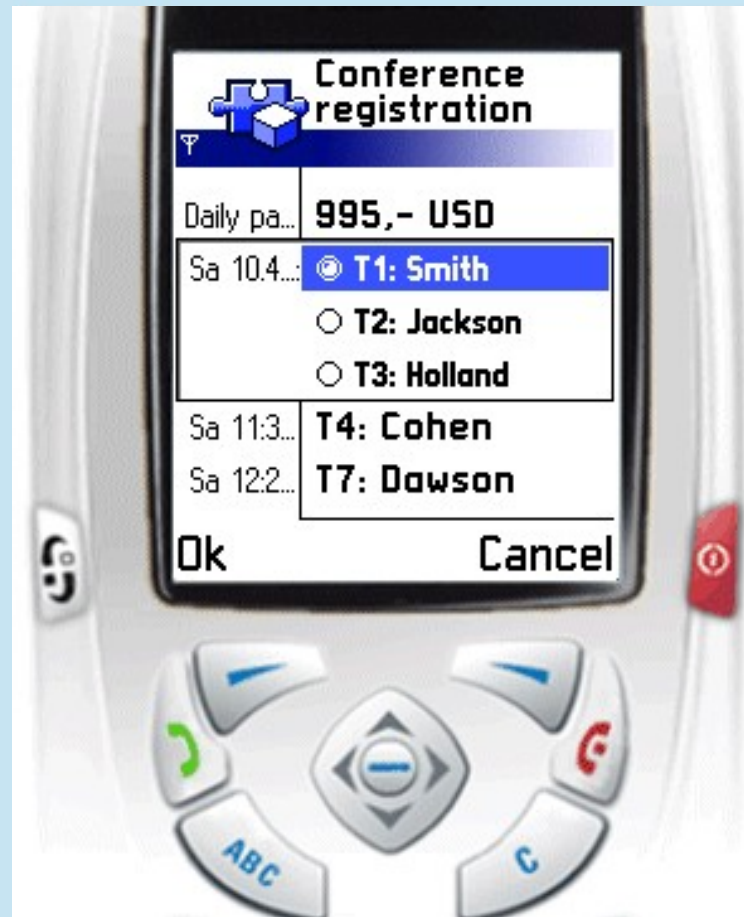
Steven Kelly and Juha-Pekka Tolvanen. Domain-Specific Modeling: Enabling Full Code Generation. Wiley, 2008.

Laurent Safa. The practice of deploying DSM, report from a Japanese appliance maker trenches. In Proceedings of the 6th

OOPSLA Workshop on Domain-Specific Modeling (DSM'06), pp. 185-196, 2006.

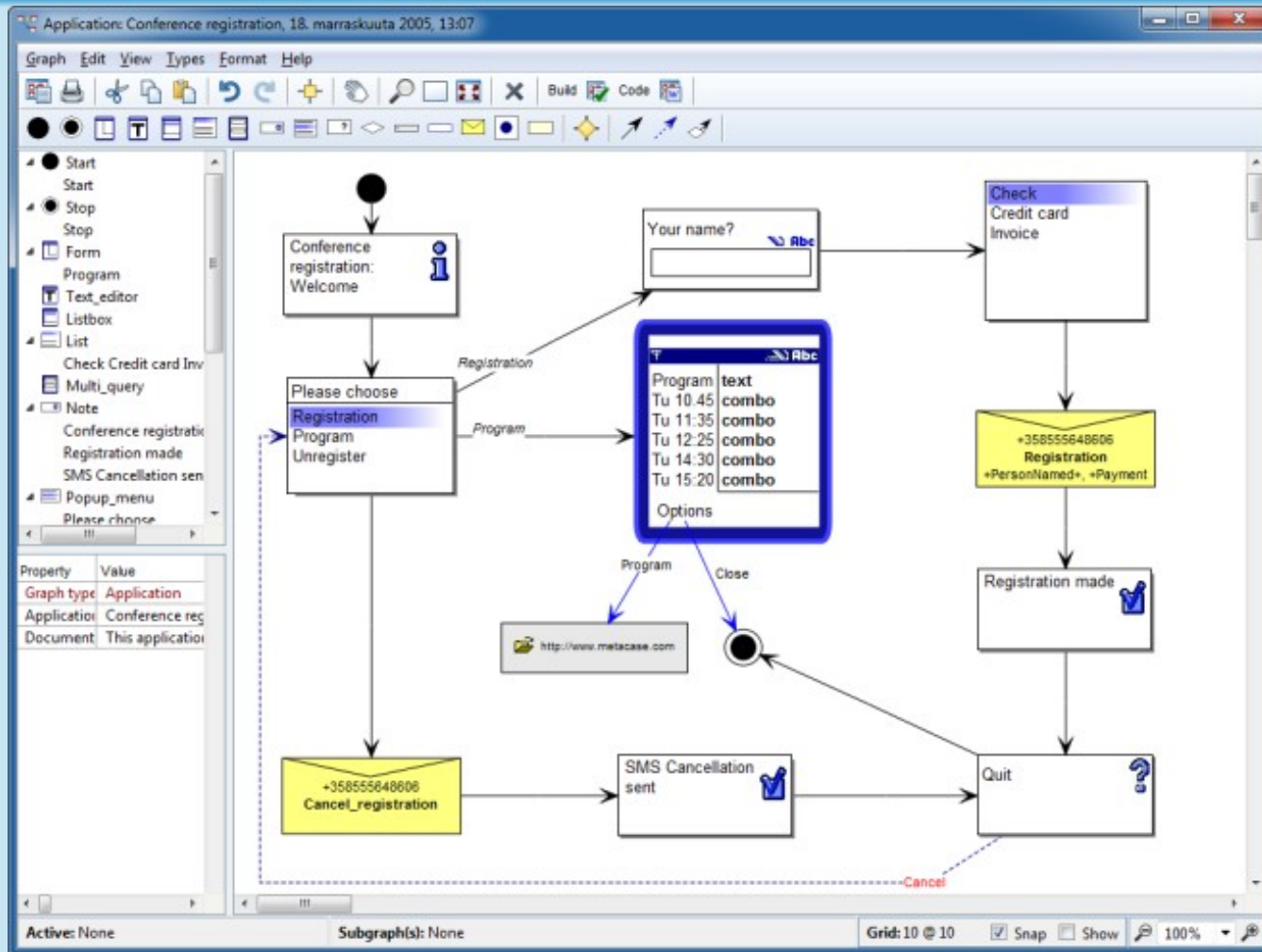
DS(V)M Example in Software Domain

smart phones, the application

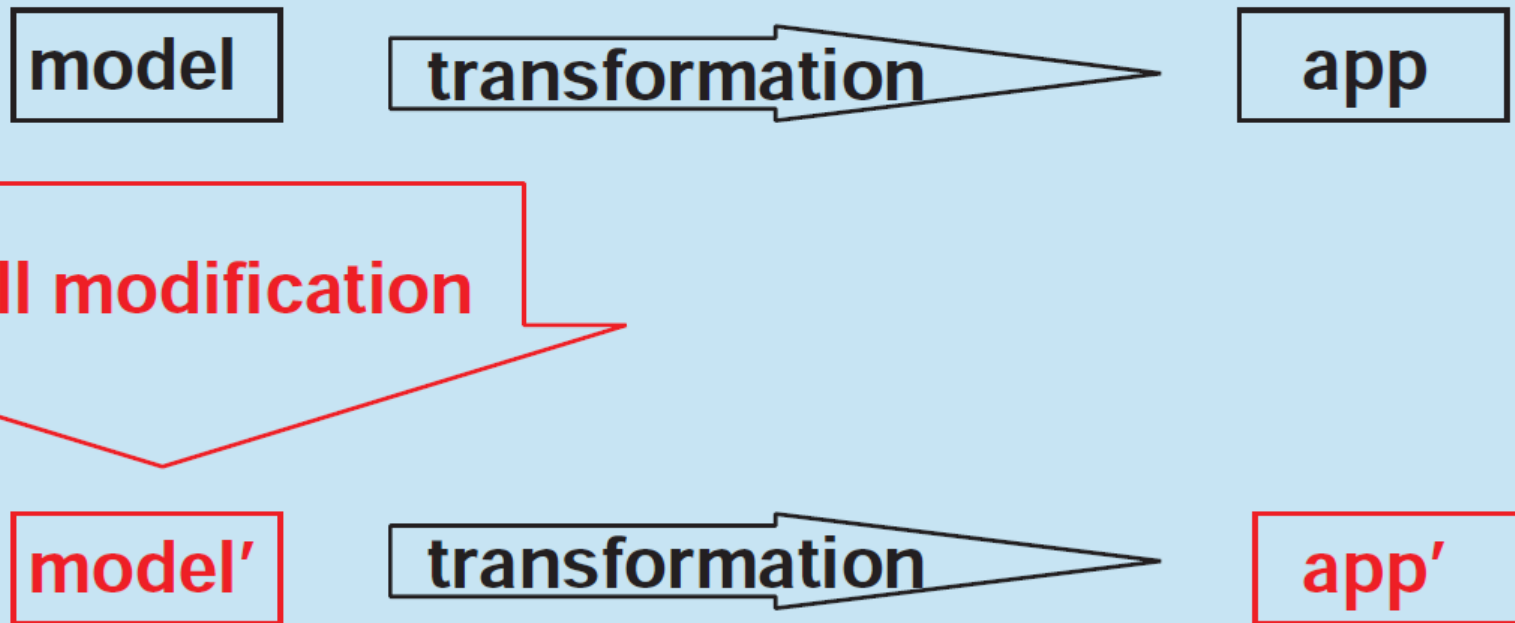


MetaEdit+ (www.metacase.com)

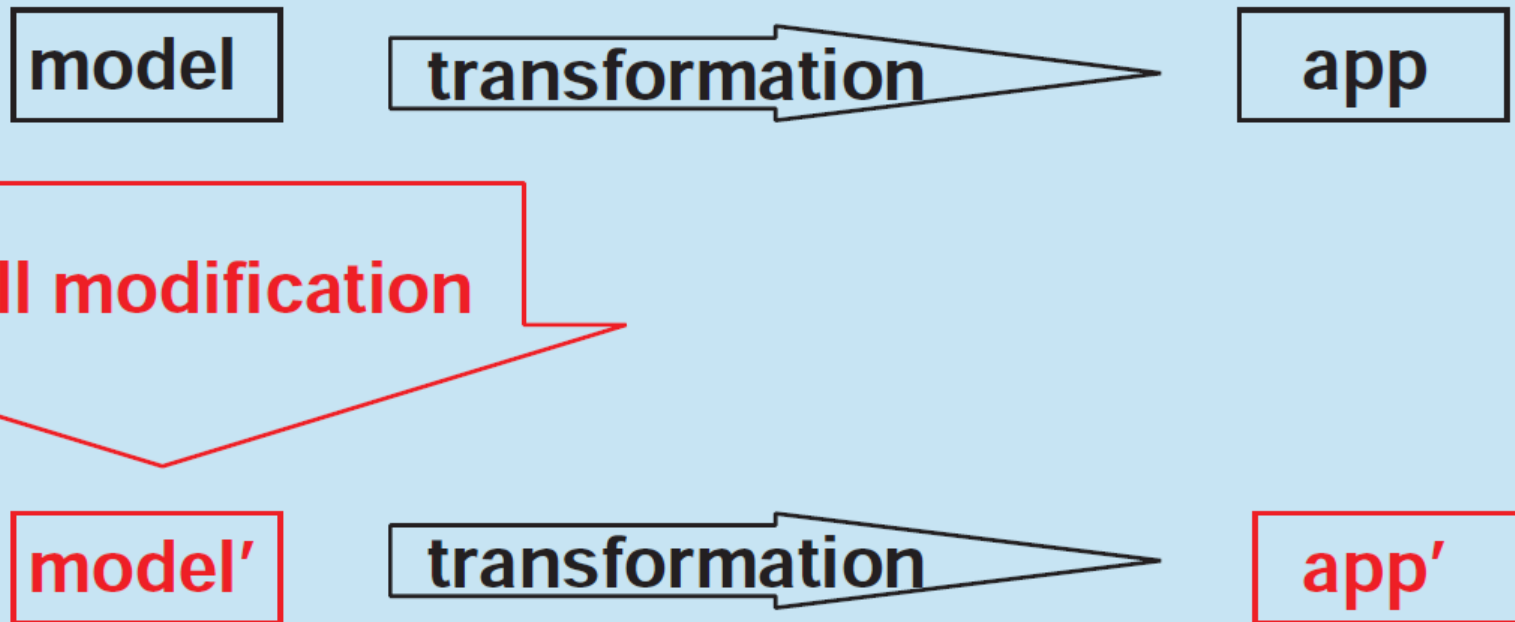
DS(V)M Example: smart phones, the Domain-Specific model



Model-Based Development:
Modify the Model
(e.g., based on feature model of product family)

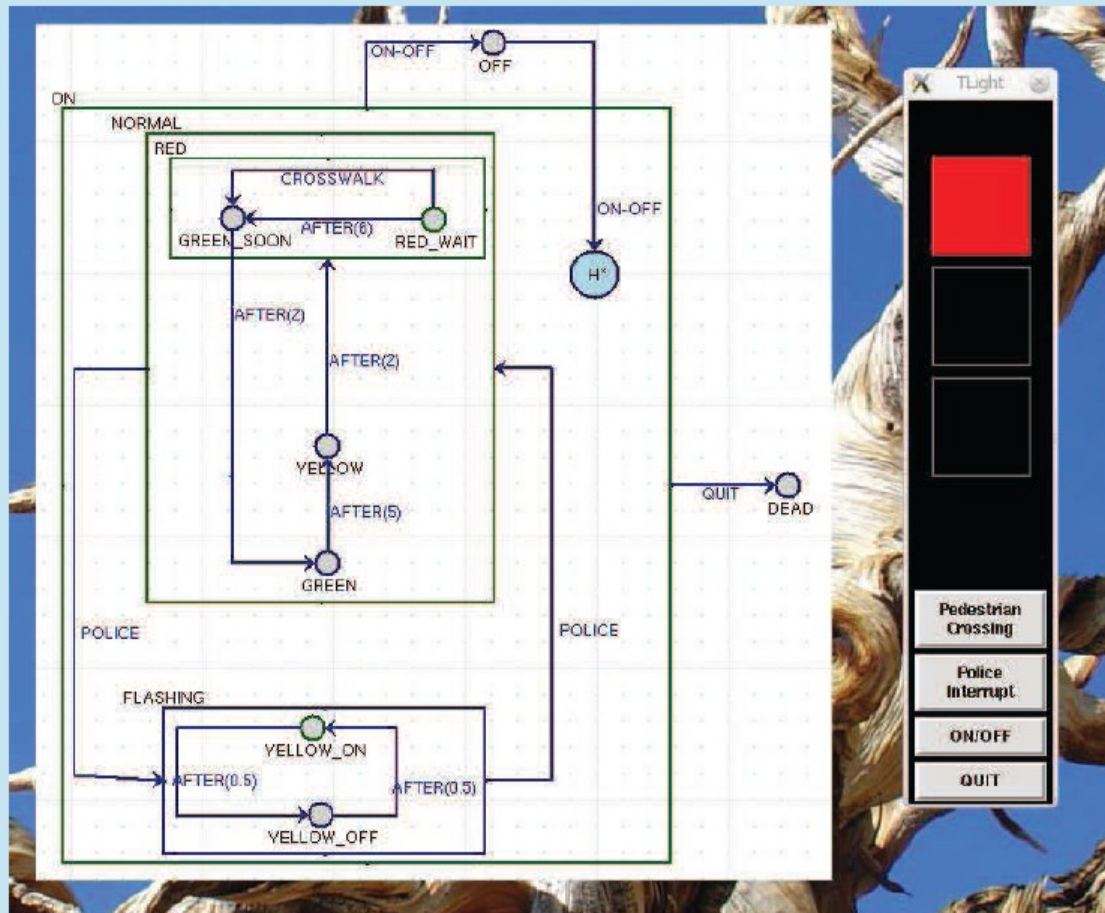


Model-Based Development:
Modify the Model
(e.g., based on feature model of product family)

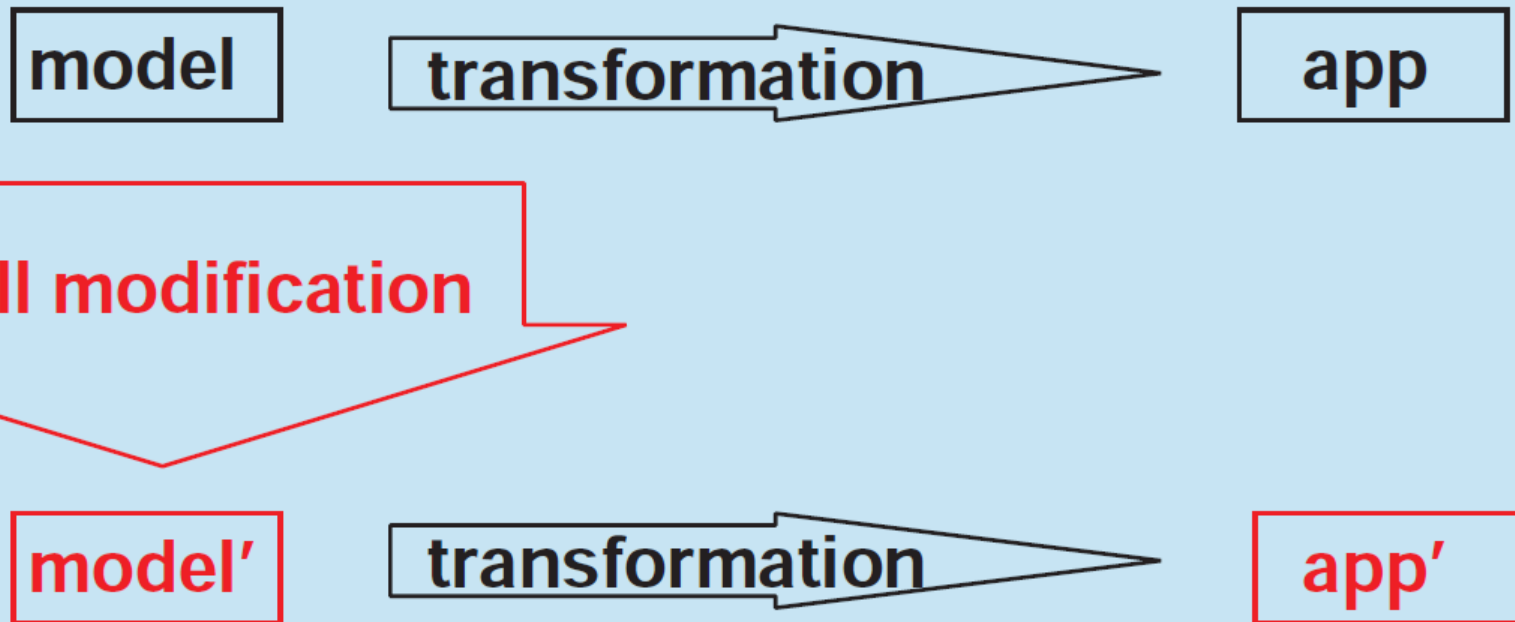


small **modification** in model may lead to large change in **app**
~ choice of formalism (e.g., Statecharts)

Statecharts

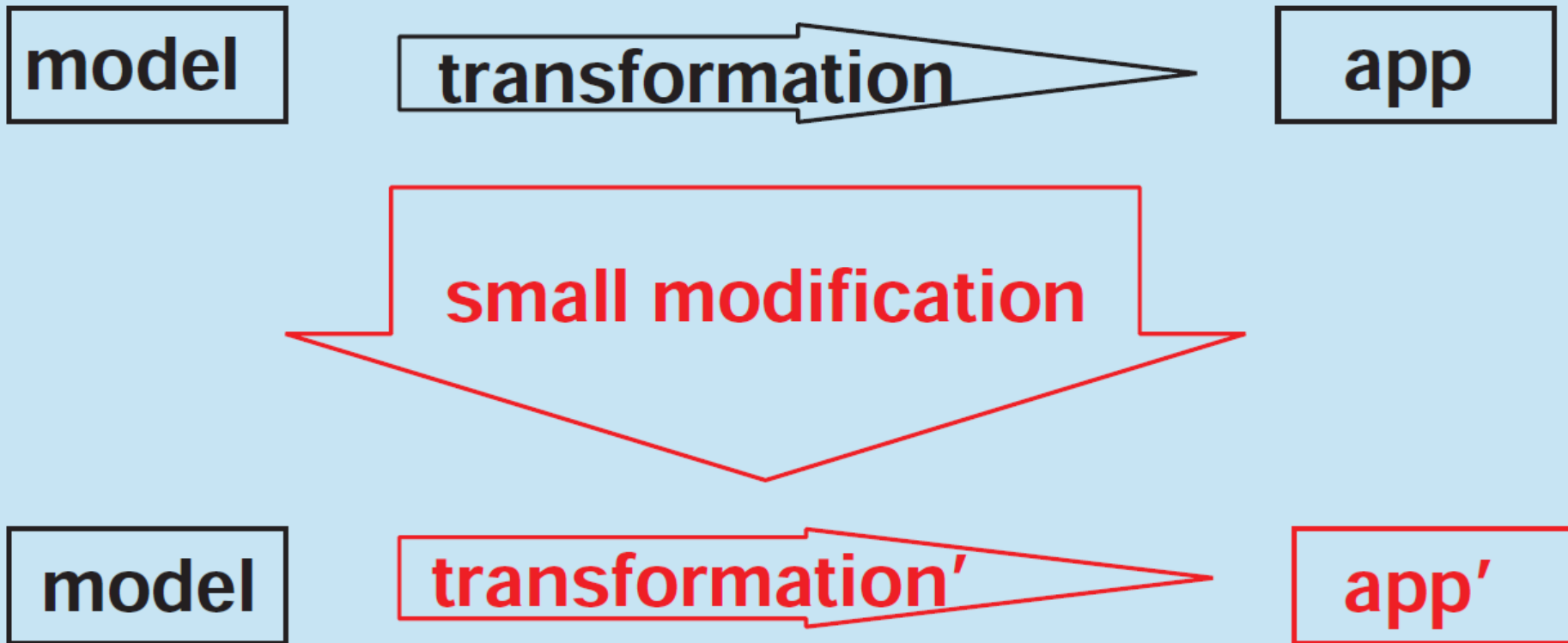


Model-Based Development:
Modify the Model
(e.g., based on feature model of product family)

