

# (Domain-Specific) Modelling Language Engineering

**Hans Vangheluwe**

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



**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!

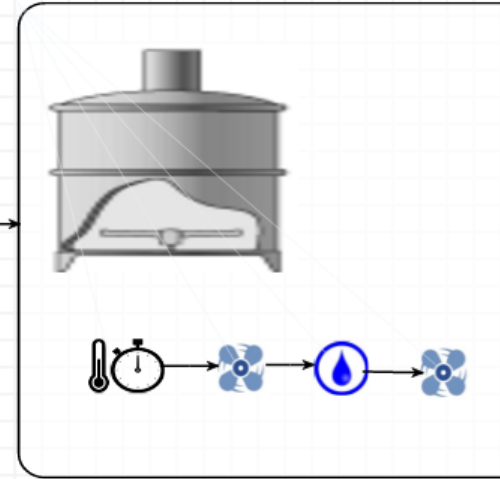
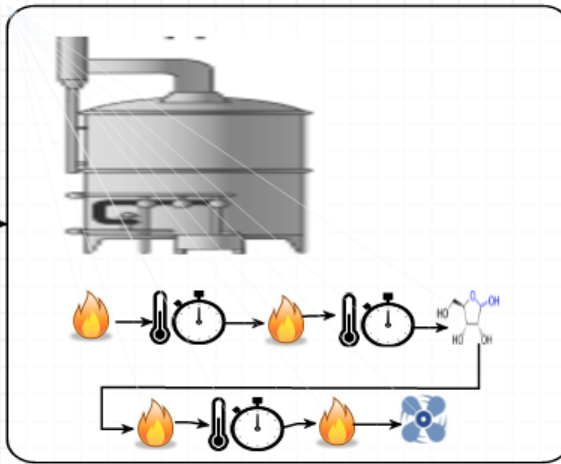
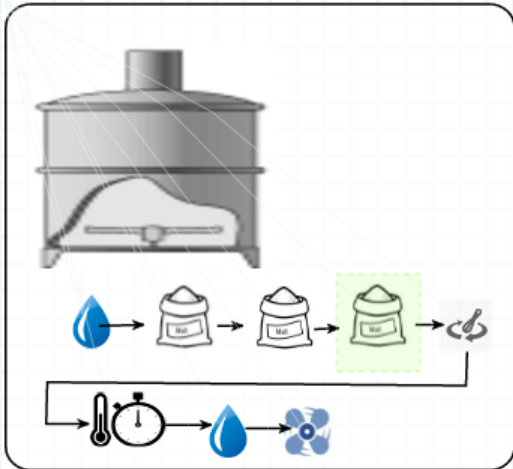


**DSM TP 2014**  
Theory and Practice

**5<sup>th</sup> International Summer School**  
on Domain Specific Modeling

**Antwerp, Belgium**  
25 - 29 August

**Thomas Kühne**

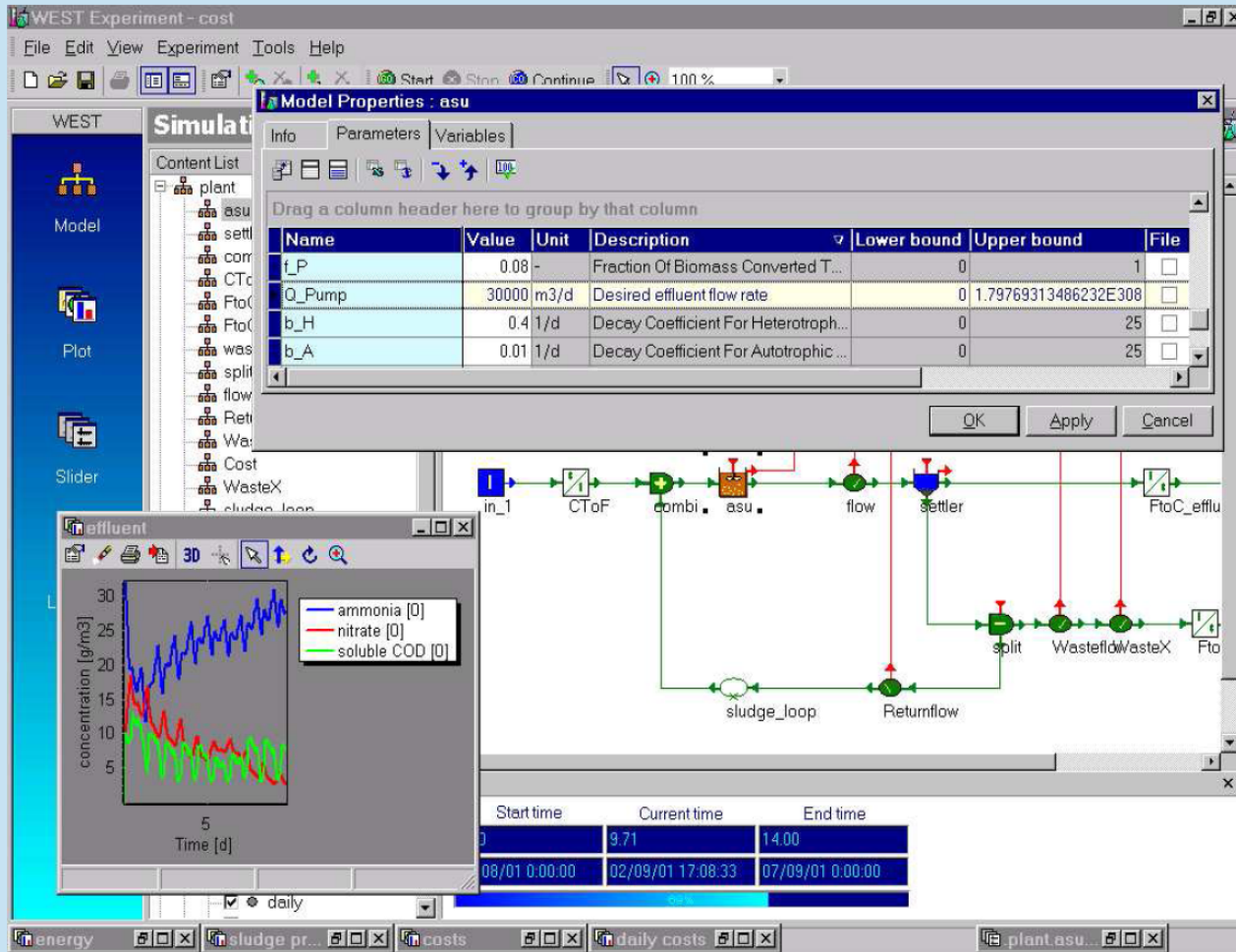


Show Chat  
send screenshare invitation  
send modelshare invitation

Joachim Denil



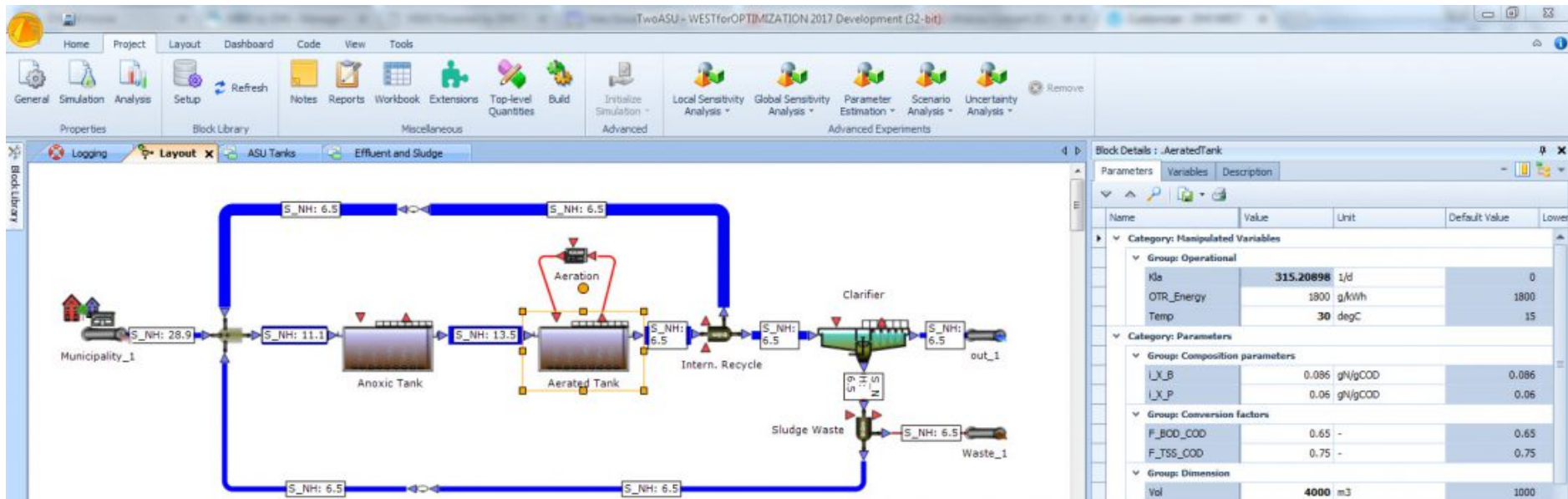
# DS(V)M Environment



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 Sfa. 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.



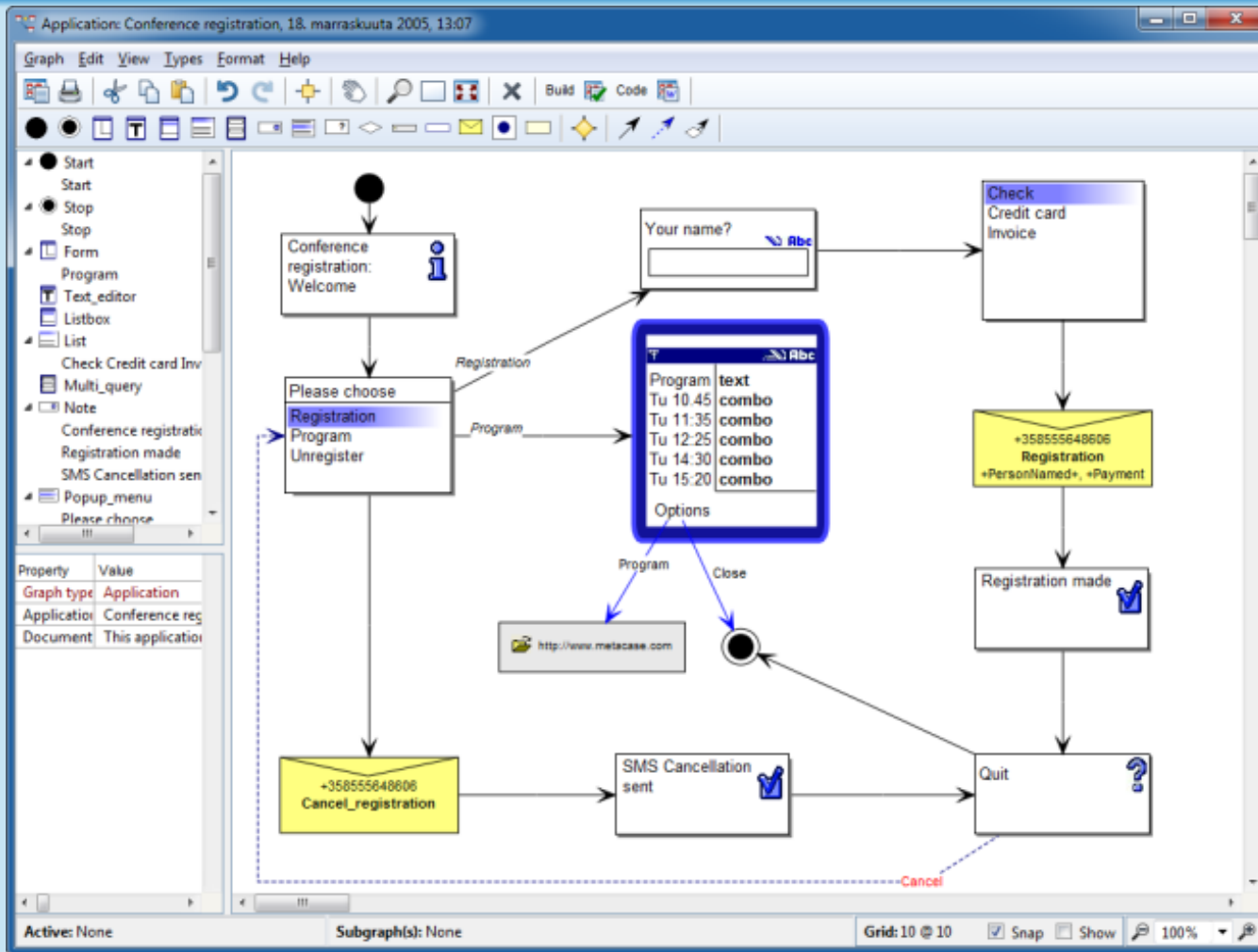
# DS(V)M Example in Software Domain

## smart phones, the application

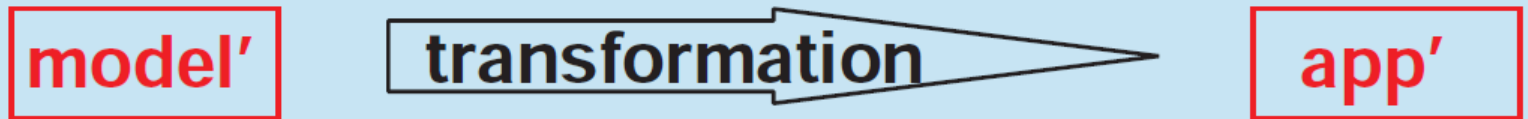
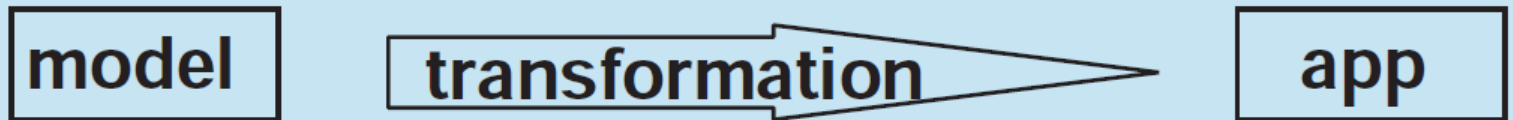


MetaEdit+ ([www.metacase.com](http://www.metacase.com))

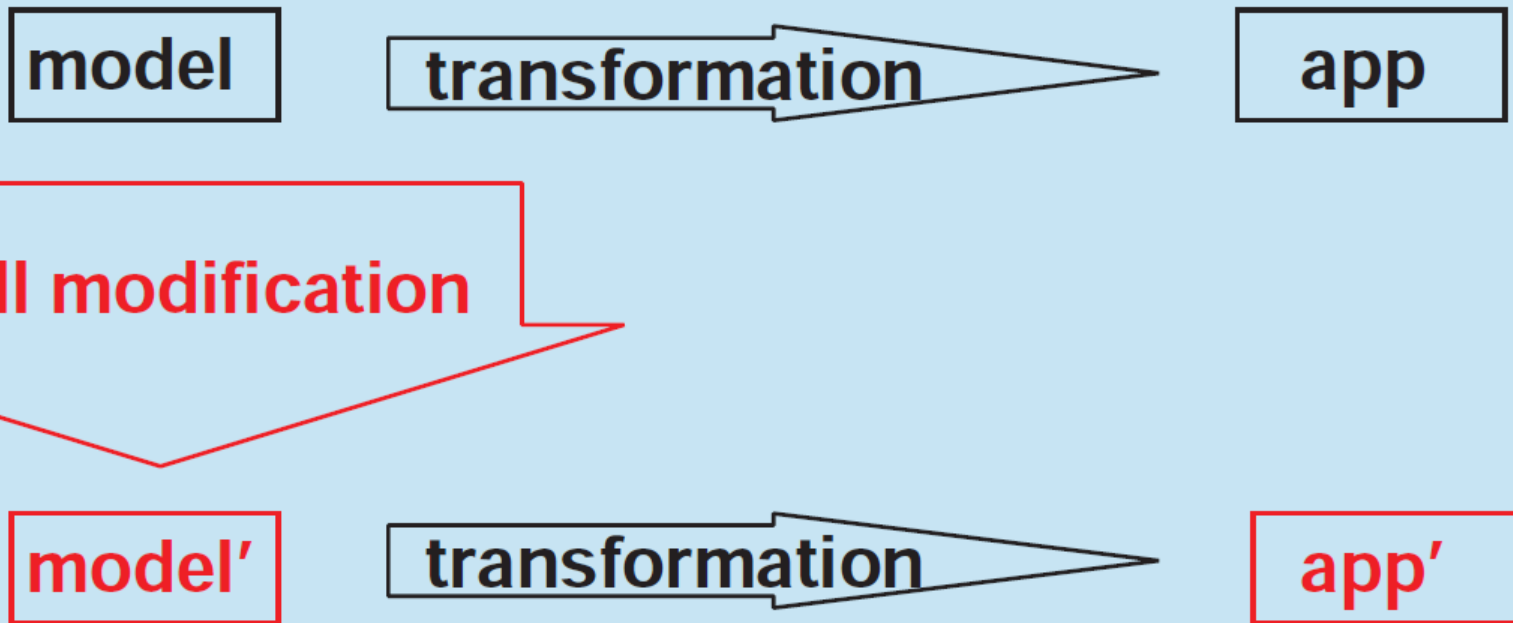
# DS(V)M Example: smart phones, the Domain-Specific model



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

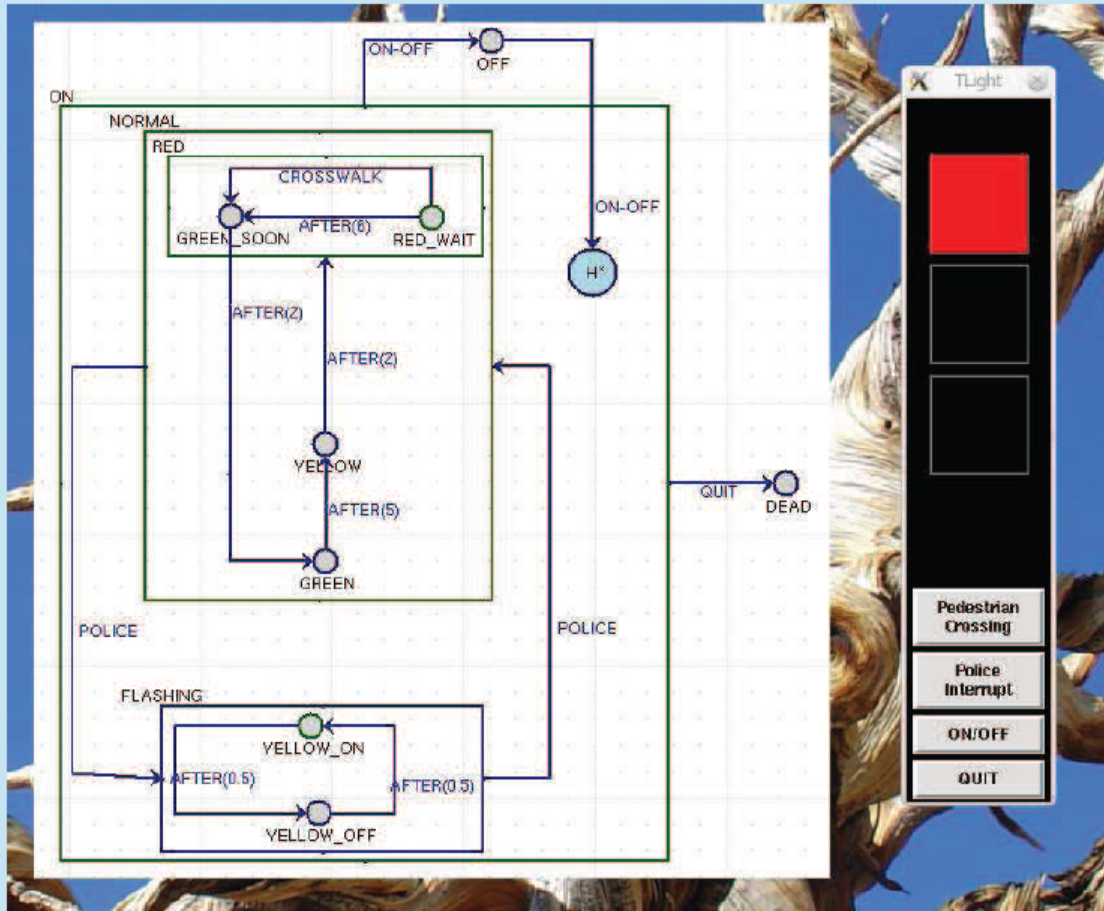


Model-Based Development:  
Modify the Model  
(e.g., based on feature model of product family)

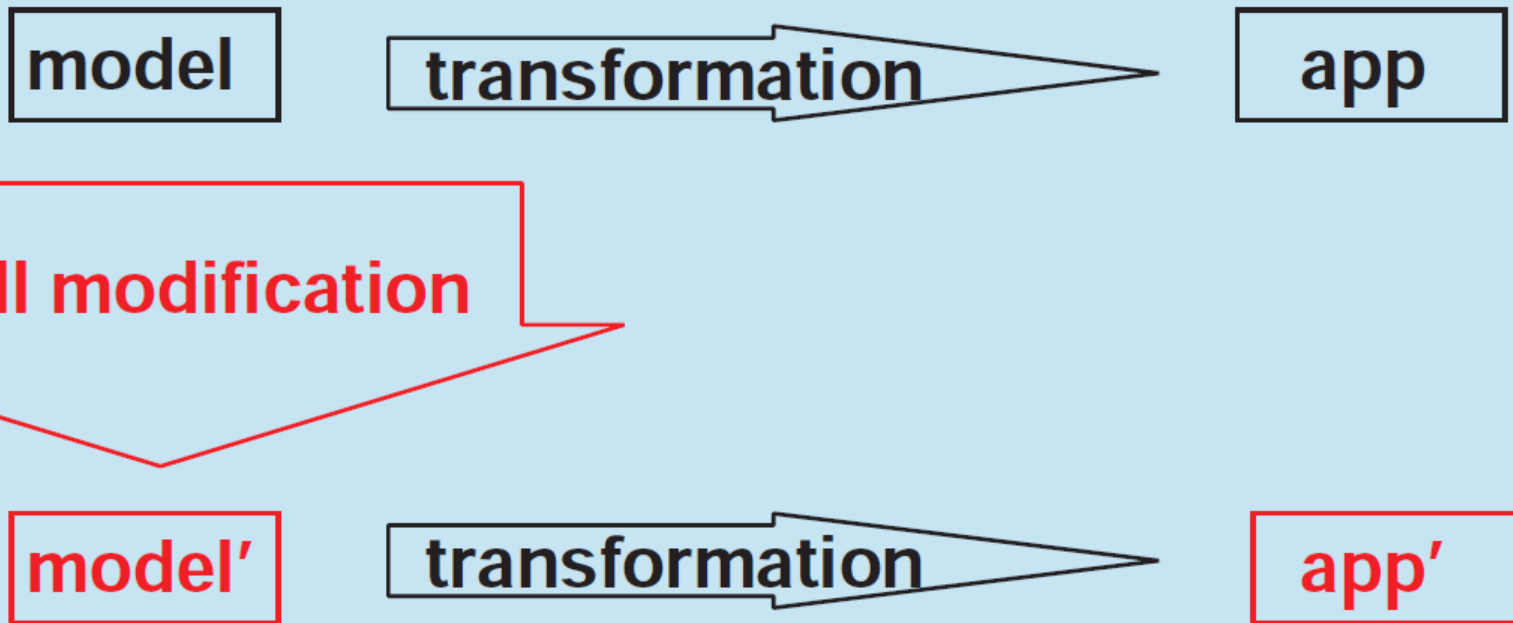


small **modification** in model may lead to large change in **app**  
~ choice of formalism (e.g., Statecharts)

# 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)

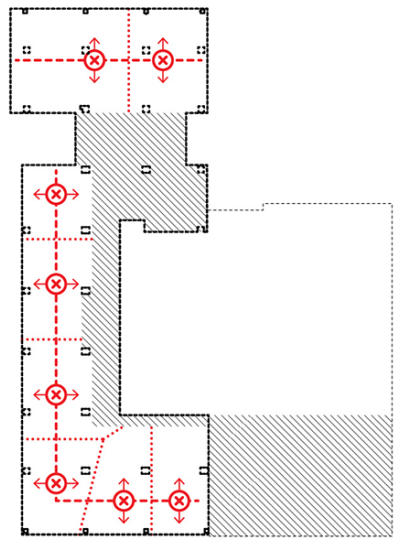




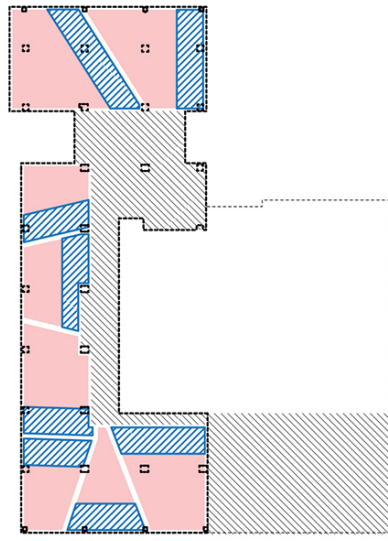
Nagy, Danil, Damon Lau, John Locke, Jim Stoddart, Lorenzo Villaggi, Ray Wang, Dale Zhao and David Benjamin. "Project Discover : An Application of Generative Design for Architectural Space Planning." (2017).



