MODEL-INTEGRATION AND CYBER PHYSICAL SYSTEMS: A SEMANTICS PERSPECTIVE

Janos Sztipanovits
with Ted Bapty, Gabor Karsai and Sandeep Neema

Institute for Software Integrated Systems
Vanderbilt University
Email: janos.sztipanovits@vanderbilt.edu

FM 2011
Limerick, Ireland
22 June 2011

About the Topic

CPS is a rapidly emerging, cross-disciplinary field with well-understood and urgent need for **formal methods** driven by challenges in

- model-based design
- system verification and
- manufacturing

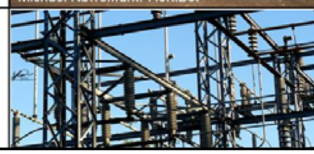
Overview

- Cyber-Physical Systems (CPS)
  - CPS and Domain Specific Modeling Languages
  - Model Integration Challenge
- Formal Semantics of DSMLs
  - Structural Semantics
  - Behavioral Semantics
- Practical Use of Formal Semantics
  - Addressing Horizontal Heterogeneity
  - Addressing Vertical Heterogeneity
- Summary

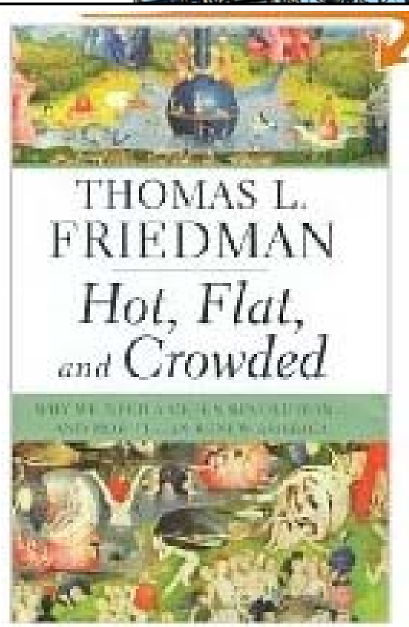
Overview

- Cyber-Physical Systems (CPS)
  - CPS and Domain Specific Modeling Languages
  - Model Integration Challenge
- Formal Semantics of DSMLs
  - Structural Semantics
  - Behavioral Semantics
- Practical Use of Formal Semantics
  - Addressing Horizontal Heterogeneity
  - Addressing Vertical Heterogeneity
- Summary

CPS is About Engineered Systems

Sectors	Opportunities	
Health and Biomedical	In-home healthcare delivery. More capable biomedical devices for measuring health. New prosthetics for use within and outside the body. Networked biomedical systems that increase automation and extend the biomedical device beyond the body.	 Goldman: Operating Rooms of the Future
Agriculture	Energy efficient technologies. Increased automation. Closed-loop bioengineering processes. Resource and environmental impact optimization. Improved safety of food products.	 Michael Nørremark: HortiBot
Smart Grid	Highway systems that allow traffic to become denser while also operating more safely. A national power grid that is more reliable and efficient.	

Sectors	Goals	
Aerospace	• Aircraft that fly faster and further on less energy. • Air traffic control systems that make more efficient use of airspace.	
Automotive	• Automobiles that are more capable and safer but use less energy. • Highways that are safe, higher throughput and energy efficient.	
Defense	• Fleets of autonomous, robotic vehicles • More capable defense systems • Integrated, maneuverable, coordinated, energy efficient • Resilient to cyber attacks	



Energy Internet: When IT Meets ET

Known Drivers of CPS

- Networking and Information Technology (NIT) have been increasingly used as *universal system integrator* in human – scale and societal – scale systems
- Functionality and salient system characteristics emerge through the interaction of *networked physical and computational objects*
- Engineered products turn into **Cyber-Physical Systems (CPS)**: networked interaction of physical and computational processes

The Good News...

Networking and computing delivers precision and flexibility in **interaction** and **coordination**

Computing/Communication

- Rich time models
- Precise interactions across highly extended spatial/temporal dimension
- Flexible, dynamic communication mechanisms
- Precise time-variant, nonlinear behavior
- Introspection, learning, reasoning

→

Integrated CPS

- Elaborate coordination of physical processes
- Hugely increased system size with controllable, stable behavior
- Dynamic, adaptive architectures
- Adaptive, autonomic systems
- Self monitoring, self-healing system architectures and better safety/security guarantees.

...and the Challenges

Fusing networking and computing with physical processes brings new unsolved problems

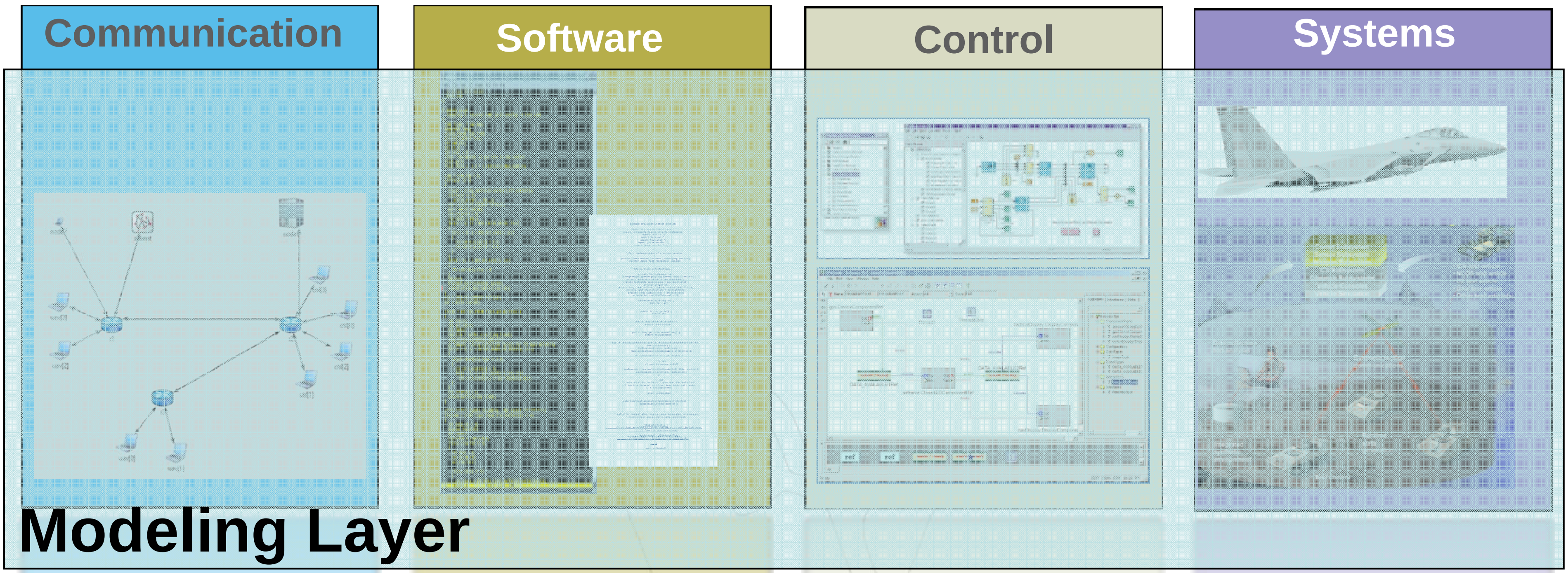
Computing/Communication

- Cyber vulnerability
- New type of interactions across highly extended spatial/temporal dimension
- Flexible, dynamic communication mechanisms
- Precise time-variant, nonlinear behavior
- Introspection, learning, reasoning

Integrated CPS

- Physical behavior of systems can be manipulated
- Lack of composition theories for heterogeneous systems: much unsolved problems
- Vastly increased complexity and emergent behaviors
- Lack of theoretical foundations for CPS dynamics
- Verification, certification, predictability has fundamentally new challenges.