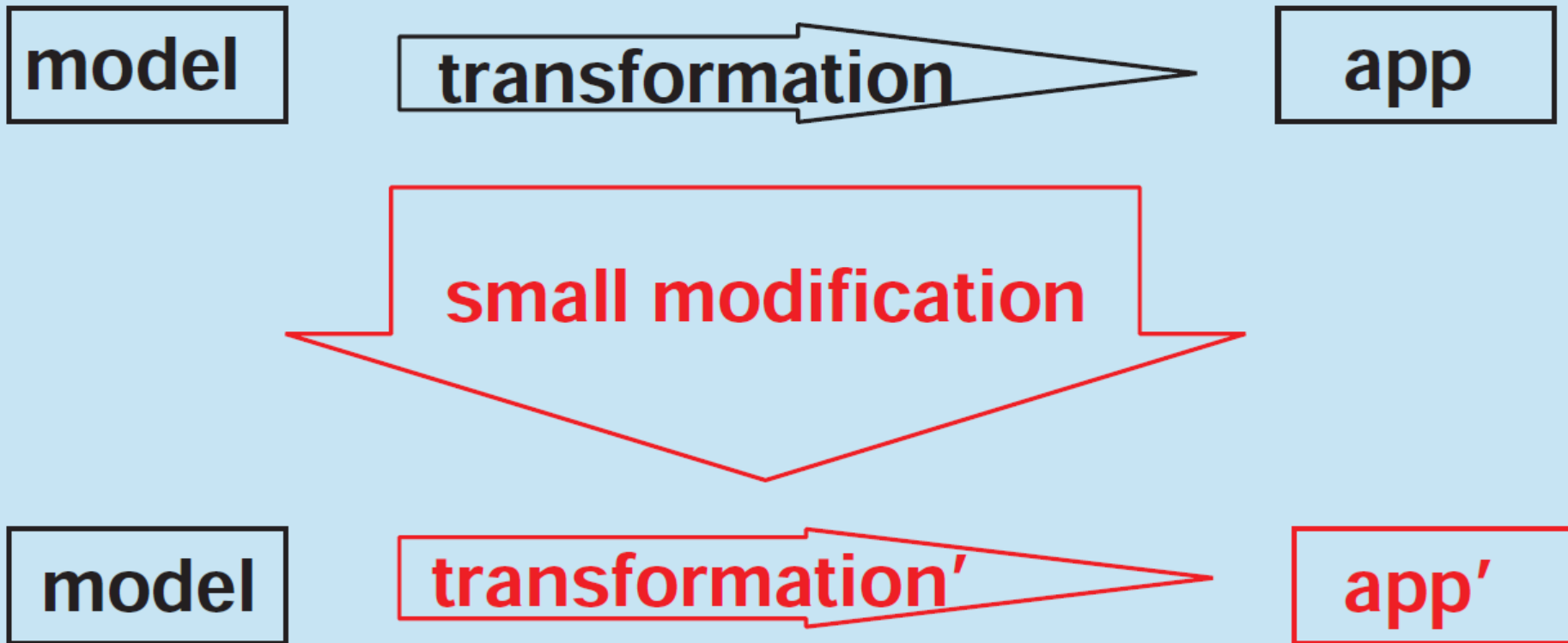


small **modification** in model may lead to large change in **app**
~ choice of formalism (e.g., Statecharts)

Model-Based Development: Modify the Transformation

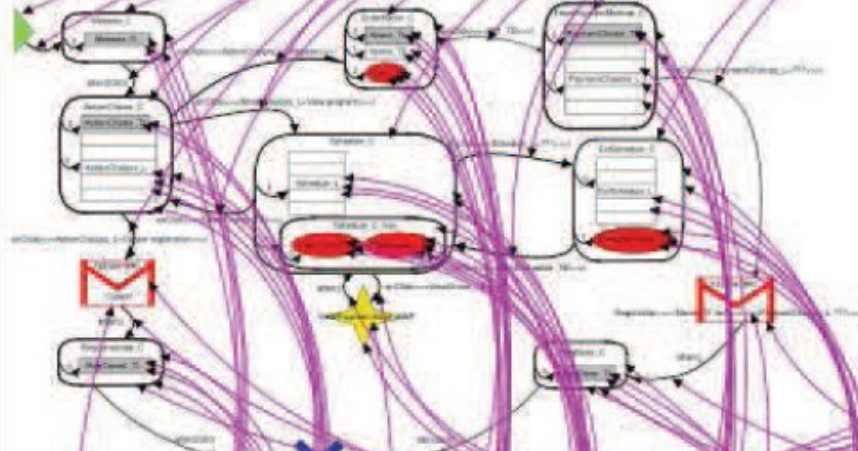


Model-Based Development:
Modify the Transformation
(e.g., target platform changes, or optimization)

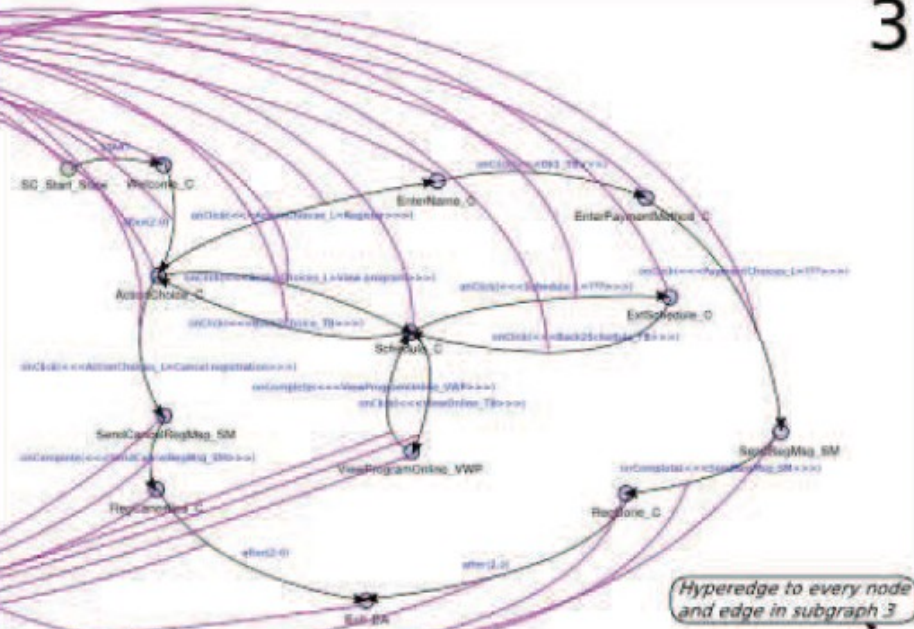


Can be Multi-Step/Multi-Formalism

1



3



2



4

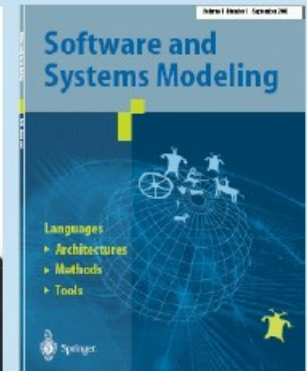


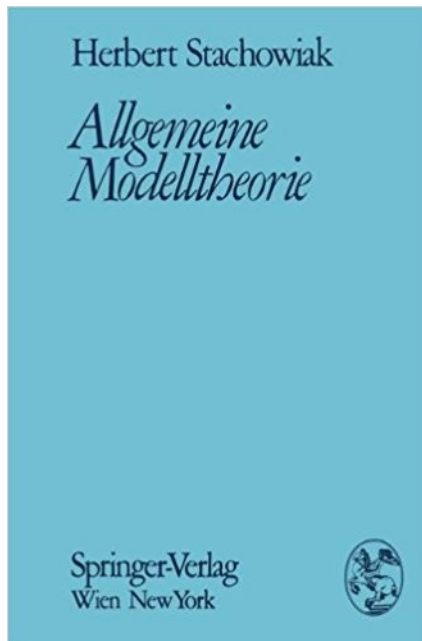
Building DS(V)M Tools Effectively ...

- **development cost** of DS(V)M Tools may be prohibitive!
- $\Rightarrow$ need **Modelling Language Engineering**

Dissecting Modelling

Thomas Kühne. *Matters of (Meta-)Modeling*.
Software and Systems Modeling 5(4): 369-385 (2006)

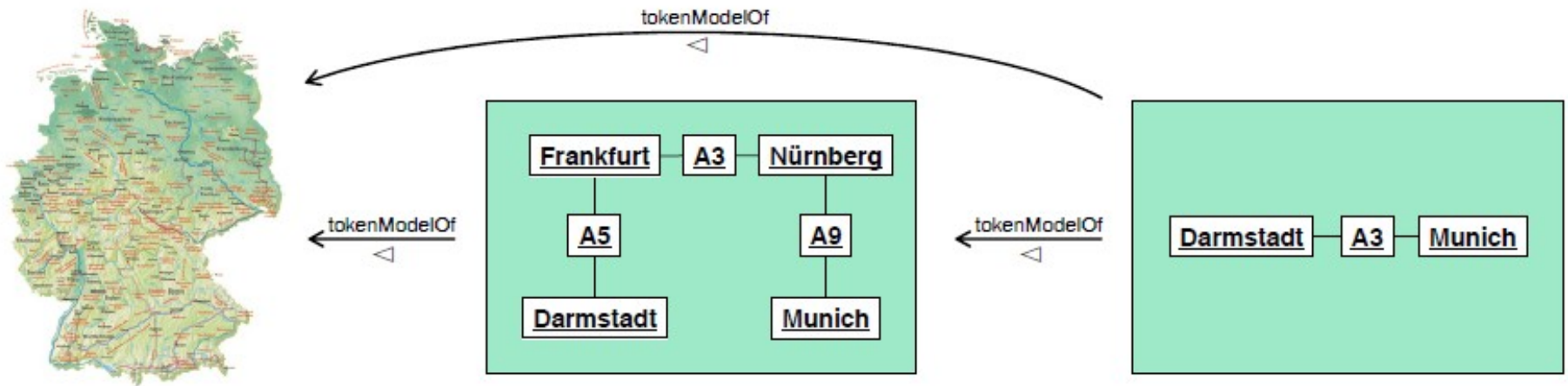




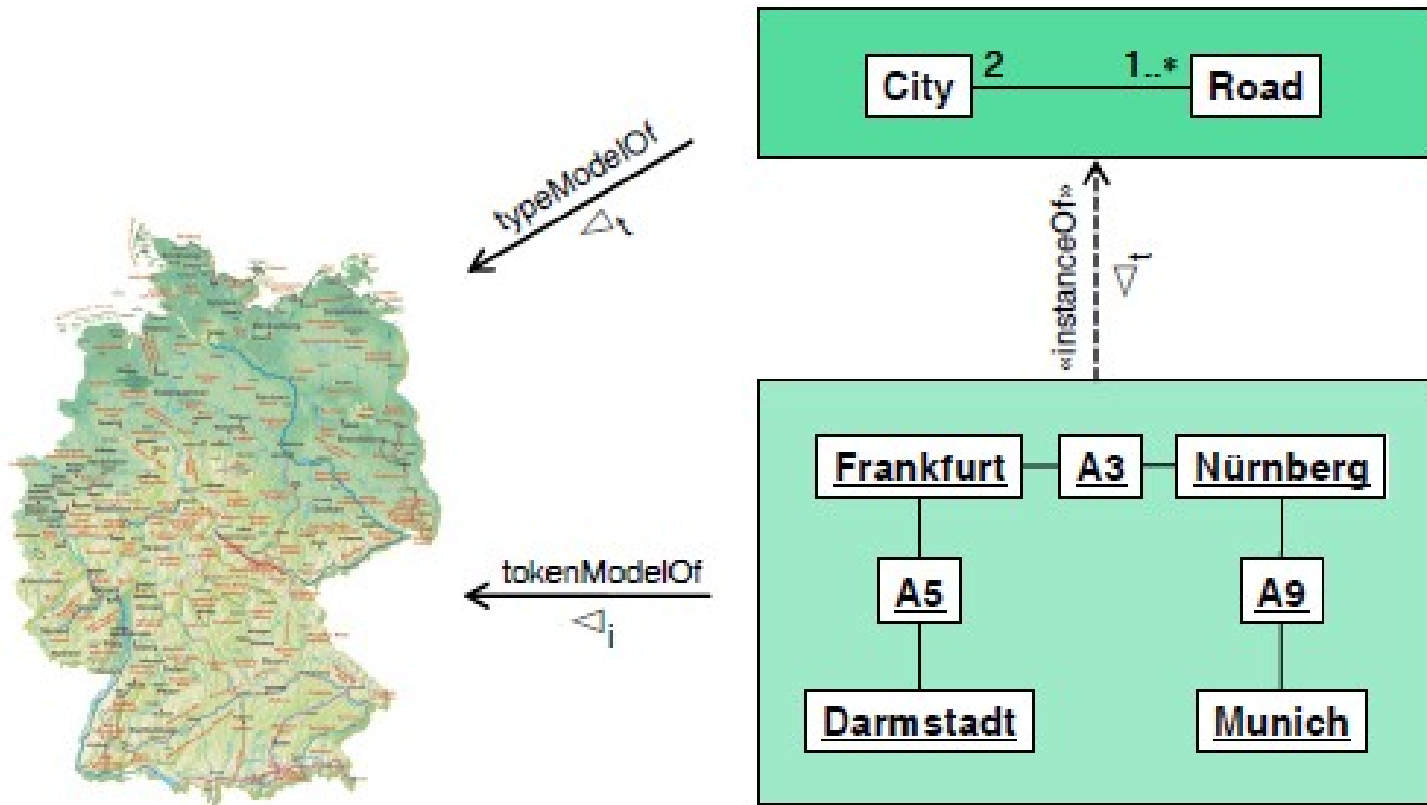
Model Features

mapping feature	A model is based on an original. ⁴
reduction feature	A model only reflects a (relevant) selection of an original's properties.
pragmatic feature	A model needs to be usable in place of an original with respect to some purpose.

Token Models

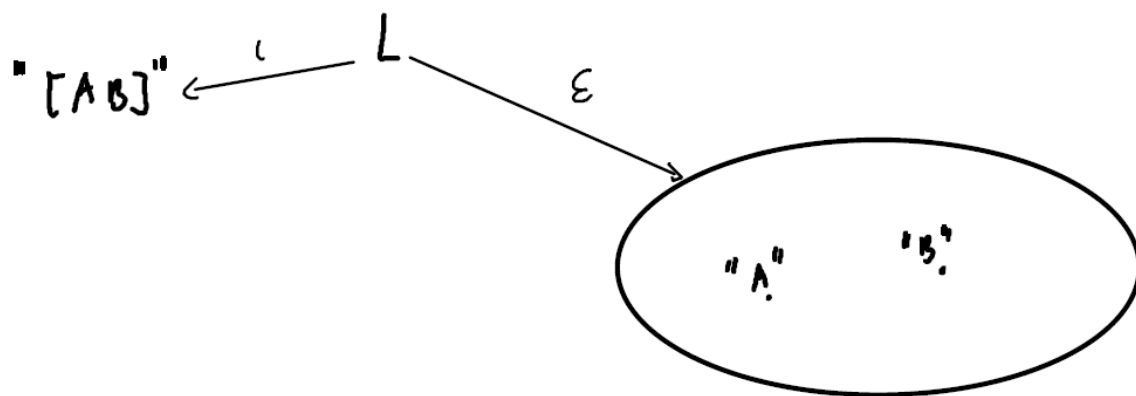
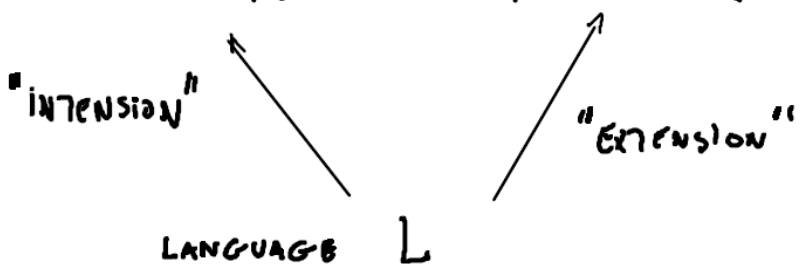


Rôles a Model may Play



$$\llbracket "A + B*" \rrbracket_{RE} = \{ "A", "AA", \dots, "AB", "AAB", \dots \}$$

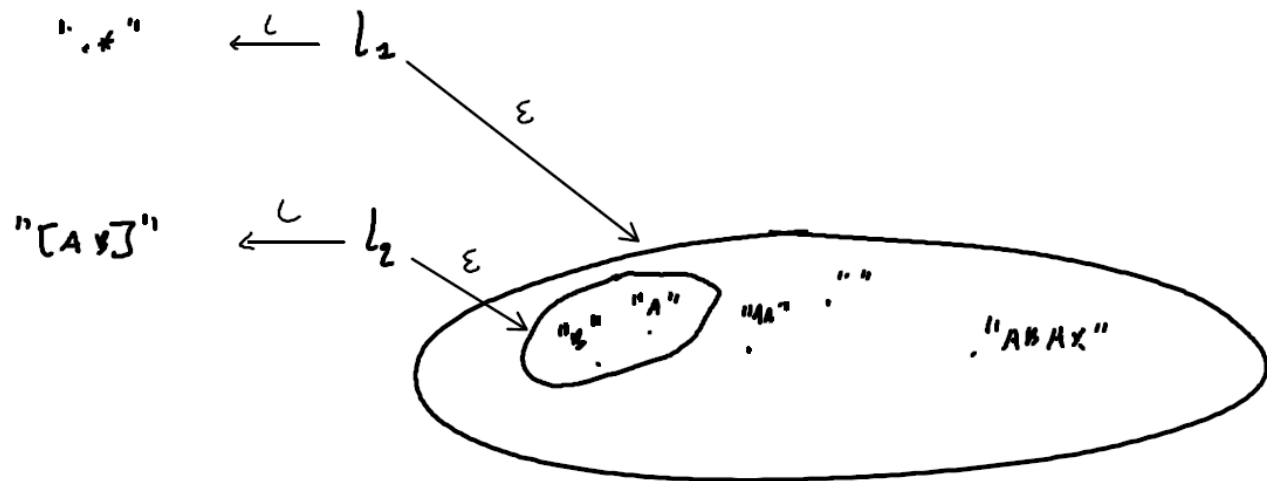
$$\llbracket "[AB]" \rrbracket_{RE} = \{ "A", "B" \}$$



• ϵ ANY CHAR FROM ALPHABET

$$l(l_1) = ".*" \quad \epsilon(l_1) = \{ "", "A", "AA", \dots \}$$

$$l(l_2) = "[AB]" \quad \epsilon(l_2) = \{ "A", "B" \}$$



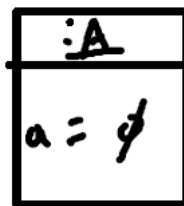
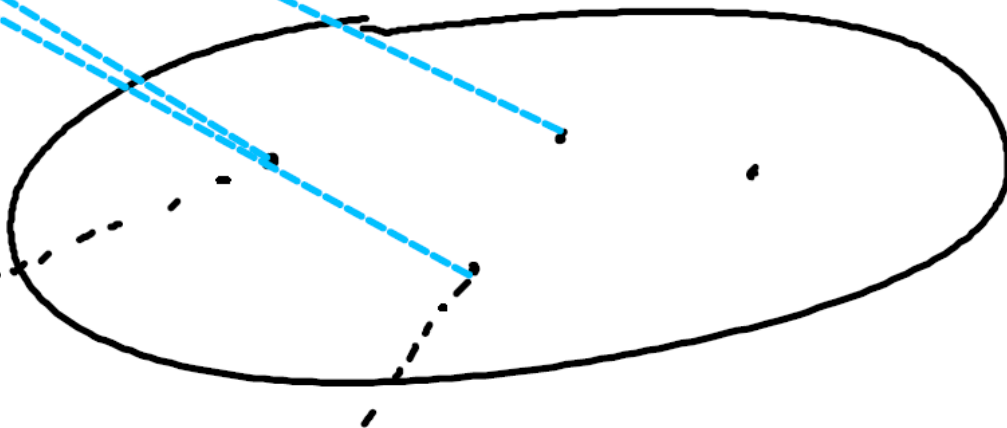
$$l(l_2) \subset l(l_1) \Leftrightarrow \epsilon(l_2) \subseteq \epsilon(l_1)$$

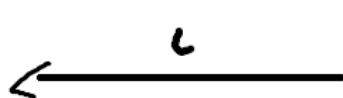
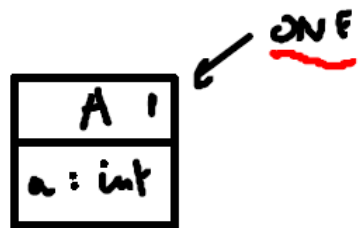


$\leftarrow \mathcal{L}$

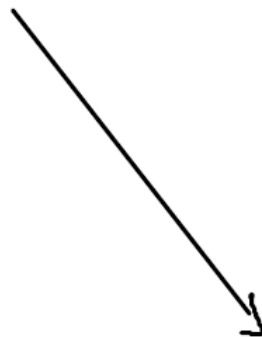
$\mathcal{L}$

$\mathcal{E}$

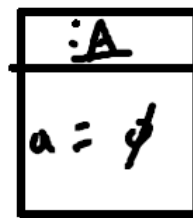
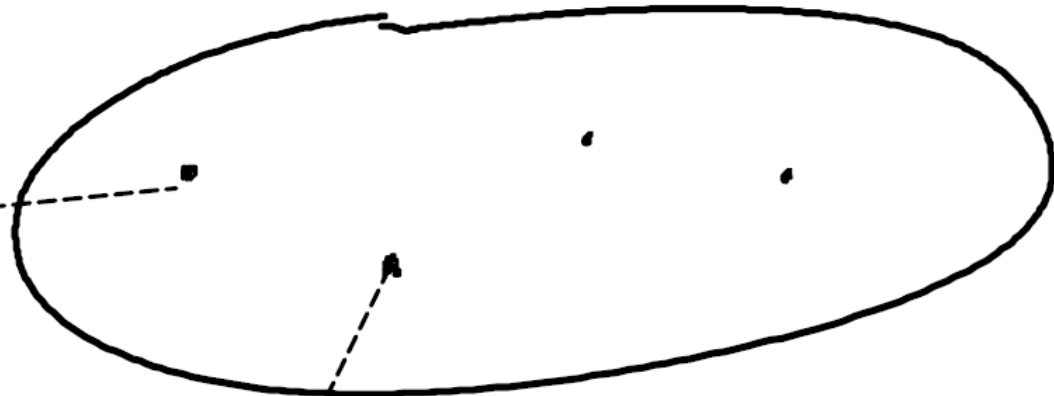
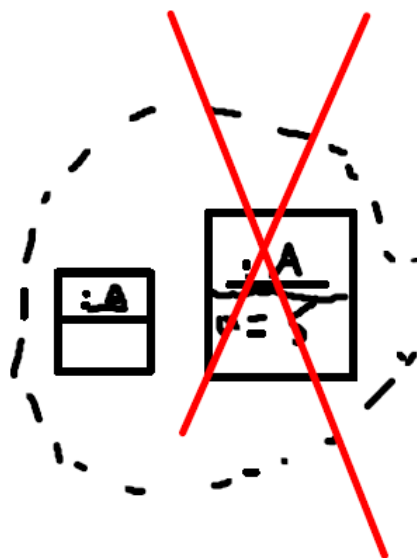