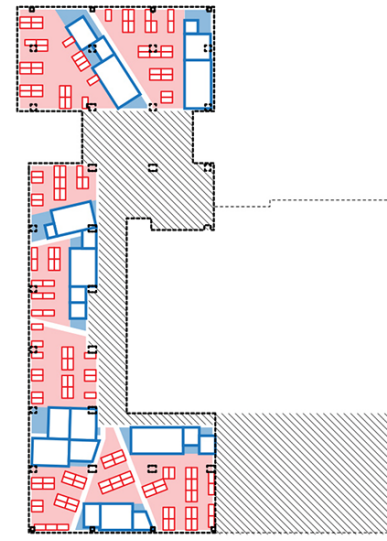
# candidate design



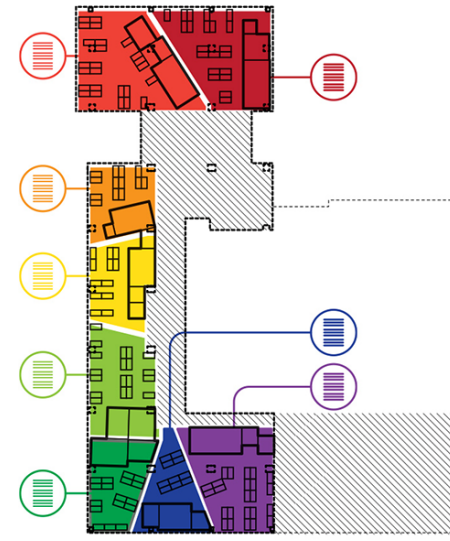
① A variable number of neighborhoods are seeded along spine, and given a parameterized range of motion.



② One edge from each neighborhood is selected to generate zone for amenity clusters.



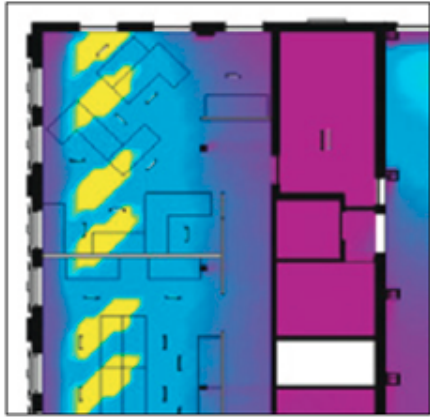
③ Automated "test fit" generates amenity rooms from space matrix and desk layout.



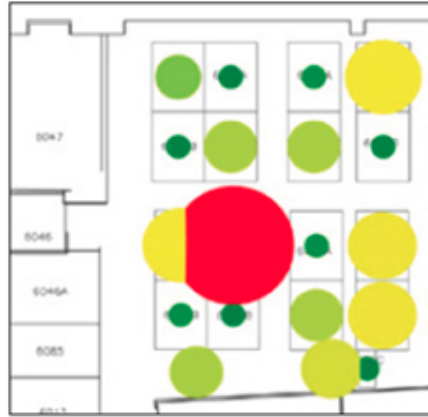
④ Teams are assigned by best-fit algorithm. Neighborhood amenities are assigned by team preferences.

Figure 1. Description of specification of geometric model

# “fitness” (multi-criteria)



1. Daylight



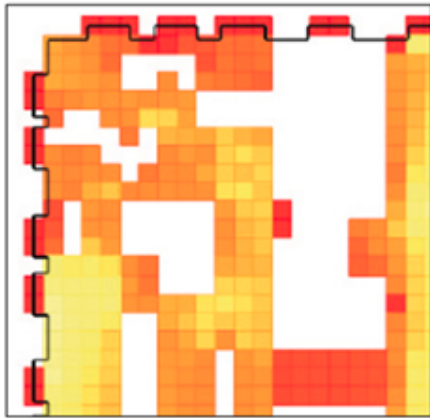
2. Low Visual Distraction



3. Views to Outside



4. Adjacency Preference



5. Circulation



6. Work Styles

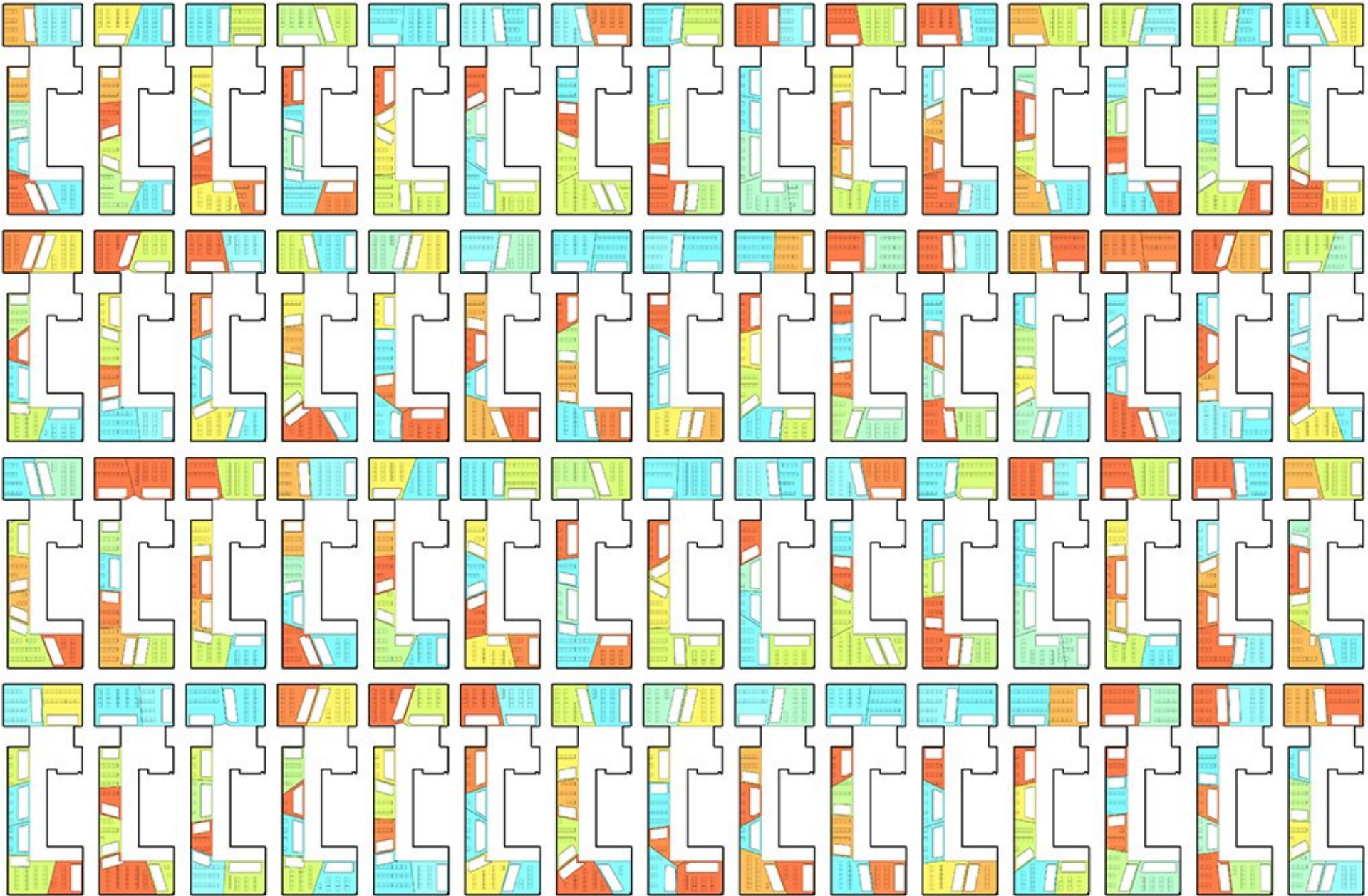


7. Low Acoustic Distraction

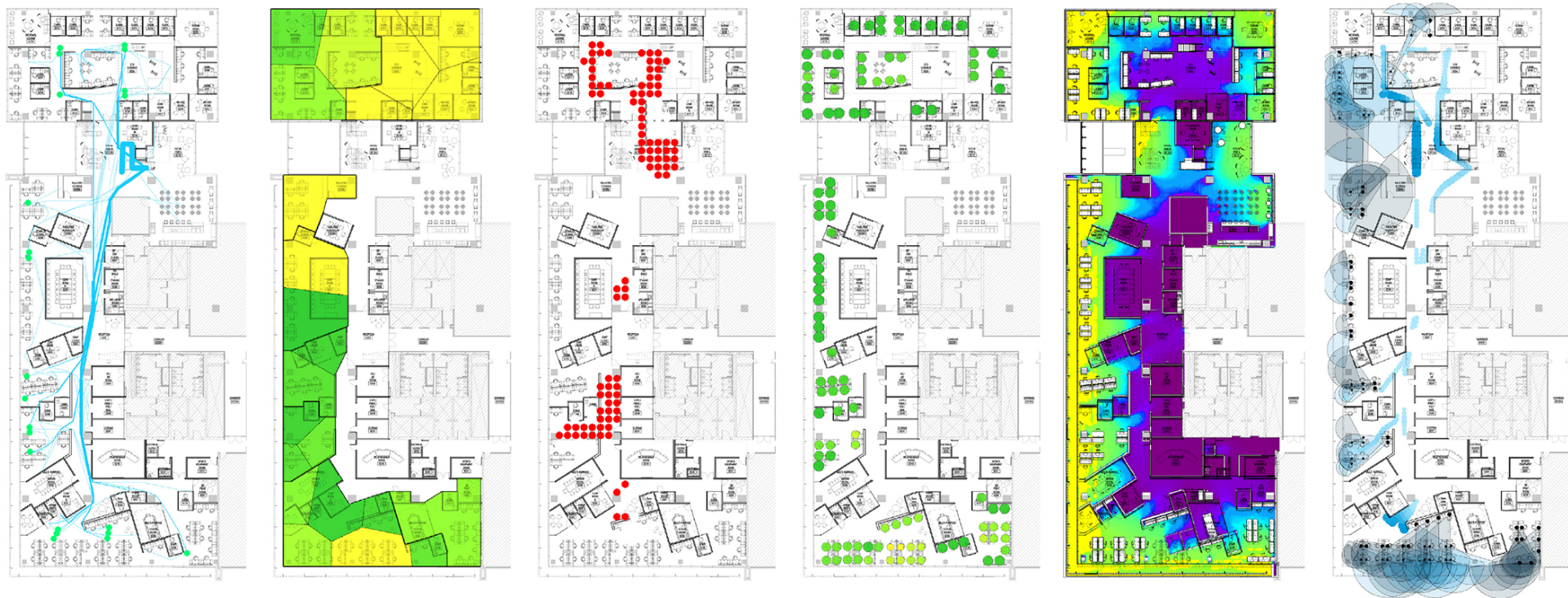


8. Low Density

# Generated designs (evolutionary – e.g., Genetic Algorithms)



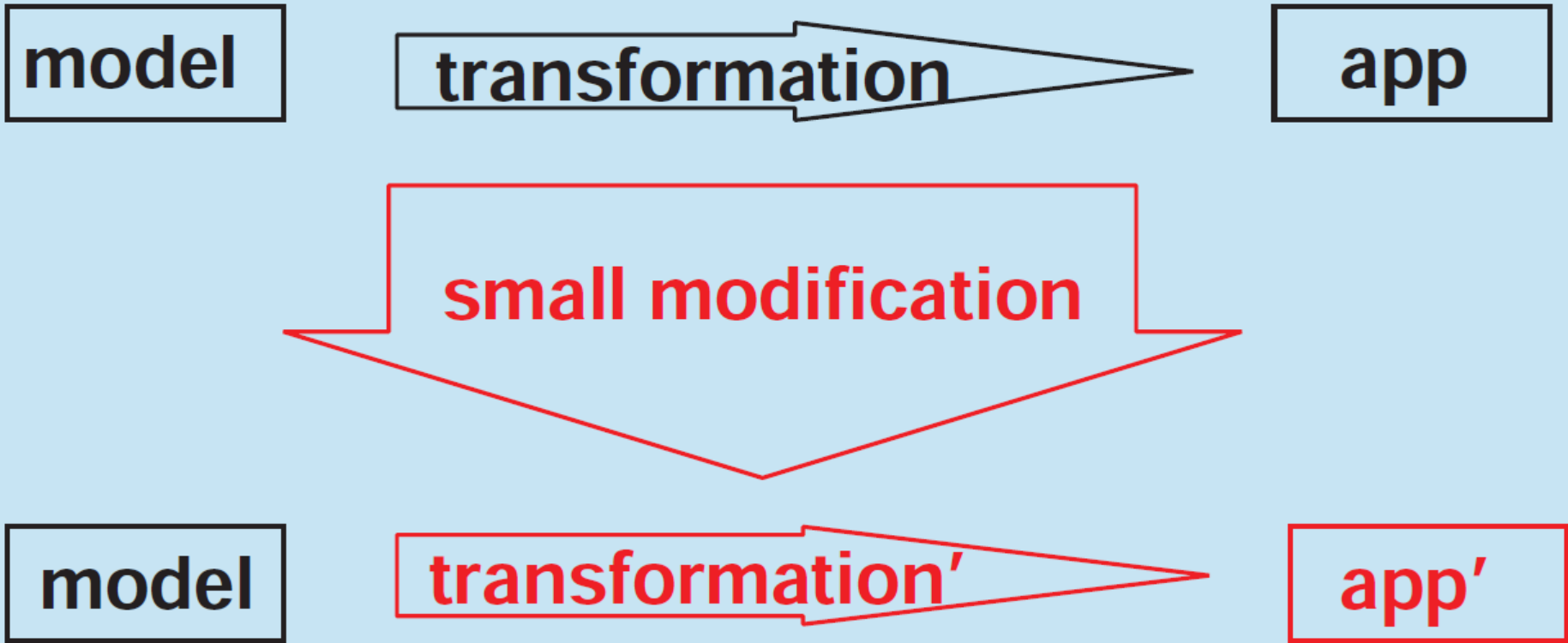
# “fitness” evaluation for each of the generated designs



**Figure 2.** Design metrics (from left to right: adjacency preference, work style preference, buzz, productivity, daylight, and views to outside)



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





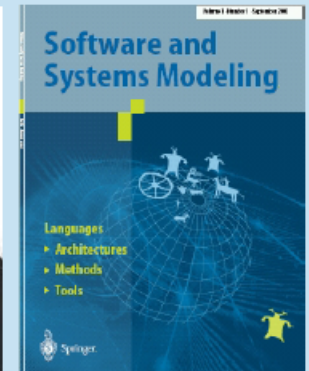
## Building DS(V)M Tools Effectively ...

- **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 System Modeling 5(4): 369-385 (2006)



Herbert Stachowiak

*Allgemeine  
Modelltheorie*

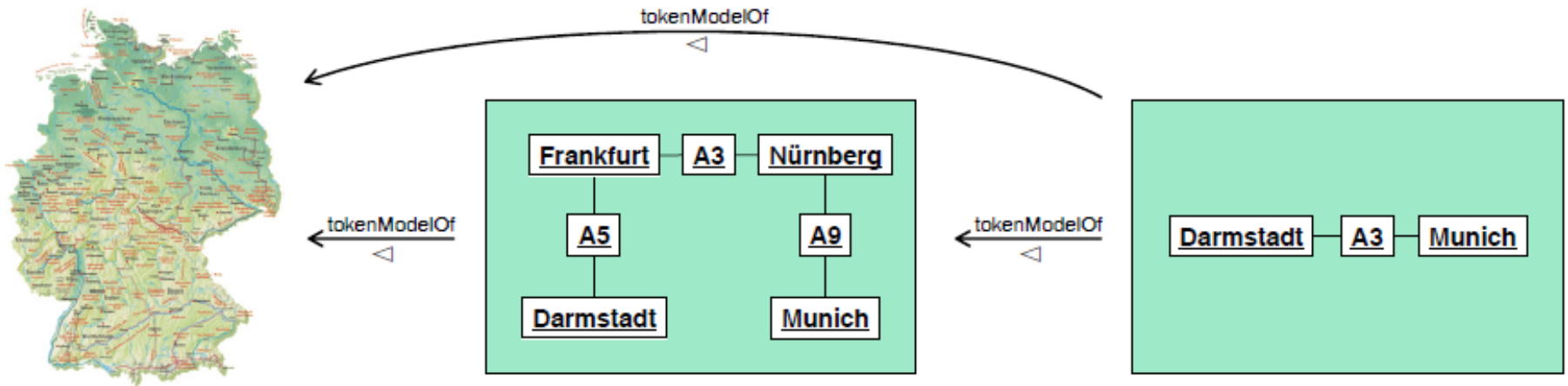
Springer-Verlag  
Wien New York



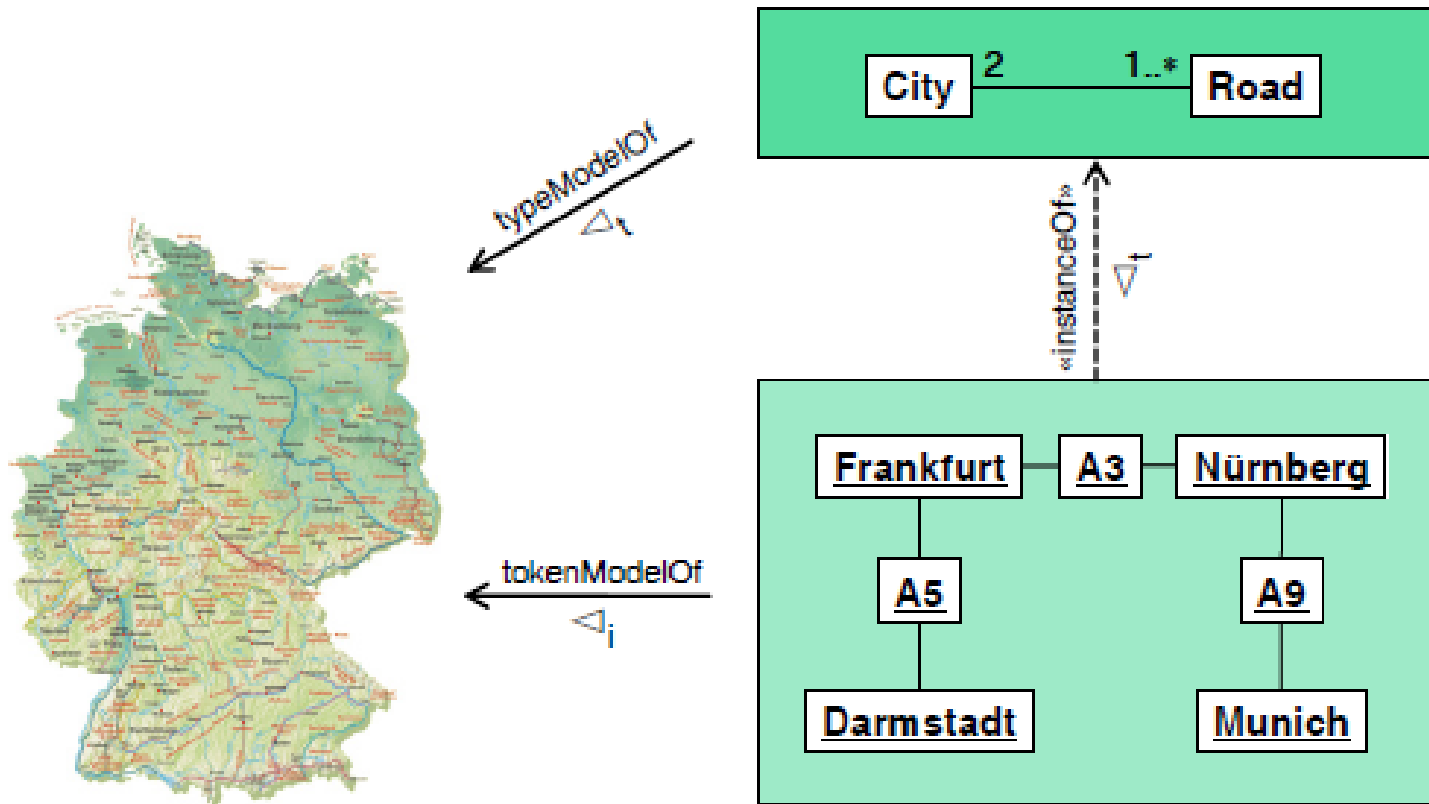
## Model Features

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

# Token Models

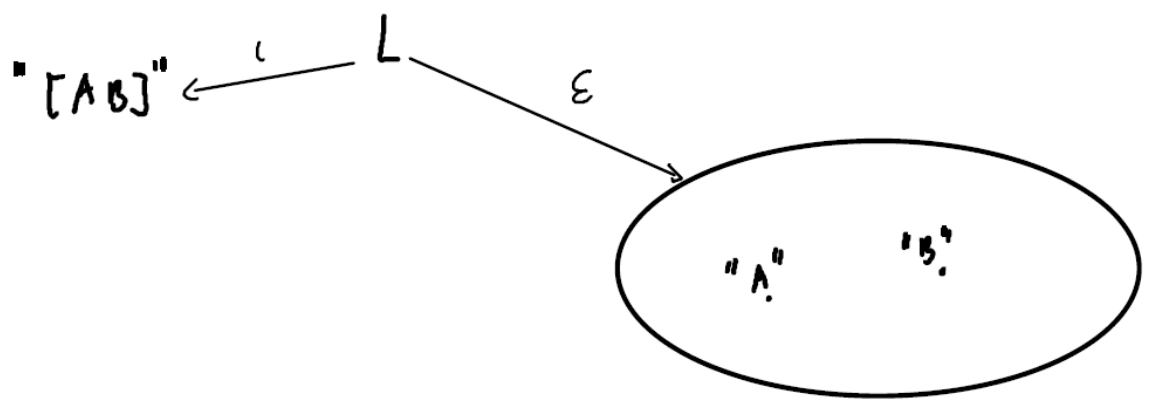
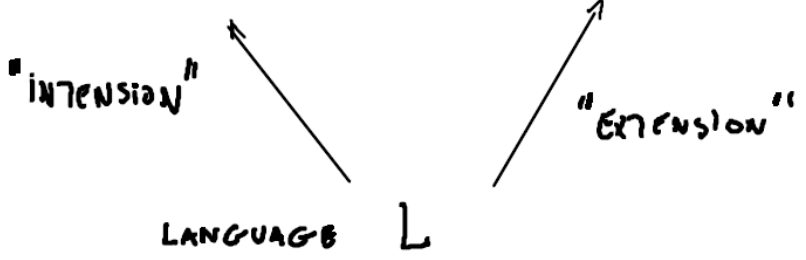