Foundation for Convergence: Model-Based Design



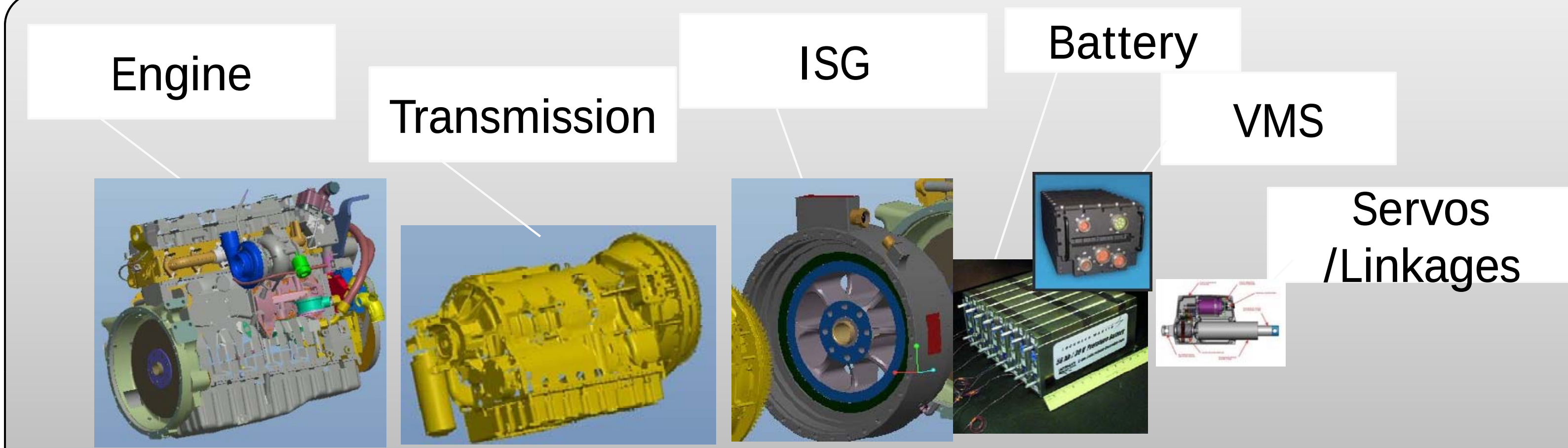
- **Systems Engineering**: Operation research, Reliability, Requirement spec.,..
- **Control Engineering**: Foundation of system theory: Linear, Nonlinear, …
- **Software Engineering**: Formal methods, Model-based SE, RT software, ..
- **Communication Engineering:** Information theory, Layered protocols, …

(Re)-convergence of Systems, Control, Software, Communication Engineering

Overview

- Cyber-Physical Systems (CPS)
  - CPS and Domain Specific Modeling Languages
  - Model Integration Challenge
- Formal Semantics of DSMLs
  - Structural Semantics
  - Behavioral Semantics
- Practical Use of Formal Semantics
  - Addressing Horizontal Heterogeneity
  - Addressing Vertical Heterogeneity
- Summary

Components of a CPS

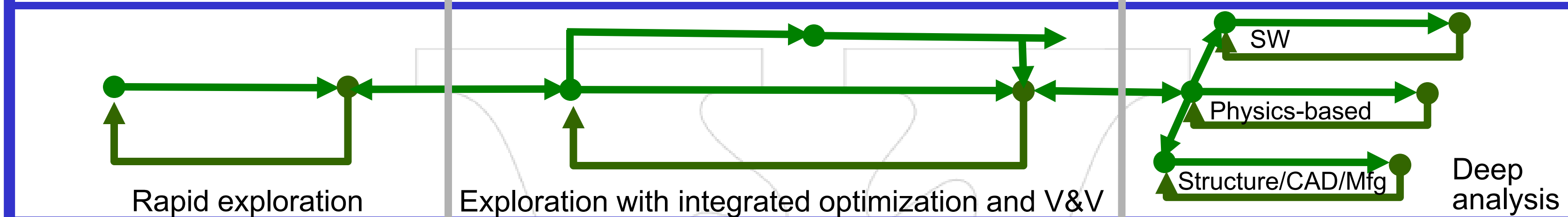


Components span:
- Multiple physics
- Multiple domains
- Multiple tools

- Physical
  - Functional: implements some function in the design
  - Interconnect: acts as the facilitators for physical interactions
- Cyber
  - Computation and communication that implements some function
  - Requires a physical platform to run/to communicate
- Cyber-Physical
  - Physical with deeply embedded computing and communication

CPS Design Flow Requires Model Integration

Architecture Design	Integrated Multi-physics/Cyber Design	Detailed Design
Modeling Exploration	Modeling Simulation V&V	Modeling Analysis



Architecture Design

- Design Space + Constraint Modeling
- Architecture Modeling
- Low-Res Component Modeling

Integrated Multi-physics/Cyber Design

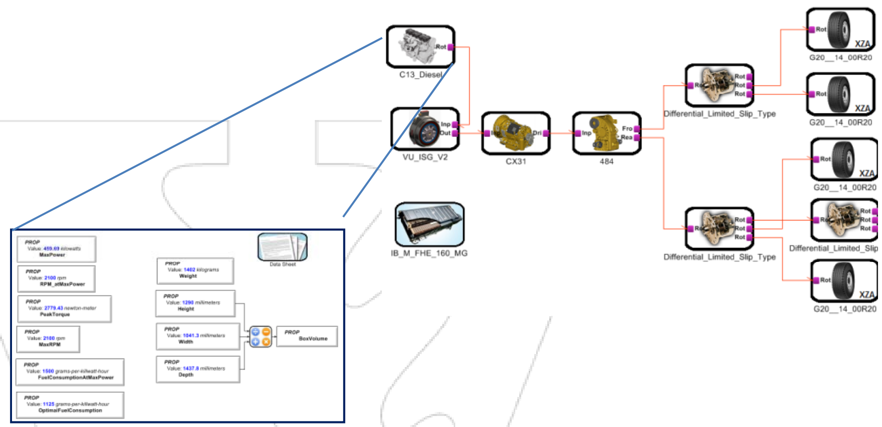
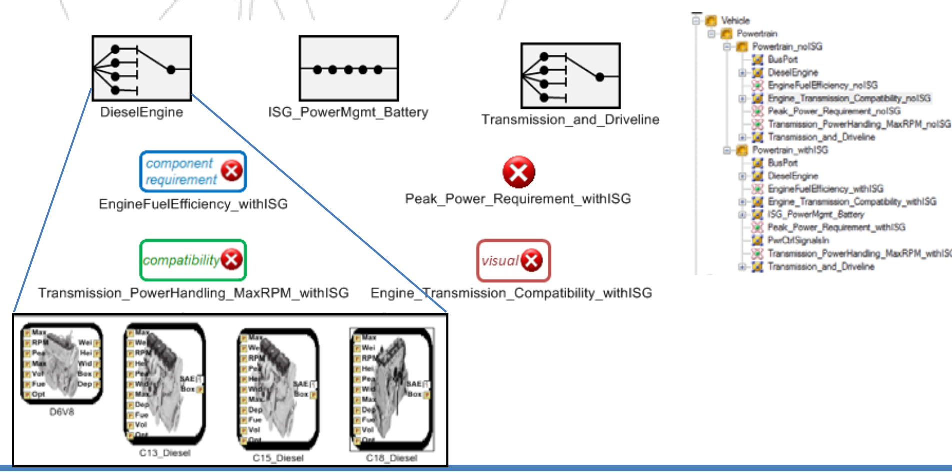
- Design Space + Constraint Modeling
- Architecture Modeling
- Dynamics Modeling
- Computational Behavior Modeling
- CAD/Thermal Modeling
- Manufacturing Modeling