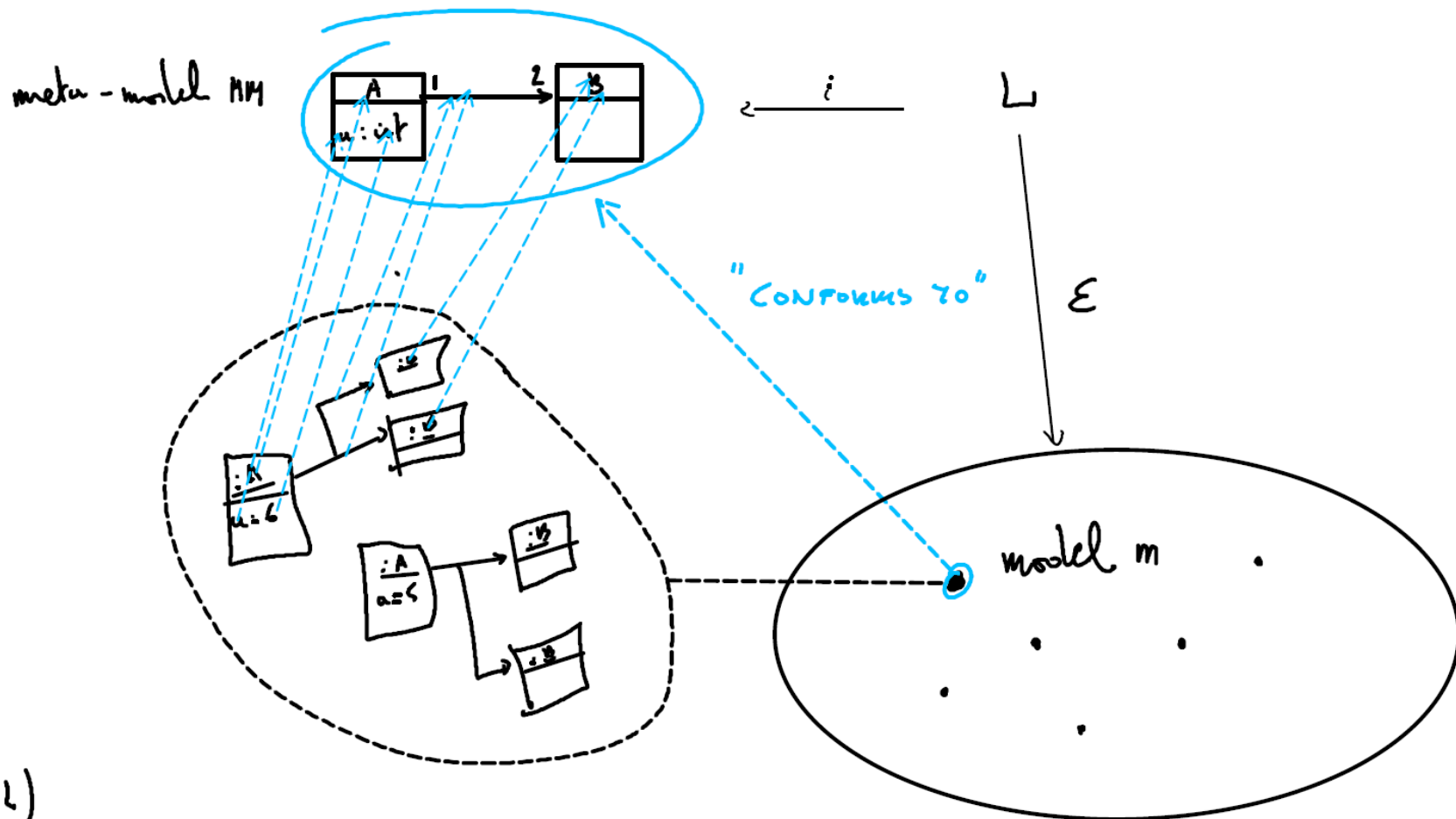


L



Σ





$$MM = i(L)$$

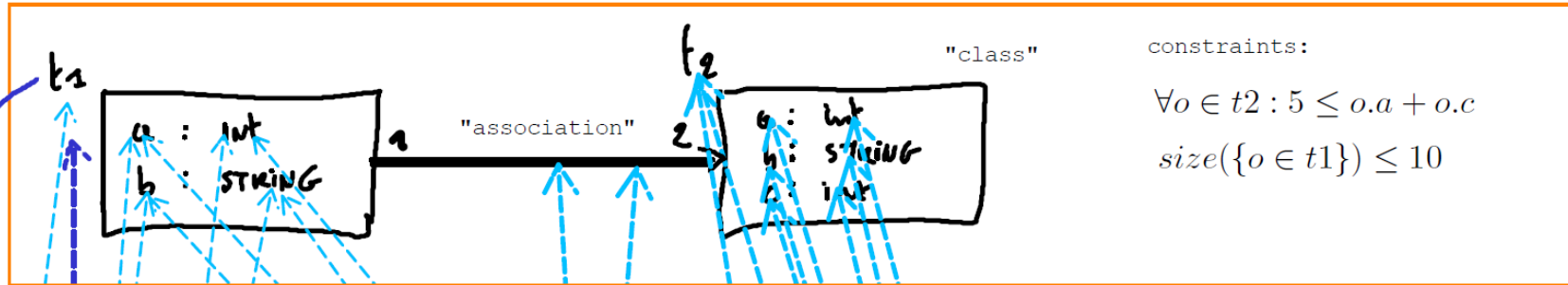
$$M \text{ CONFORMS TO } MM \equiv m \in E(L)$$

$$\text{verify}(m, MM) = \text{True} \quad (\text{WITH PARTICULAR BINDING / MORPHISM} \text{ ???})$$

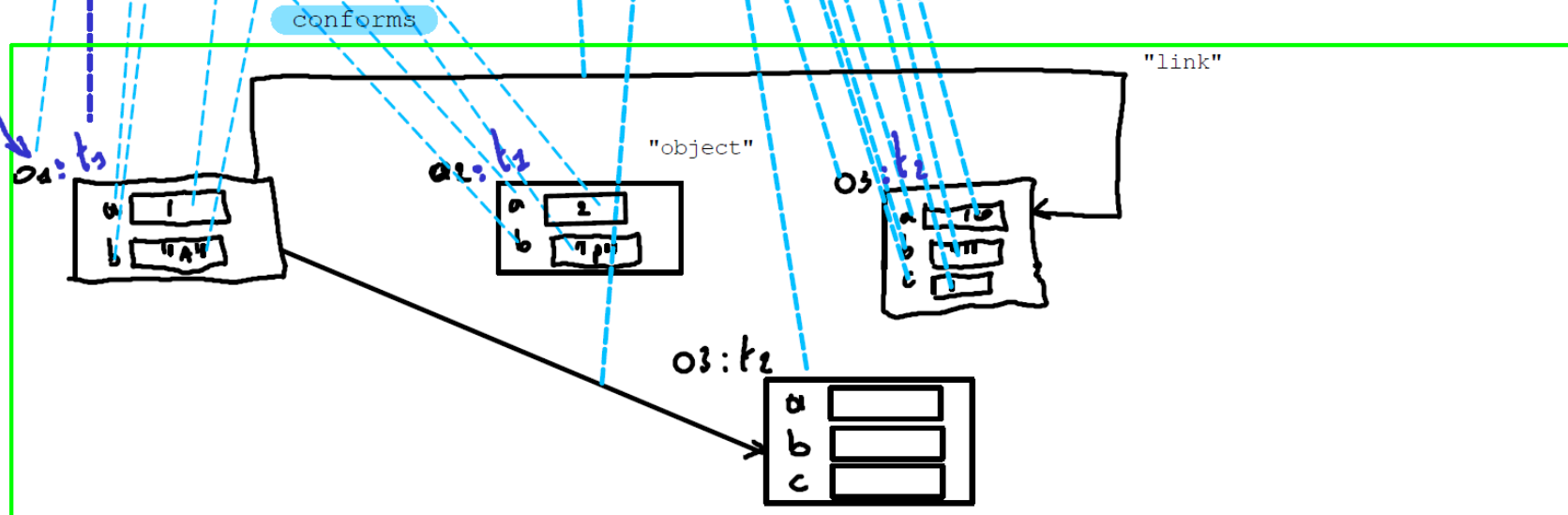
$$\mu(MM) = L$$

$$[MM]_{\text{SEM}} = E(L)$$

meta-model (intensional specification of language L)



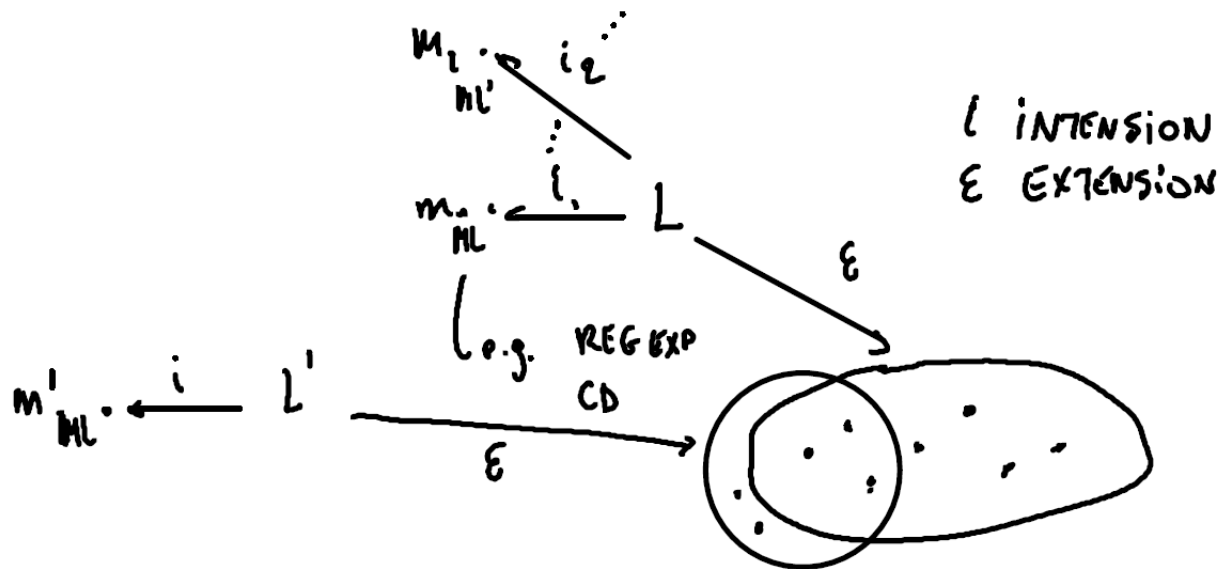
instantiate



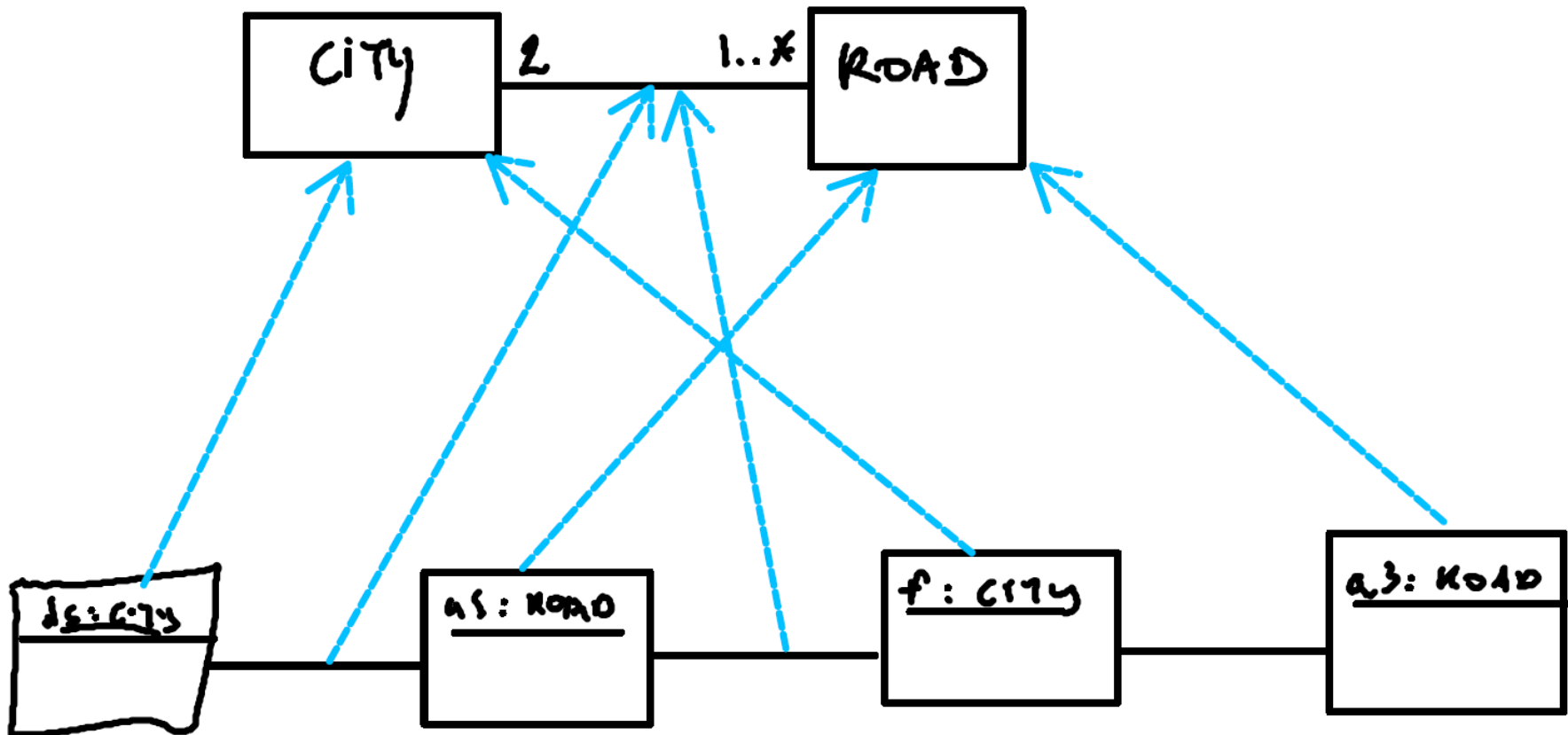
model in language L

note that there may be different (types of)
 conforms_to relationships:
 name-based (also known as nominal), structure based, ...

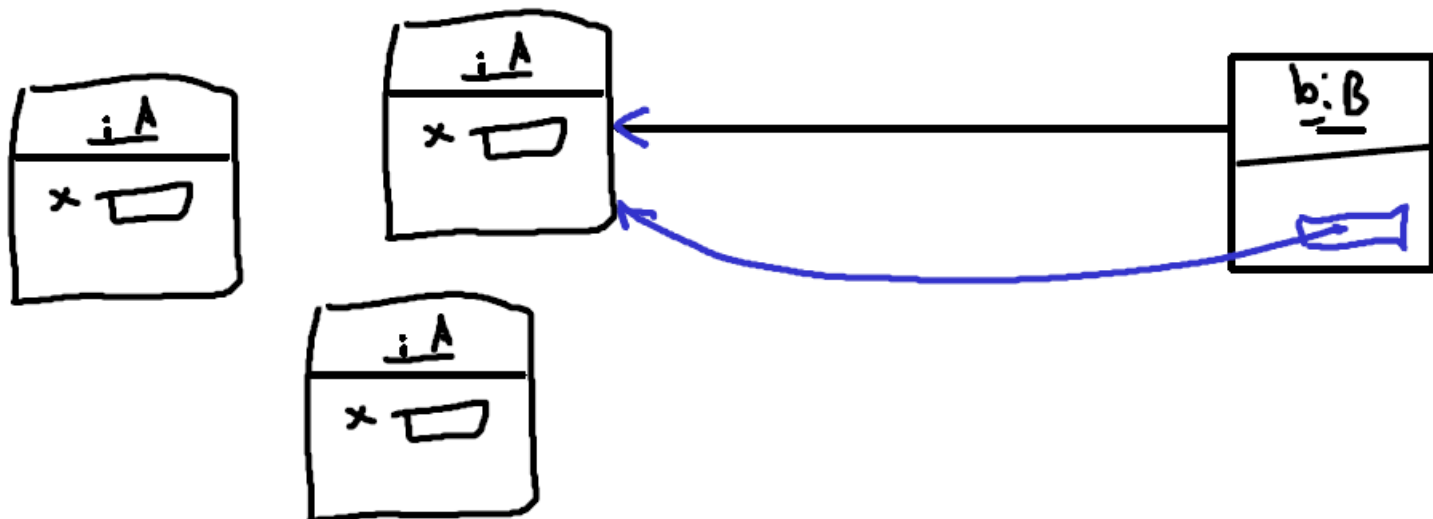
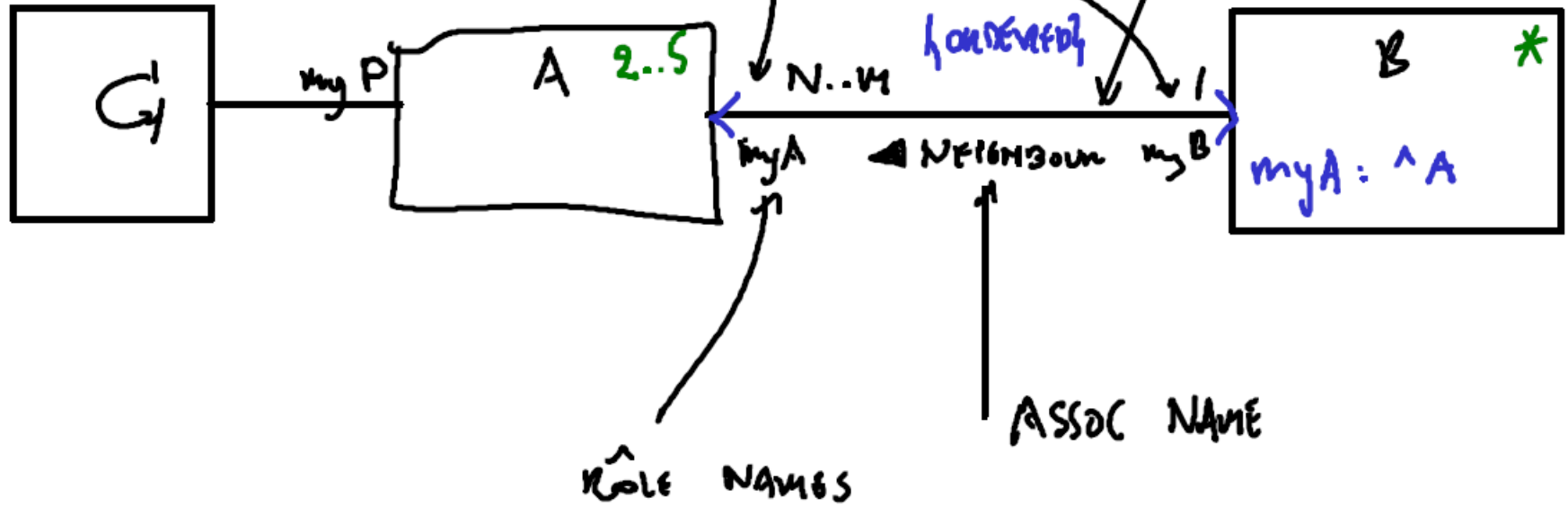
a single language may have multiple intensional descriptions