# Rôles a Model may Play



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

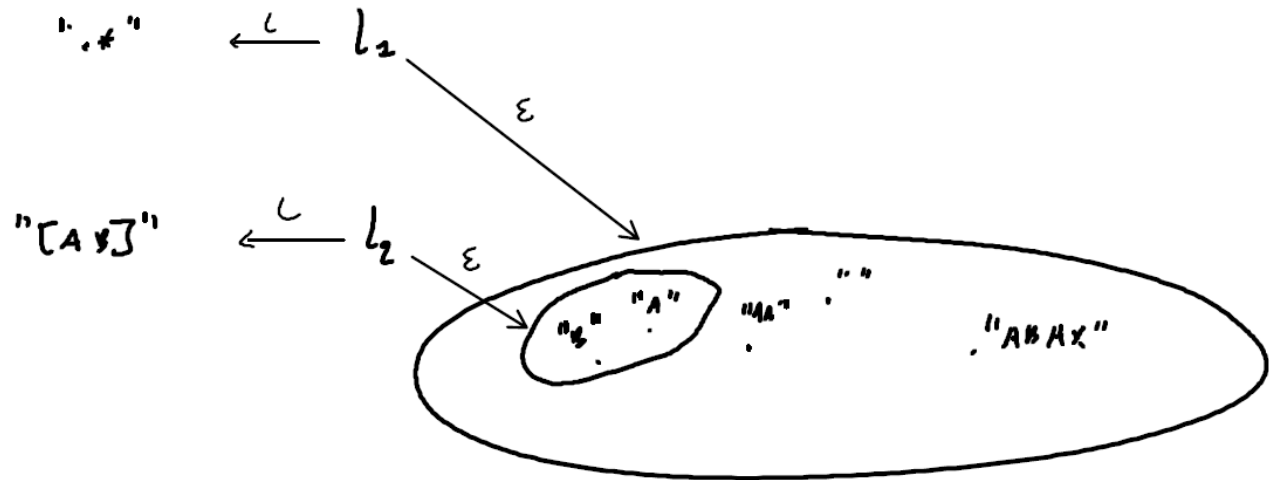
$$\llbracket "[AB]" \rrbracket_{RE} = \{ "A", "B" \}$$



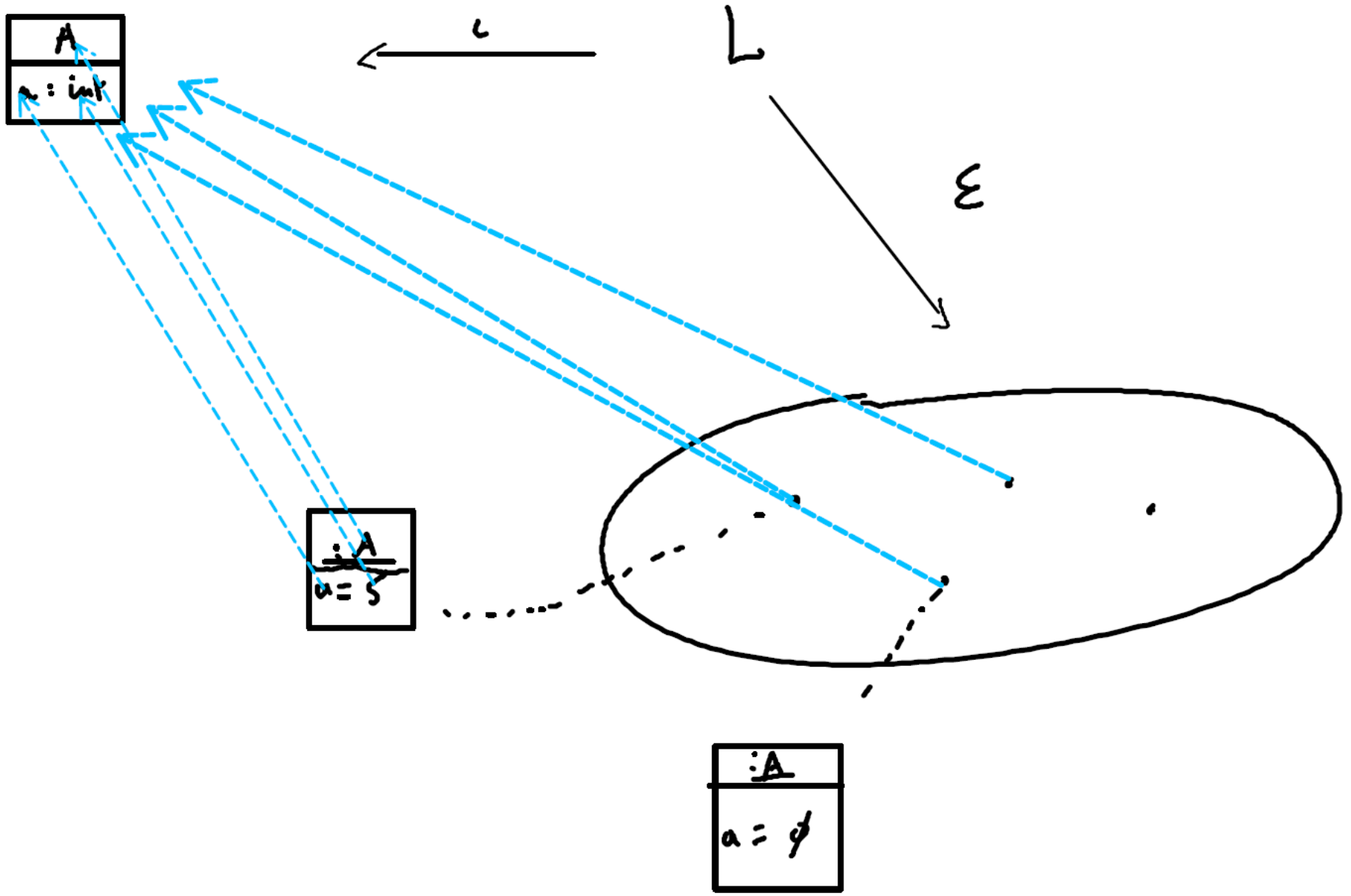
•  $\epsilon$  ANY CHAR FROM ALPHABET

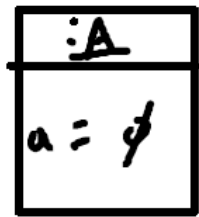
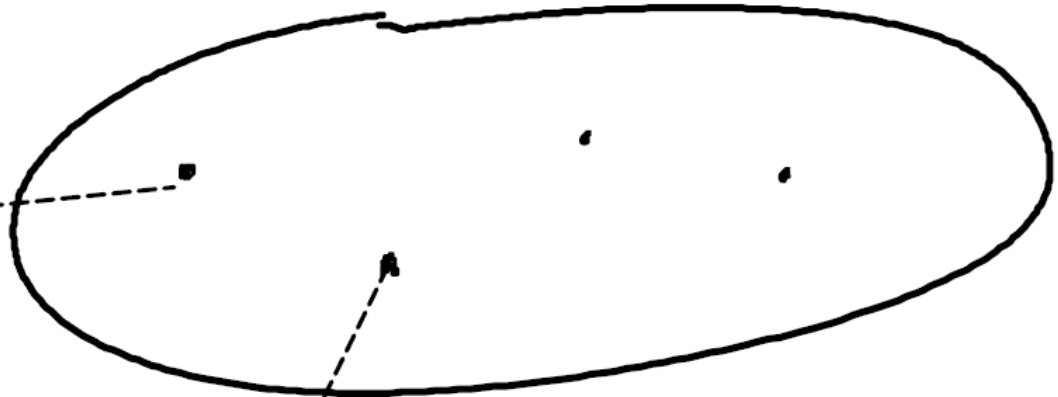
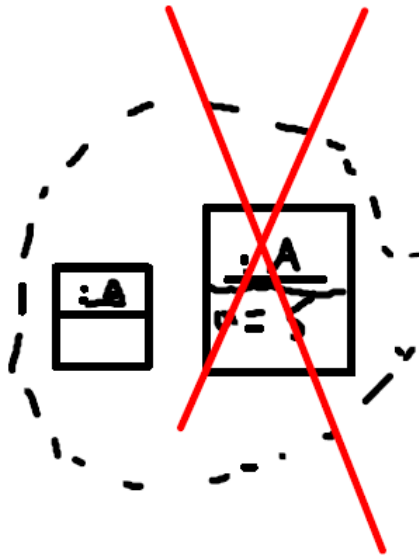
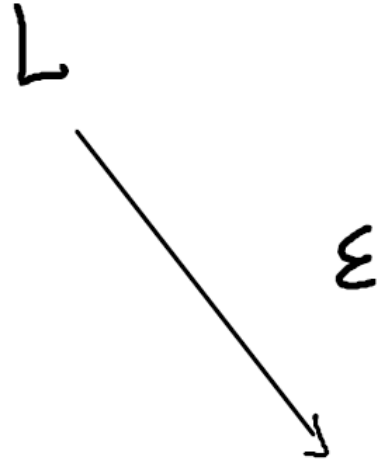
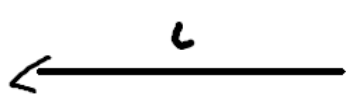
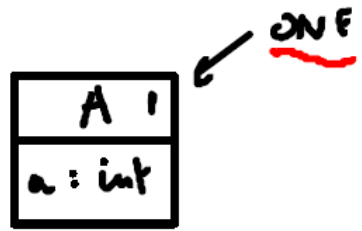
$$L_1 = ".*" \quad \mathcal{E}(L_1) = \{ "", "A", "AA", \dots \}$$

$$L_2 = "[AB]" \quad \mathcal{E}(L_2) = \{ "A", "B" \}$$

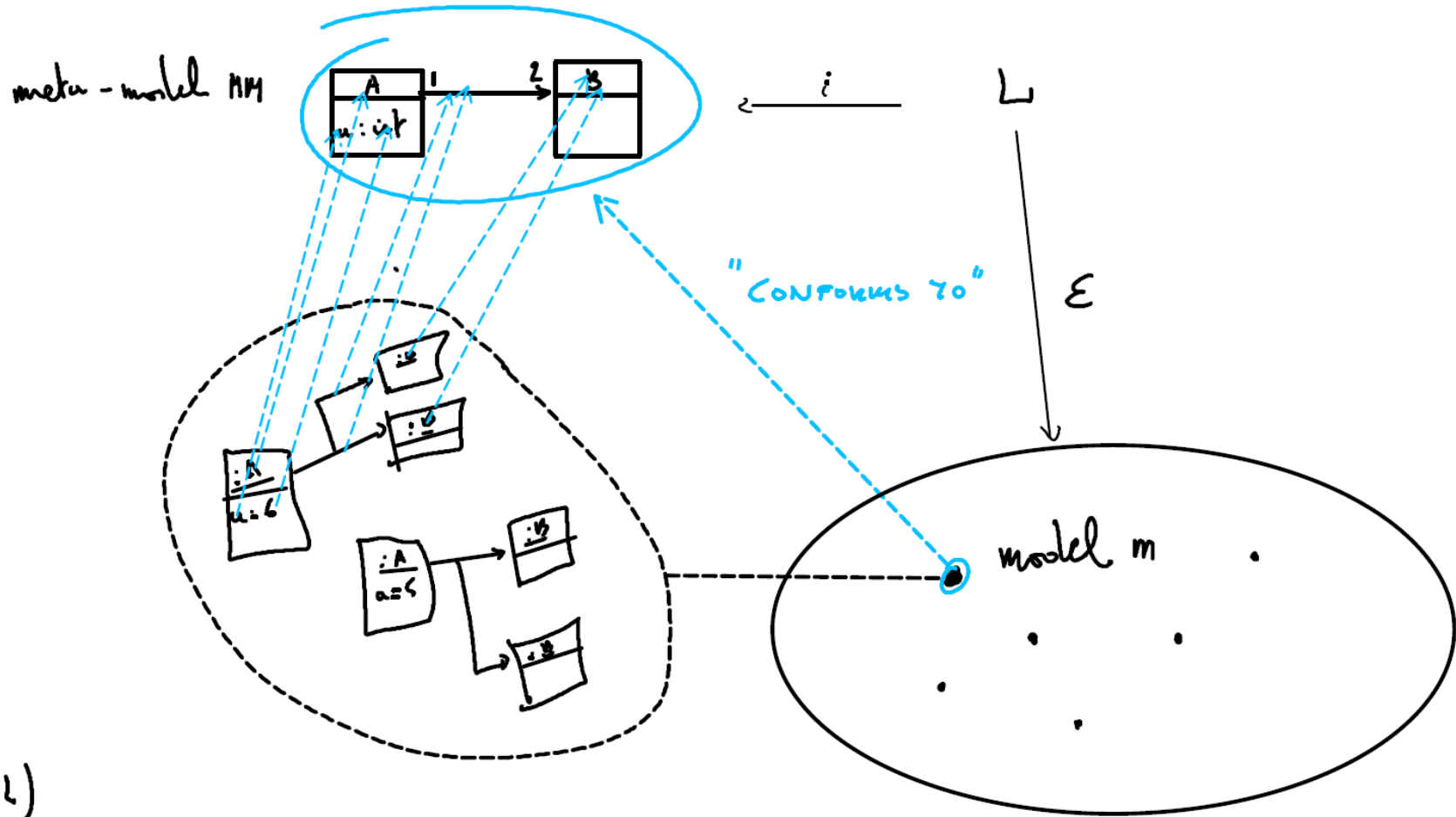


$$L_1 \cup L_2 \Leftrightarrow \mathcal{E}(L_1) \cup \mathcal{E}(L_2)$$









$$MM = i(L)$$

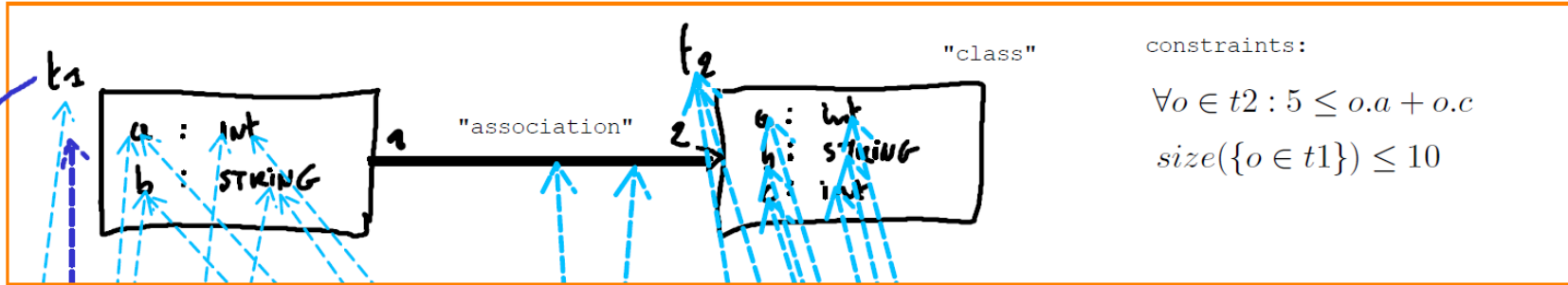
$$M \text{ CONFORMS TO } MM \equiv m \in \varepsilon(L)$$

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

$$\mu(MM) = L$$

$$[MM]_{\text{CONFORM}} = \varepsilon(L)$$

meta-model (intensional specification of language L)

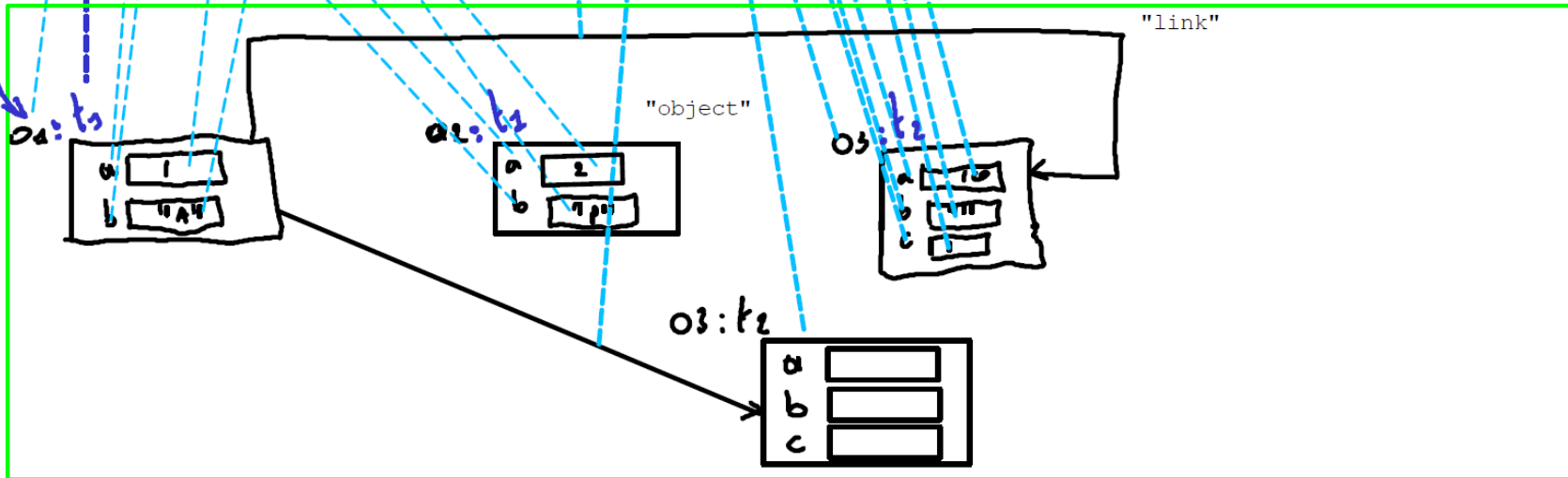


constraints:

$$\forall o \in t2 : 5 \leq o.a + o.c$$

$$size(\{o \in t1\}) \leq 10$$

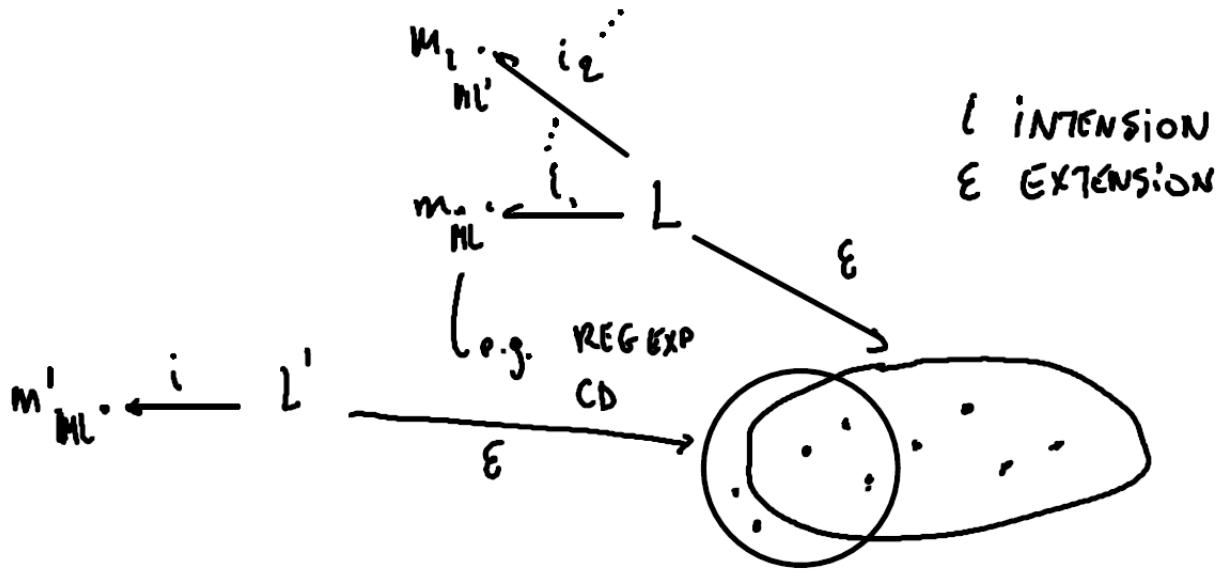
instantiate



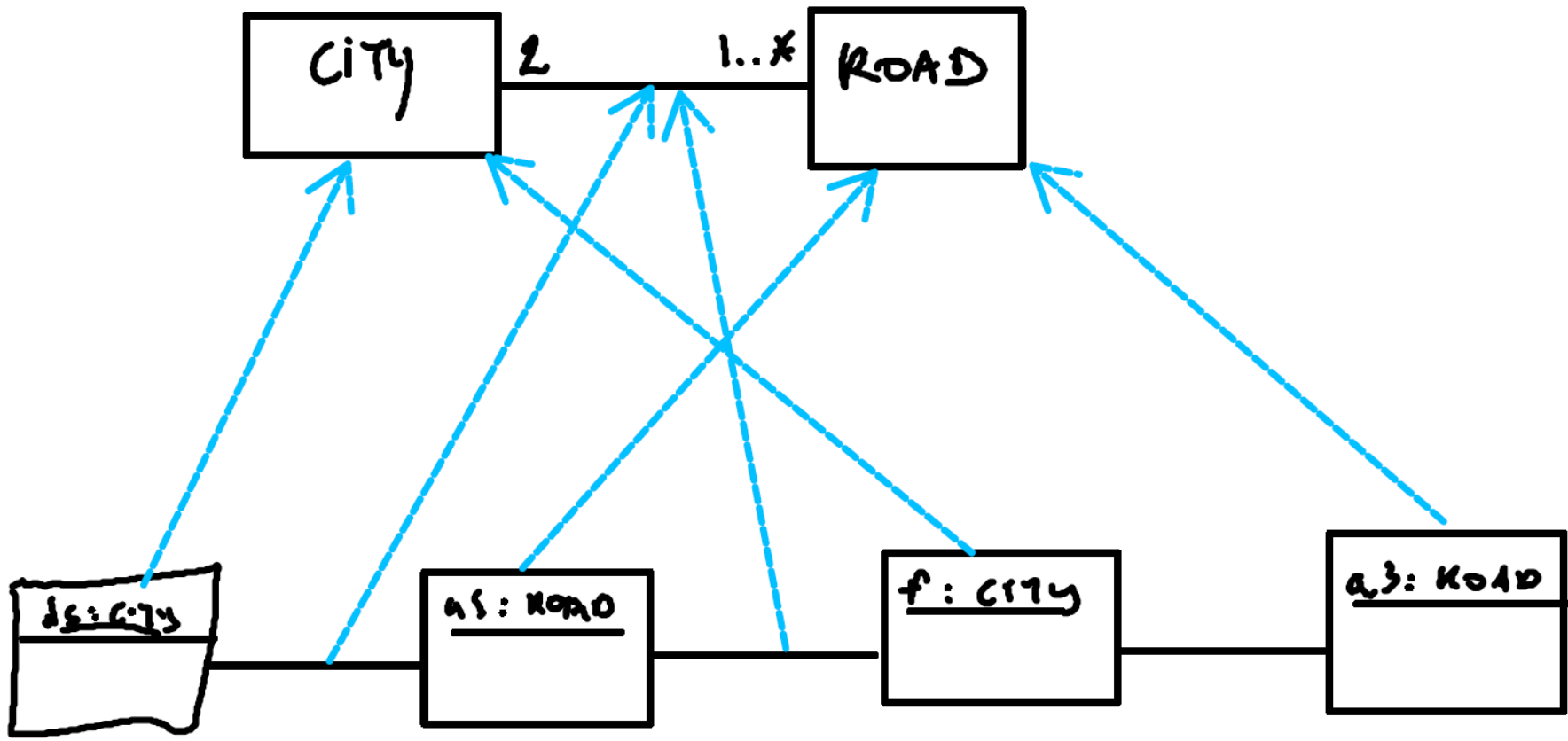
model in language L

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

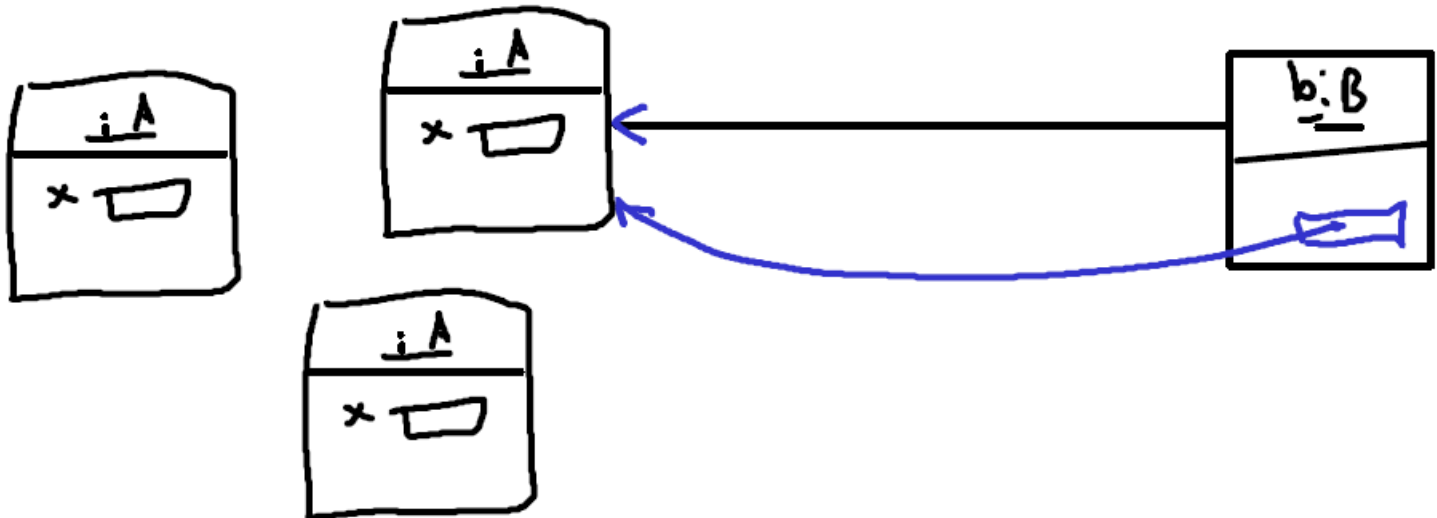
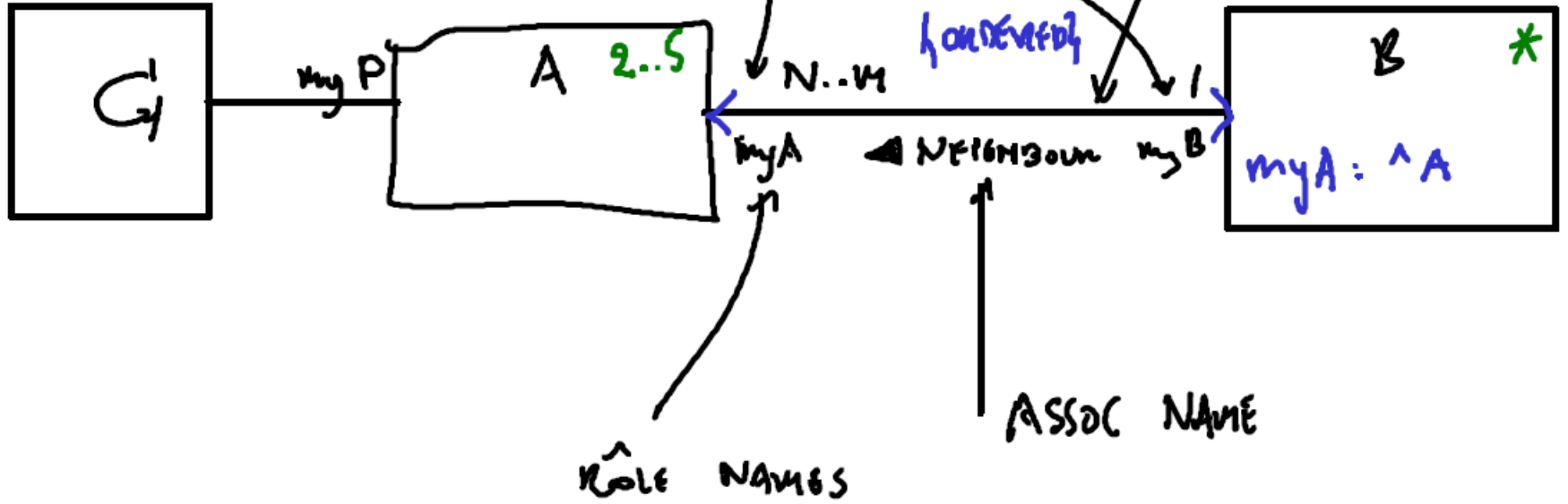
a single language may have multiple intensional descriptions



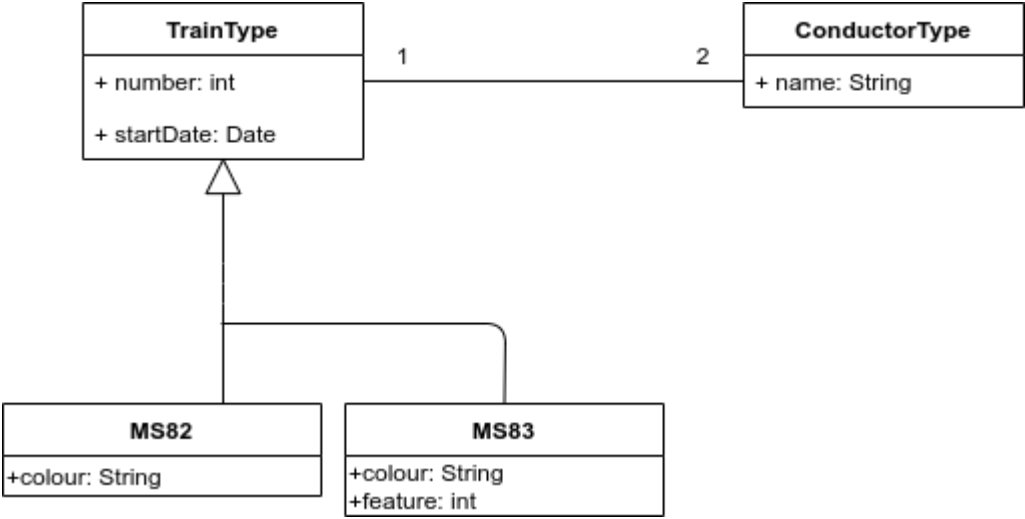
languages may (partially) overlap



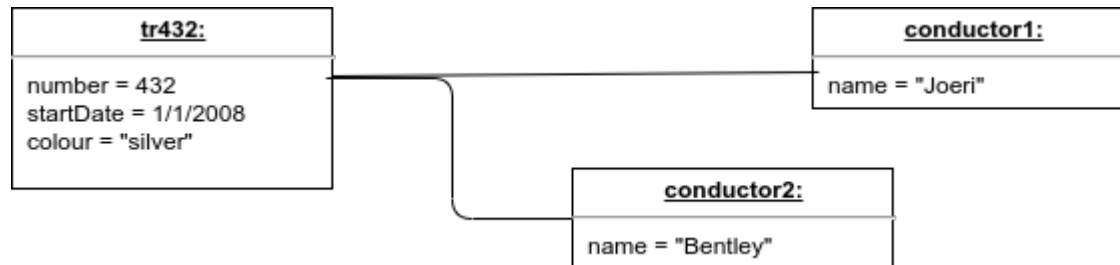
# Class Diagrams



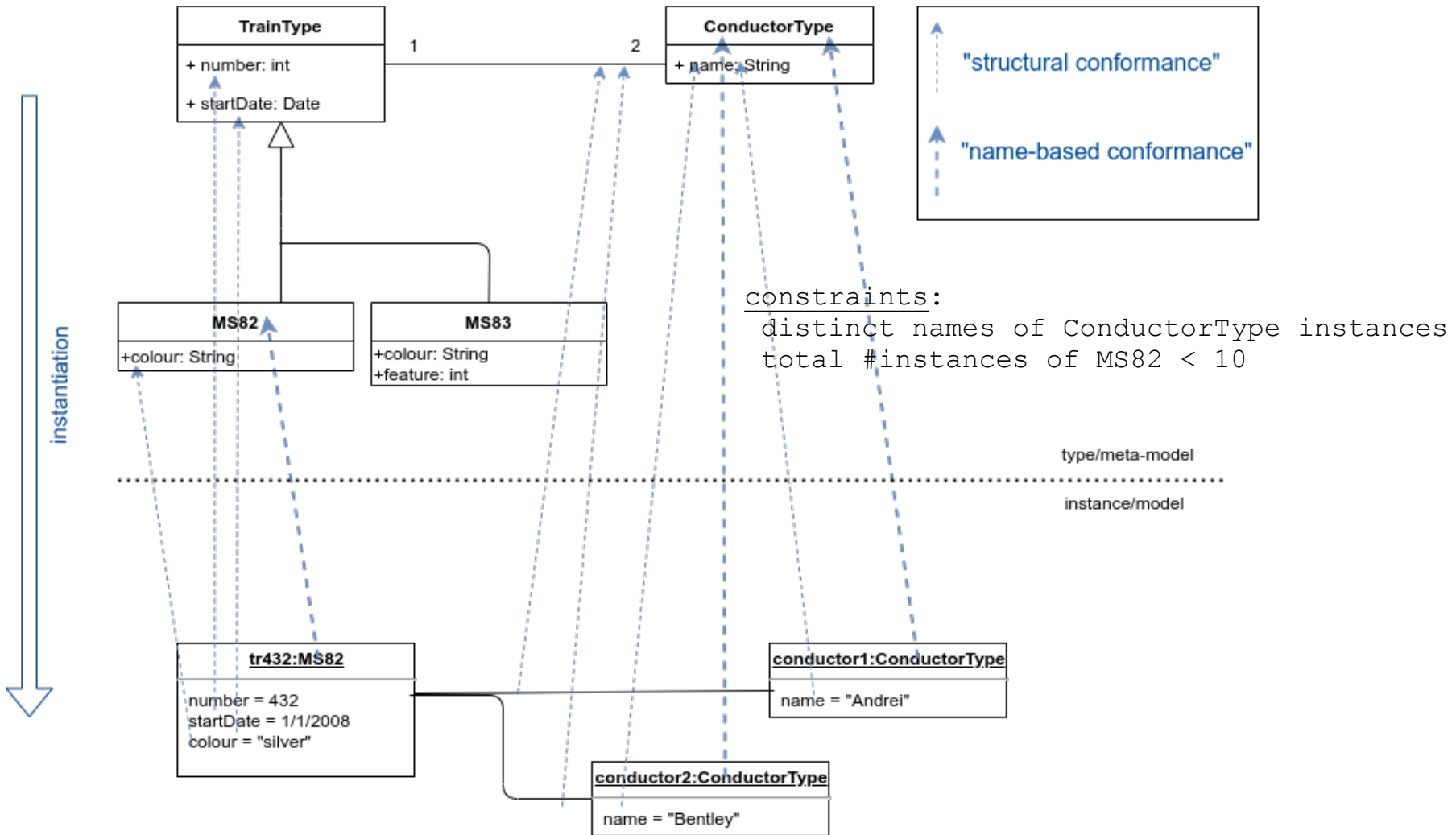
# Class Diagram



# Object Diagram

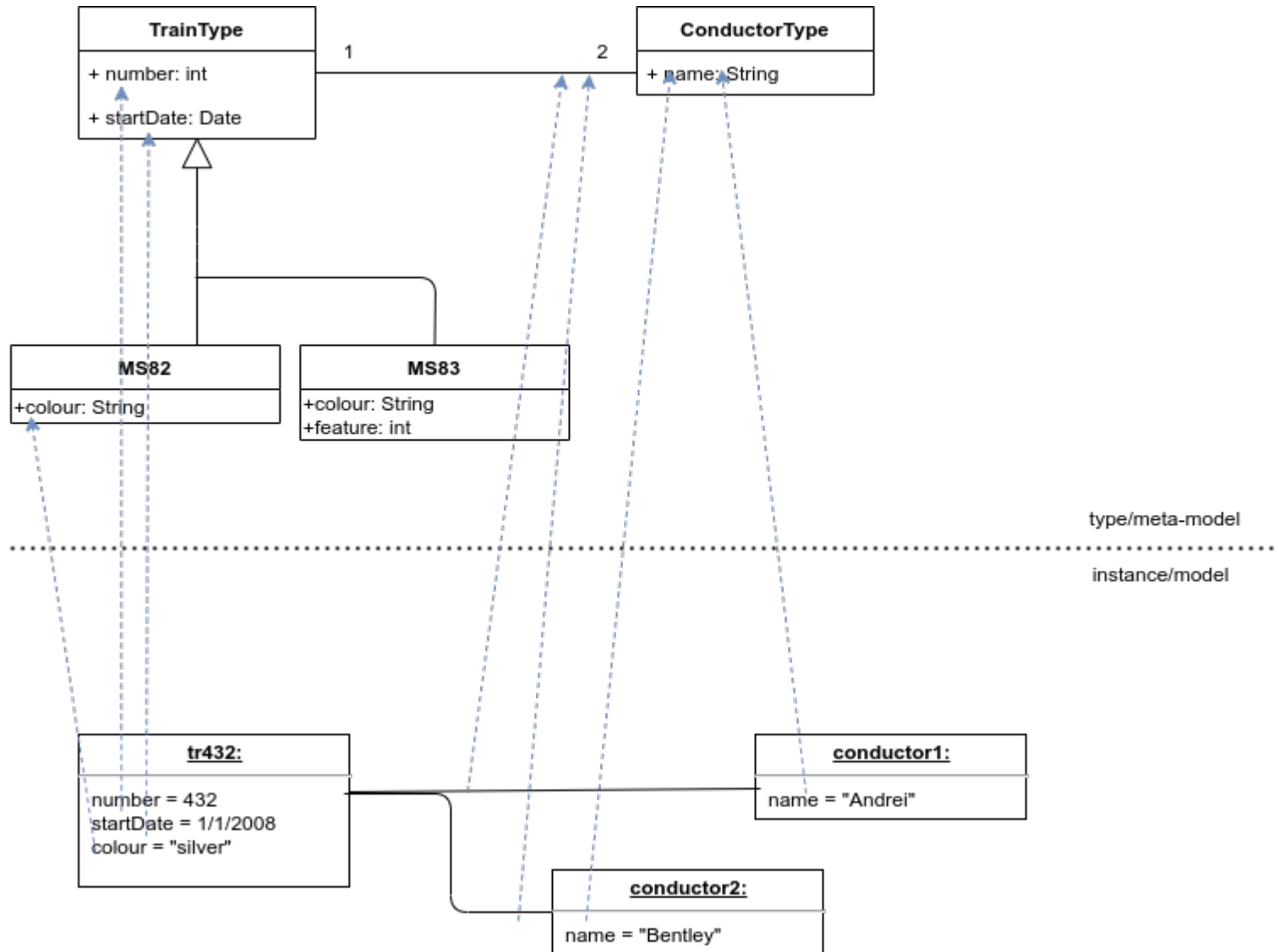


# Class vs. Instance

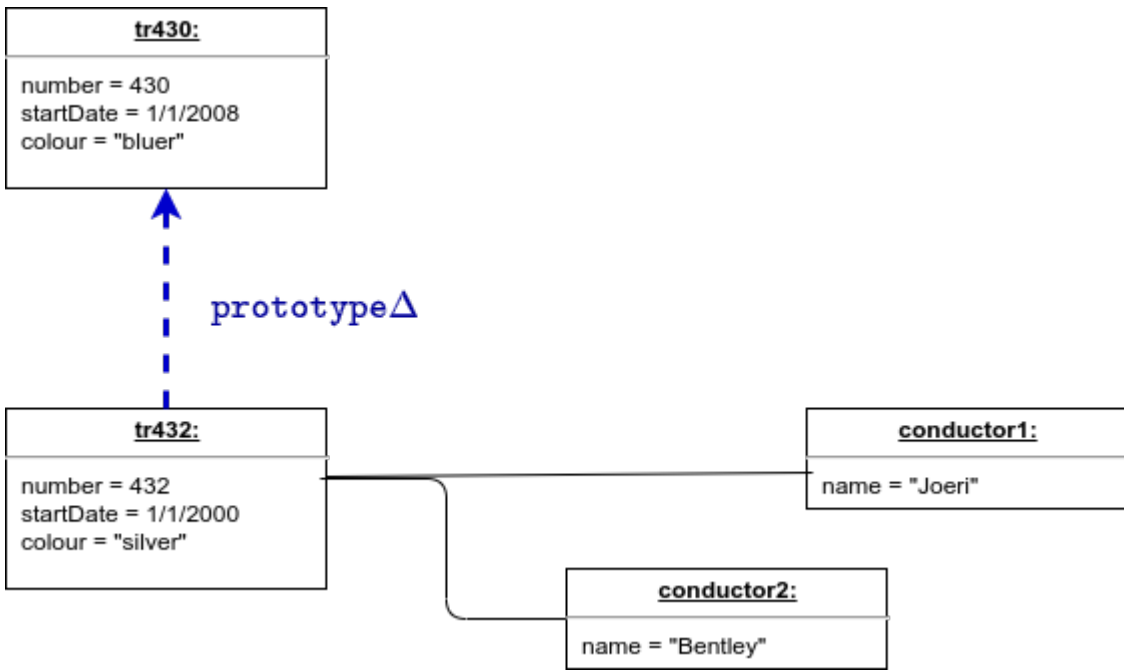


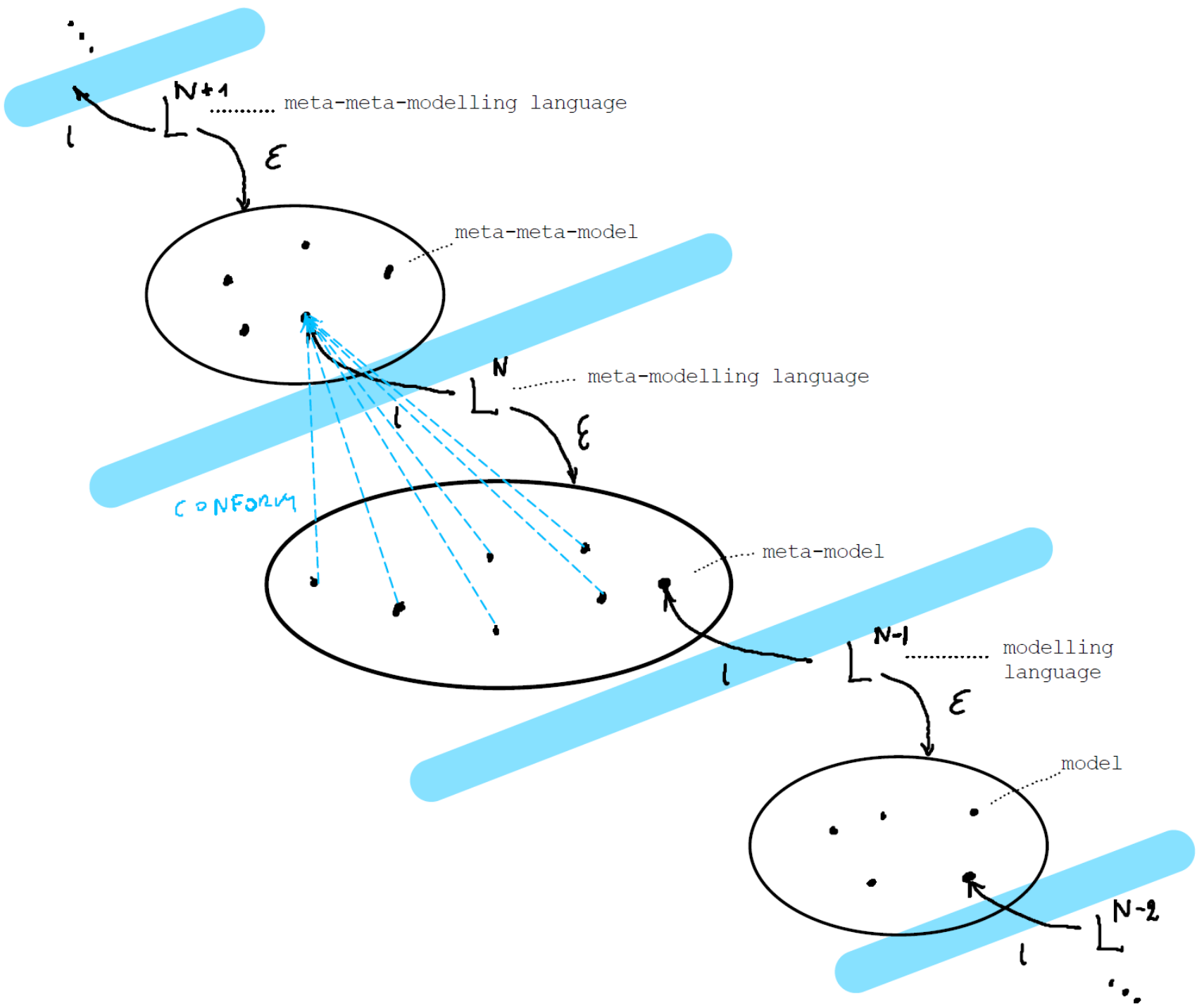


# “duck typing”



# delegation, object-based





meta-ness

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

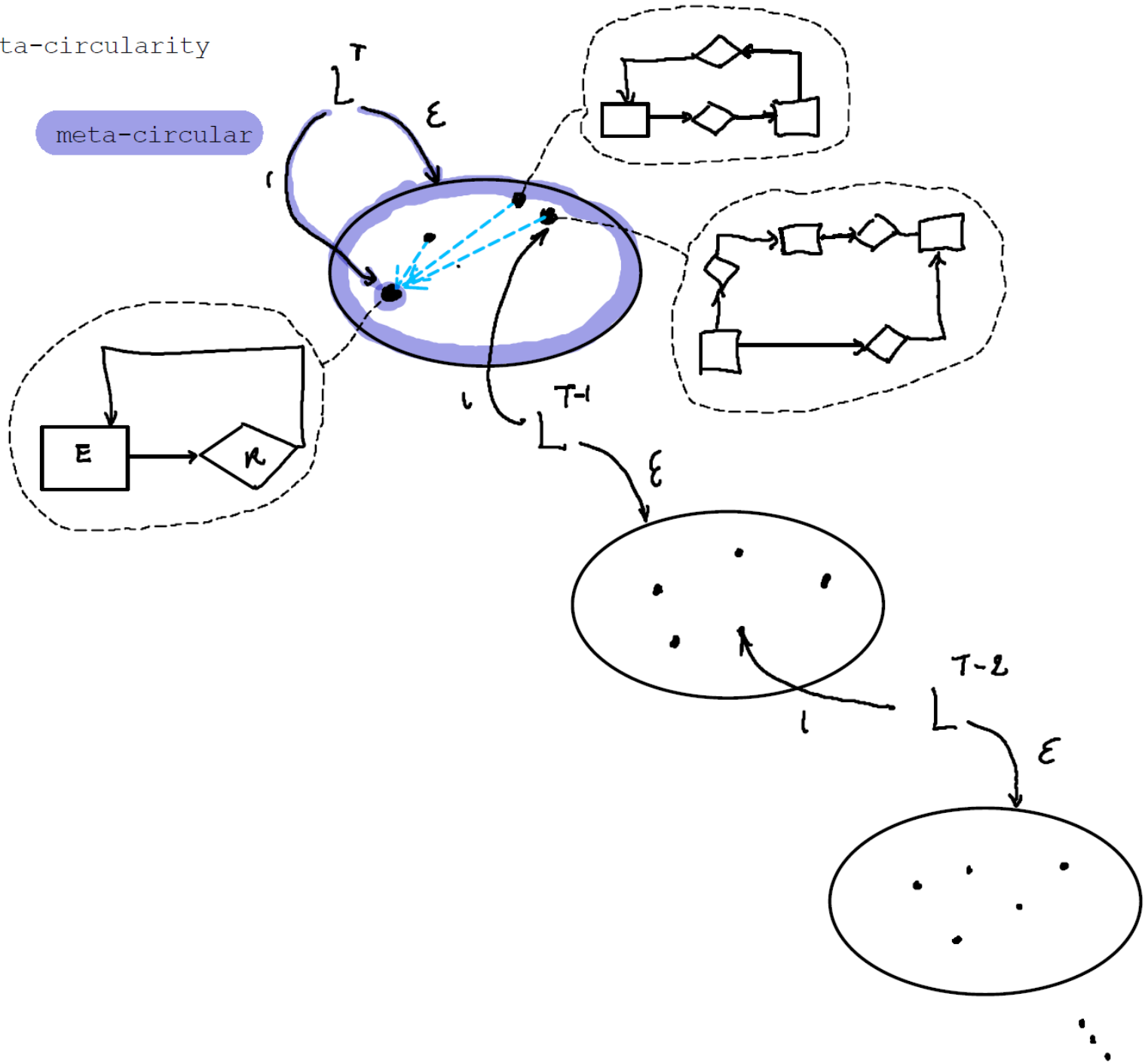
**acyclic**  $\forall e_1, e_2, n : e_1 R^n e_2 \rightarrow \neg e_2 R e_1$ , and