Detailed Design

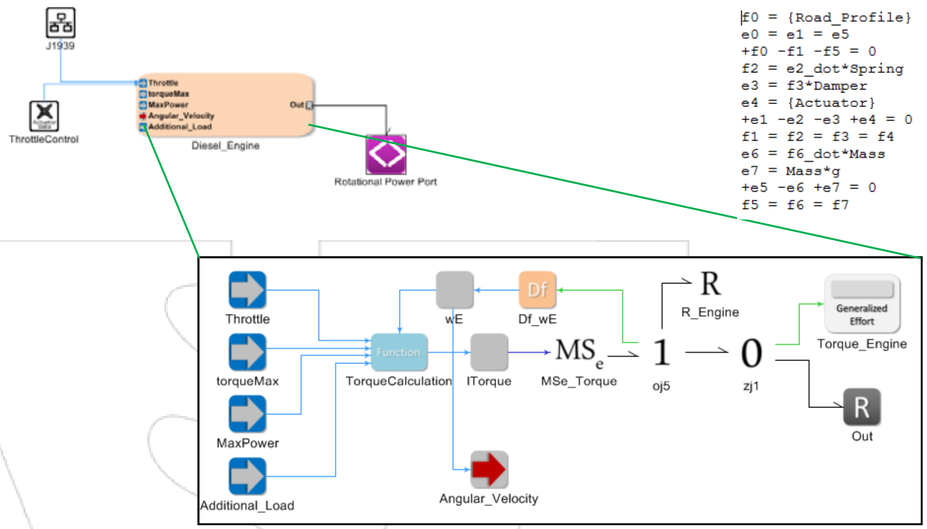
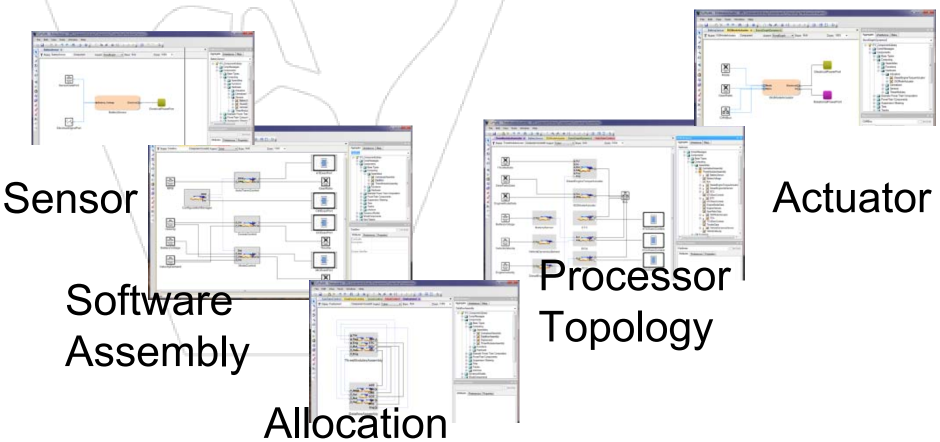
- Architecture Modeling
- Dynamics, RT Software, CAD, Thermal, …
- Detailed Domain Modeling

Domain Specific Modeling Languages

Example: Architecture Modeling

Sublanguage / Capability	Formalism, Language Constructs, Examples	Usage
Architecture Modeling	Hierarchical Module Interconnect - Components - Interfaces - Interconnects - Parameters - Properties 	Systems Architect - Explore Design Space - Derive Candidate Designs
Design Space Modeling	Hierarchically Layered Parametric Alternatives - Alternatives/ Options - Parameters - Constraints 	Systems Architect - Define Design Space - Define Constraint

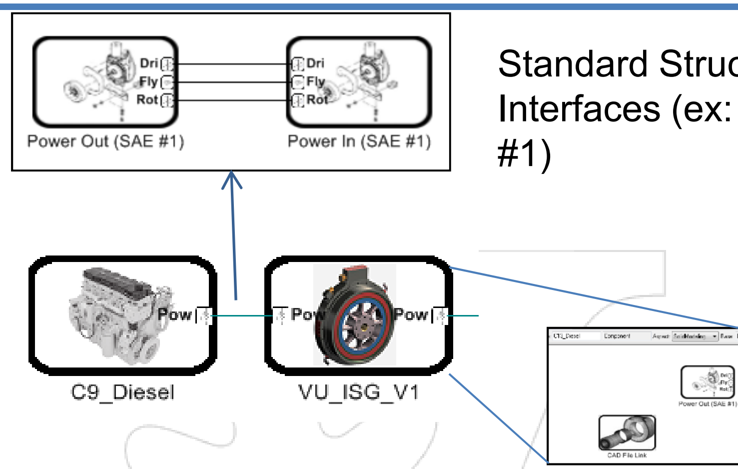
Example: Dynamics Modeling

Physical Dynamics Modeling	**Hybrid Bond Graphs** - Efforts, Flows, - Sources, Capacitance, Inductance, - Resistance, - Transformers Gyrators,  	Component Engineer - model dynamics with Hybrid Bond Graphs System Engineers - Compose system dynamics
Computational Dynamics Modeling	**Dataflow + Stateflow + TT Schedule** - Interaction with Physical Components - Cyber Components - Processing Components  	Domain Engineers - design controller System Engineers - Processor allocate - Platform Effects

Example: Physical Structure and Manufacturing Modeling

Solid Modeling (CAD / Geometry)

Structural Interfaces
- Defined with Peer Roles:
  - Axis
  - Point
  - Surface
- CAD Links



Standard Structural Interfaces (ex: SAE #1)

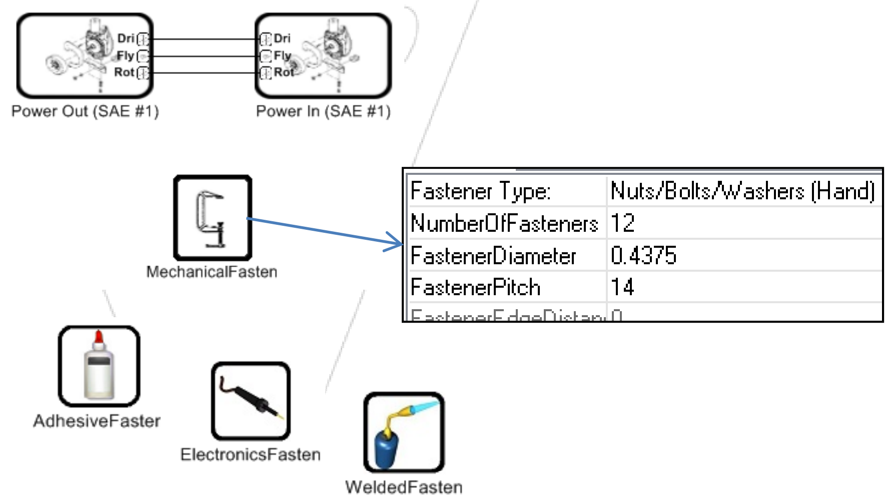
Component Engineer
- Defines Structural Interface