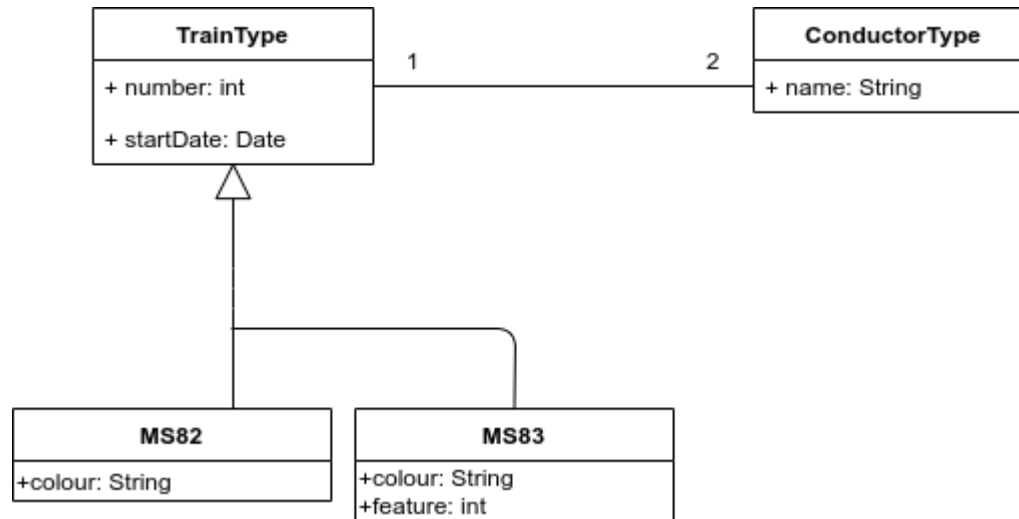
languages may (partially) overlap



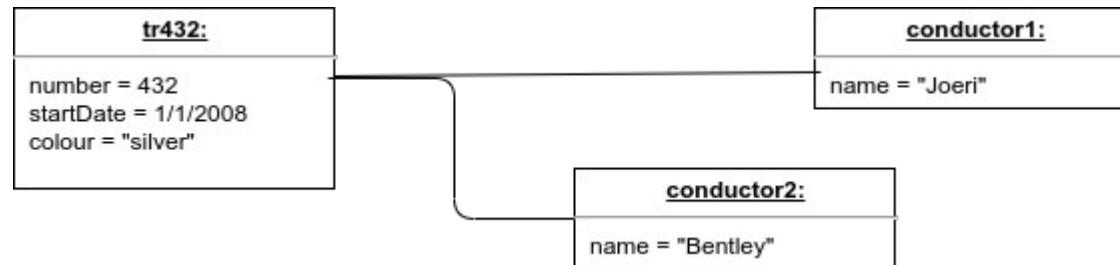
Class Diagrams



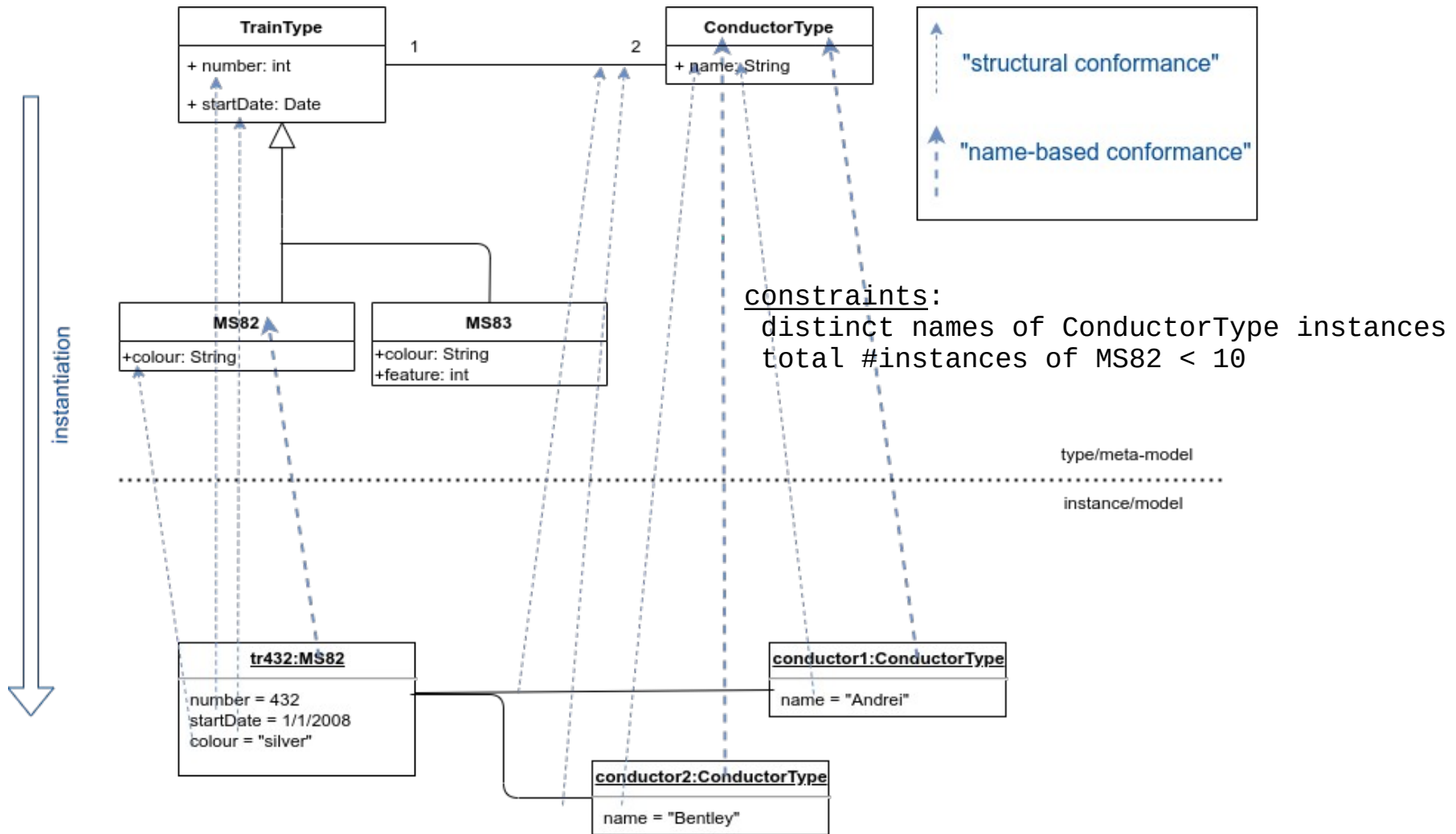
Class Diagram



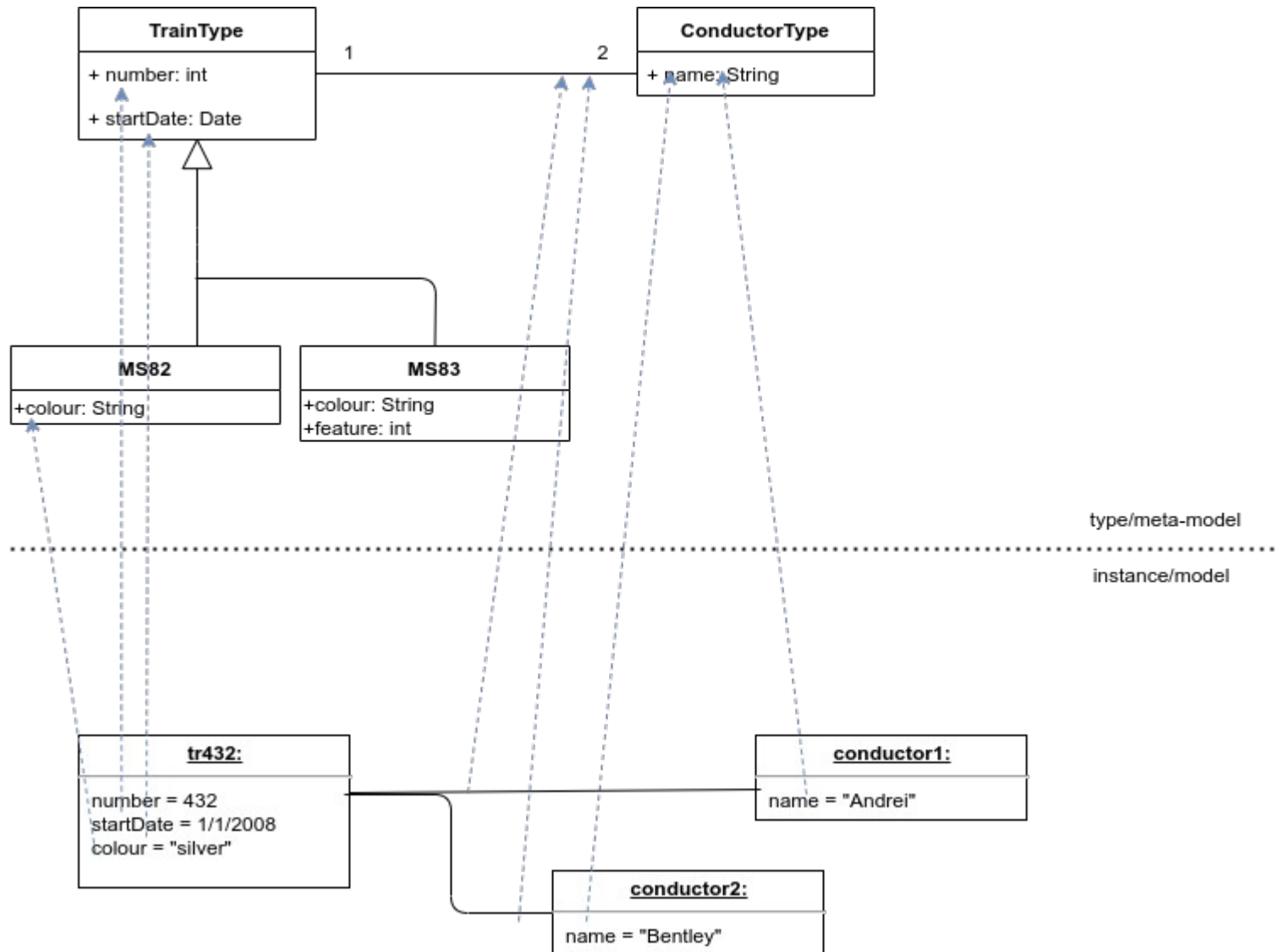
Object Diagram



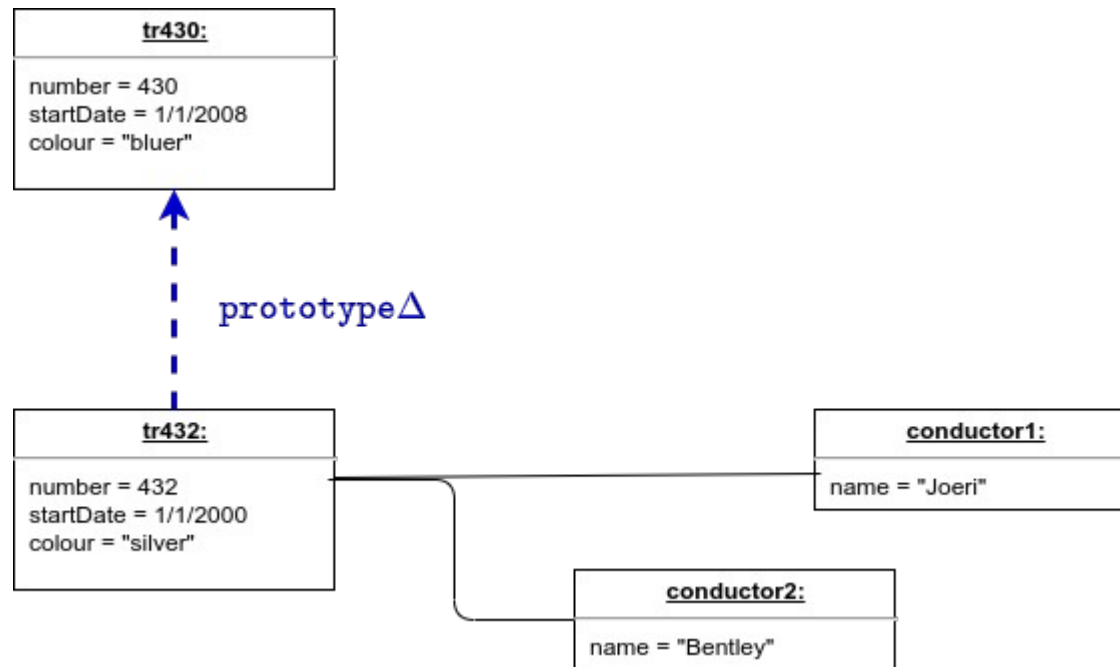
Class vs. Instance

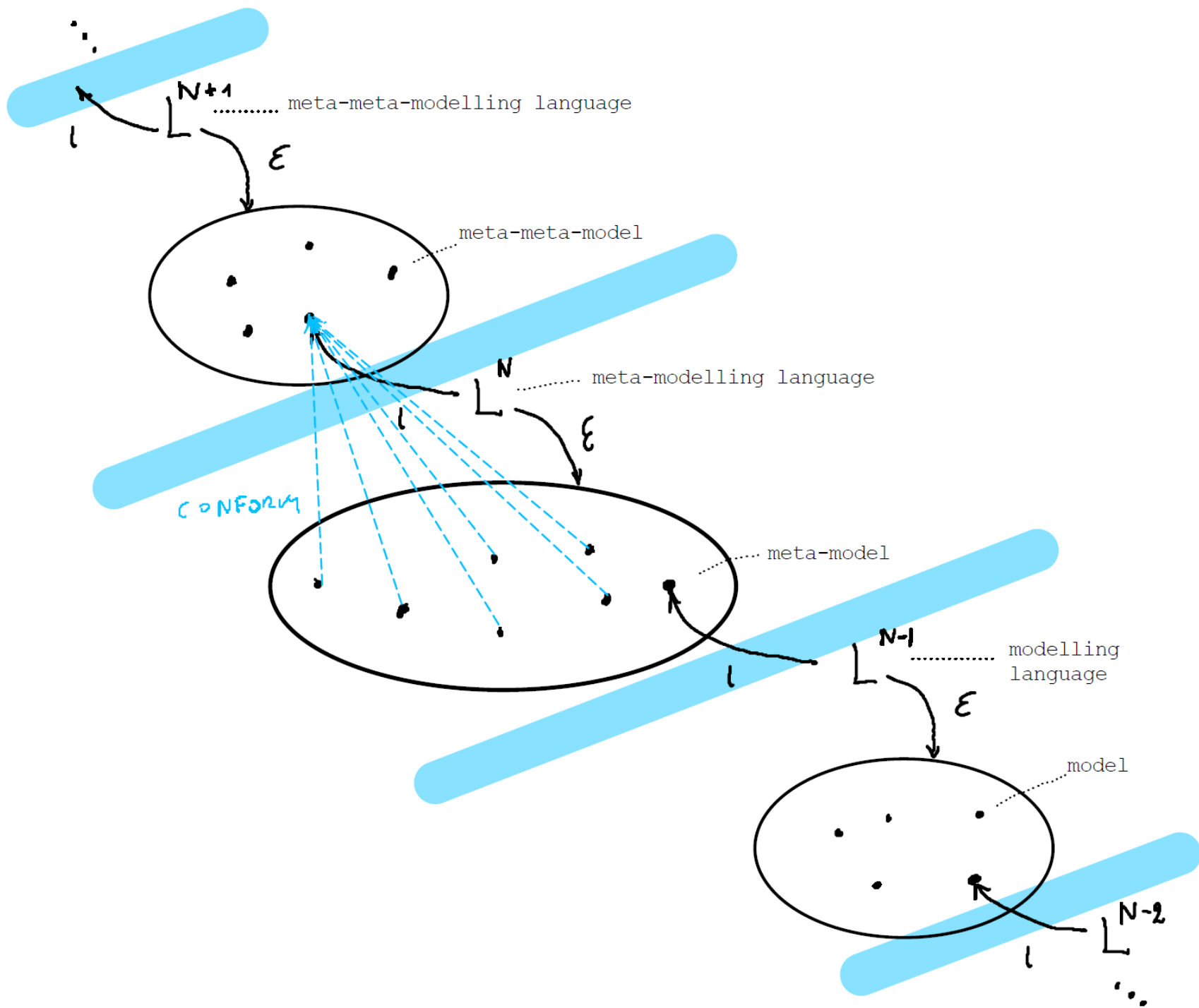


“duck typing”



delegation, object-based





meta-ness

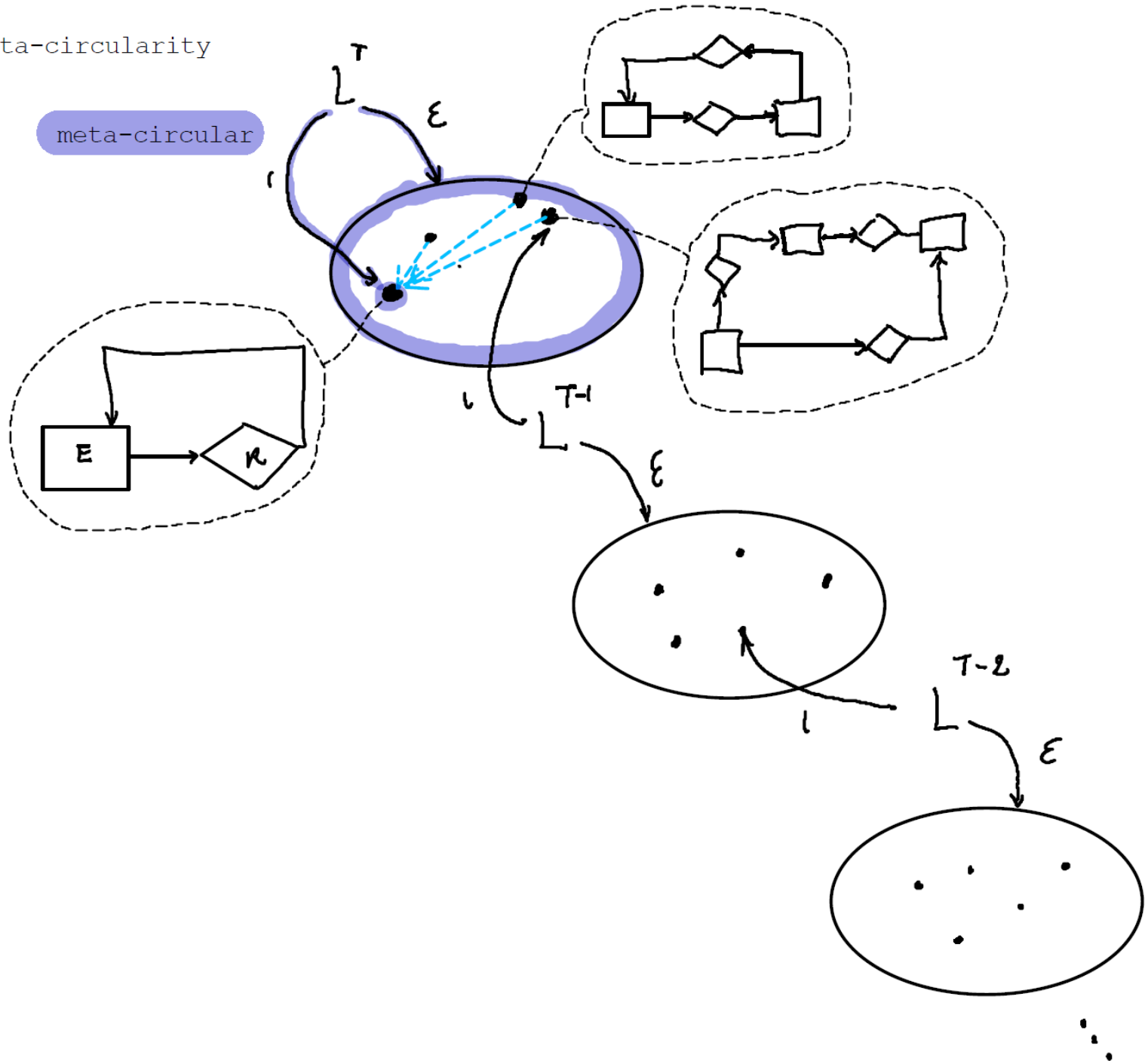
$$e_1 R^n e_2 = \begin{cases} \exists e : e_1 R^{n-1} e \wedge e R e_2, & n > 1 \\ e_1 R e_2 & , n = 1 \end{cases}$$

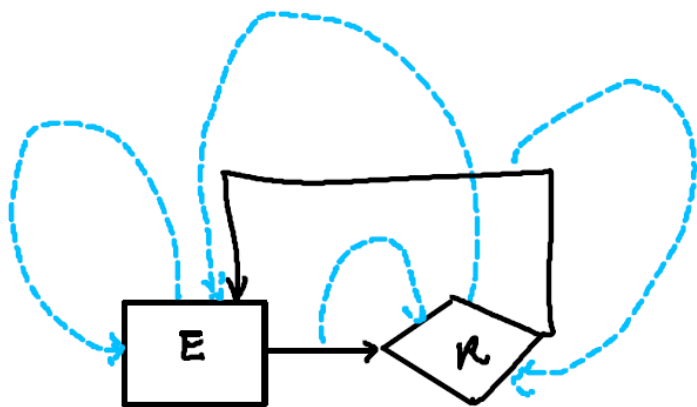
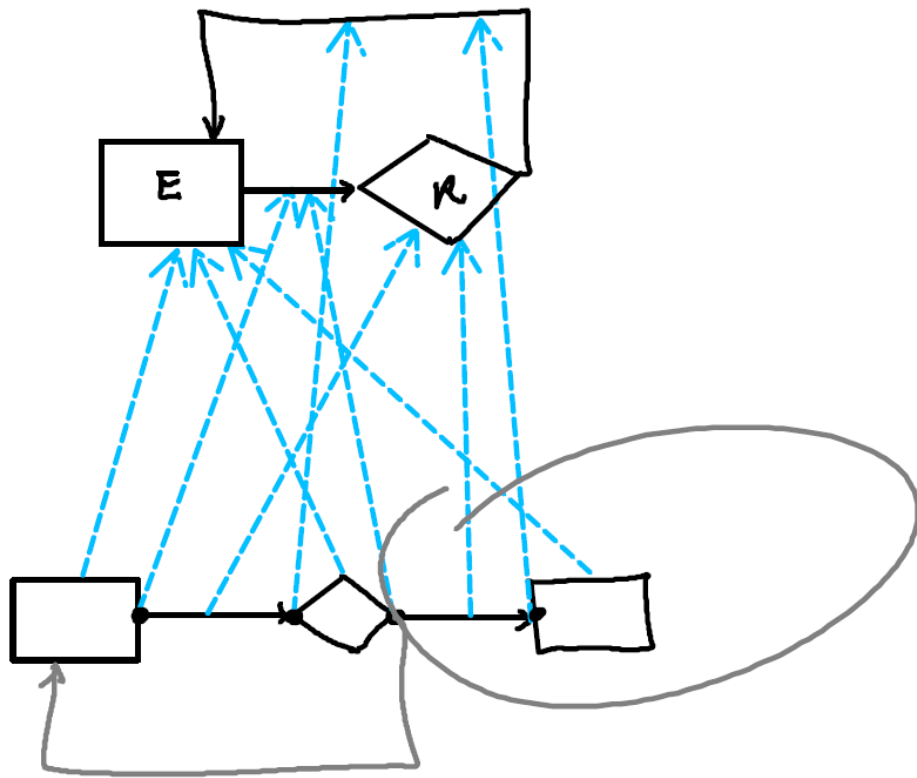
acyclic $\forall e_1, e_2, n : e_1 R^n e_2 \rightarrow \neg e_2 R e_1$, and

anti-transitive $\forall n \geq 2 : R^n \cap R = \emptyset$.

level respecting $\forall n, m :$
 $(\exists e_1, e_2 : e_1 R^n e_2 \wedge e_1 R^m e_2) \rightarrow$
 $n = m$

meta-circularity





Entity Relationship Diagram (ERD)

can be used as the MMMCL
Meta-Model at the Meta-Circular Level

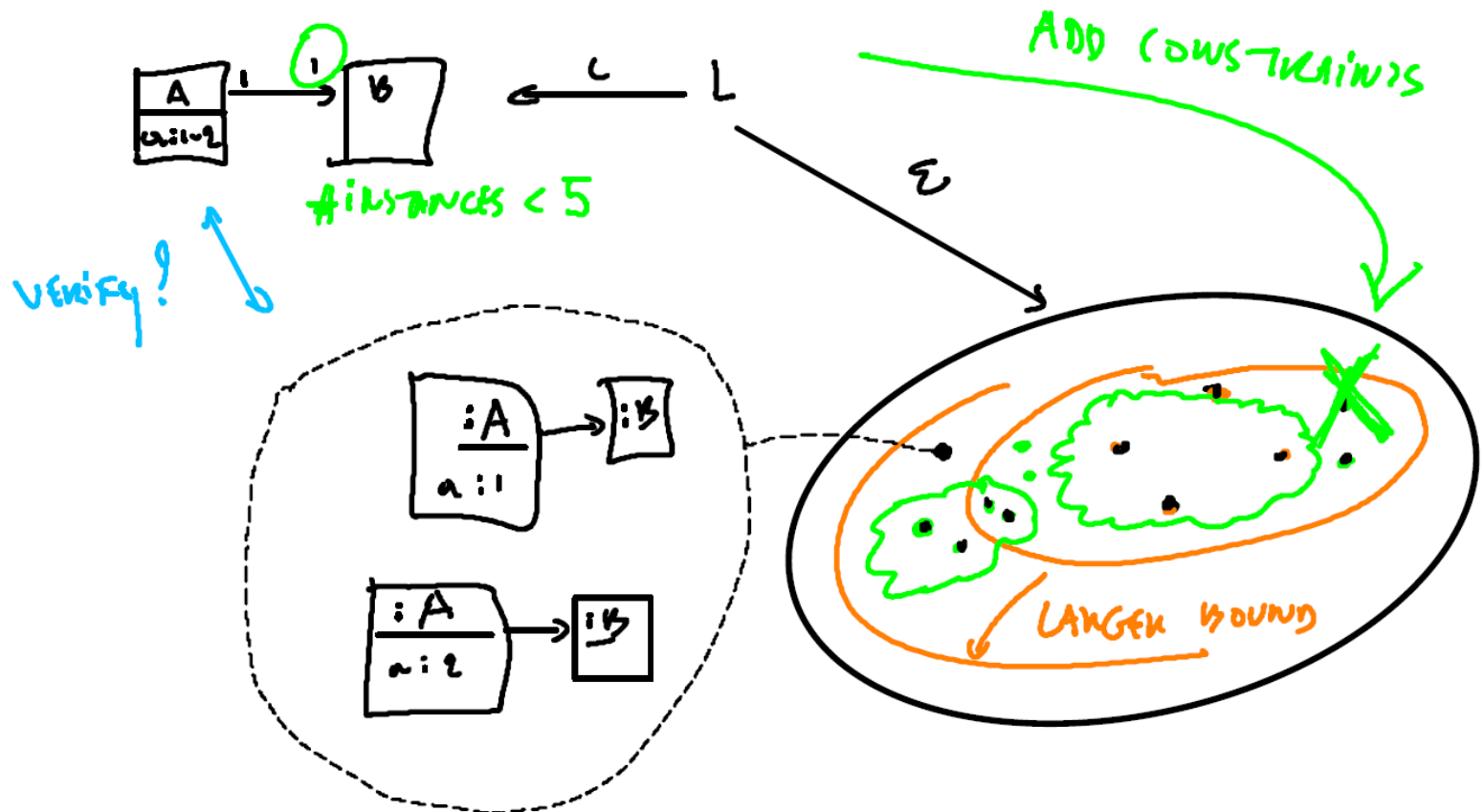
MMMCL can be used to "bootstrap"
a meta-modelling infrastructure.