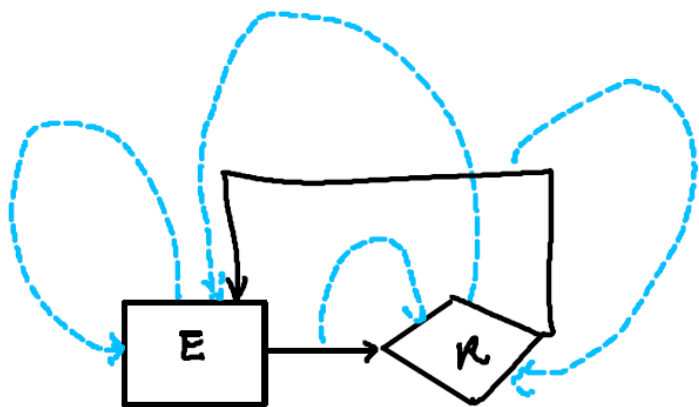
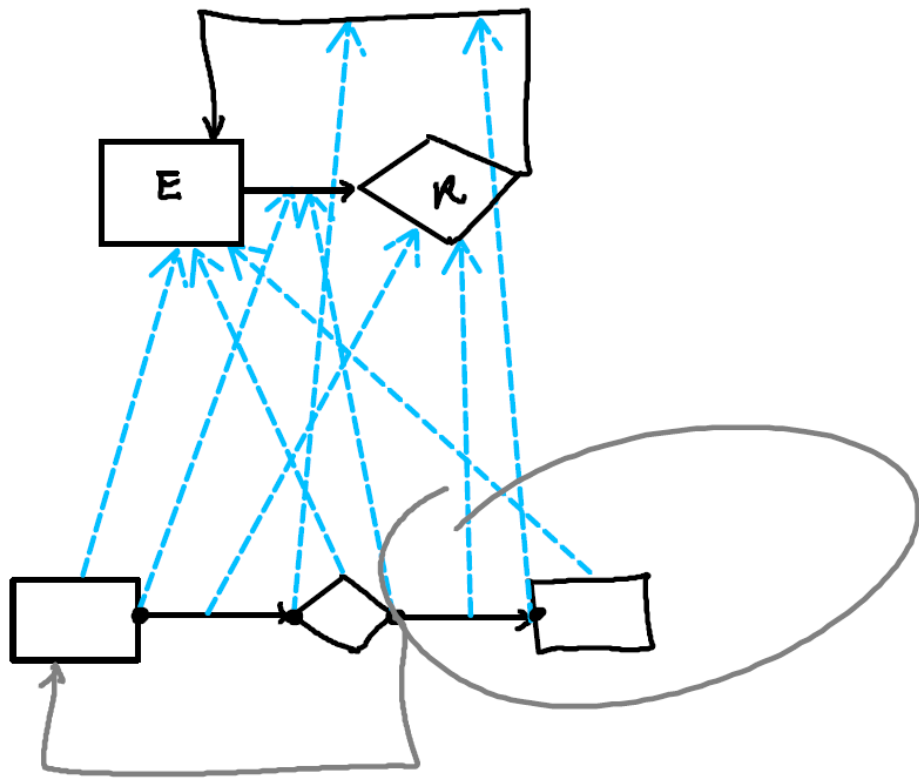
**anti-transitive**  $\forall n \geq 2 : R^n \cap R = \emptyset$ .

**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

meta-circular





Entity Relationship Diagram (ERD)

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

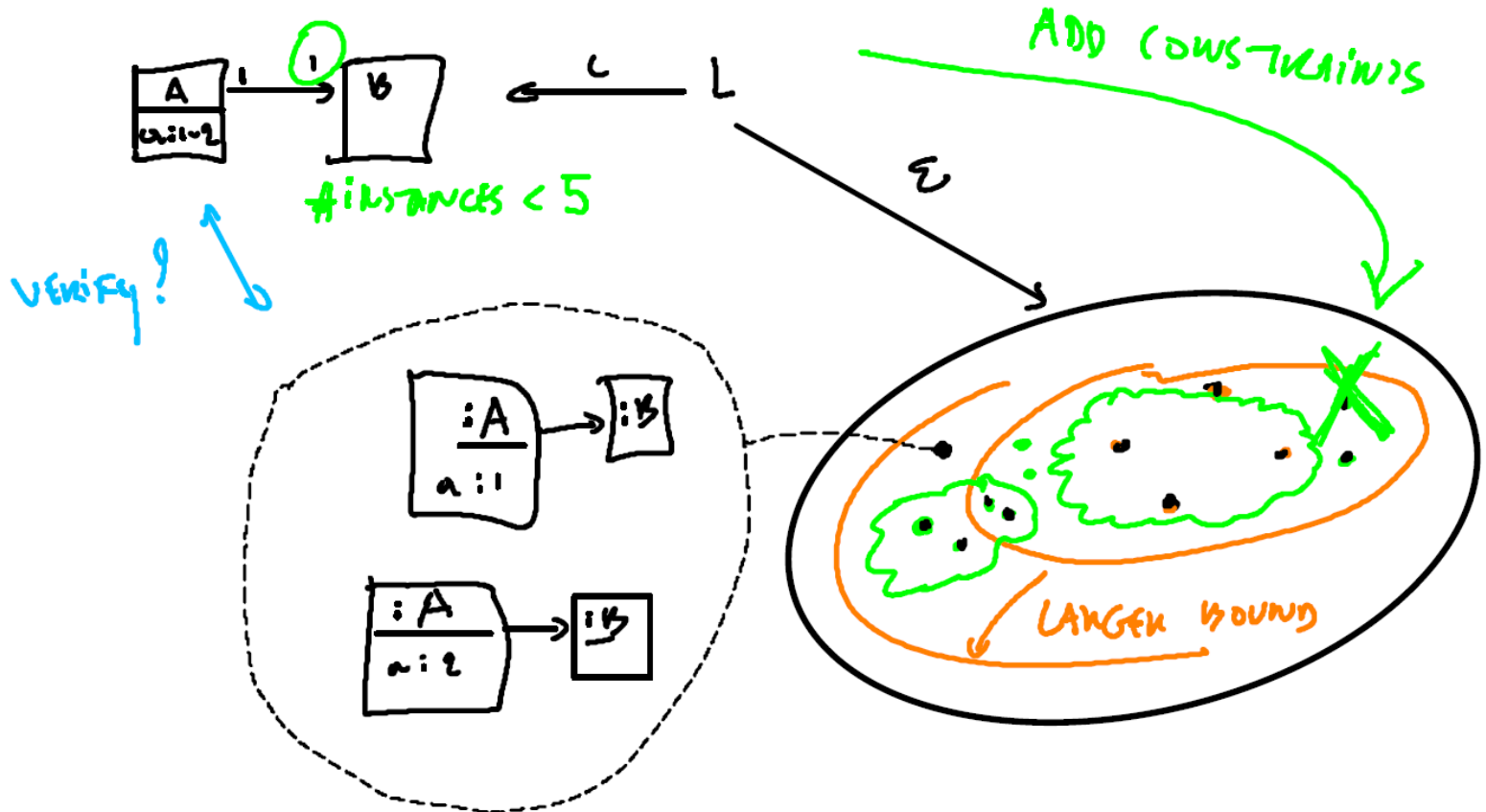
MMMCL can be used to "bootstrap"  
a meta-modelling infrastructure.  
e.g., Node and Edge in metaDepth

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

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

Alloy

MAX INSTANCES  
"BOUNDED EXPLORATION"



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

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

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

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

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)

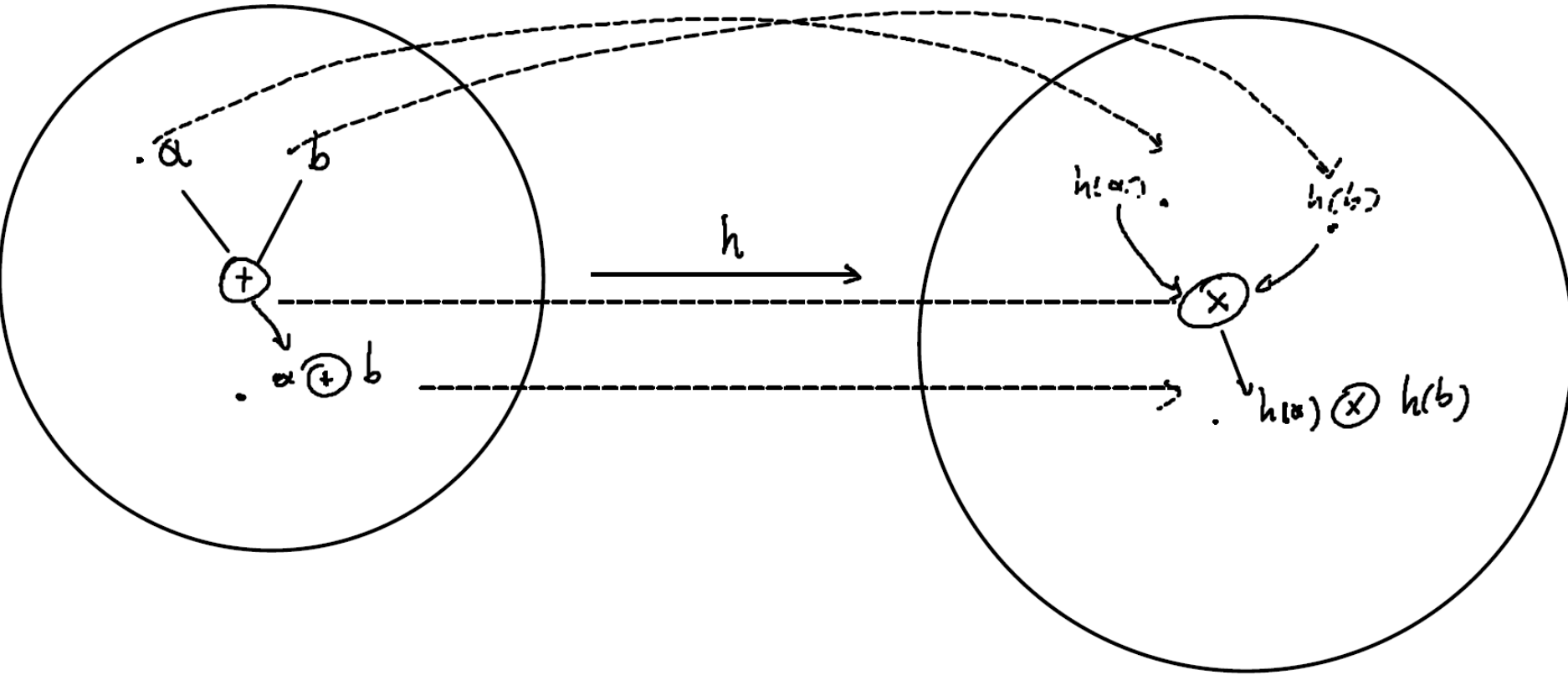
## Model is abstraction of System

<b>mapping feature</b>	A model is based on an original. <sup>4</sup>
<b>reduction feature</b>	A model only reflects a (relevant) selection of an original's properties.
<b>pragmatic feature</b>	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





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

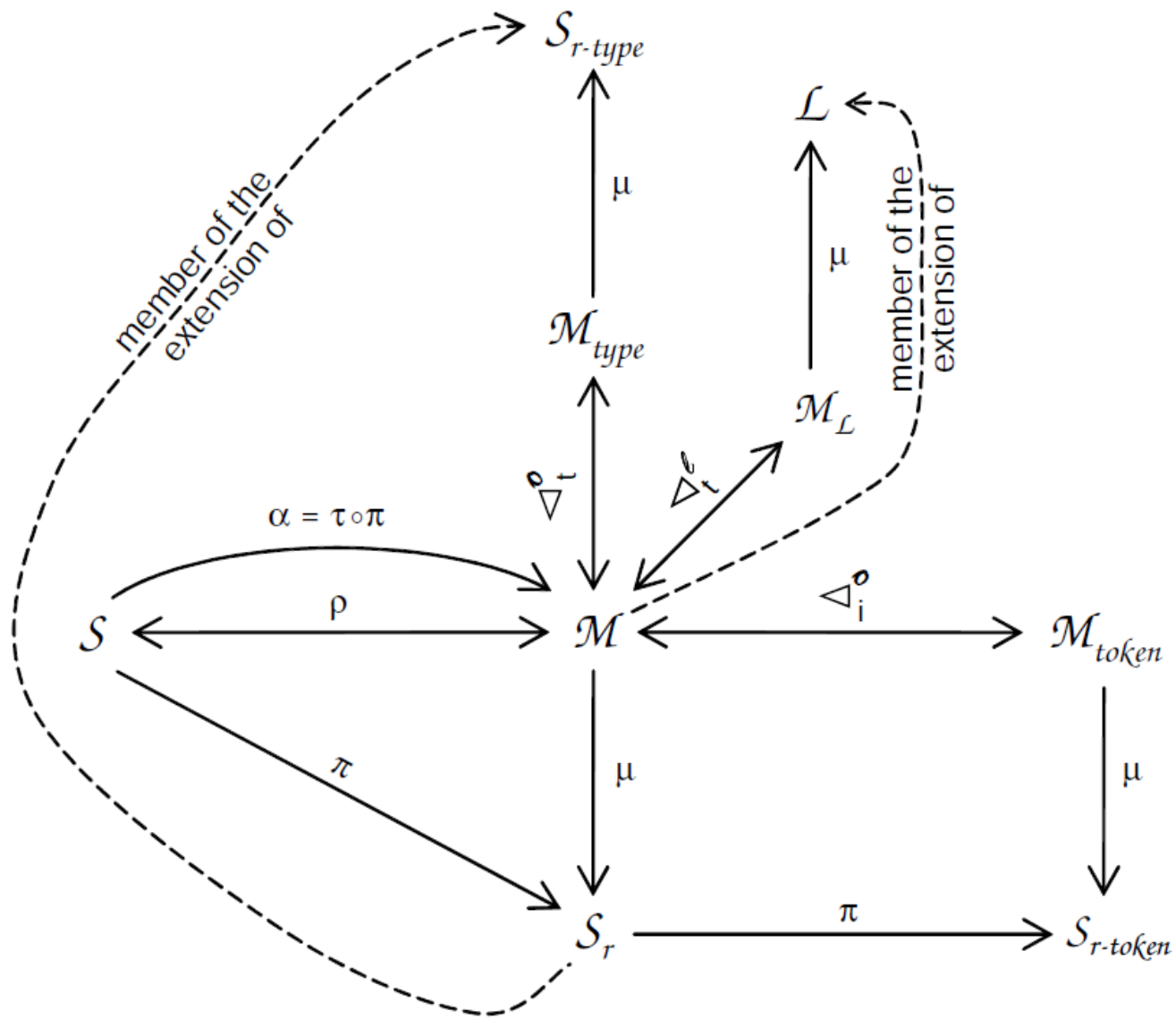
$$\varepsilon(C) = \{x \mid P(x)\}, \quad \text{where } P = \iota(C)$$

generalization vs. specialization

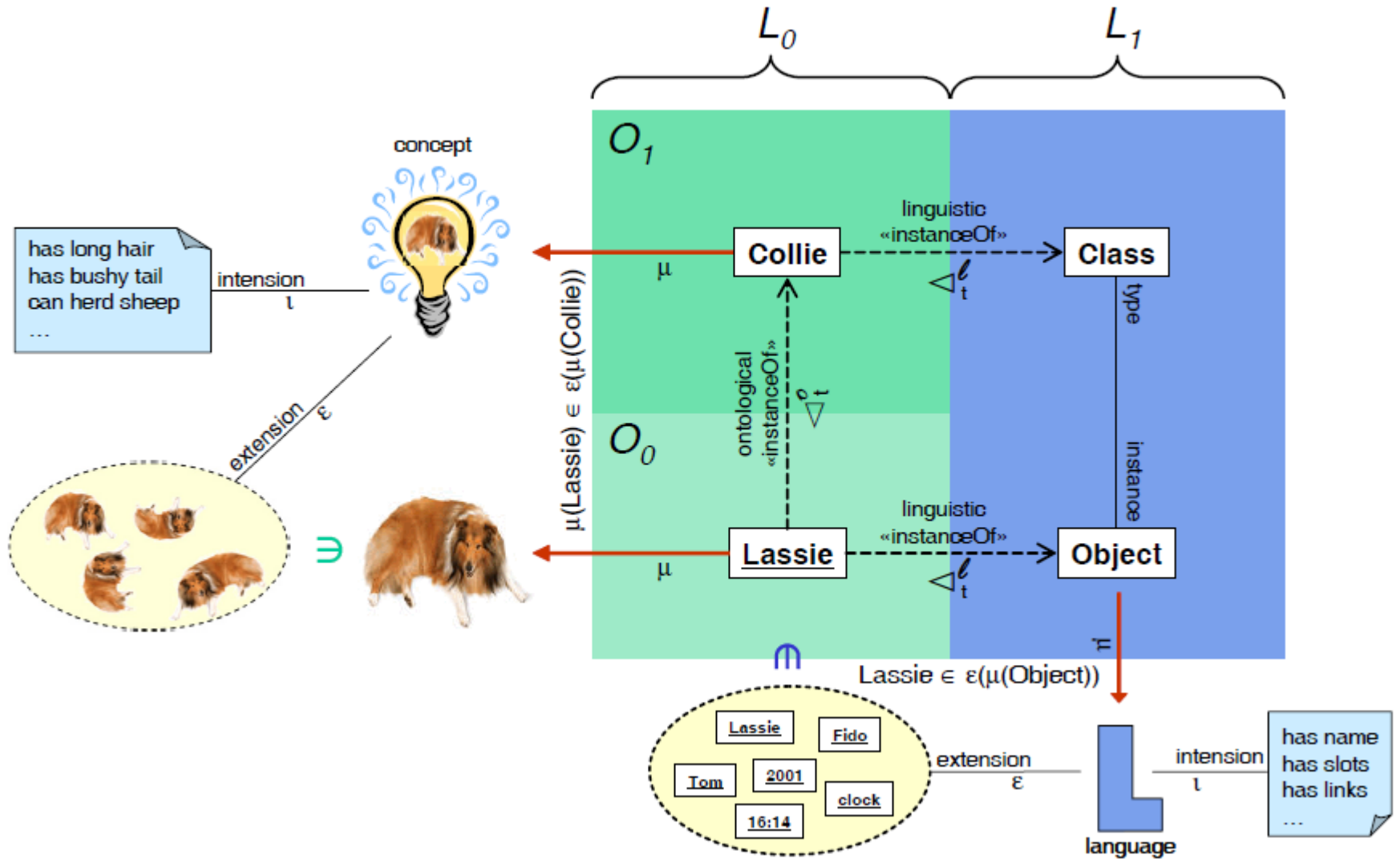
$$\forall x : \iota(C_{special})(x) \rightarrow \iota(C_{general_1})(x)$$