System Engineer
- Defines Architecture

Manufacturing Modeling

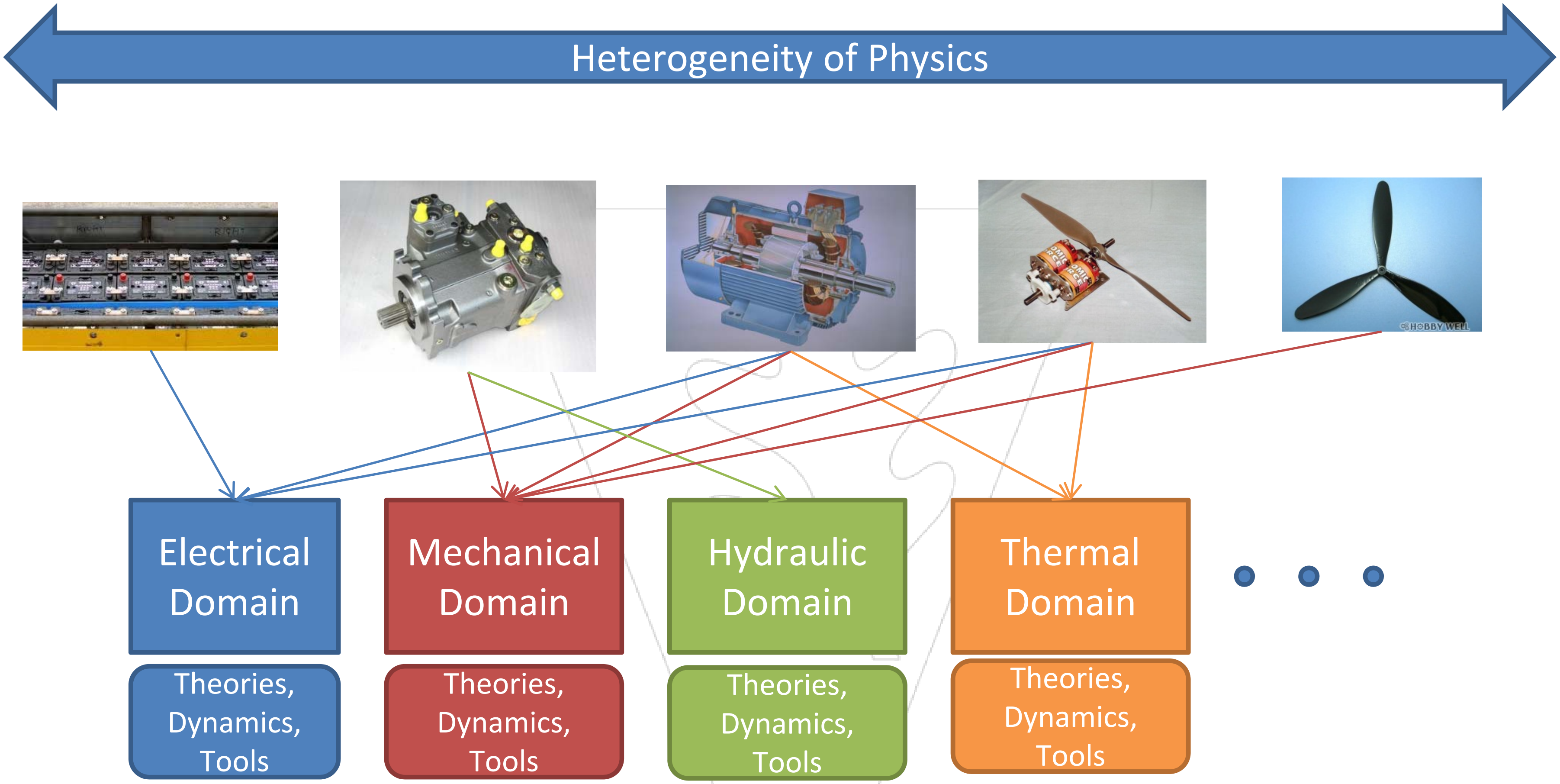
Component Manuf. Cost
- Make
  - Material
  - Fab Proc
  - Complxity
  - Shape/Wt
- OTS: Cost/unit

Structural Interfaces
- Fastener Types, Num# …



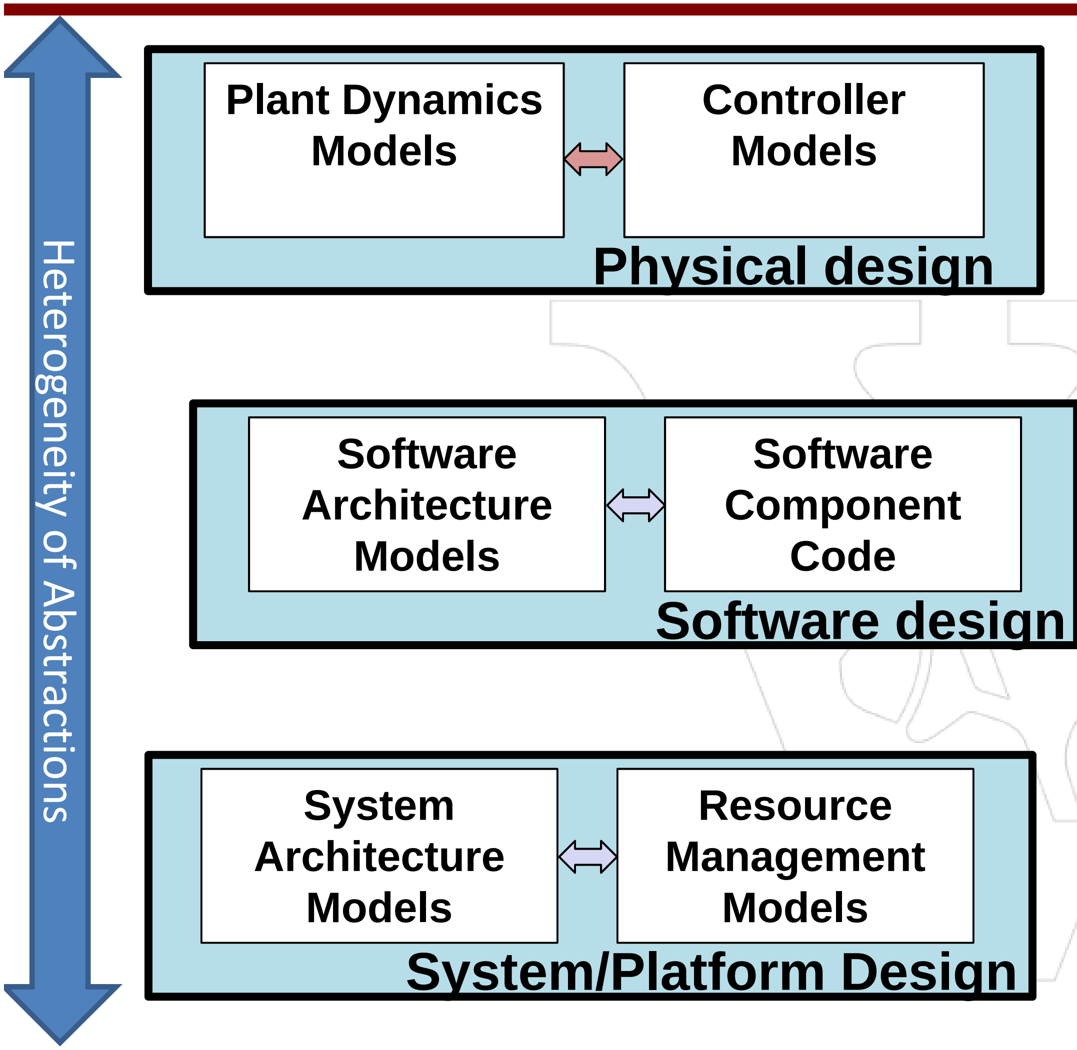
Component Engineer
- Defines Part Cost
- Defines Structural Interface, Fastener

Model Integration Challenge: Physics



Physical components are involved in multiple physical interactions (multi-physics)
Challenge: How to compose multi-models for heterogeneous physical components

Model Integration Challenge: Abstraction Layers



Dynamics: $B(t) = \kappa_p(B_1(t), \ldots, B_j(t))$

- *Properties*: stability, safety, performance
- *Abstractions*: continuous time, functions, signals, flows,…