Bounded Exploration (of a language defined through a meta-model)

<http://alloytools.org/> Daniel Jackson @ MIT

Alloy

MAX INSTANCES
"BOUNDED EXPLORATION"



<https://refinery.services>

```
% Metamodel

abstract class CompositeElement {
  contains Region[] regions
}

class Region {
  contains Vertex[] vertices opposite region
}

abstract class Vertex {
  container Region region opposite vertices
  contains Transition[] outgoingTransition opposite source
  Transition[] incomingTransition opposite target
}

class Transition {
  container Vertex source opposite outgoingTransition
  Vertex[1] target opposite incomingTransition
}

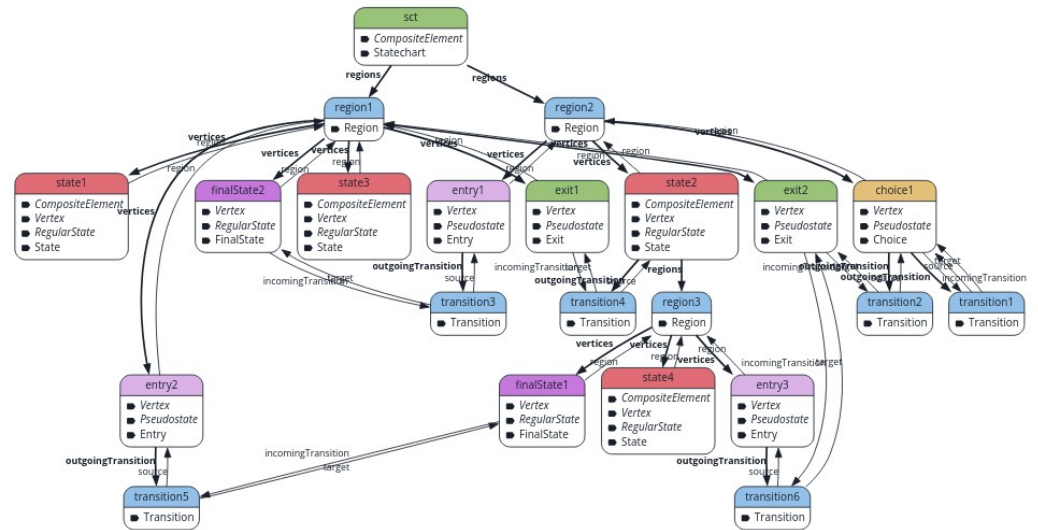
abstract class Pseudostate extends Vertex.

abstract class RegularState extends Vertex.

class Entry extends Pseudostate.

class Exit extends Pseudostate.
```

INITIAL MODEL GENERATED AT 20:14:01 (1)



Statechart(sct).

% Scope

scope node = 20..30, Region = 2..*, Choice = 1..*, Statechart += 0.

$$\mathcal{S} \triangleleft \mathcal{M}.$$

$$\rho(\mathcal{S}, \mathcal{M}) \rightarrow \mathcal{S} \triangleleft \mathcal{M}$$

$$\mathcal{S} \triangleleft \mathcal{M}_1 \wedge \rho(\mathcal{S}, \mathcal{M}_1)$$

$$\mathcal{M}_1 \triangleleft \mathcal{M}_2 \wedge \rho(\mathcal{S}, \mathcal{M}_2)$$

as opposed to $\rho(\mathcal{M}_1, \mathcal{M}_2)$.

$$\mathcal{M} = \alpha(\mathcal{S}).$$

$$\alpha = \tau \circ \alpha' \circ \pi$$

translation o abstraction o projection

$$h(a \oplus b) = h(a) \otimes h(b)$$

System vs. Model

representation relationship
(representative, placeholder for)

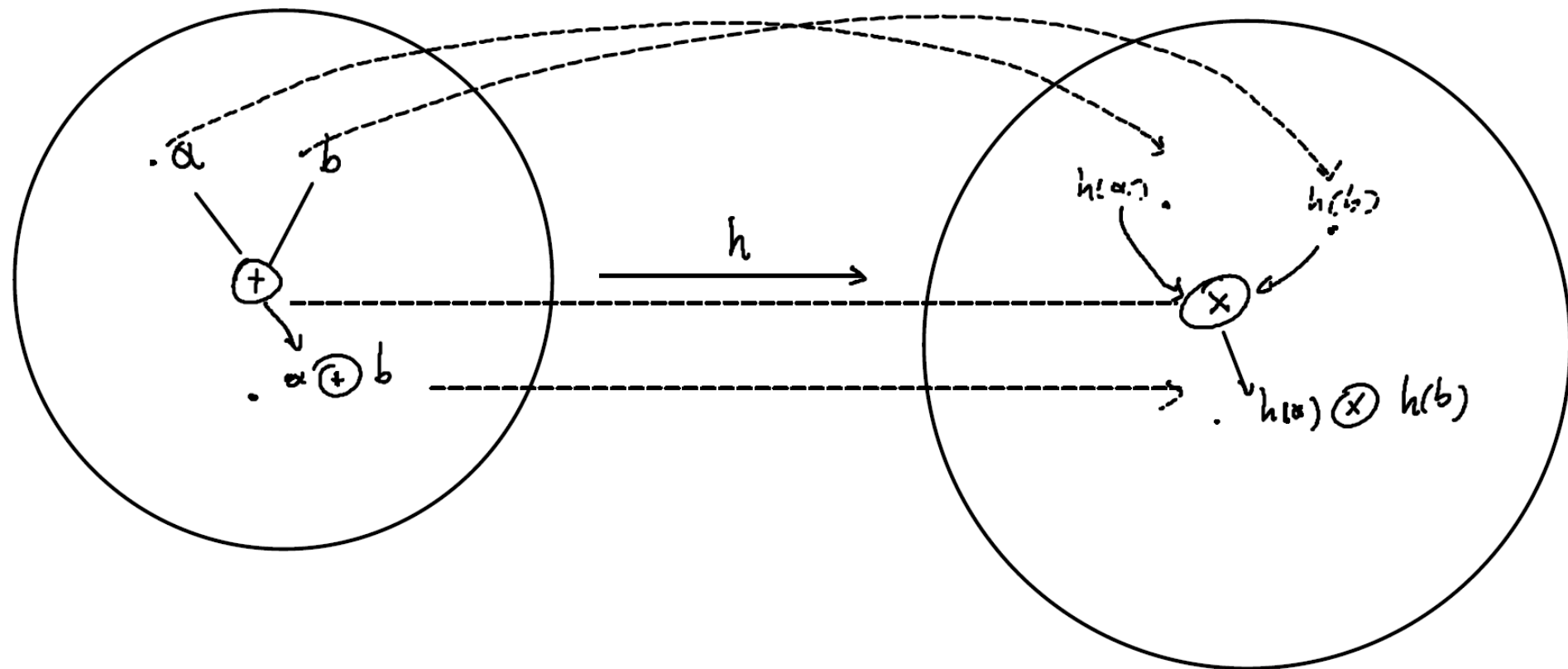
Model is abstraction of System

mapping feature	A model is based on an original. ⁴
reduction feature	A model only reflects a (relevant) selection of an original's properties.
pragmatic feature	A model needs to be usable in place of an original with respect to some purpose.

projection is an injective homomorphism

maps distinct elements of its domain to distinct elements of its codomain

morphism



token (aka “snapshot”) model: projection and translation $\alpha' = id$

$$\pi(e_1) = \pi(e_2) \not\Rightarrow e_1 = e_2$$

$$e_1 \sim_{\pi} e_2 \rightarrow \pi(e_1) = \pi(e_2) \quad \text{equivalence (wrt. property)}$$

for concept C pertaining to property satisfaction P:
extension vs. intension

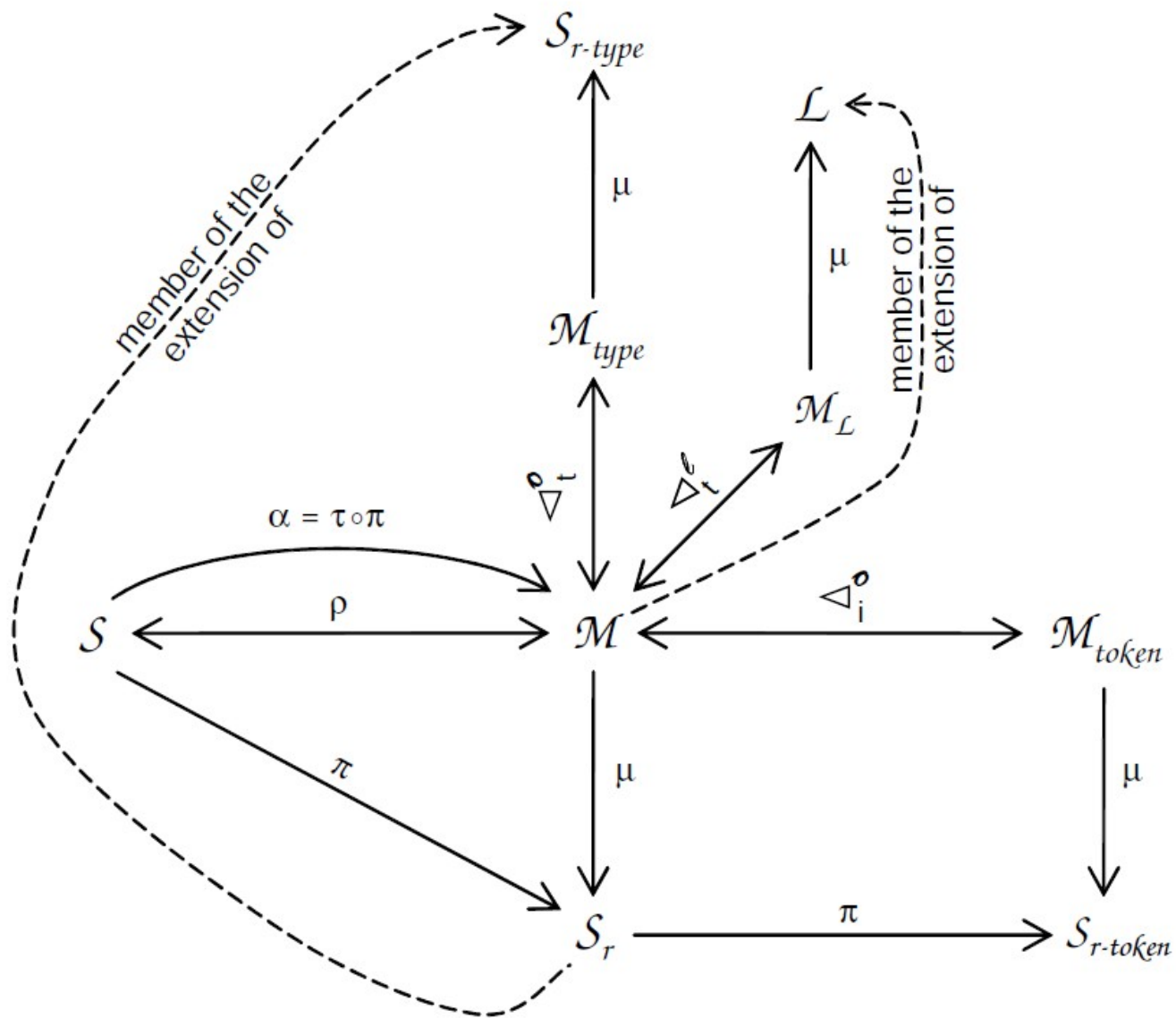
$$\varepsilon(C) = \{x \mid P(x)\}, \quad \text{where } P = \iota(C)$$

generalization vs. specialization

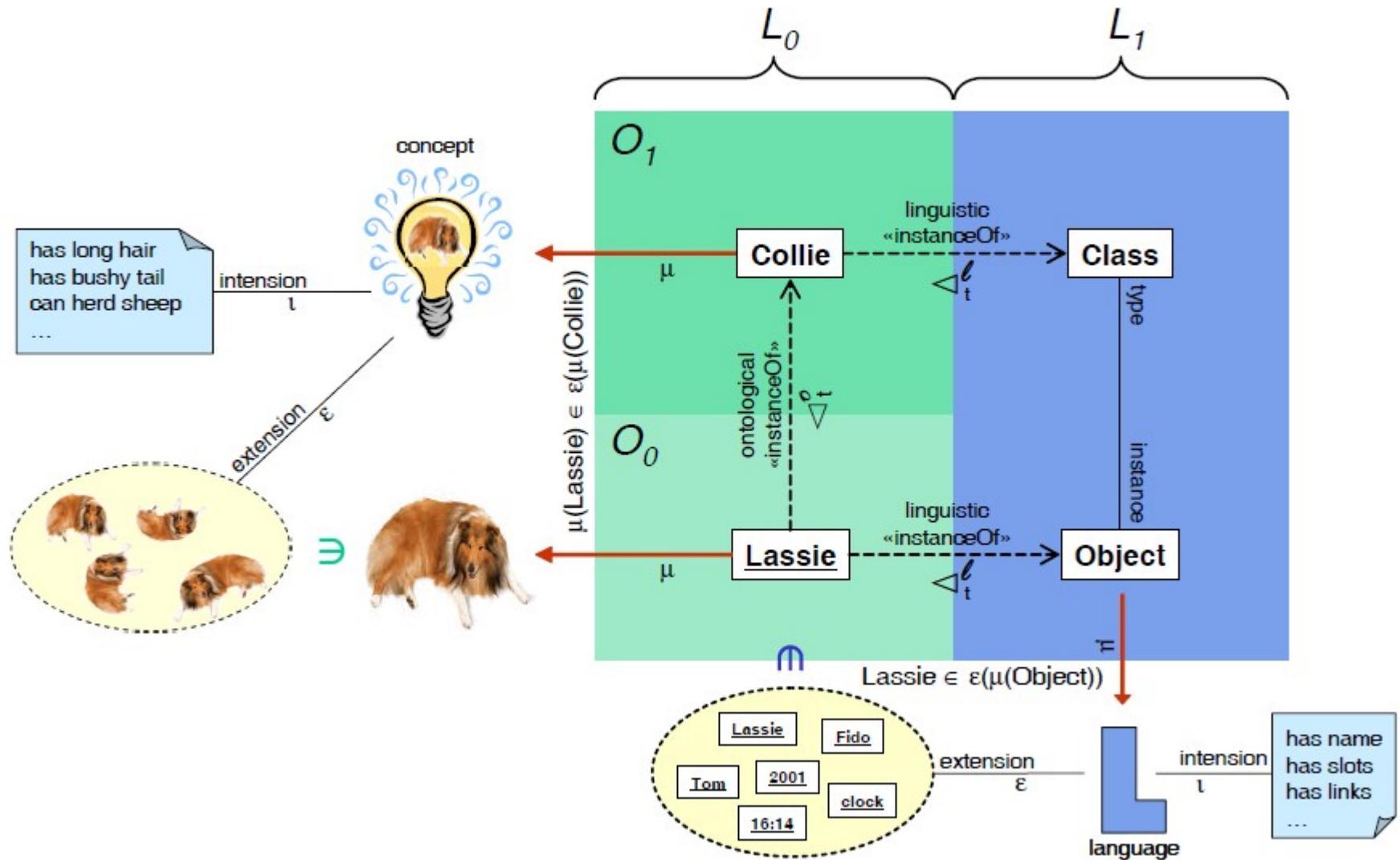
$$\forall x : \iota(C_{special})(x) \rightarrow \iota(C_{general_1})(x)$$

$$\varepsilon(C_{special}) \subseteq \varepsilon(C_{general}).$$

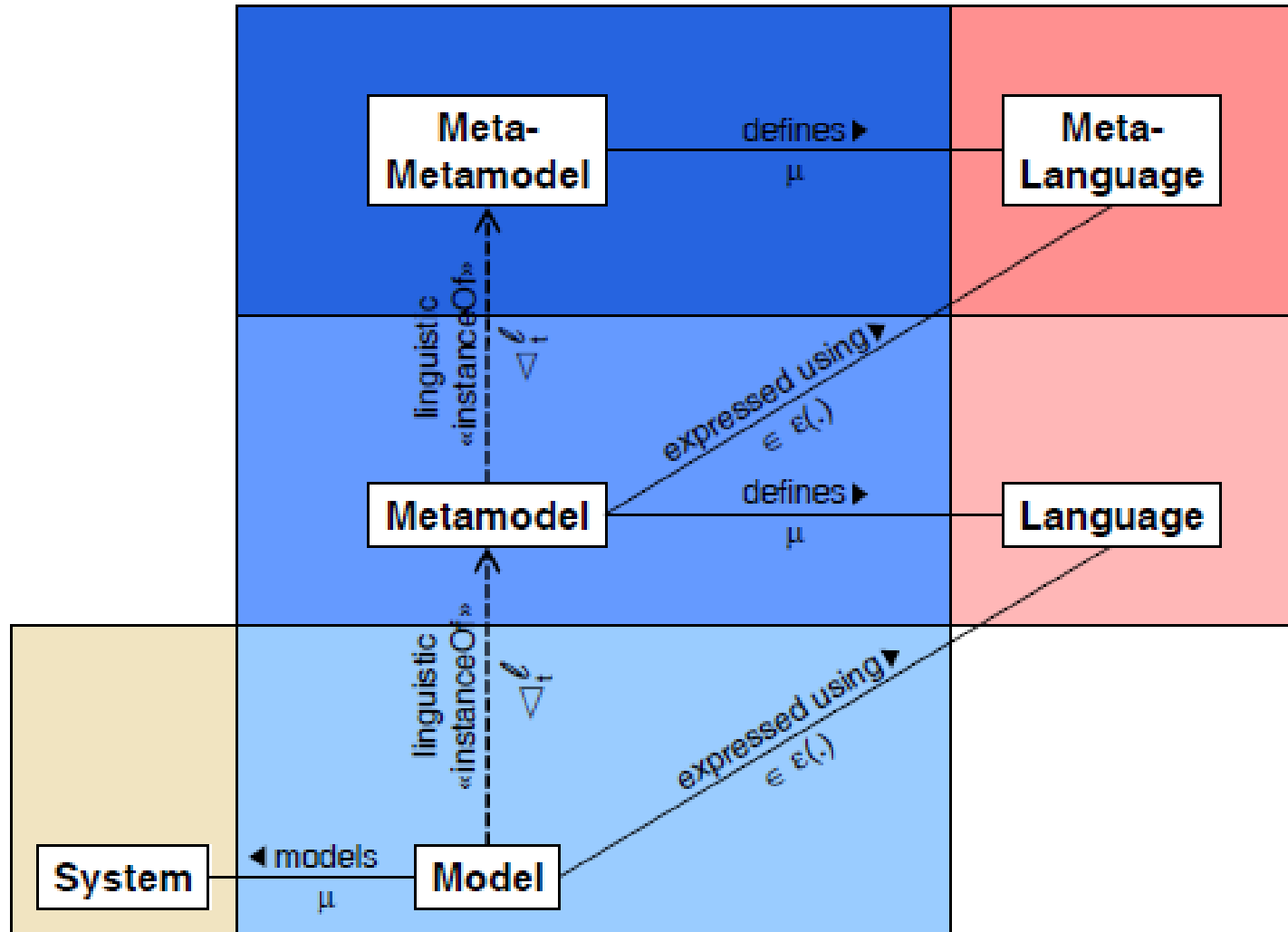
<i>notation</i>	<i>name</i>	<i>description</i>
α	abstraction	creates a model from a system using projection (π) and possibly classification (Λ) and generalization (Γ), hence $\mathcal{S} \triangleleft \alpha(\mathcal{S})$.
Λ	classification	creates a type model, hence $\mathcal{M} \triangleleft_t \Lambda(\mathcal{M})$
Γ	generalization	creates a supermodel, hence $\mathcal{S} \triangleleft_t \mathcal{M} \rightarrow \mathcal{S} \triangleleft_t \Gamma(\mathcal{M})$
π	projection	a homomorphic mapping creating a reduced system from a given system, using selection and reduction of information.
ρ	represents	records the intention of a model to represent a system.
μ	meaning	assigns meaning to a model (element). If $\rho(\mathcal{S}, \mathcal{M})$ then one may define $\mu(\mathcal{M}) = \pi(\mathcal{S})$.
$\triangleleft$	model-of	holds between a system and a model describing the former.
$\triangleleft_i$	token model-of	holds between a system and a model representing the former in a one-to-one fashion. Model elements may be regarded as designators for system elements.
$\triangleleft_t$	type model-of	holds between a system and a model classifying the former in a many-to-one fashion. Model elements are regarded as classifiers for system elements.
$\triangleleft^o$	ontological model-of	indicates that the model controls the <i>content</i> of its elements, hence $\mathcal{S} \triangleleft_t^o \mathcal{M} \Leftrightarrow \mu(\mathcal{S}) \in \varepsilon(\mu(\mathcal{M}))$ and $\mathcal{S} \triangleleft_i^o \mathcal{M} \Leftrightarrow \mu(\mathcal{M}) = \pi(\mu(\mathcal{S}))$. Assuming $\mu(\mathcal{S}) = \mathcal{S}$, for systems which do not model anything, we have $\mathcal{S} \triangleleft_i^o \mathcal{M} \Leftrightarrow \mu(\mathcal{M}) = \pi(\mathcal{S})$ and thus $\rho(\mathcal{S}, \mathcal{M}) \rightarrow \mathcal{S} \triangleleft_i^o \mathcal{M}$ (see definition of μ above).
$\triangleleft^l$	linguistic model-of	indicates that the model controls the <i>form</i> of its elements. This automatically implies $\triangleleft_t^l$ and, hence $\mathcal{S} \triangleleft_t^l \mathcal{M} \Leftrightarrow \mathcal{S} \in \varepsilon(\mu(\mathcal{M}))$



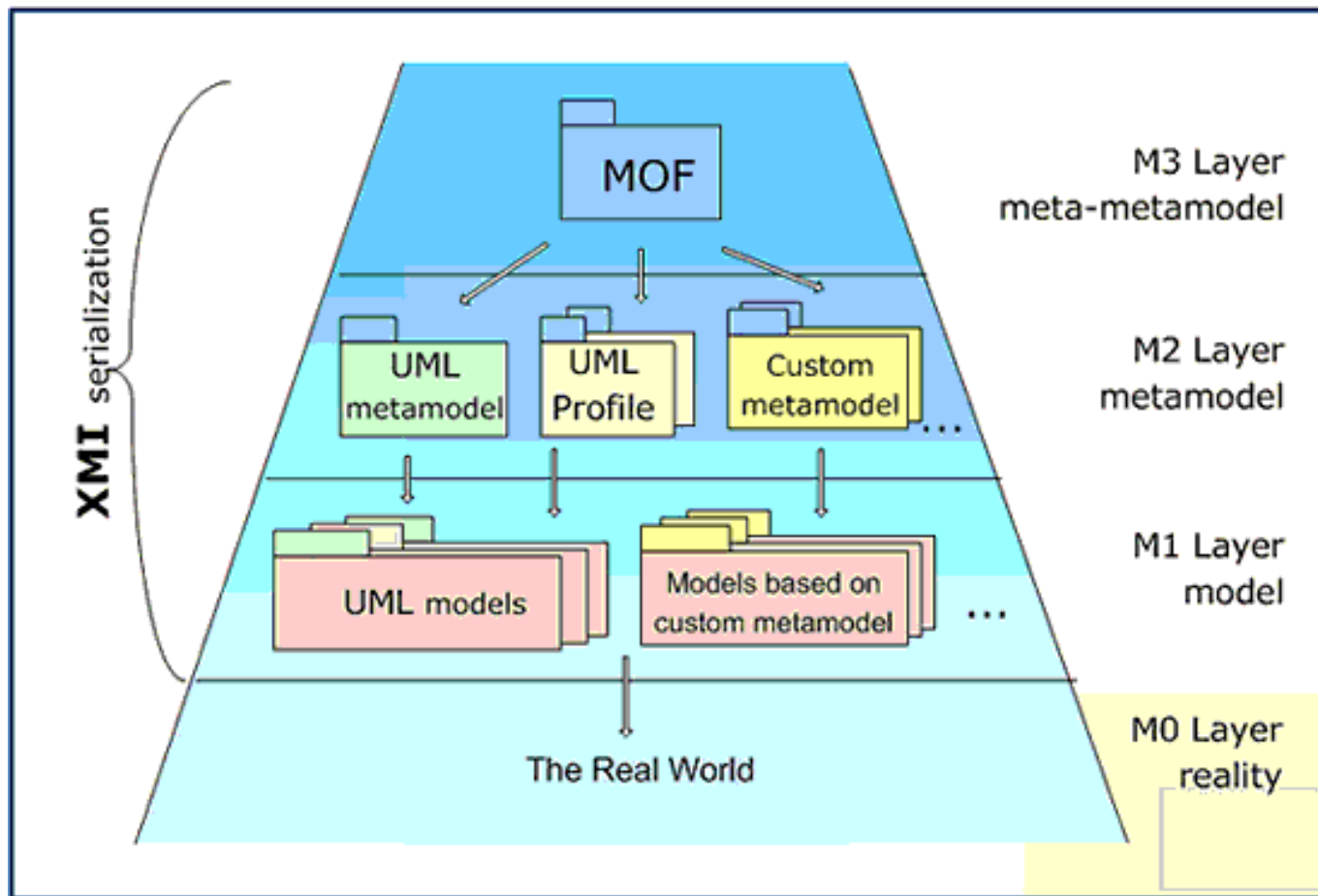
Ontological vs. Linguistic Conformance/Instantiation