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

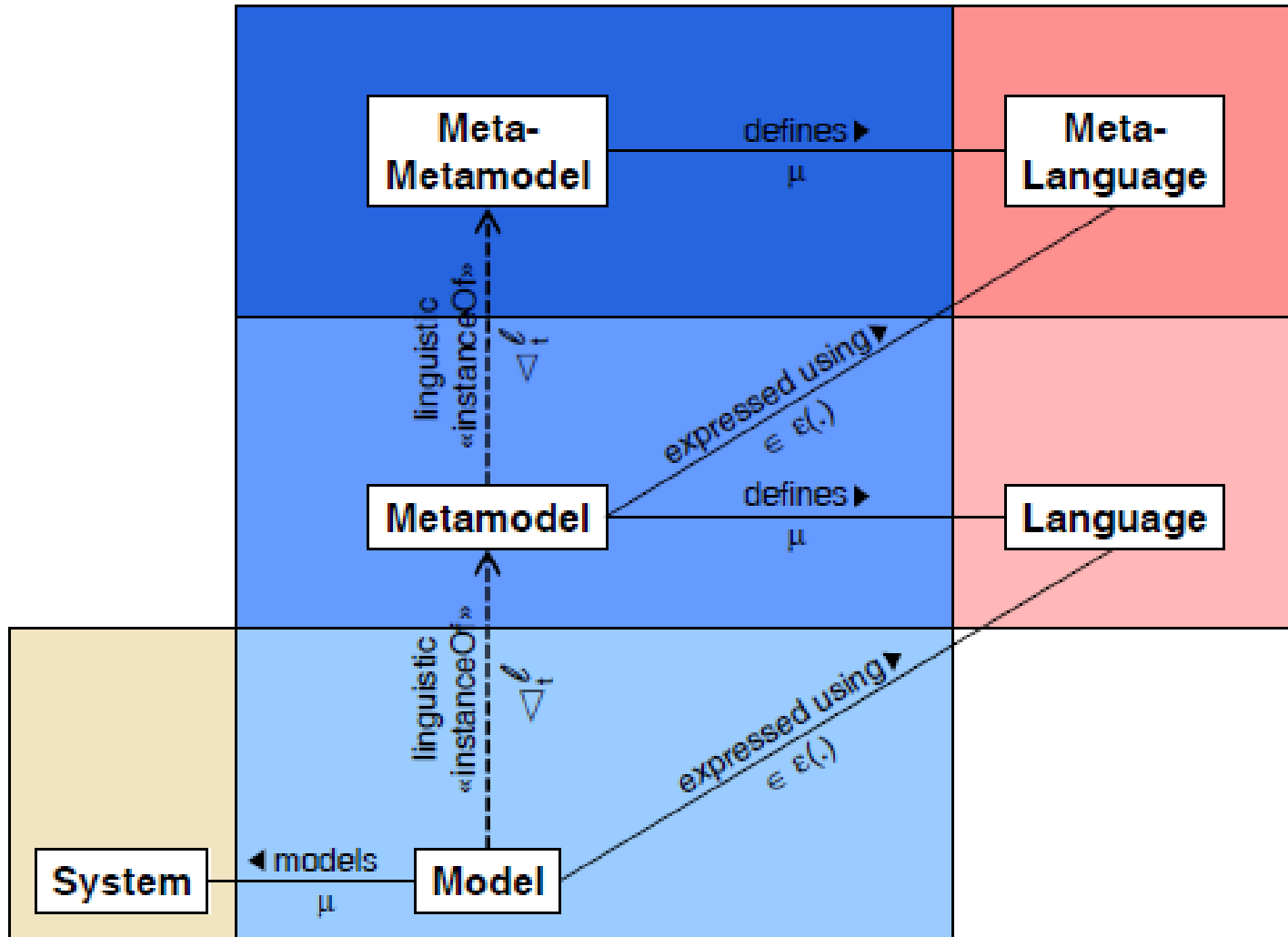
<i>notation</i>	<i>name</i>	<i>description</i>
$\alpha$	abstraction	creates a model from a system using projection ( $\pi$ ) and possibly classification ( $\Lambda$ ) and generalization ( $\Gamma$ ), hence $\mathcal{S} \triangleleft \alpha(\mathcal{S})$ .
$\Lambda$	classification	creates a type model, hence $\mathcal{M} \triangleleft_t \Lambda(\mathcal{M})$
$\Gamma$	generalization	creates a supermodel, hence $\mathcal{S} \triangleleft_t \mathcal{M} \rightarrow \mathcal{S} \triangleleft_t \Gamma(\mathcal{M})$
$\pi$	projection	a homomorphic mapping creating a reduced system from a given system, using selection and reduction of information.
$\rho$	represents	records the intention of a model to represent a system.
$\mu$	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 $\mu$ 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 $\mathcal{S} \triangleleft_t^l \mathcal{M} \Leftrightarrow \mathcal{S} \in \varepsilon(\mu(\mathcal{M}))$



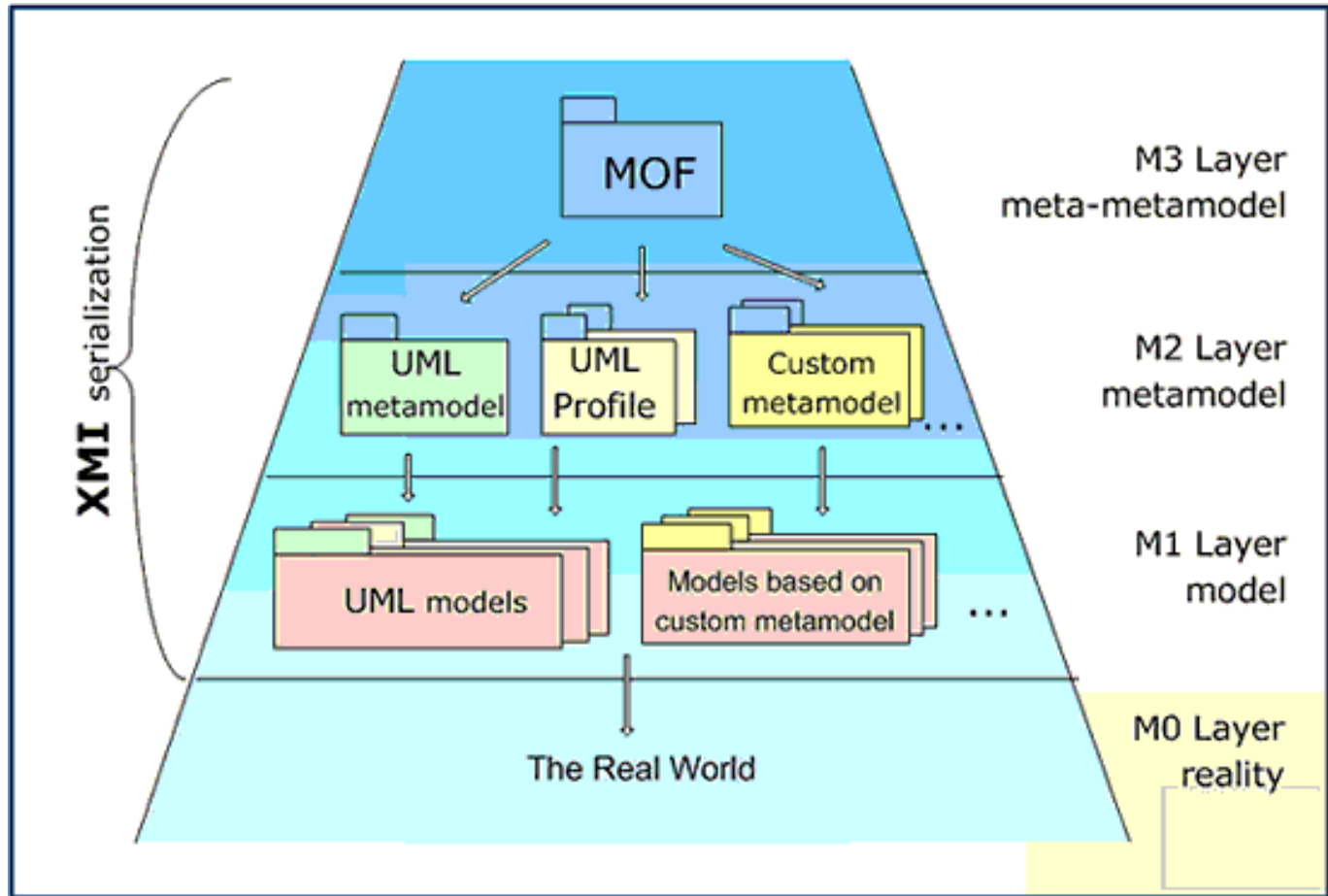
# Ontological vs. Linguistic Conformance/Instantiation

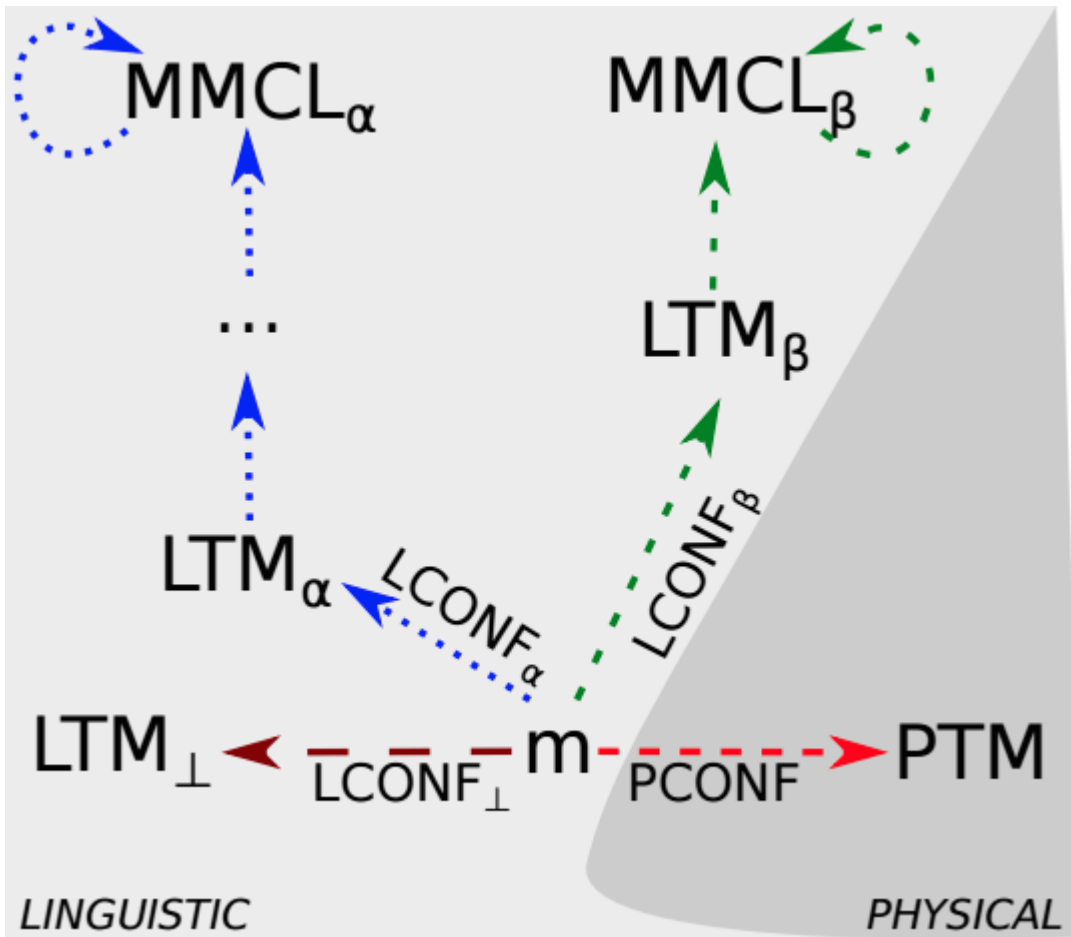


# Language Definition Stack



## 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 17<sup>th</sup> 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>



# metaDepth

<http://metadepth.org/>

Textual language for (multi-level) (meta-)modelling

Constraint and operations in the Epsilon Object Language (EOL)

<https://www.eclipse.org/epsilon/>

EOL is similar the UML's Object Constraint Language (OCL)

EOL can be used in metaDepth for:

- Queries within the model (limited to navigation)
- Computation/assignment, for example to give operational semantics

# metaDepth: meta-model

```
Model TrainNet {  
  
  Node Train{  
    train_name : String {id};  
    maxVelocity : int;  
    maxVConstraint: $self.maxVelocity > 0 and self.maxVelocity <= 300$  
    currSegment : Segment[1];  
  }  
  
  Node Segment{  
    currTrain : Train[0..1];  
    nextSegment : Segment[0..1];  
  }  
  
  Node FastTrain : Train {}  
  Node IntercityExpress: Train {}  
}
```

# metaDepth: model

```
TrainNet tn {  
  Train train1 {train_name = "T3456"; maxVelocity = 300; currSegment = seg1;}  
  
  IntercityExpress train2 {train_name = "ICE1234"; maxVelocity = 250;}  
  
  Segment seg1 {currTrain = train1;}  
  Segment seg2 {}  
}
```

Verifying conformance between `tn` and `TrainNet`:

**Constraint fails:** cardinality 0 not allowed for `currSegment` in `train2`

**Reason:** `train2` is not on a `Segment`

# metaDepth - Epsilon Object Language (EOL)

**Example query:** What are the trains in each segment?

```
operation main(){
  for (s in Segment.allInstances()){
    if (Train.allInstances.excludes(s.currTrain)){
      ('Segment ` + s + "has no train.>").println();
    }
    else{
      ("Segment ` + s + ` has train: `` + s.currTrain.toString + ``").println();
    }
  }
}
```

**Result:**

```
Segment seg2 has no train.
Segment seg1 has train 'train1'
```

**Caveat:** explicit navigation (via . . .), need to know types  
(or be able to query: `Types.allInstances()`)

Meta-model



model

```
strict Model PetriNets@1 {
  abstract Node NamedElement{
    name : String {id};
  }
  Node Place : NamedElement {
    tokens : int = 0;
    outTrans : Transition[*] {ordered,unique};
    inTrans : Transition[*] {ordered,unique};
    minTokens: $self.tokens>=0$
  }
  Node Transition : NamedElement {
    inPlaces : Place[*] {ordered,unique};
    outPlaces: Place[*] {ordered,unique};
  }
  Edge ArcPT (Place.outTrans, Transition.inPlaces) {
    weight: int = 1;
  }
  Edge ArcTP (Transition.outPlaces, Place.inTrans) {
    weight: int = 1;
  }
  minWeight(ArcTP, ArcPT): $self.weight>0$
  minPlaces:$Place.allInstances().size(>0$
}
```

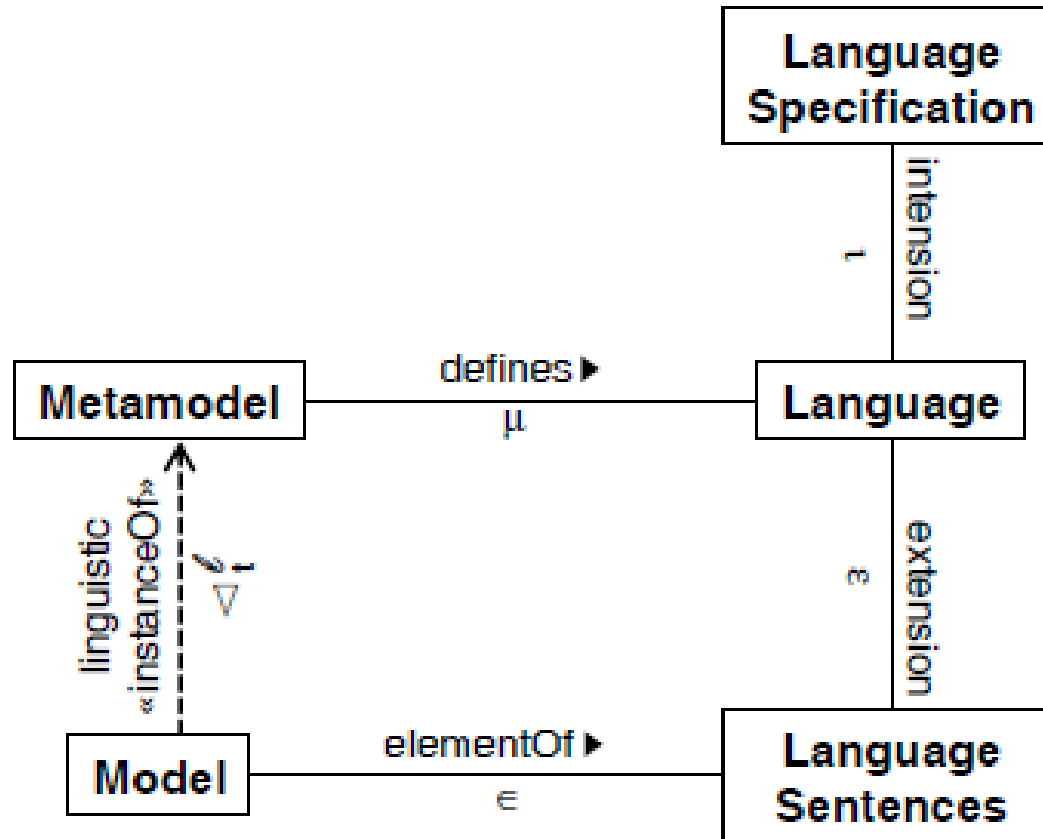
```
load "PetriNets"
PetriNets Test{
  Place p0{name="p0"; tokens=2;}
  Place p1{name="p1"; tokens=0;}
  Place p2{name="p2"; tokens=2;}
  Transition t1{name="t1";}
  ArcPT pt0(p0,t1){weight=1;}
  ArcPT pt1(p2,t1){weight=1;}
  ArcTP tp(t1,p1){weight=2;}
}
```

# MetaDepth

## operational semantics

```
operation main() {
  'Simulating the Petri Net'.println();
  while (Transition.allInstances()->exists(t|t.enabled() and t.fire())) {}
}
operation Transition enabled() : Boolean {
  ('checking enabledness of '+self.name).println();
  return self.ArcPT->forall(arc| arc.inPlaces.tokens>=arc.weight);
}
operation Transition fire() : Boolean {
  ('Firing '+self.name).println();
  for (arc in self.ArcPT)
    arc.inPlaces.tokens := arc.inPlaces.tokens-arc.weight;
  for (arc in self.ArcTP)
    arc.outPlaces.tokens := arc.outPlaces.tokens+arc.weight;
  return true;
}
```

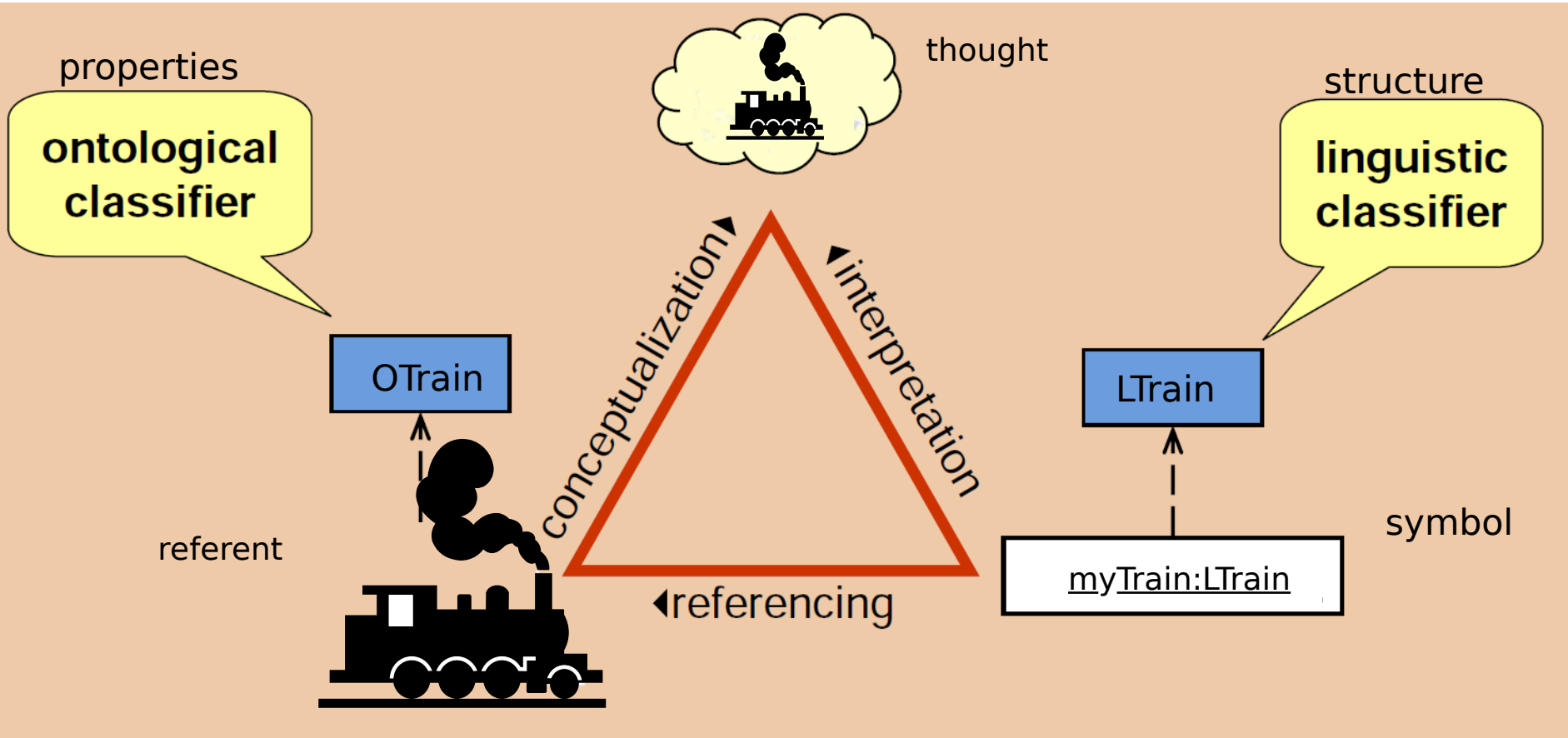
# Meta-models as Language Definitions



# 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

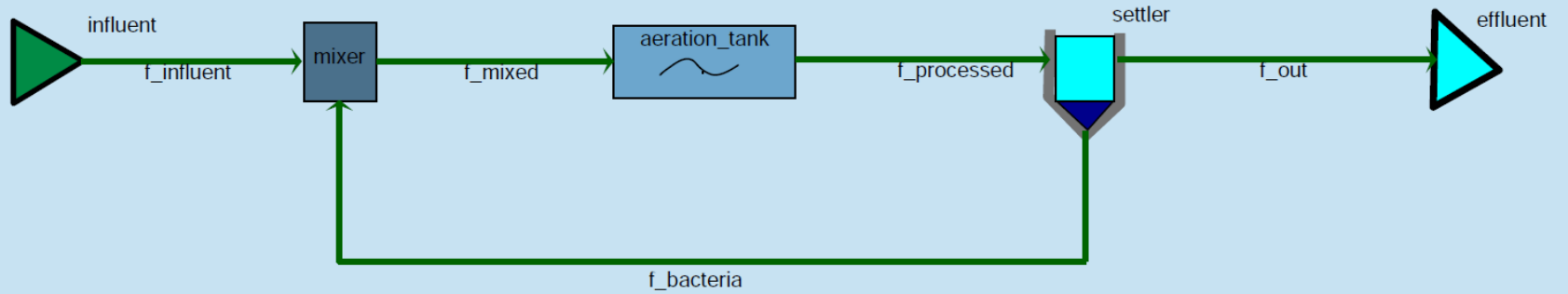
## Operational vs. Denotational (Translational) semantics