Software : $B(i) = \kappa_c(B_1(i), \ldots, B_k(i))$

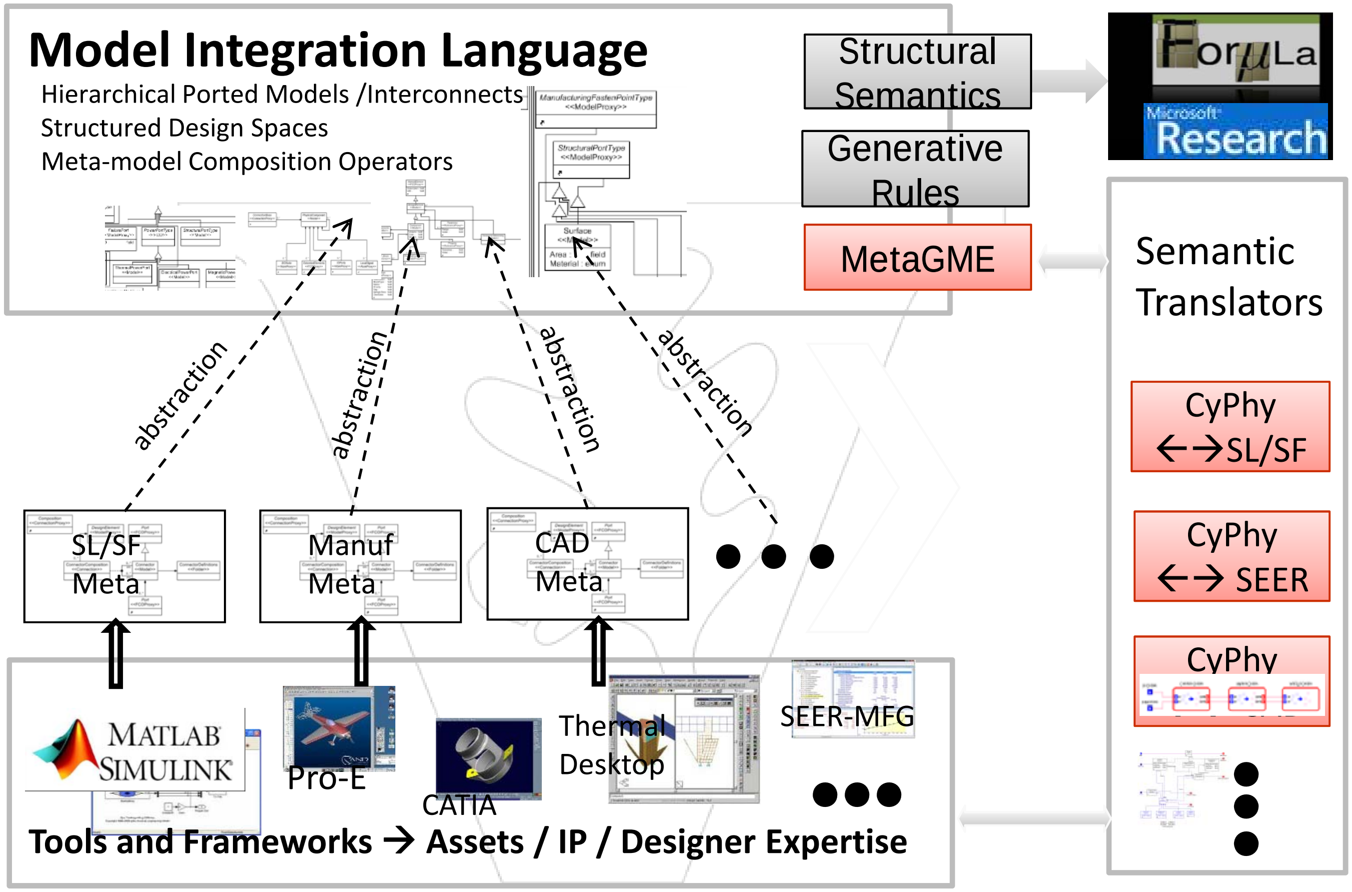
- *Properties*: deadlock, invariants, security,…
- *Abstractions*: logical-time, concurrency, atomicity, ideal communication,..

Systems : $B(t_j) = \kappa_p(B_1(t_i), \ldots, B_k(t_i))$

- *Properties*: timing, power, security, fault tolerance
- *Abstractions*: discrete-time, delays, resources, scheduling,

Cyber-physical components are modeled using multiple abstraction layers
Challenge: How to compose abstraction layers in heterogeneous CPS components?

A Pragmatic Approach: Model Integration Language



Impact: Open Language Engineering Environment → Adaptability of Process/Design Flow → Accommodate New Tools/Frameworks , Accommodate New Languages

Overview

- Cyber-Physical Systems (CPS)
  - CPS and Domain Specific Modeling Languages
  - Model Integration Challenge
- Formal Semantics of DSMLs
  - Structural Semantics
  - Behavioral Semantics
- Practical Use of Formal Semantics
  - Addressing Horizontal Heterogeneity
  - Addressing Vertical Heterogeneity
- Summary

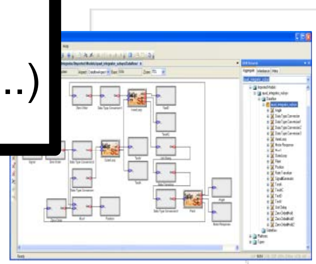
What Do We Expect From Formal Semantics?

- Specify
- Unambiguate
- **Compute**

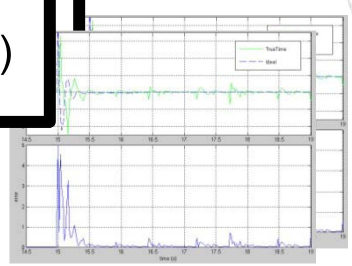
DSML Semantics

Models represent:

Structure
(logical, physical,..)



Behavior
(cont. discrete,..)



Modeling Language Semantics:

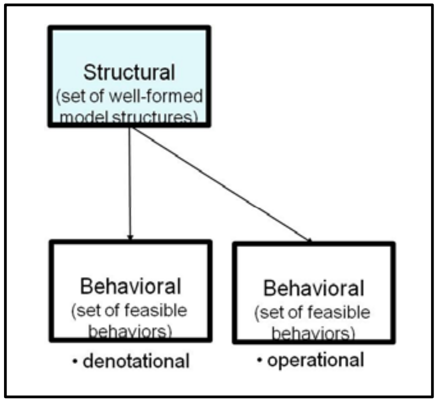
Structural
(set of well-formed model structures)

Mathematical Domains:
- graphs
- term algebra + logic

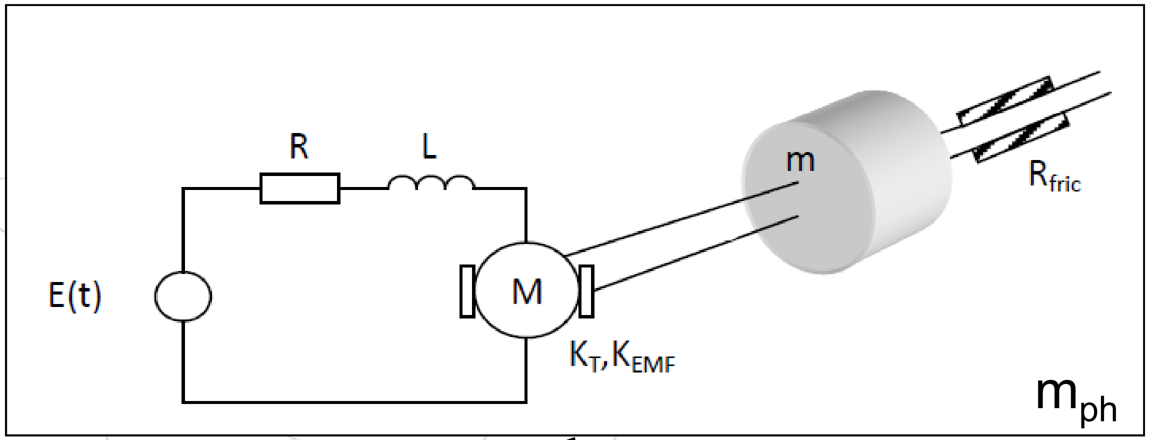
Behavioral
(set of feasible behaviors)
- denotational