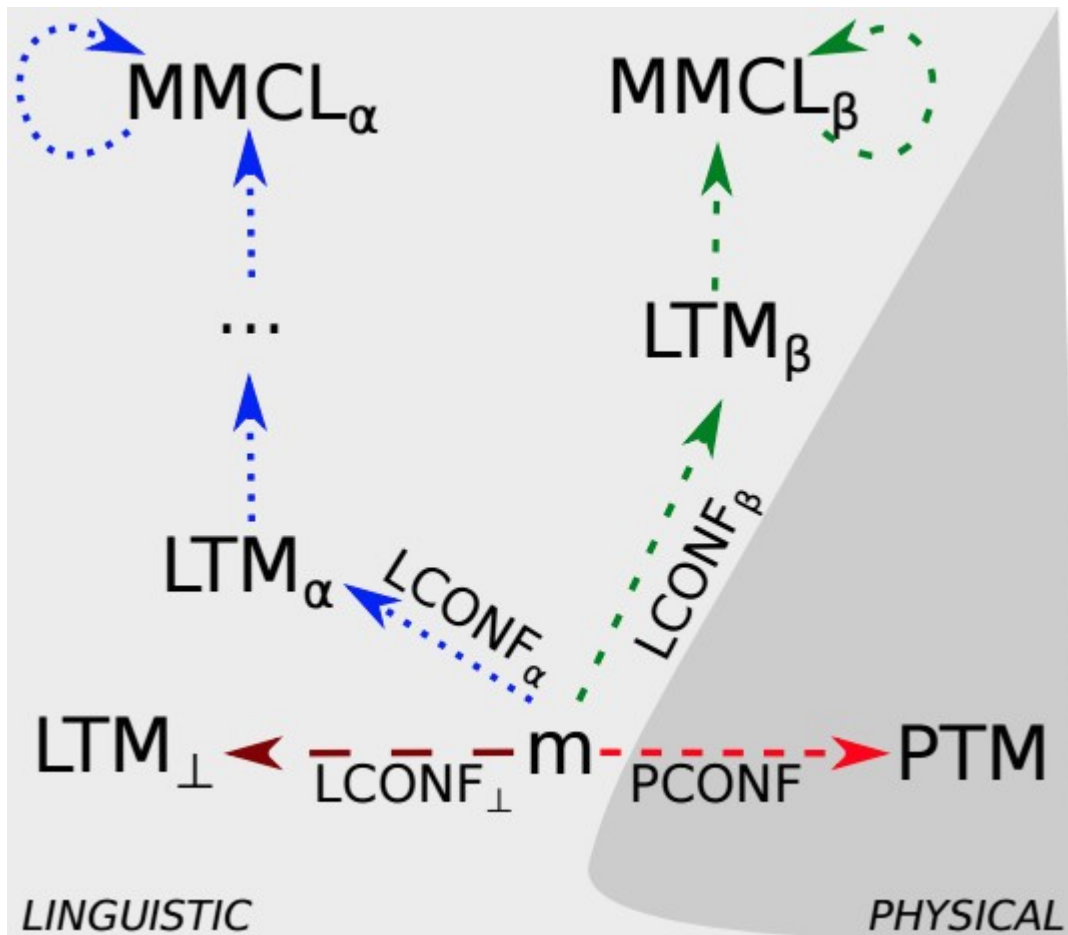


Language Definition Stack



Meta-hierarchy – OMG's 4 Layer Architecture

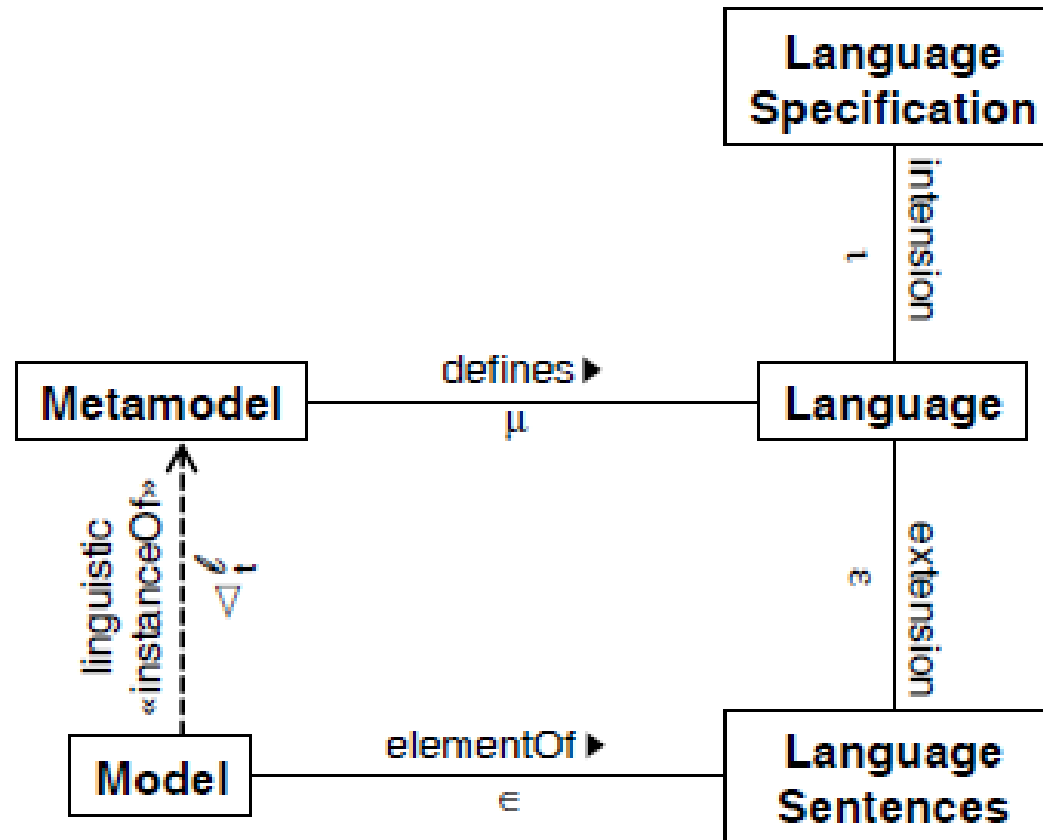




Simon Van Mierlo, Bruno Barroca, Hans Vangheluwe, Eugene Syriani, and Thomas Kuehne. Multi-level modelling in the Modelverse. In The first Workshop on Multi-Level Modelling (MULTI), co-located with the 17th International Conference on Model Driven Engineering Languages and Systems (MoDELS), Valencia, Spain, volume 1286 of CEUR Workshop Proceedings , pages 83 - 92, 2014.

Yentl Van Tendeloo, Bruno Barroca, Simon Van Mierlo, Hans Vangheluwe. Modelverse specification. 2016.
<http://msdl.cs.mcgill.ca/people/yentl/files/Modelverse.pdf>

Meta-models as Language Definitions

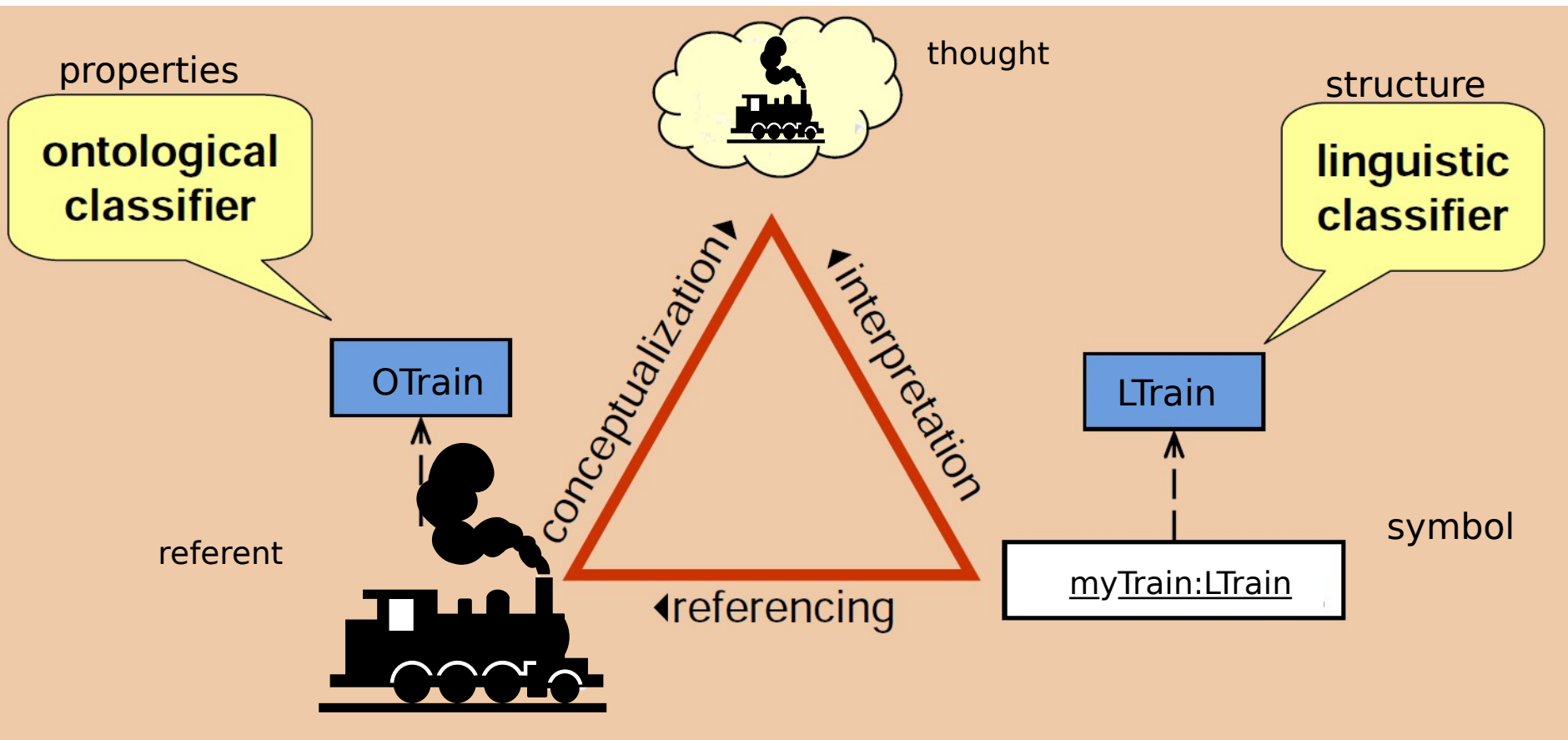


Triangle of Meaning

aka triangle of reference or semiotic triangle

Ogden and Richards in “the meaning of meaning” (1923)

caveat: collaborative modelling only works if collaborators have a shared interpretation



Syntax, **Semantics**, and all that Stuff

David Harel, Bernhard Rumpe.

Meaningful Modeling: What's the Semantics of "Semantics"?

IEEE Computer, vol. 37, no. 10, pp. 64-72, October, 2004.



- “operational” semantics
- “denotational” (transformational) semantics

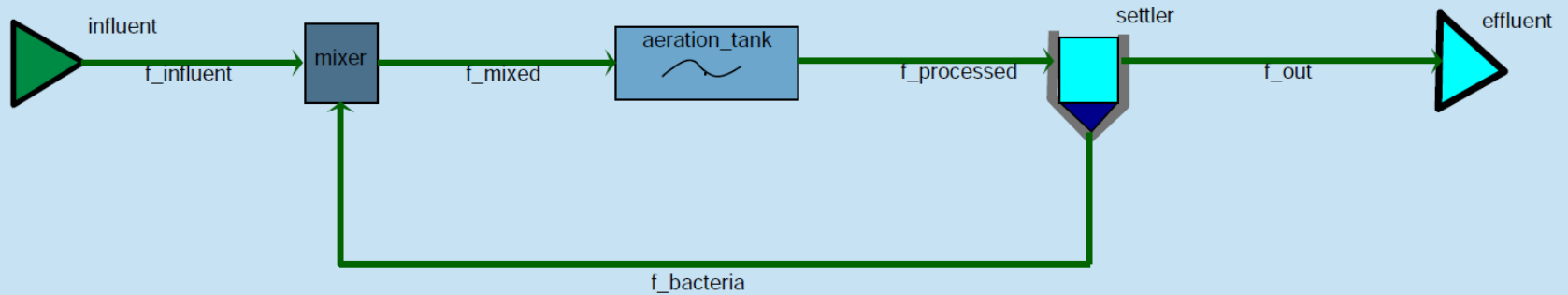
Operational vs. Denotational (Translational) semantics



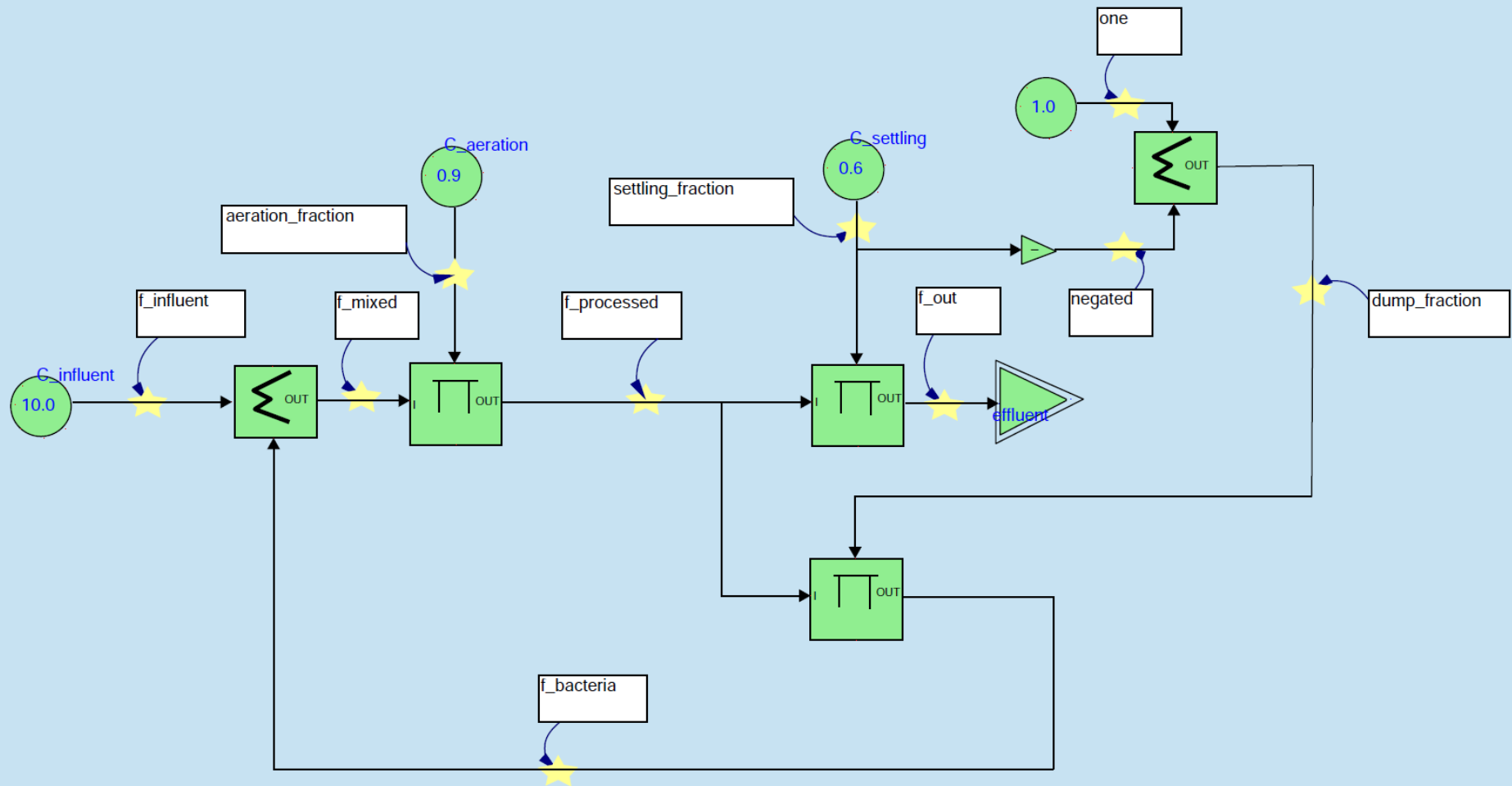
NATO's Sarajevo Waste Water Treatment Plant

www.nato.int/sfor/cimic/env-pro/waterpla.htm

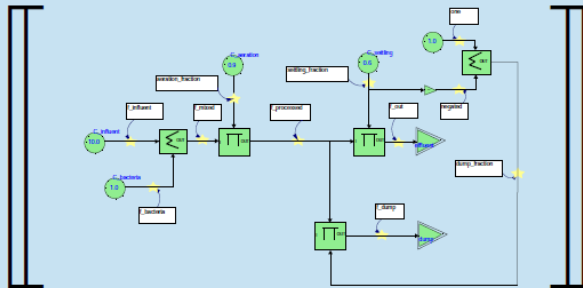
What does this WWTP model mean?



... its meaning (steady-state abstraction): Causal Block Diagram (CBD)



Meaning of the CBD ... semantic mapping onto algEqns

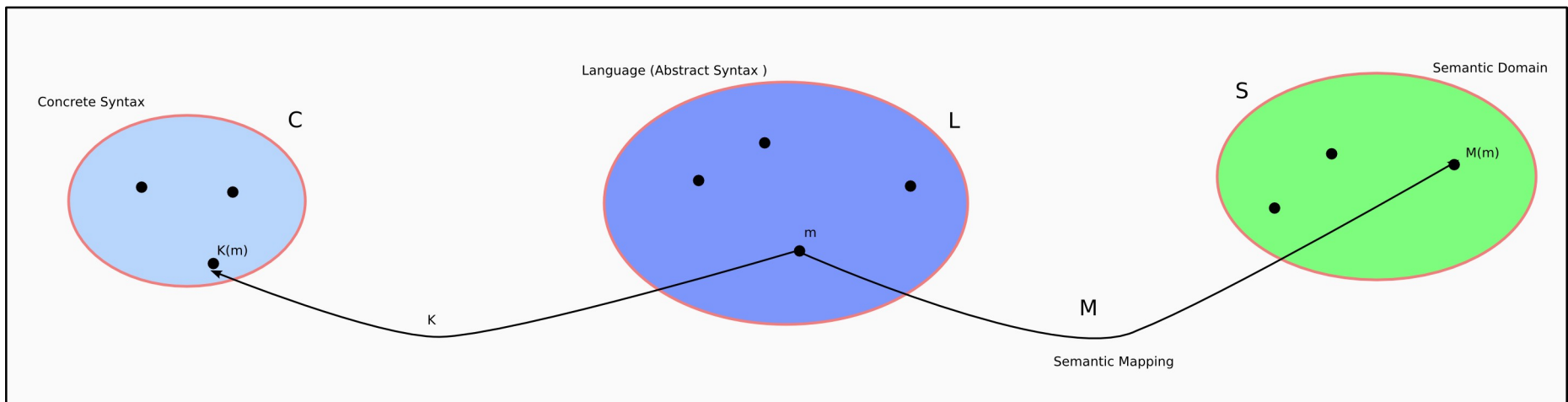


= {

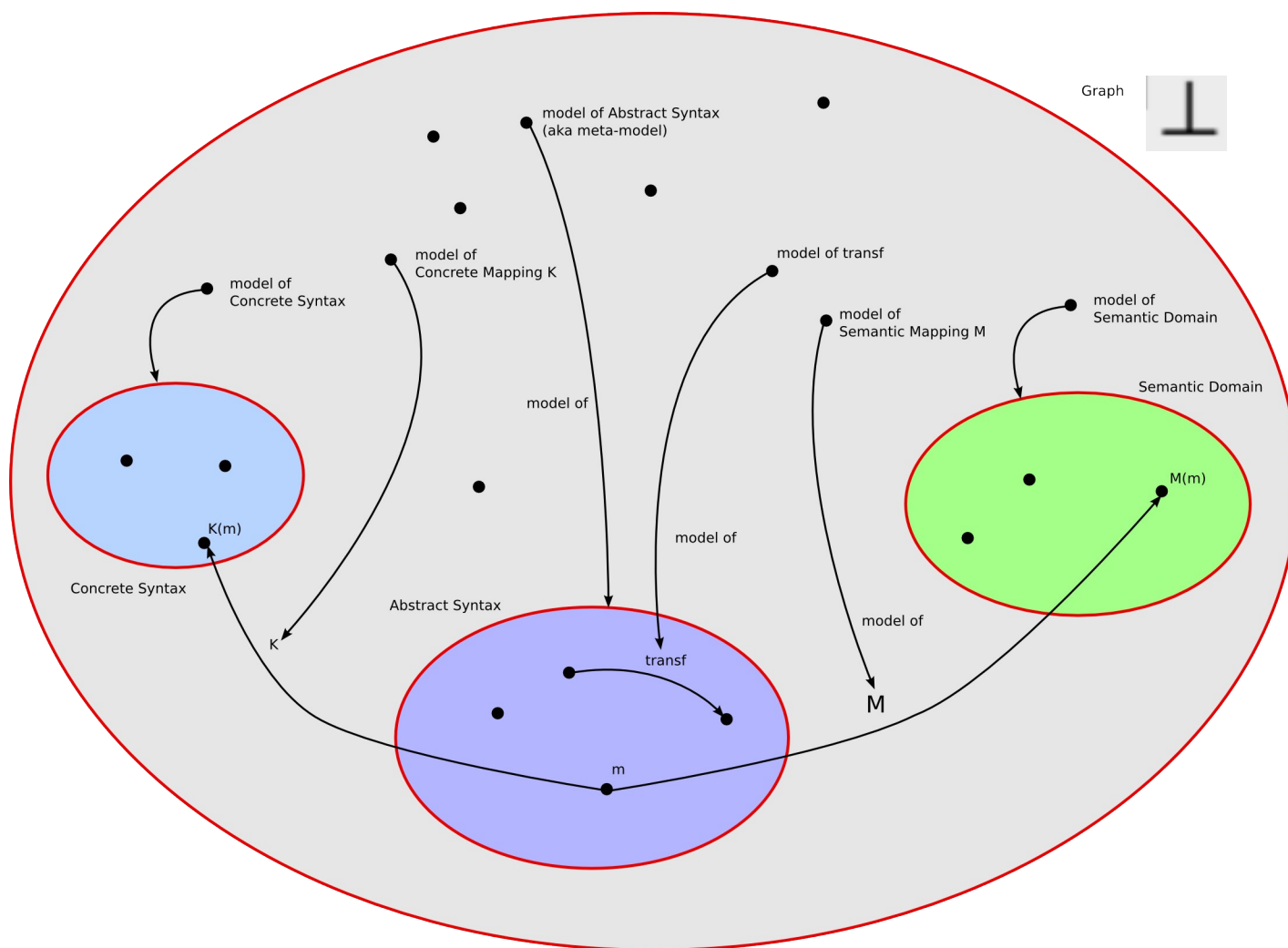
$f_influent$	=	$C_influent$
$f_bacteria$	=	$C_bacteria$
f_mixed	=	$f_influent + f_bacteria$
$aeration_fraction$	=	$C_aeration$
$f_processed$	=	$aeration_fraction * f_mixed$
$setling_fraction$	=	$C_setling$
$negated$	=	$-setling_fraction$
one	=	1
$dump_fraction$	=	$one + negated$
f_dump	=	$f_processed * dump_fraction$
f_out	=	$setling_fraction * f_processed$