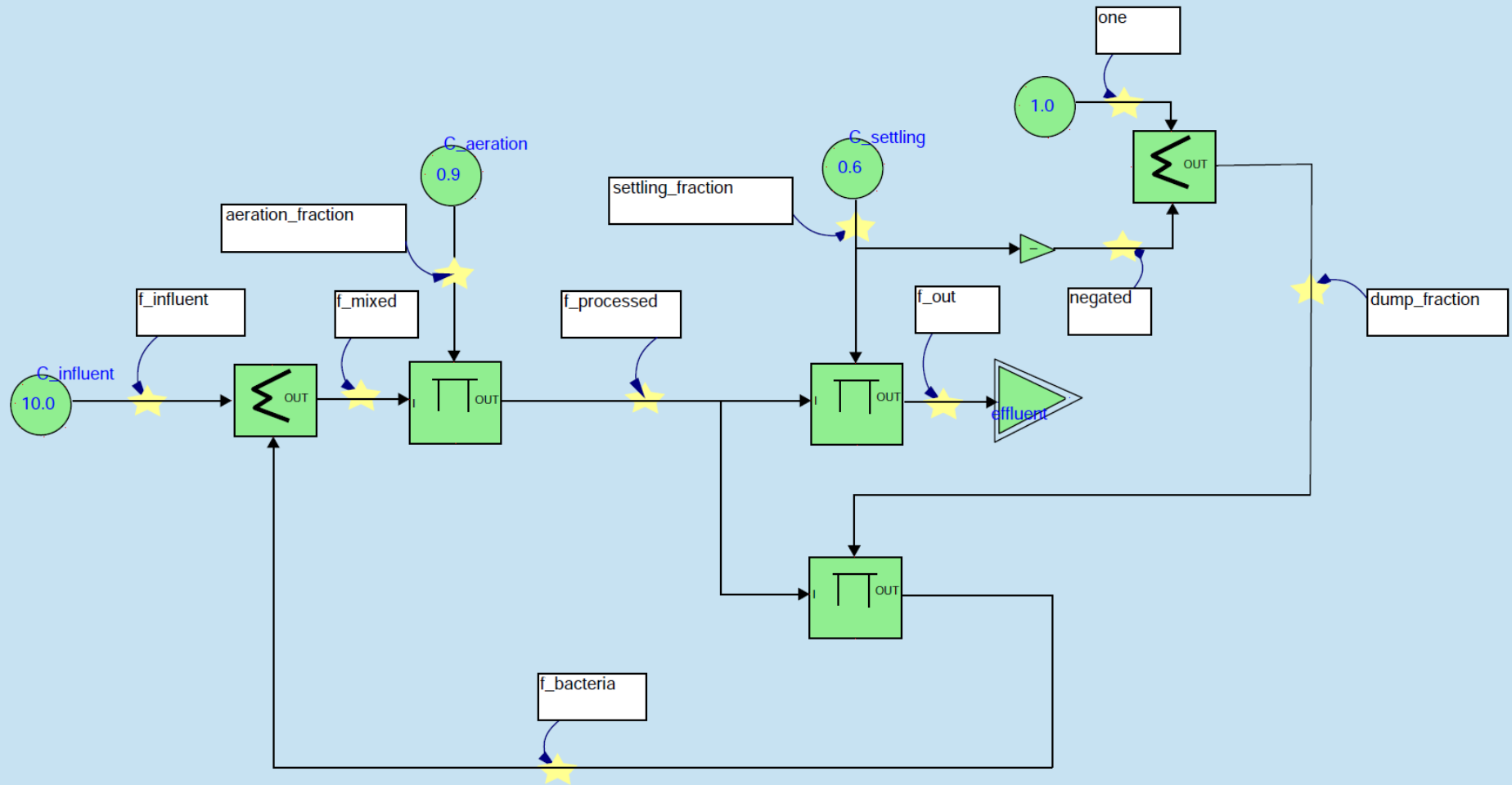
NATO's Sarajevo Waste Water Treatment Plant

[www.nato.int/sfor/cimic/env-pro/waterpla.htm](http://www.nato.int/sfor/cimic/env-pro/waterpla.htm)

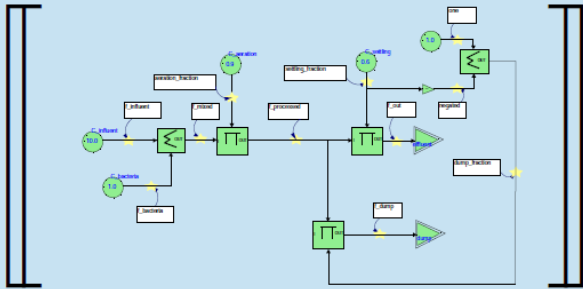
# What does this WWTP model mean?



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



## Meaning of the CBD ... semantic mapping onto algEqns

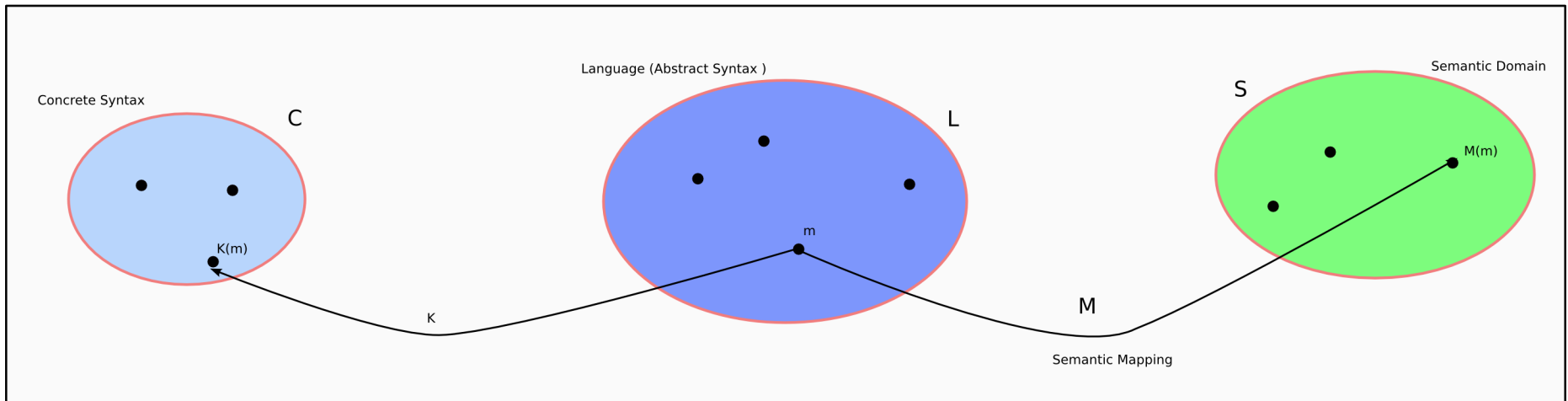


= {

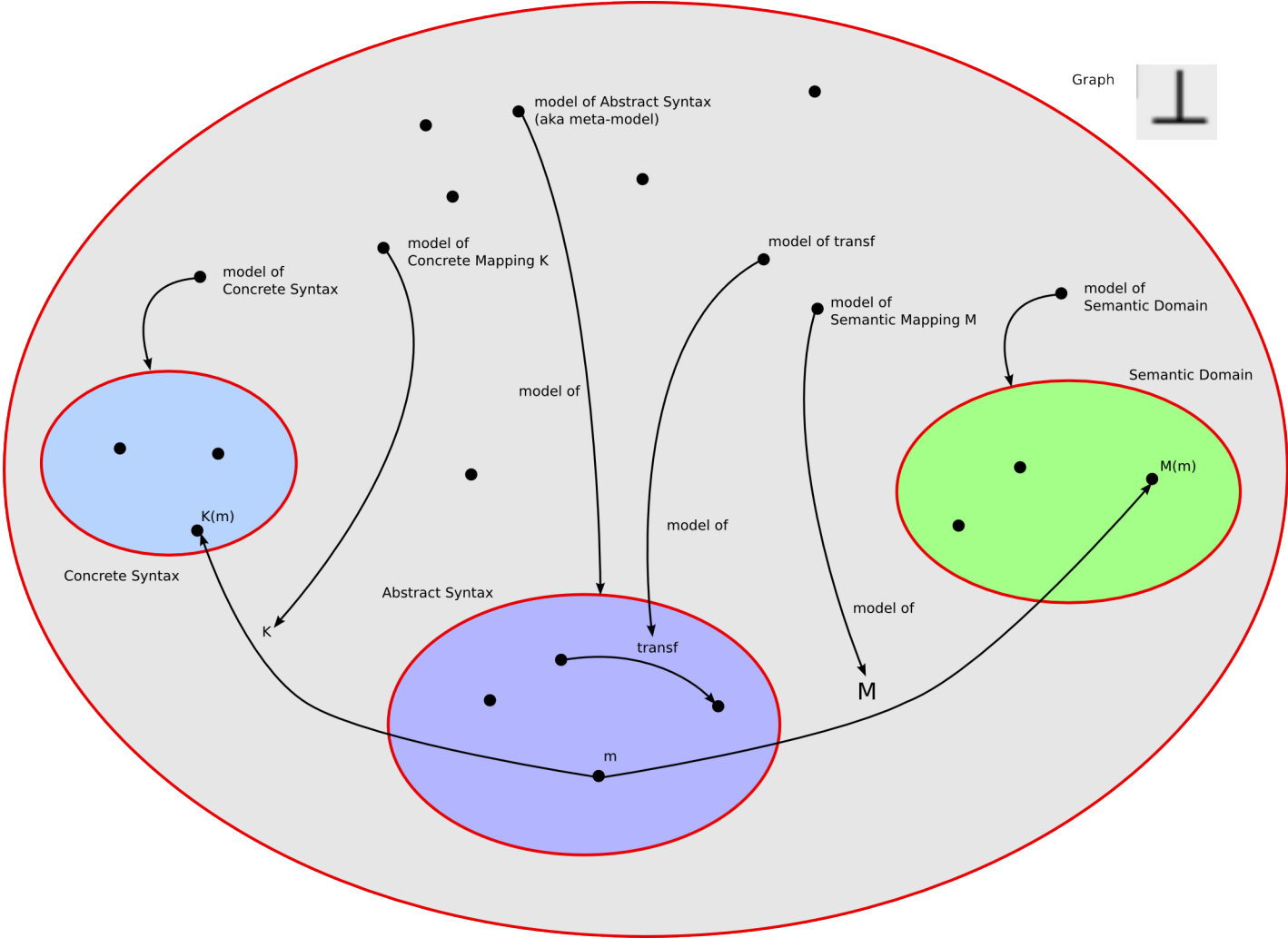
$f\_influent$	=	$C\_influent$
$f\_bacteria$	=	$C\_bacteria$
$f\_mixed$	=	$f\_influent + f\_bacteria$
$aeration\_fraction$	=	$C\_aeration$
$f\_processed$	=	$aeration\_fraction * f\_mixed$
$settling\_fraction$	=	$C\_settling$
$negated$	=	$-settling\_fraction$
$one$	=	$1$
$dump\_fraction$	=	$one + negated$
$f\_dump$	=	$f\_processed * dump\_fraction$
$f\_out$	=	$settling\_fraction * f\_processed$

# "linguistic" view on Modelling Languages/Formalisms: Syntax and Semantics

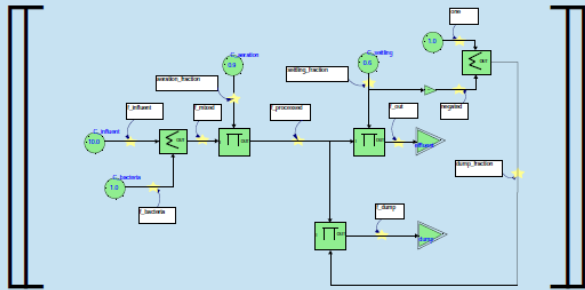
Concrete Formalism F



# Explicit "linguistic" Modelling of Modelling Languages/Formalisms



## Meaning of the CBD ... semantic mapping onto algEqns

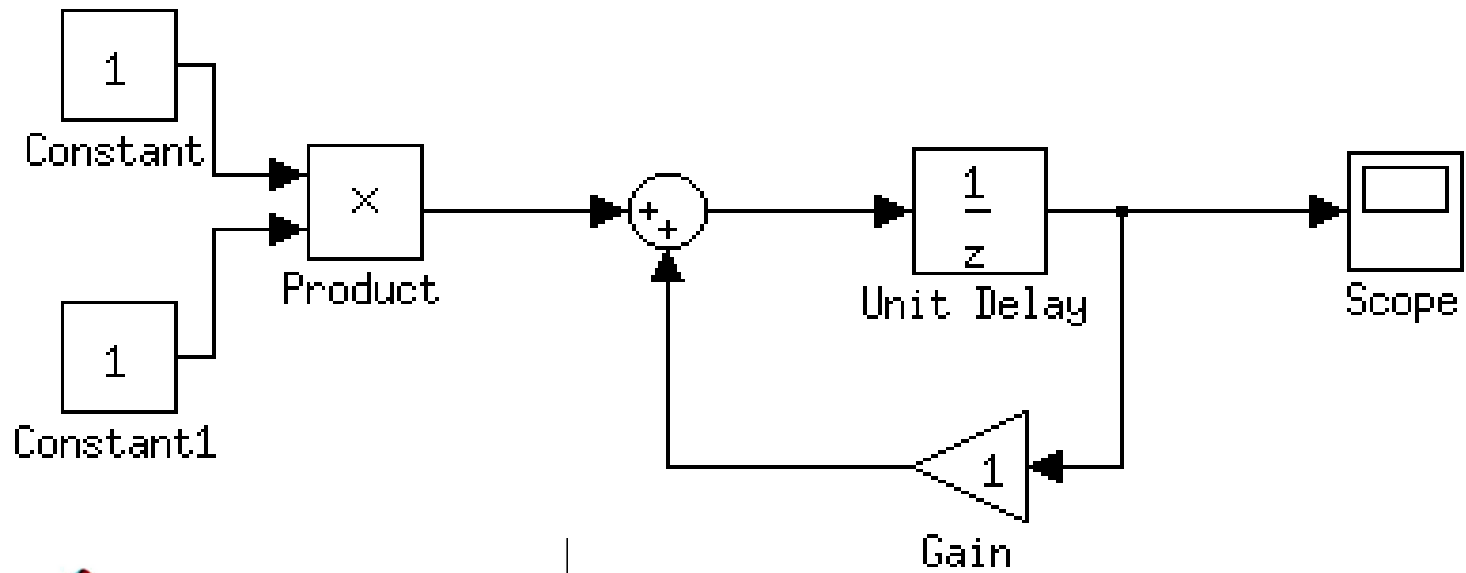


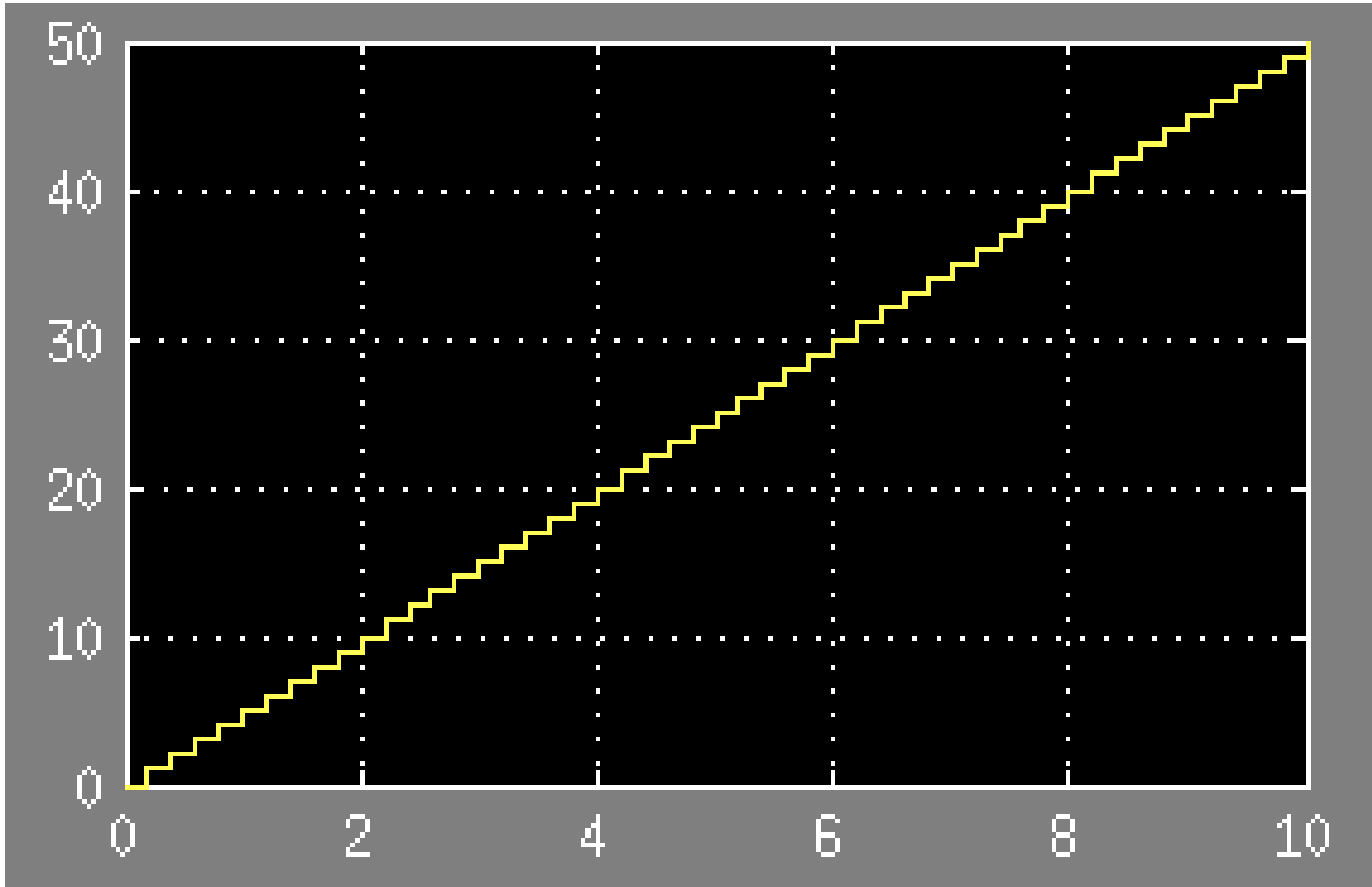
= {

$f\_influent$	=	$C\_influent$
$f\_bacteria$	=	$C\_bacteria$
$f\_mixed$	=	$f\_influent + f\_bacteria$
$aeration\_fraction$	=	$C\_aeration$
$f\_processed$	=	$aeration\_fraction * f\_mixed$
$settling\_fraction$	=	$C\_settling$
$negated$	=	$-settling\_fraction$
$one$	=	$1$
$dump\_fraction$	=	$one + negated$
$f\_dump$	=	$f\_processed * dump\_fraction$
$f\_out$	=	$settling\_fraction * f\_processed$



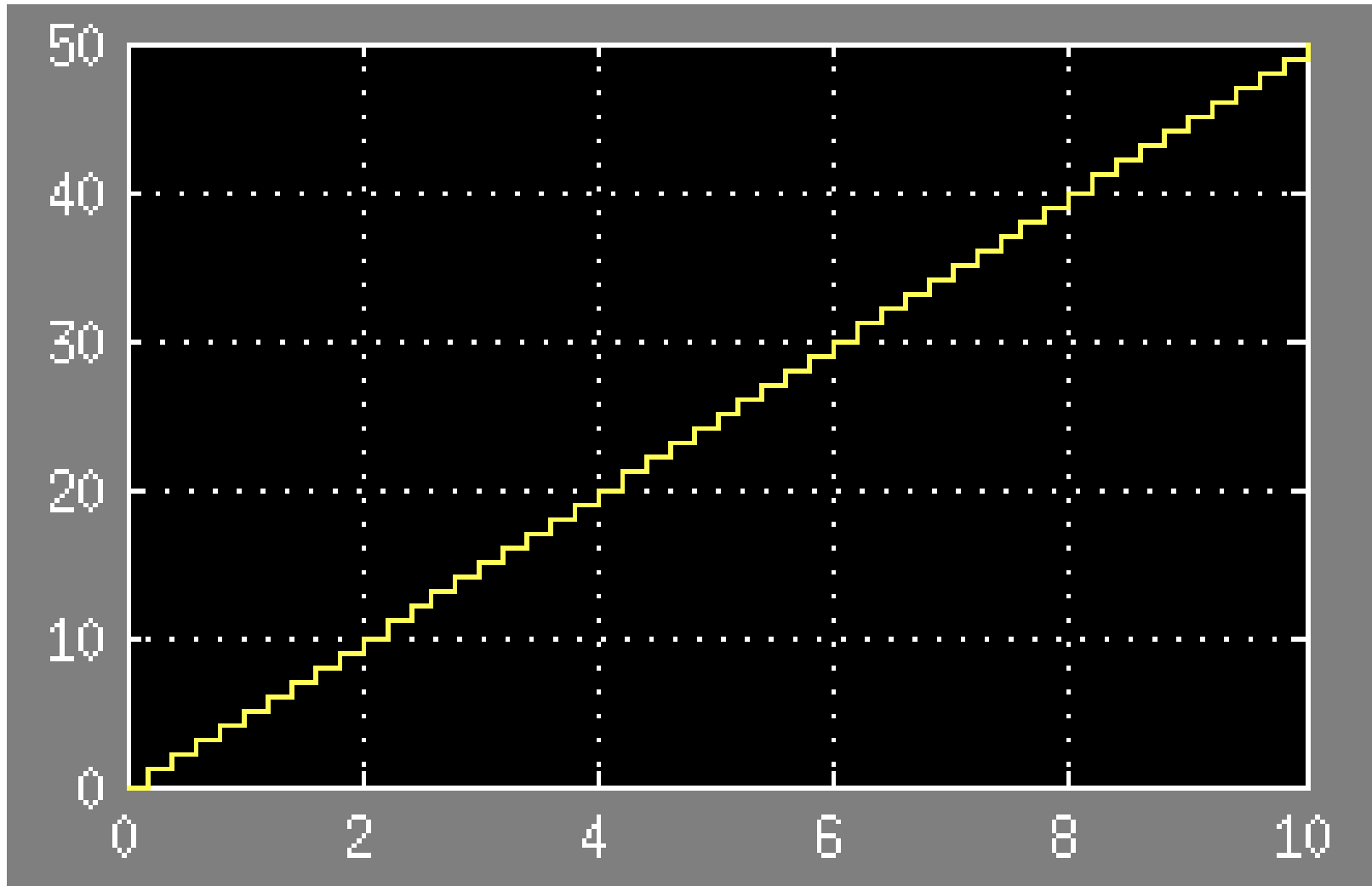
Causal Block Diagrams  
(syntax)