Behavioral
(set of feasible behaviors)
- operational

Example 1/2

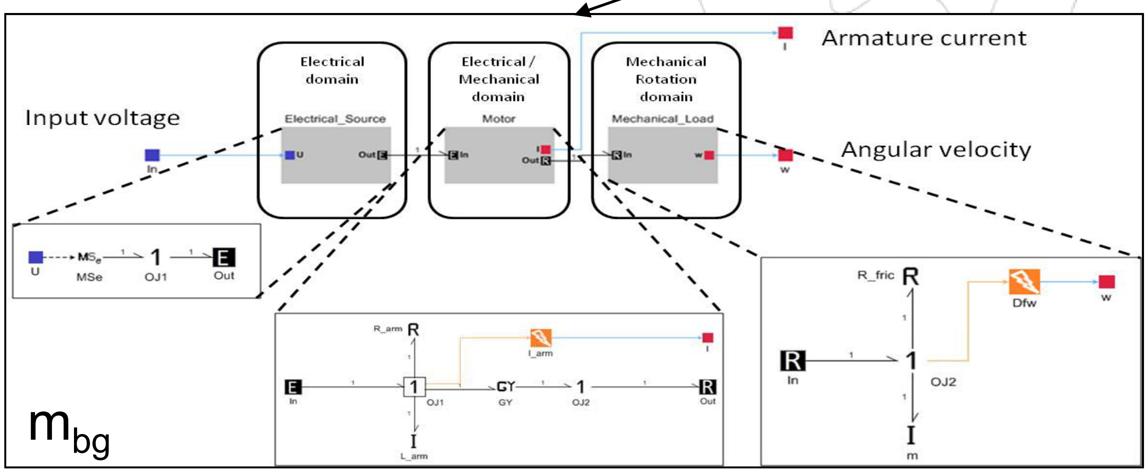


Physical Structure (components and terminals)



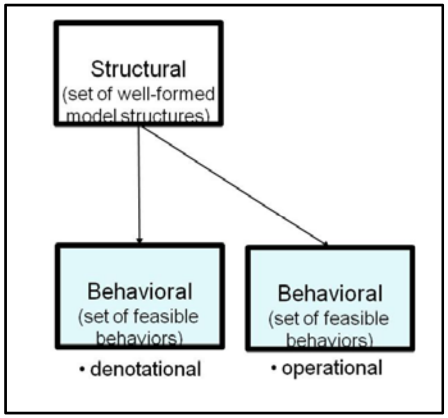
Transformation:

$$m_{bg} = T(m_{ph})$$



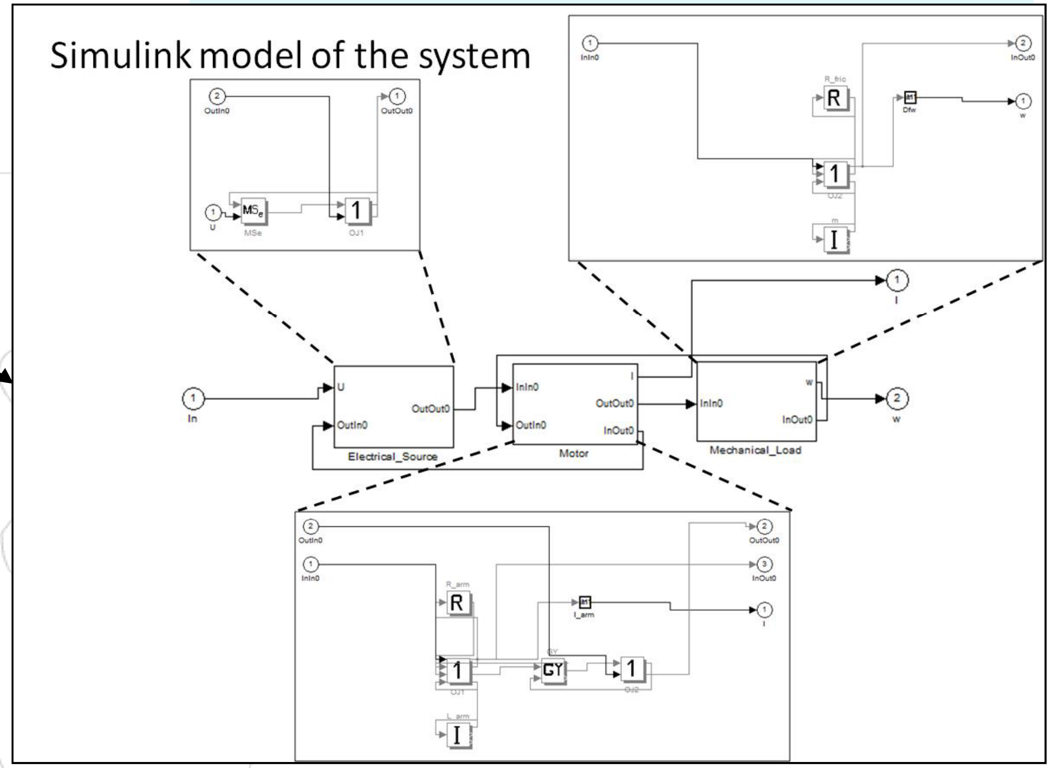
Bond Graph model (energy flows)

Example 2 / 2



operational: simulated trajectories

$m_{sl} = T(m_{bg})$



$m_{de} = T(m_{bg})$

$OJ1: e_1 - e_2 - e_3 - e_4 = 0$

$OJ1: f_1 = f_2 = f_3 = f_4$

$Se: e_1 = E(t)$

$R_{arm}: e_2 = R_{arm} * f_2$

$L_{arm}: e_3 = L_{arm} * \dot{f}_3$

$GY: e_5 = f_1 * K_T$

$GY: e_4 = f_2 * K_{EMF}$

$OJ2: e_5 - e_6 - e_7 = 0$

$OJ2: f_5 = f_6 = f_7$

$R_{fric}: e_6 = R_{fric} * f_6$

$m: e_7 = m * \dot{f}_2$

denotational: mathematical equations

Modeling Language Semantics Has Extensive Research History

- Broy, Rumpe '1997
- Harel '1998
- Harel and Rumpe '2000
- Tony Clark, Stuart Kent, Bernhard Rumpe, Kevin Lano, Jean-Michel Bruel and Ana Moreira - Precise UML Group
- Edward Lee, Alberto Sangiovanni-Vincentelli '2004
- Joseph Sifakis '2005
- ...

Overview

- Cyber-Physical Systems (CPS)
  - CPS and Domain Specific Modeling Languages
  - Model Integration Challenge
- Formal Semantics of DSMLs
  - Structural Semantics
  - Behavioral Semantics
- Practical Use of Formal Semantics
  - Addressing Horizontal Heterogeneity
  - Addressing Vertical Heterogeneity
- Summary

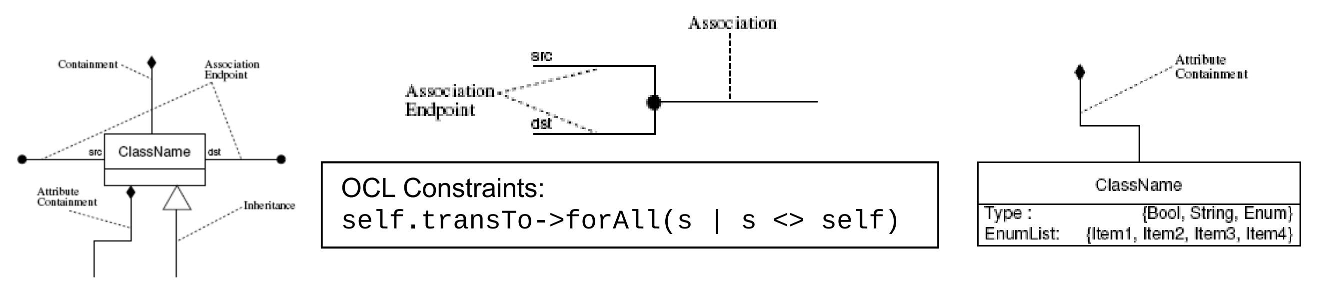
Specification of Domain-Specific Modeling Languages

Abstract syntax of DSML-s are defined by metamodels.