“linguistic” view on Modelling Languages/Formalisms: Syntax and Semantics

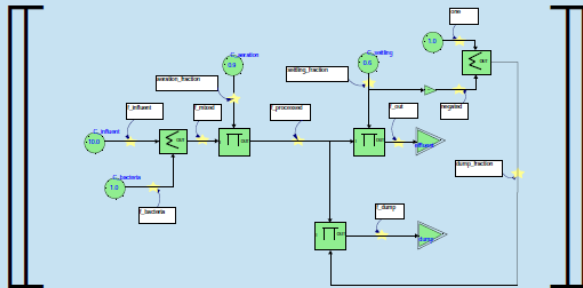
Concrete Formalism F



Explicit “linguistic” Modelling of Modelling Languages/Formalisms



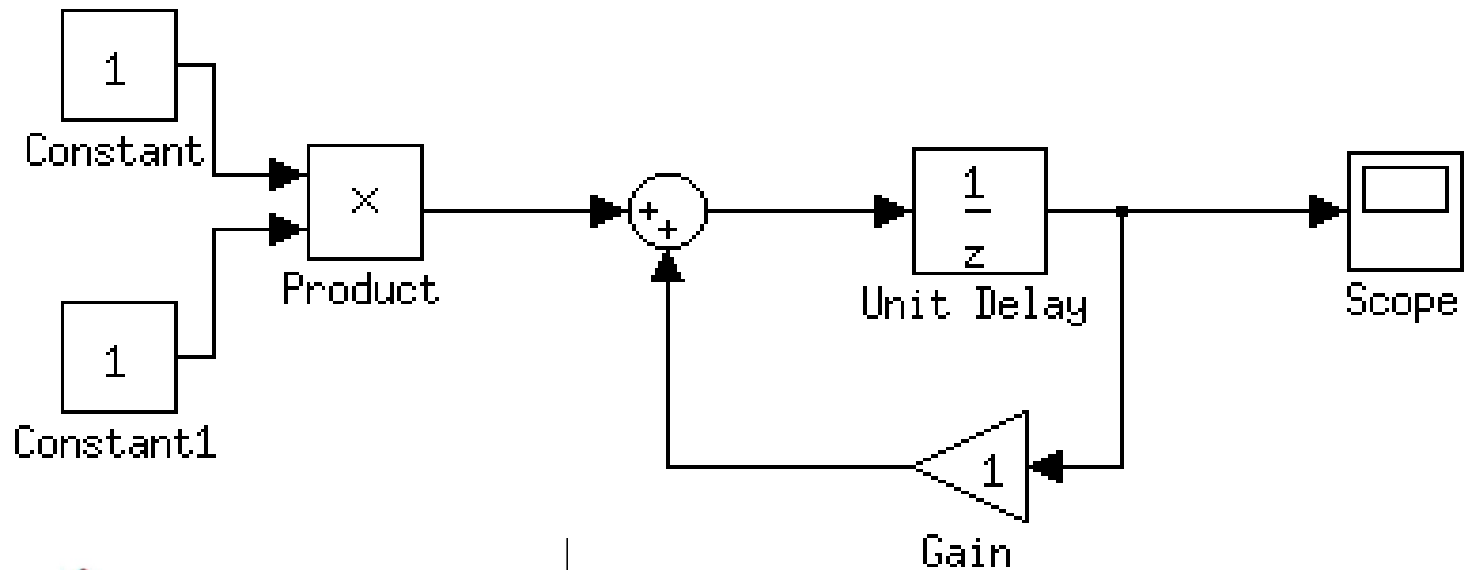
Meaning of the CBD ... semantic mapping onto algEqns



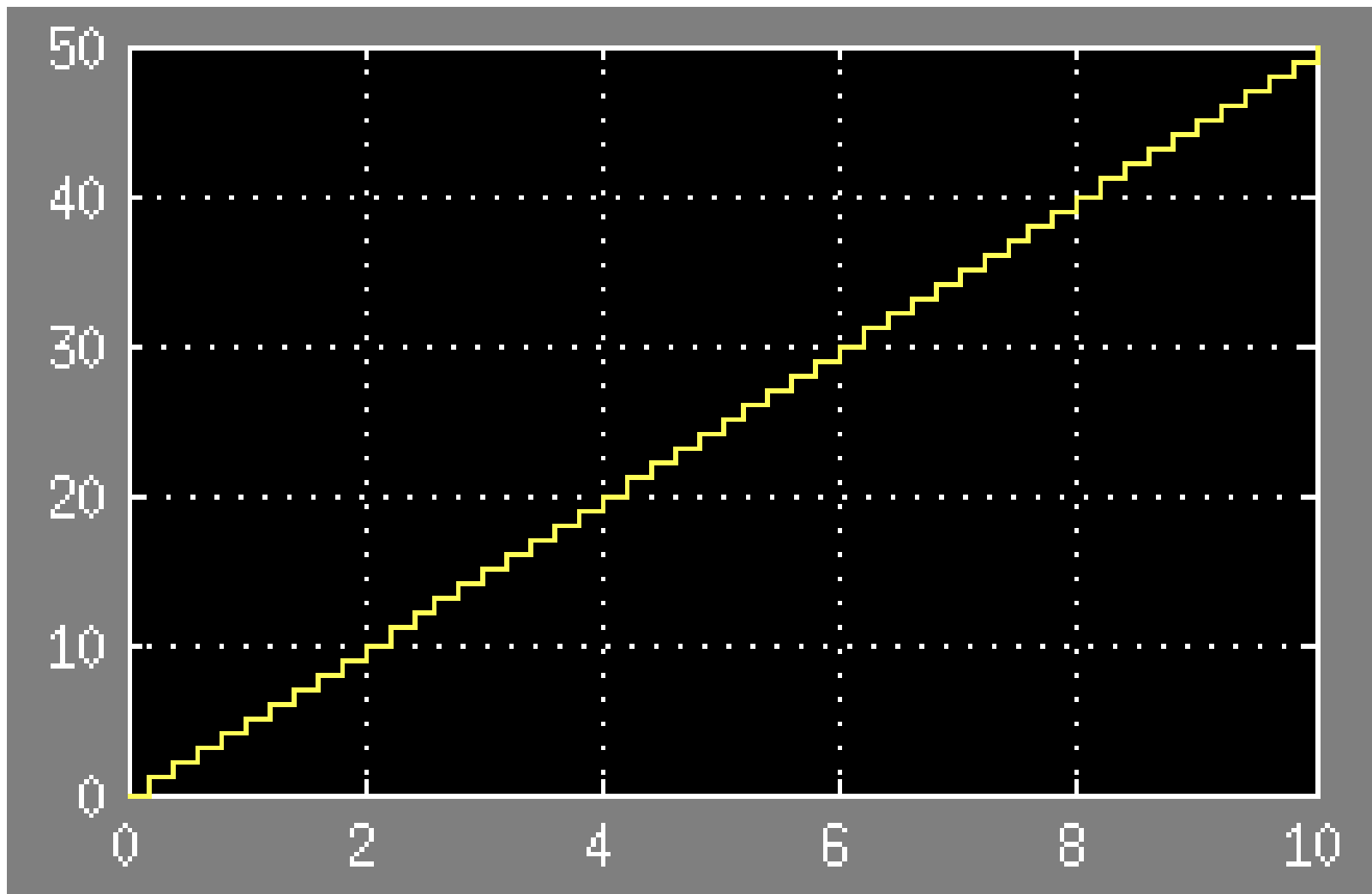
= {

$f_influent$	=	$C_influent$
$f_bacteria$	=	$C_bacteria$
f_mixed	=	$f_influent + f_bacteria$
$aeration_fraction$	=	$C_aeration$
$f_processed$	=	$aeration_fraction * f_mixed$
$setling_fraction$	=	$C_setling$
$negated$	=	$-setling_fraction$
one	=	1
$dump_fraction$	=	$one + negated$
f_dump	=	$f_processed * dump_fraction$
f_out	=	$setling_fraction * f_processed$

Causal Block Diagrams (syntax)

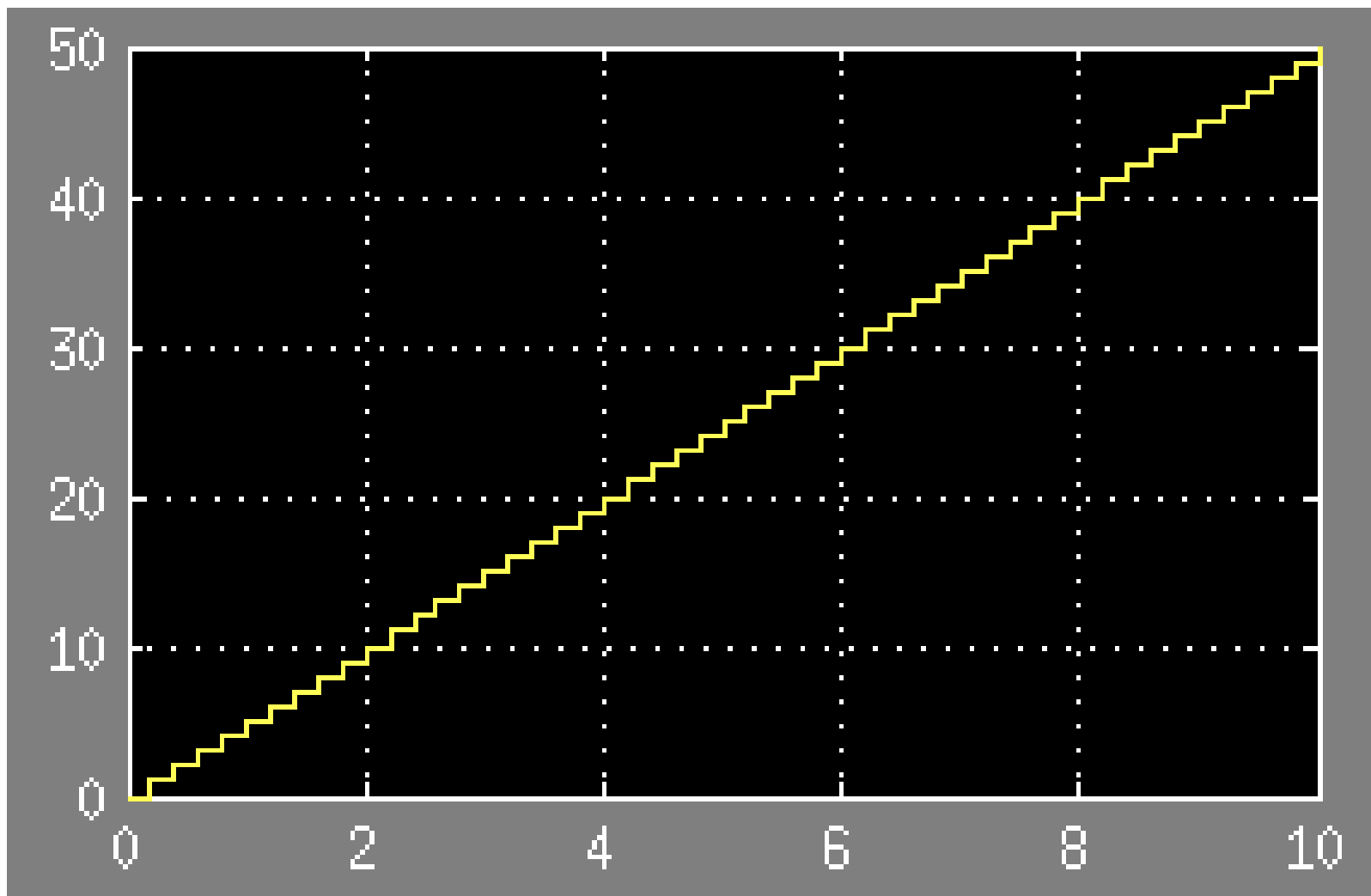


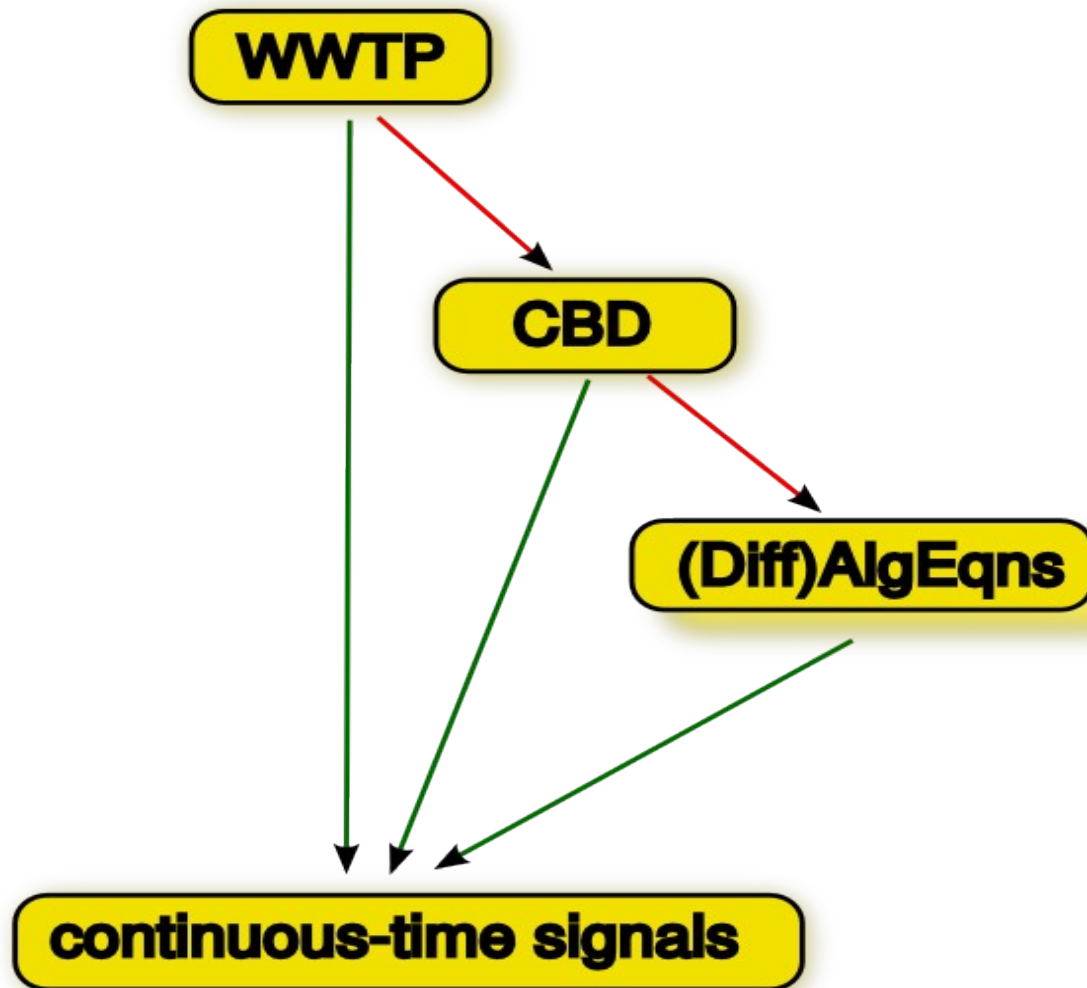
Causal Block Diagrams (semantics)



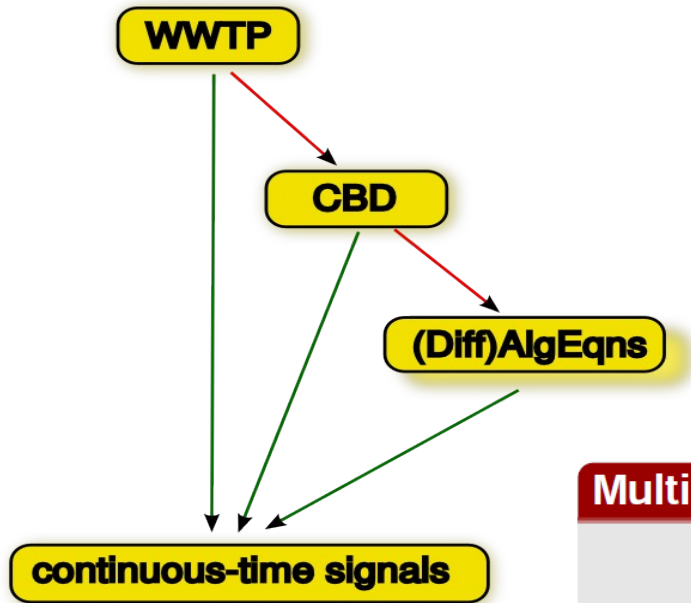
```
logicalTime  $\leftarrow$  0  
while not end_condition do  
    schedule  $\leftarrow$  LOOPDETECT(DEPGRAPH(cbd))  
    for gblock in schedule do  
        COMPUTE(gblock)  
    end for  
    logicalTime  $\leftarrow$  logicalTime +  $\Delta t$   
end while
```

Causal Block Diagrams (semantics)