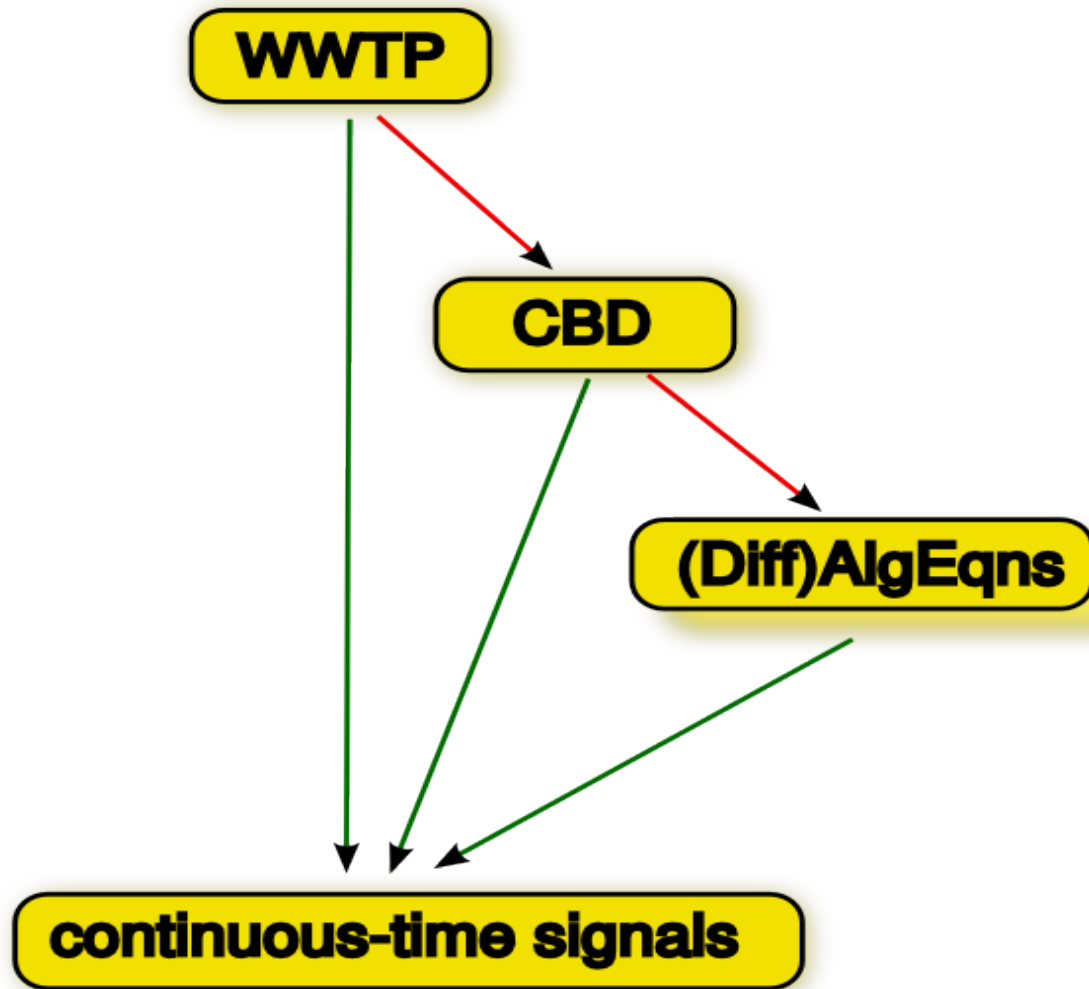




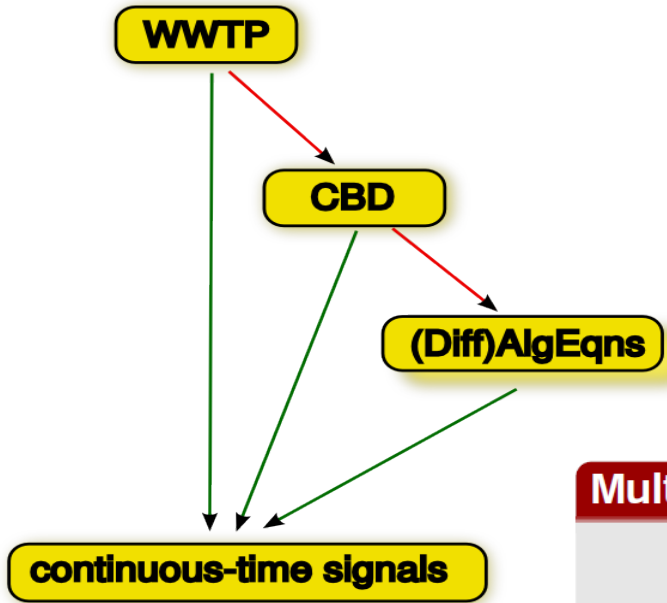
```
logicalTime  $\leftarrow$  0  
while not end_condition do  
  schedule  $\leftarrow$  LOOPDETECT(DEPGRAPH(cbd))  
  for gblock in schedule do  
    COMPUTE(gblock)  
  end for  
  logicalTime  $\leftarrow$  logicalTime +  $\Delta t$   
end while
```

## Causal Block Diagrams (semantics)

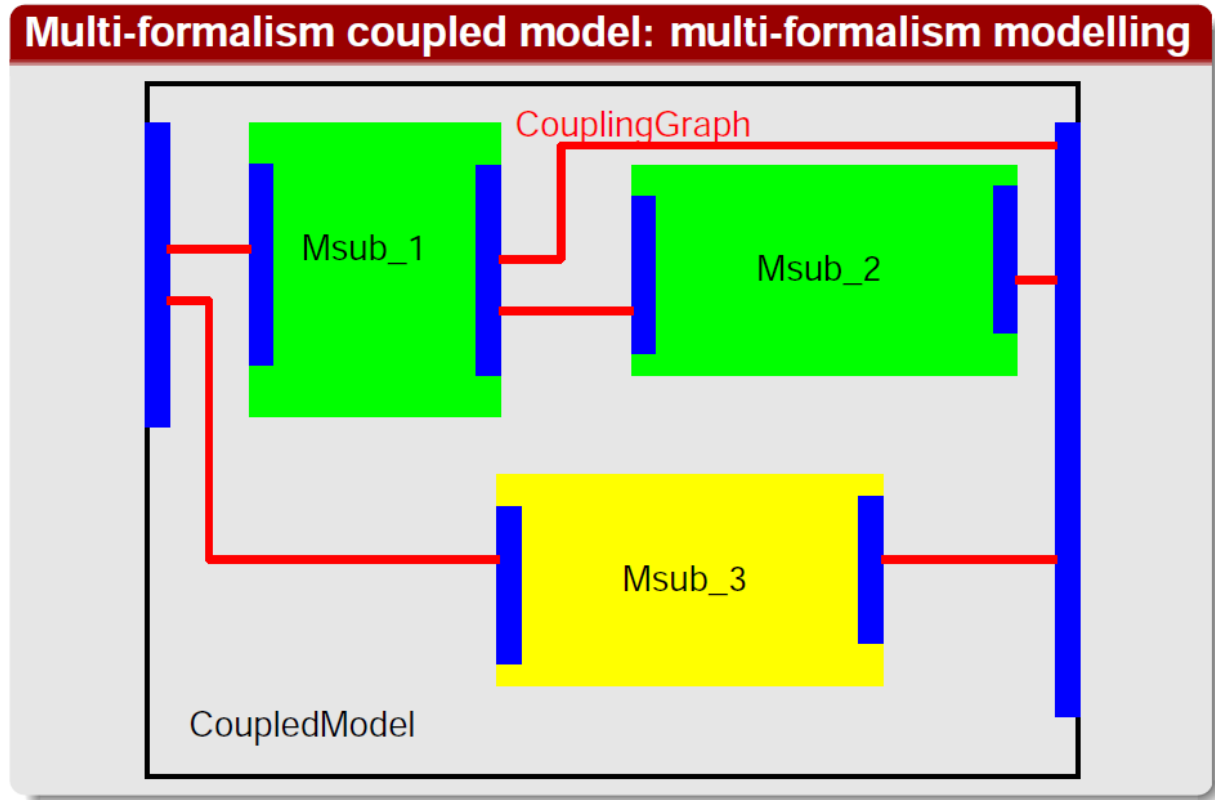


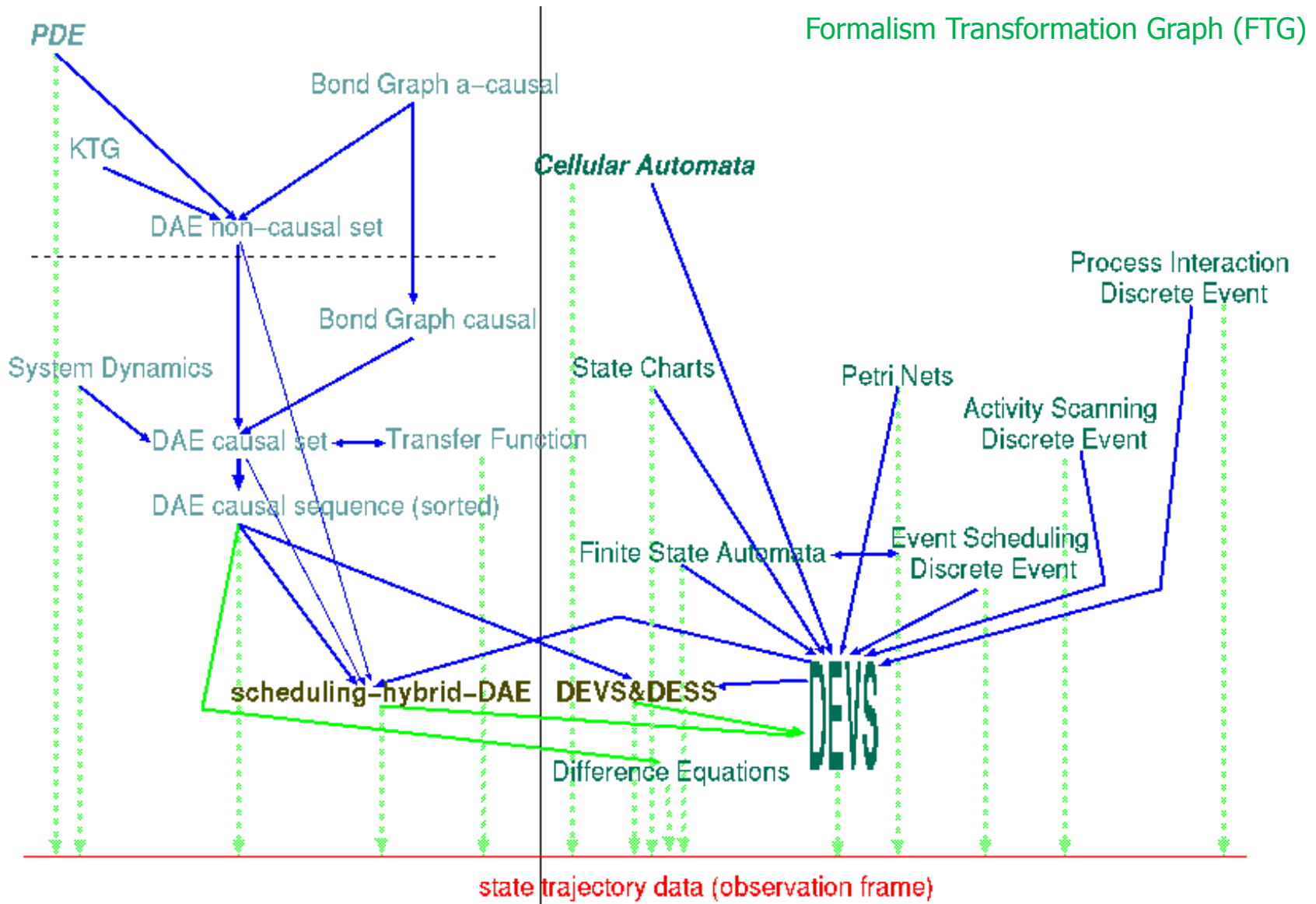


# Formalism Transformation Graph (FTG)

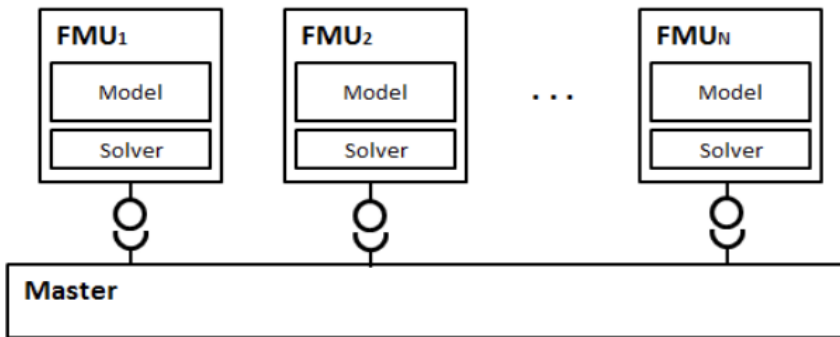
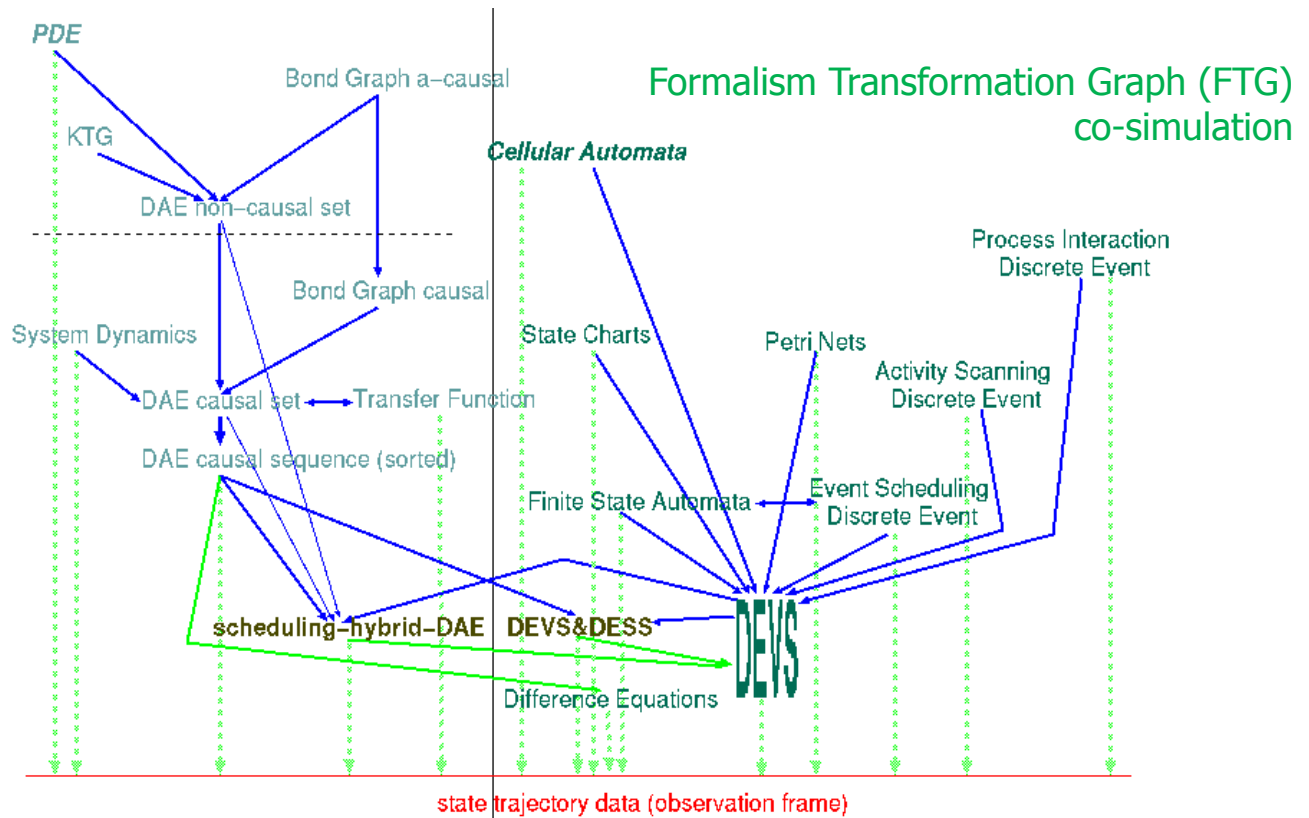


Bran Selic: "fragmentation problem"





Hans Vangheluwe and Ghislain C. Vansteenkiste. A multi-paradigm modeling and simulation methodology: Formalisms and languages. In European Simulation Symposium (ESS), pages 168 – 172. Society for Computer Simulation International (SCS), October 1996. Genoa, Italy.



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 | 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.

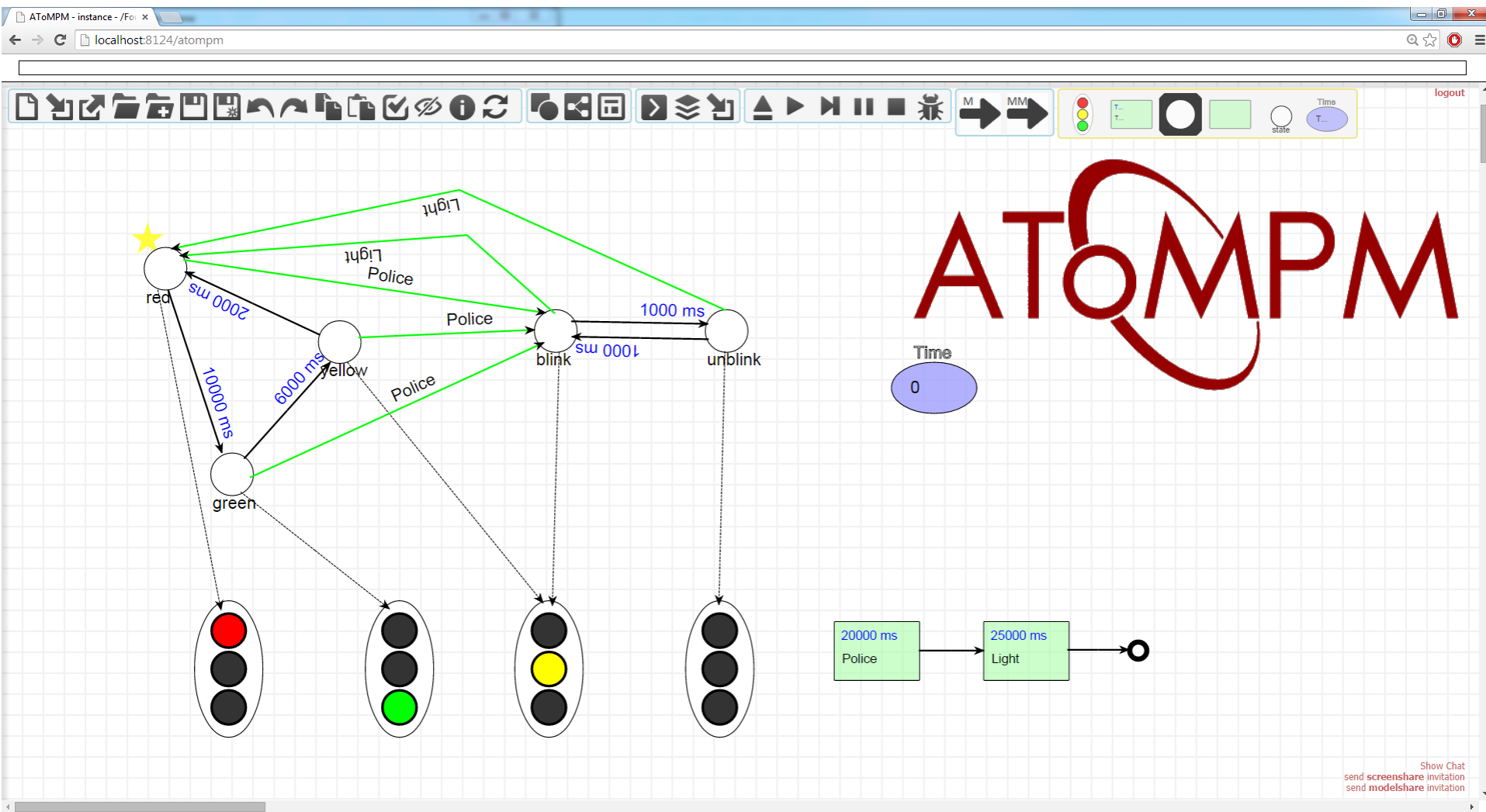
Comments: 157 pages, about 30 figures  
 Subjects: Systems and Control (cs.SY)  
 MSC classes: 65Y10  
 ACM classes: I.6.1; I.6.7  
 Cite as: arXiv:1702.00686 [cs.SY]  
 (or arXiv:1702.00686v1 [cs.SY] for this version)





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

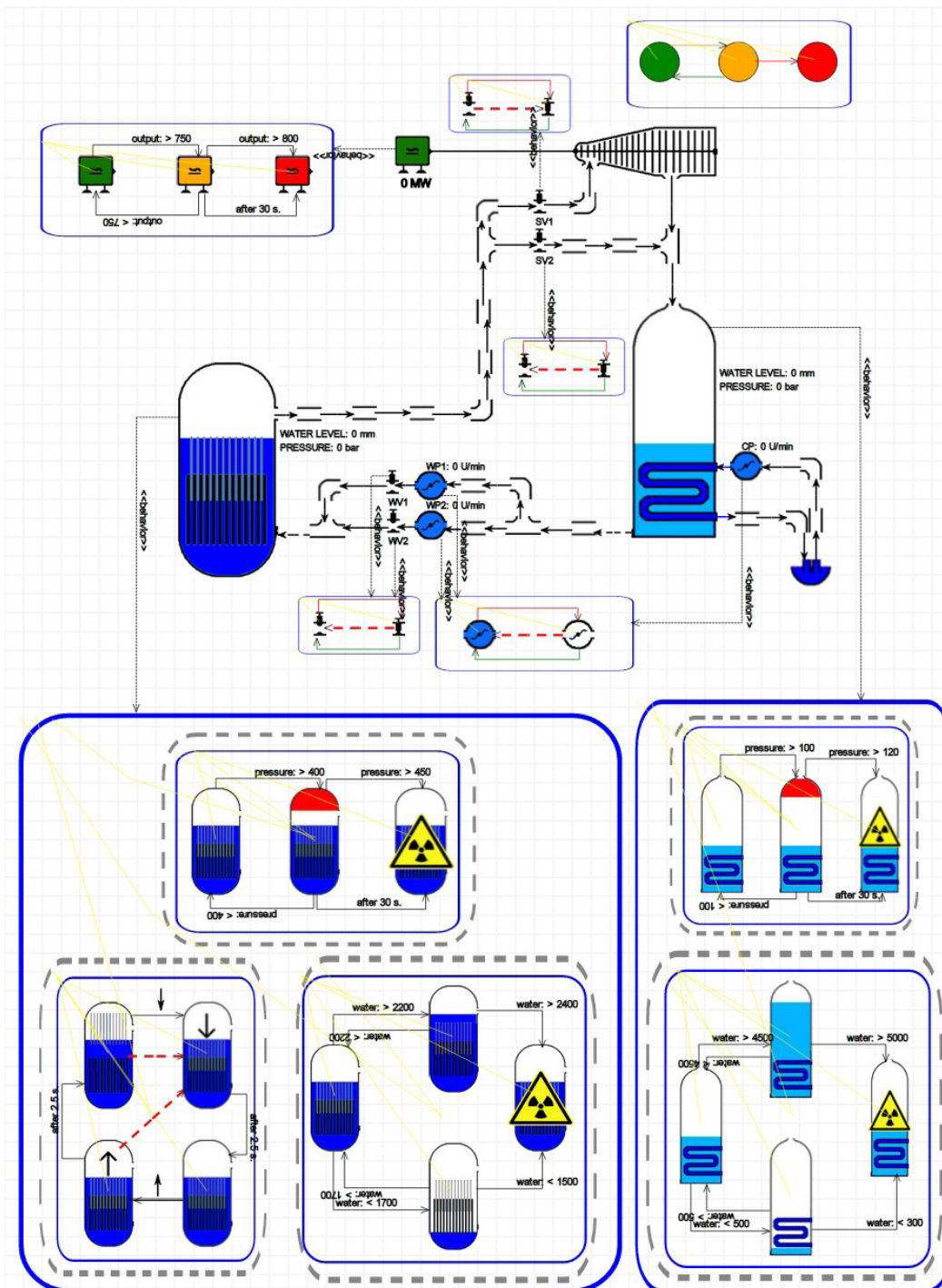
# (domain-specific) "linguistic" Modelling Language Engineering



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 16<sup>th</sup> 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](https://www.youtube.com/watch?feature=player_detailpage&v=RYtea2BiQ98)



A close-up photograph of a glass, possibly a shot glass, with the words "MODEL" and "EVERYTHING!" etched on it. "MODEL" is in a larger, green font, and "EVERYTHING!" is in a smaller, blue font. The glass is partially filled with a clear liquid, and the background is a warm, brownish-gold color. The text is centered on the glass.

**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**