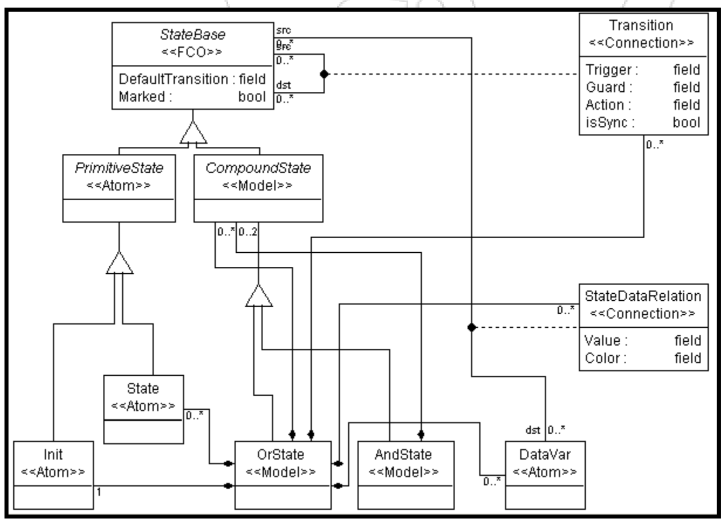
A metamodeling language is one of the DSML-s.

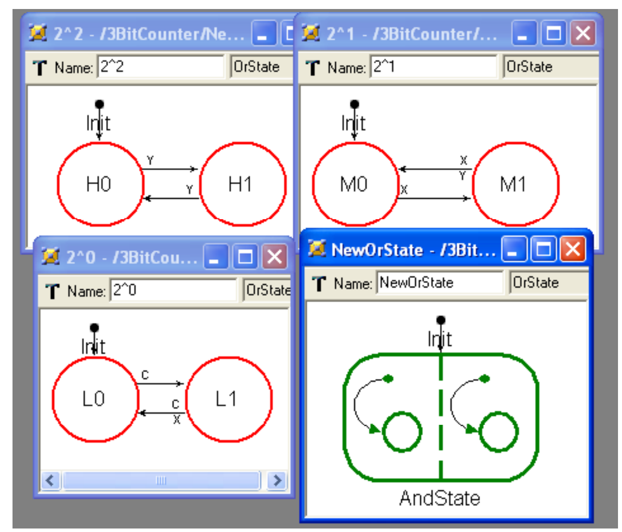
Semantics of metamodeling languages: structural semantics.

<u>Key Concept</u>: Modeling languages define a set *of well- formed models* and their *interpretations*. The interpretations are mappings from one domain to another domain.



Basic metamodeling notation: UML Class Diagram + OCL





MetaGME metamodel of simple statecharts

Model-editor generated from metamodel

Formalization of Structural Semantics

$$L = \langle Y, R_Y, C, ([\ ]_i)_{i \in J} \rangle$$
$$D(Y, C) = \{ r \in R_Y \mid r \models C \}$$
$$[\ ] : R_Y \mapsto R_{Y'}$$

Y: set of concepts,
R_Y: set of possible model realizations
C: set of constraints over R_Y
$D(Y,C)$: domain of well-formed models
[ ]: interpretations

Jackson & Sz. '2007
Jackson, Schulte, Sz. '2008
Jackson & Sz. '2009

Key Concept: DSML syntax is understood as a constraint system that identifies behaviorally meaningful models. ***Structural semantics provides mathematical formalism for interpreting models as well-formed structures*.**

Structural Semantics defines modeling domains using term algebra extended with Logic Programming. This mathematical structure is the semantic domain of metamodeling languages.

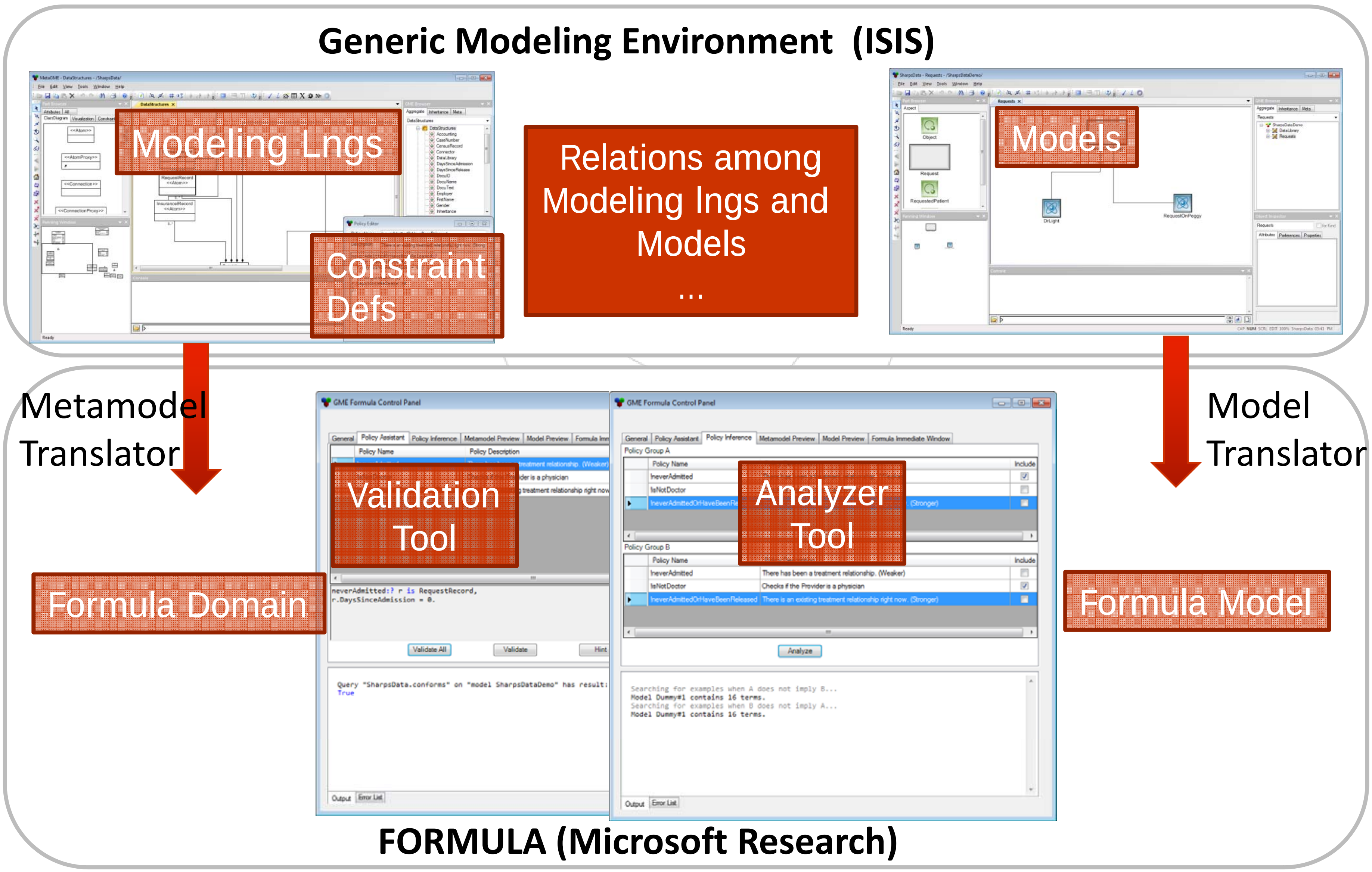
Use of structural semantics:

- Conformance testing: $x \in D$
- Non-emptiness checking: $D(Y, C) \neq \{nil\}$
- DSML composing: $D_1 * D_2 | D_1 + D_2 | D' \text{ includes } D | \ldots$
- Model finding: $S = \{ s \in D | s \models P \}$
- Transforming: $m' = T(m); m' \in X; m \in Y$

Microsoft Research Tool: FORMULA

- Fragment of LP is equivalent to full first-order logic
- Provide semantic domain for model transformations.