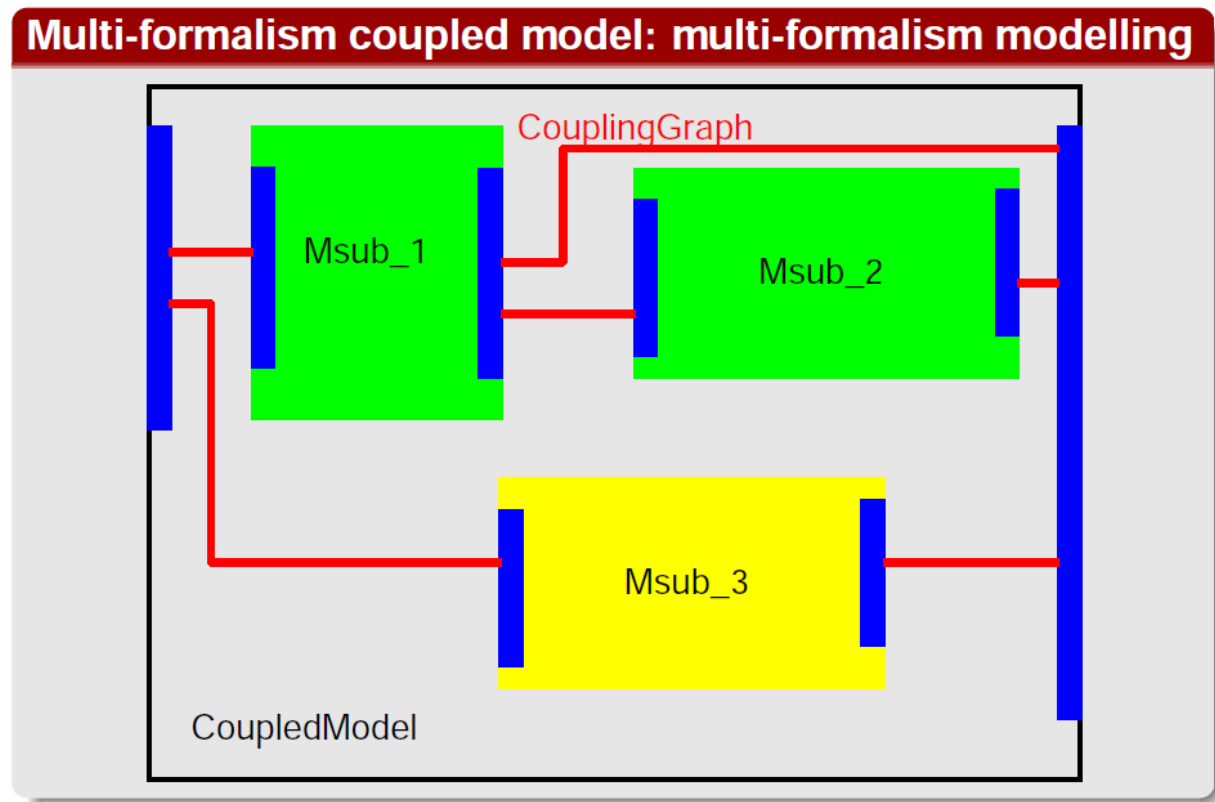


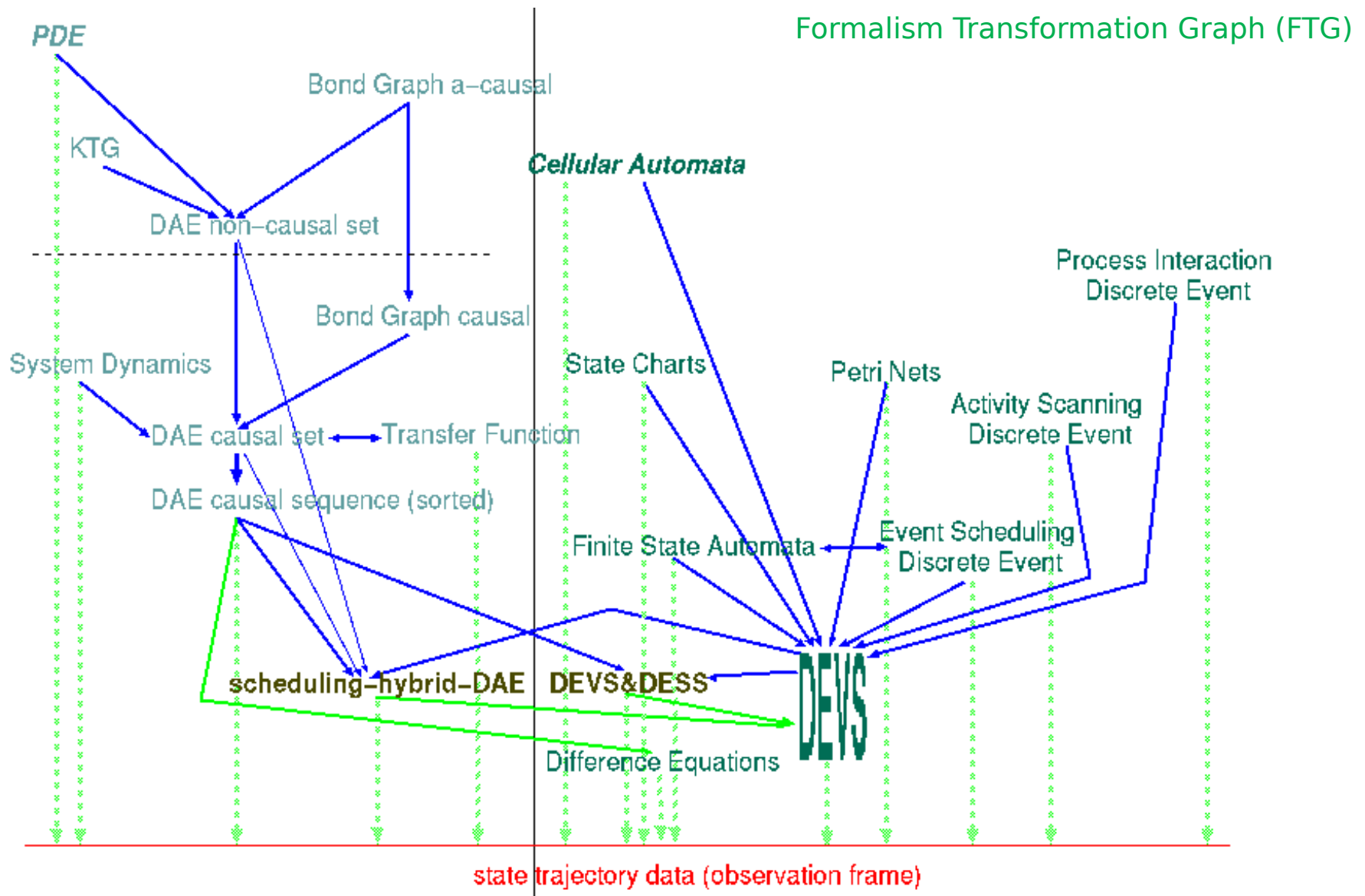


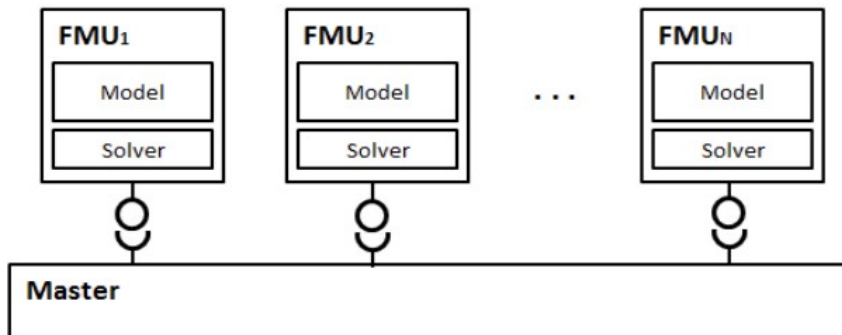
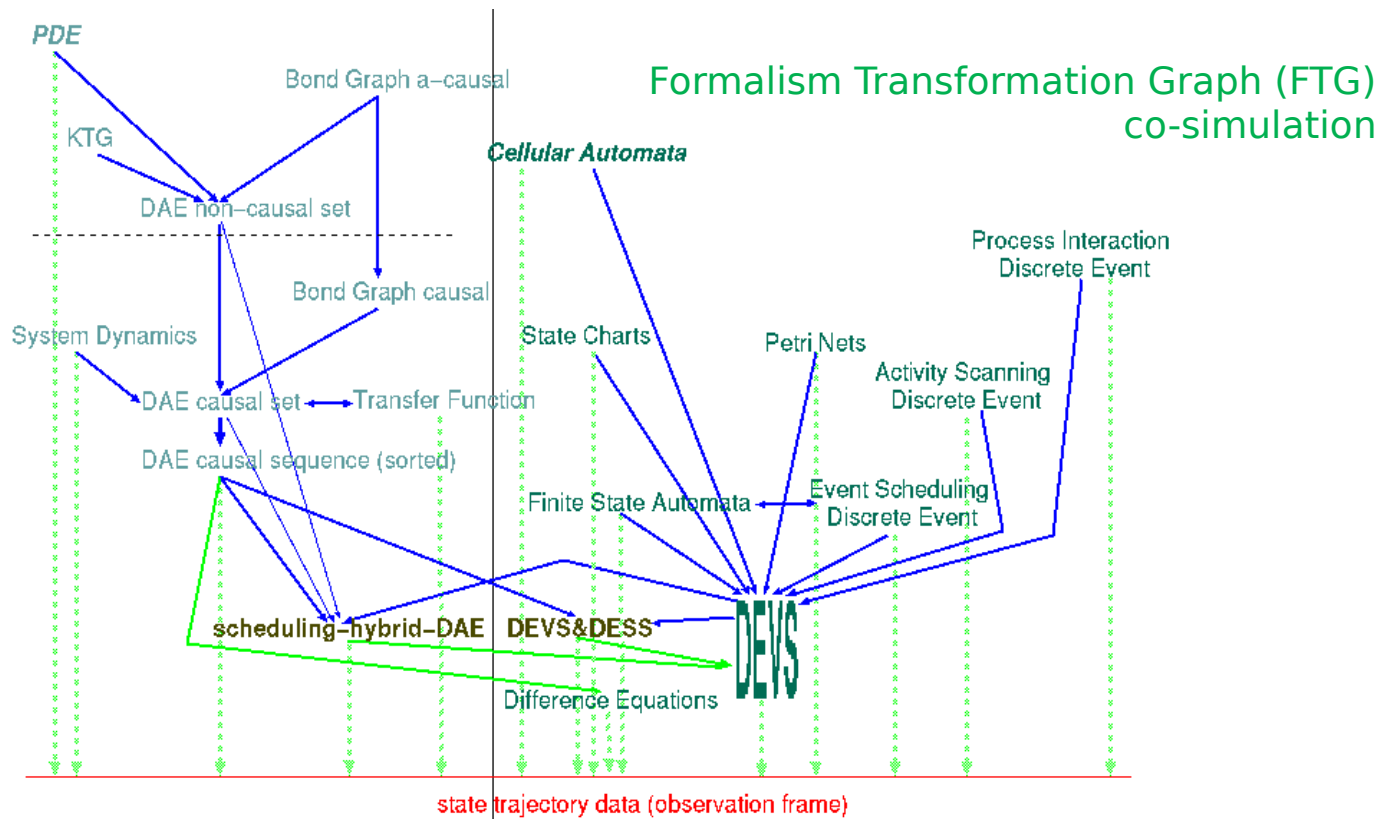
Formalism Transformation Graph (FTG)



Bran Selic: “fragmentation problem”







Cláudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, Hans Vangheluwe.
Co-simulation: State of the art. ArXiv 1702.00686. 2017.

Cornell University Library

arXiv.org > cs > arXiv:1702.00686

Search or Article ID inside arXiv All papers > Q Broaden y

(Help | Advanced search)

Computer Science > Systems and Control

Co-simulation: State of the art

Cláudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, Hans Vangheluwe

(Submitted on 1 Feb 2017)

It is essential to find new ways of enabling experts in different disciplines to collaborate more efficient in the development of ever more complex systems, under increasing market pressures. One possible solution for this challenge is to use a heterogeneous model-based approach where different teams can produce their conventional models and carry out their usual mono-disciplinary analysis, but in addition, the different models can be coupled for simulation (co-simulation), allowing the study of the global behavior of the system. Due to its potential, co-simulation is being studied in many different disciplines but with limited sharing of findings. Our aim with this work is to summarize, bridge, and enhance future research in this multidisciplinary area.

We provide an overview of co-simulation approaches, research challenges, and research opportunities, together with a detailed taxonomy with different aspects of the state of the art of co-simulation and classification for the past five years. The main research needs identified are: finding generic approaches for modular, stable and accurate coupling of simulation units; and expressing the adaptations required to ensure that the coupling is correct.

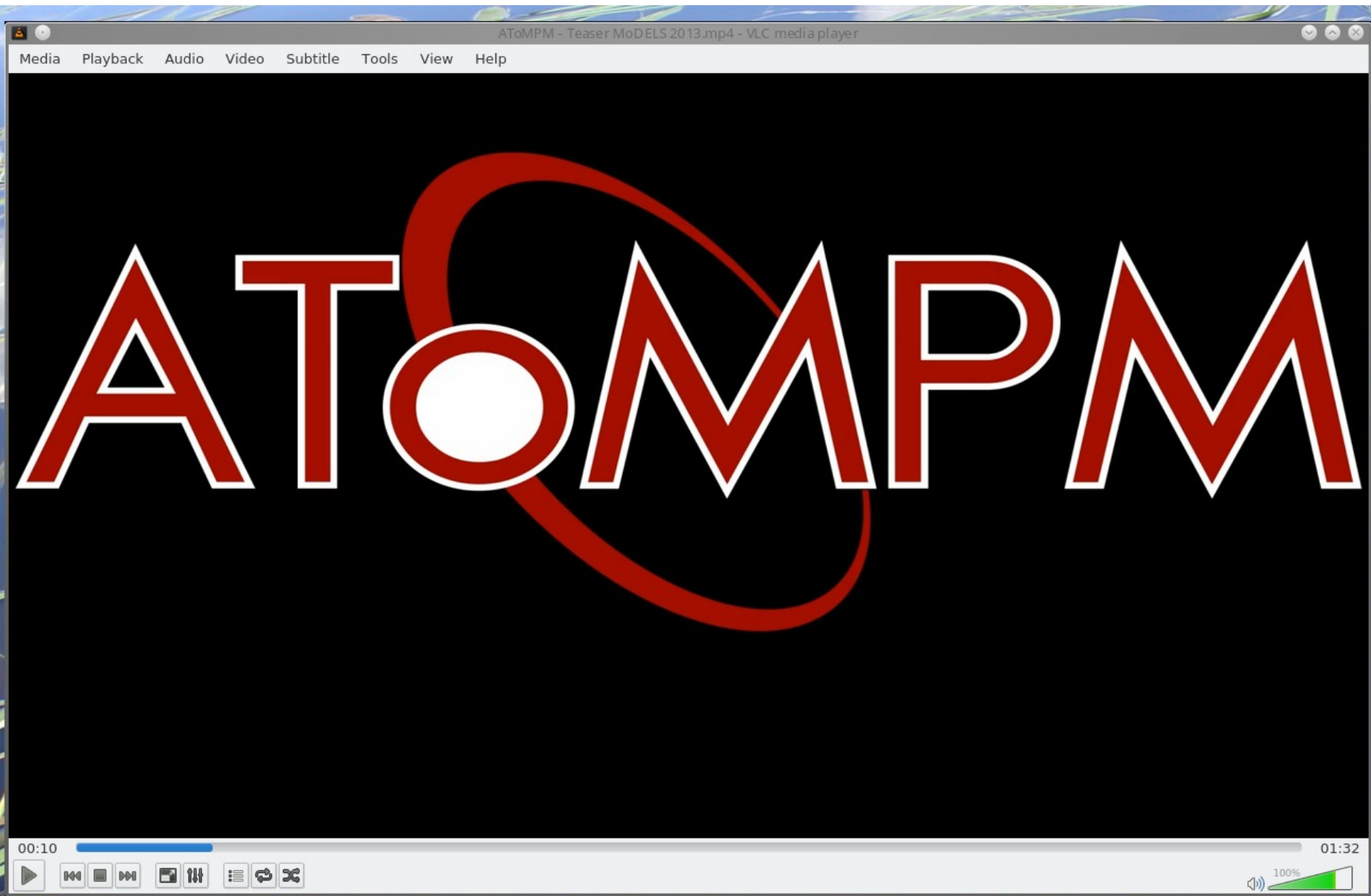
Comments: 157 pages, about 30 figures

Subjects: **Systems and Control (cs.SY)**

MSC classes: 65Y10

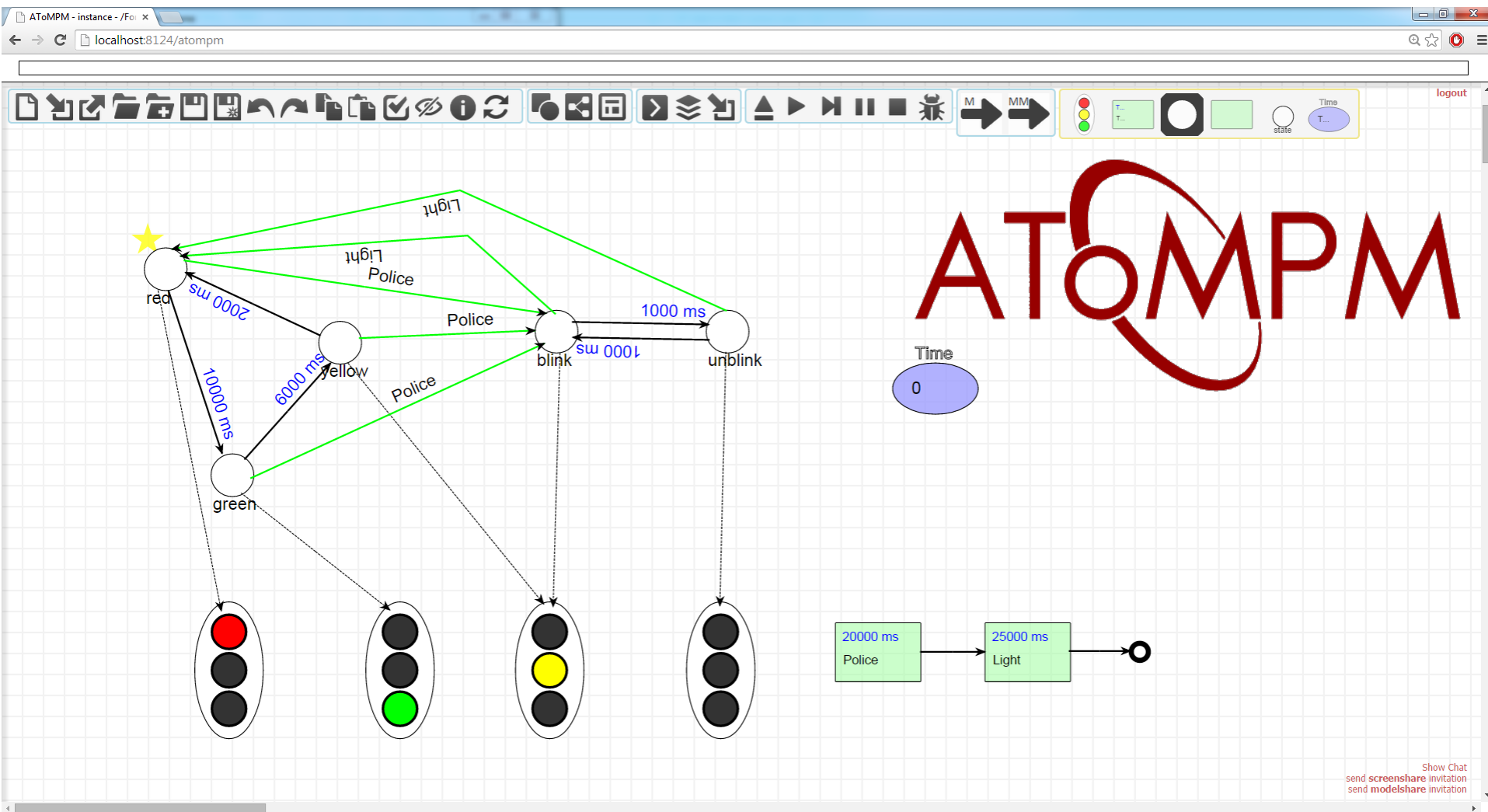
ACM classes: I.6.1; I.6.7

Cite as: **arXiv:1702.00686 [cs.SY]**
(or **arXiv:1702.00686v1 [cs.SY]** for this version)



<https://www.youtube.com/watch?v=RYtea2BiQ98>

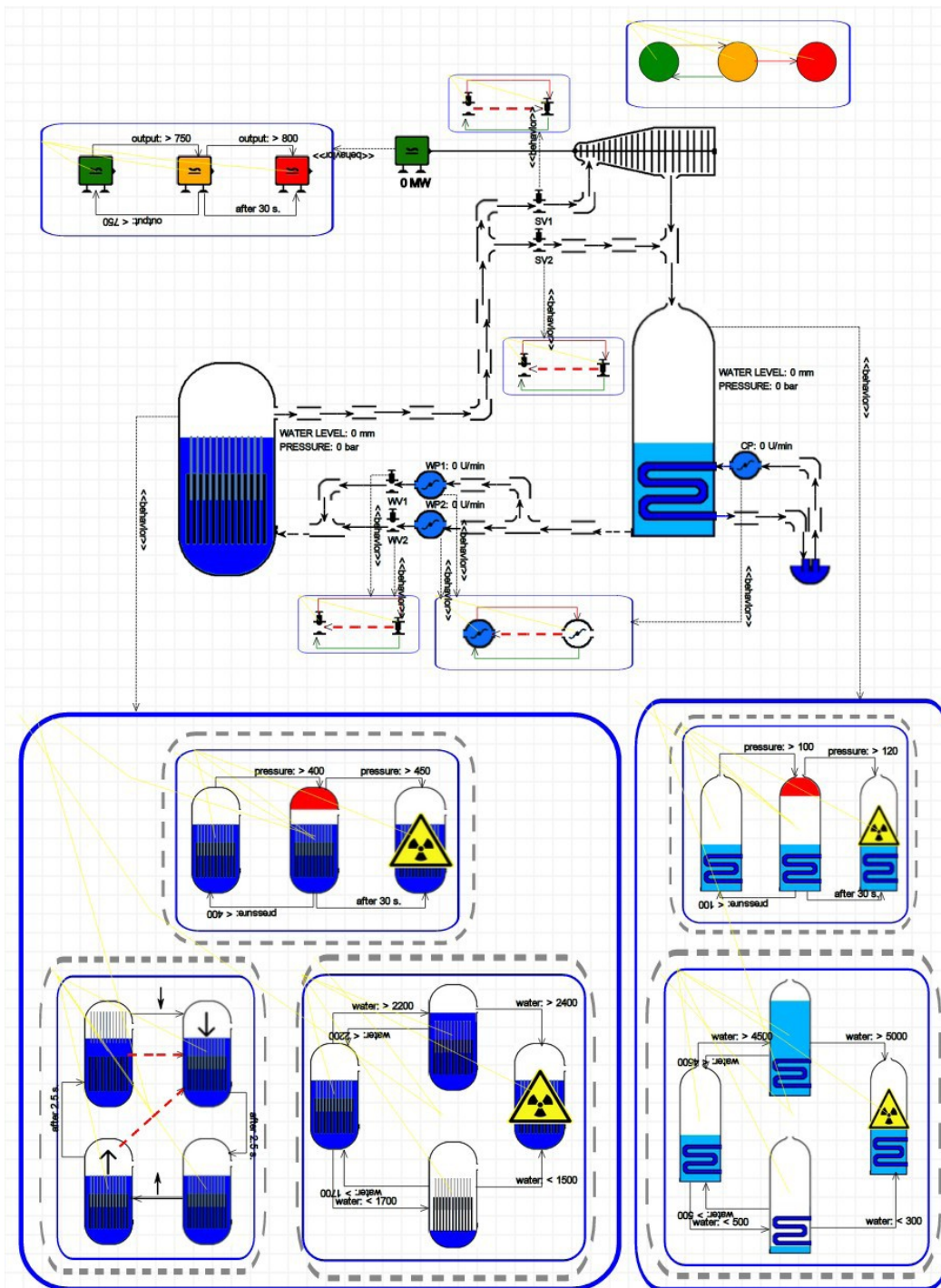
(domain-specific) “linguistic” Modelling Language Engineering

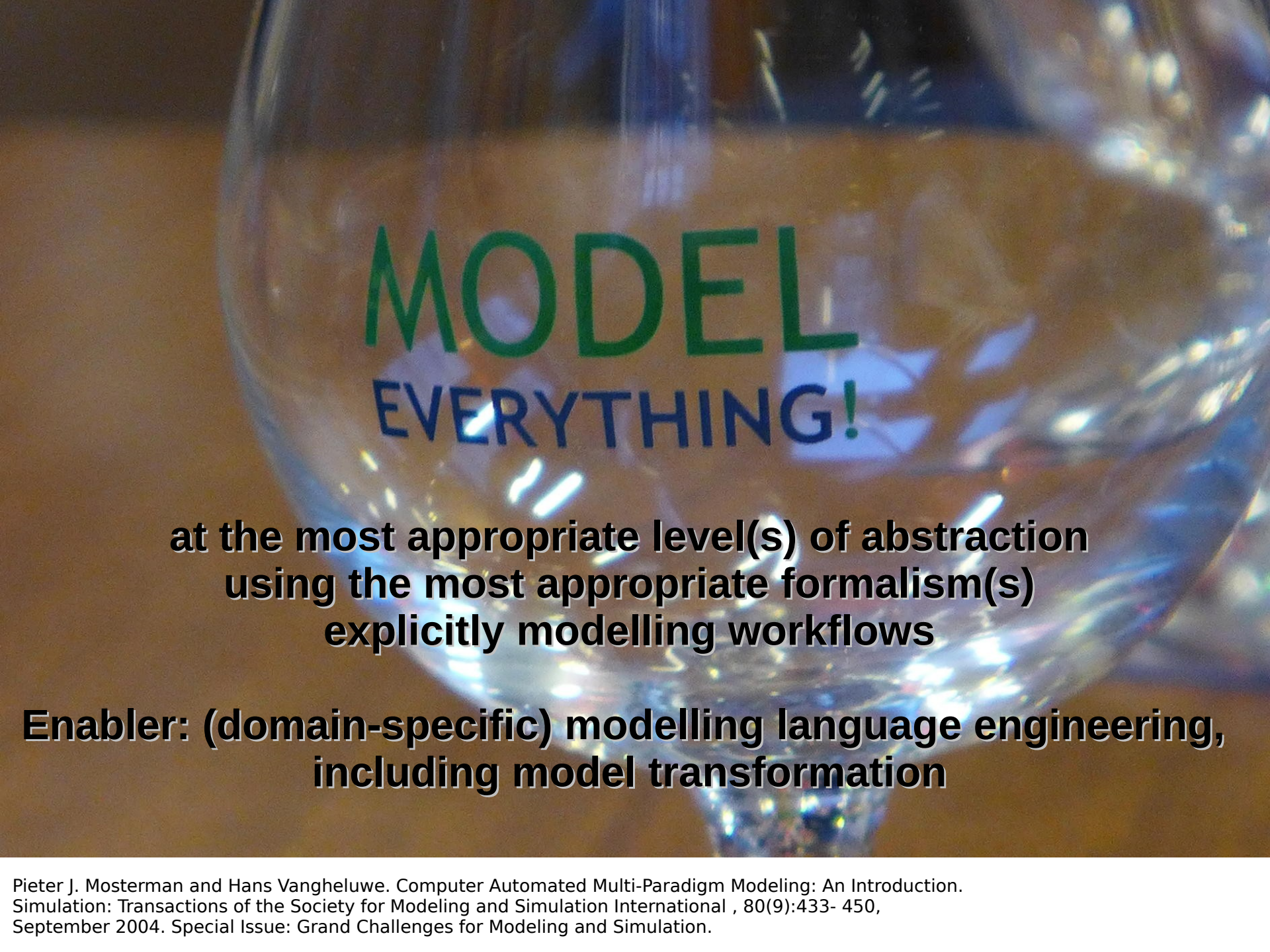


Raphael Mannadiar. A Multi-Paradigm Modelling Approach to the Foundations of Domain-Specific Modelling. PhD thesis, McGill Univ., 2012.

Eugene Syriani, Hans Vangheluwe, Raphael Mannadiar, Conner Hansen, Simon Van Mierlo, and Huseyin Ergin. AToMPM: A web-based modeling environment. In Proceedings of MODELS'13 Demonstration Session co-located with the 16th International Conference on Model Driven Engineering Languages and Systems (MODELS 2013), Miami, USA, pages 21- 25, 2013.

https://www.youtube.com/watch?feature=player_detailpage&v=Rytea2BiQ98





MODEL
EVERYTHING!

**at the most appropriate level(s) of abstraction
using the most appropriate formalism(s)
explicitly modelling workflows**

**Enabler: (domain-specific) modelling language engineering,
including model transformation**