GME-FORMULA Tool Interfaces



Ongoing Work

- FORMULA (Schulte, Jackson et al, MSR) - A tool suite for building models and analyzing their properties. Co-developed with the European Microsoft Innovation Center (EMIC), Aachen, Germany
- GME-FORMULA translator – Extension of the MIC tool suite (VU-ISIS in cooperation with MSR)
- Analysis tools – Domain and Model Equivalence, Domain Composition, Model Completion (VU-ISIS in cooperation with MSR)

Overview

- Cyber-Physical Systems (CPS)
  - CPS and Domain Specific Modeling Languages
  - Model Integration Challenge
- Formal Semantics of DSMLs
  - Structural Semantics
  - ⇨ Behavioral Semantics
- Practical Use of Formal Semantics
  - Addressing Horizontal Heterogeneity
  - Addressing Vertical Heterogeneity
- Summary

Behavioral Semantics

- Given a DSML

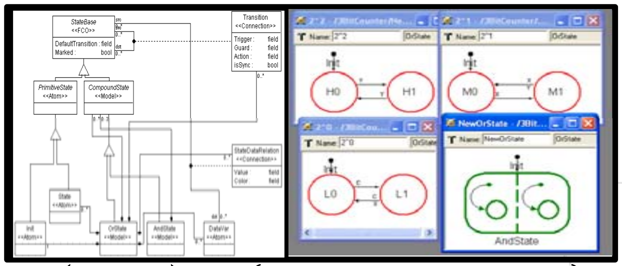
$$L = \langle Y, R_Y, C, (\llbracket\ \rrbracket_i)_{i \in J} \rangle$$

$$D(Y, C) = \{ r \in R_Y \mid r \models C \}$$

$$\llbracket\ \rrbracket : R_Y \mapsto R_{Y'}$$

- Behavioral semantics will be defined by specifying the transformation between the DSML and a modeling language with behavioral semantics.

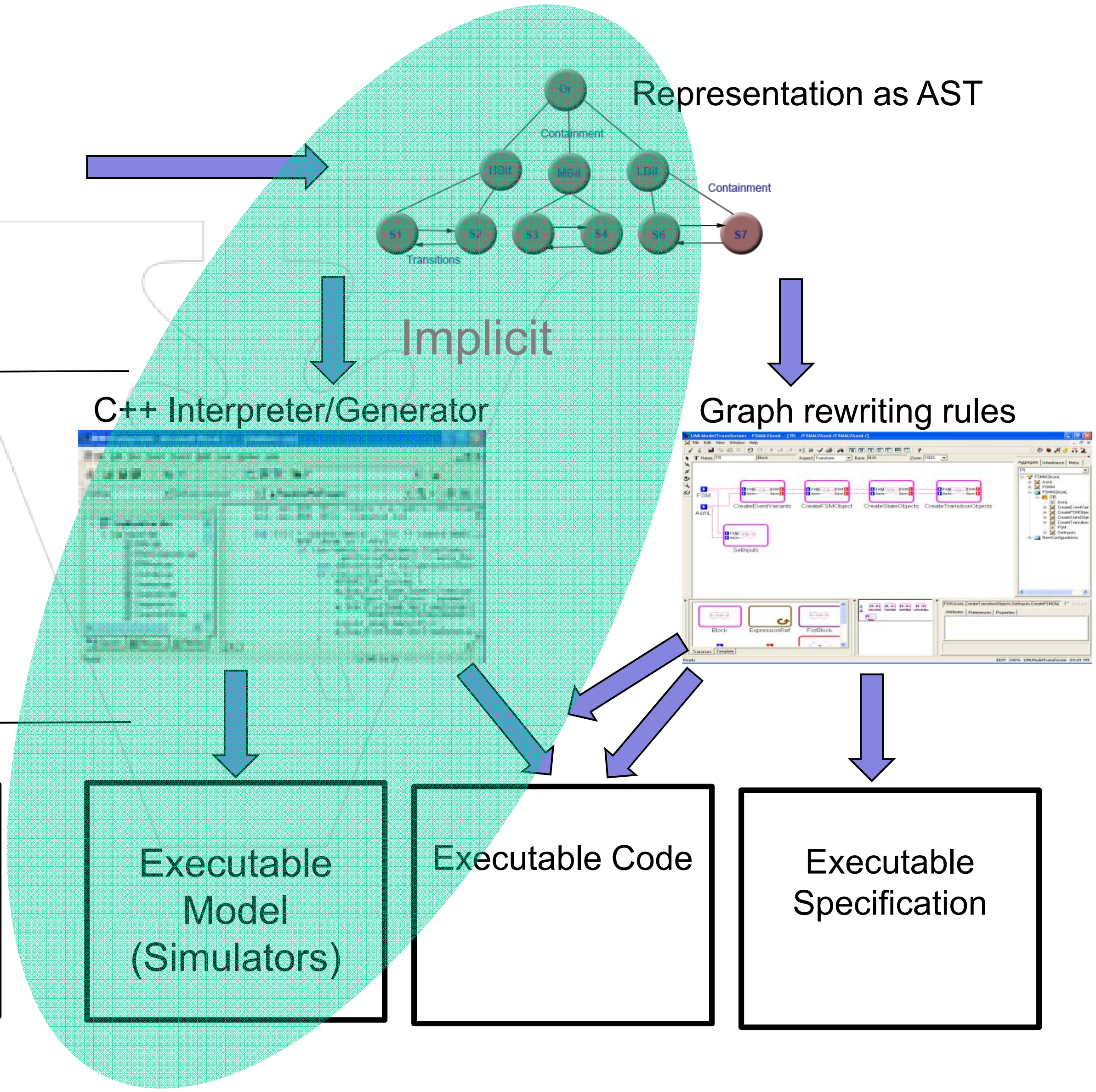
Implicit Methods for Specifying Behavioral Semantics



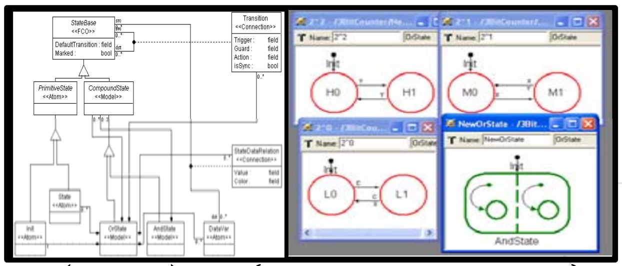
$$D(Y,C)=\{r \in R_Y \mid r \models C\}$$

$$[\;]: R_Y \mapsto R_{Y'}$$

$$D(Y',C')=\{r \in R_{Y'} \mid r \models C'\}$$
$$[\;]: R_{Y'} \mapsto R_{Y''}$$



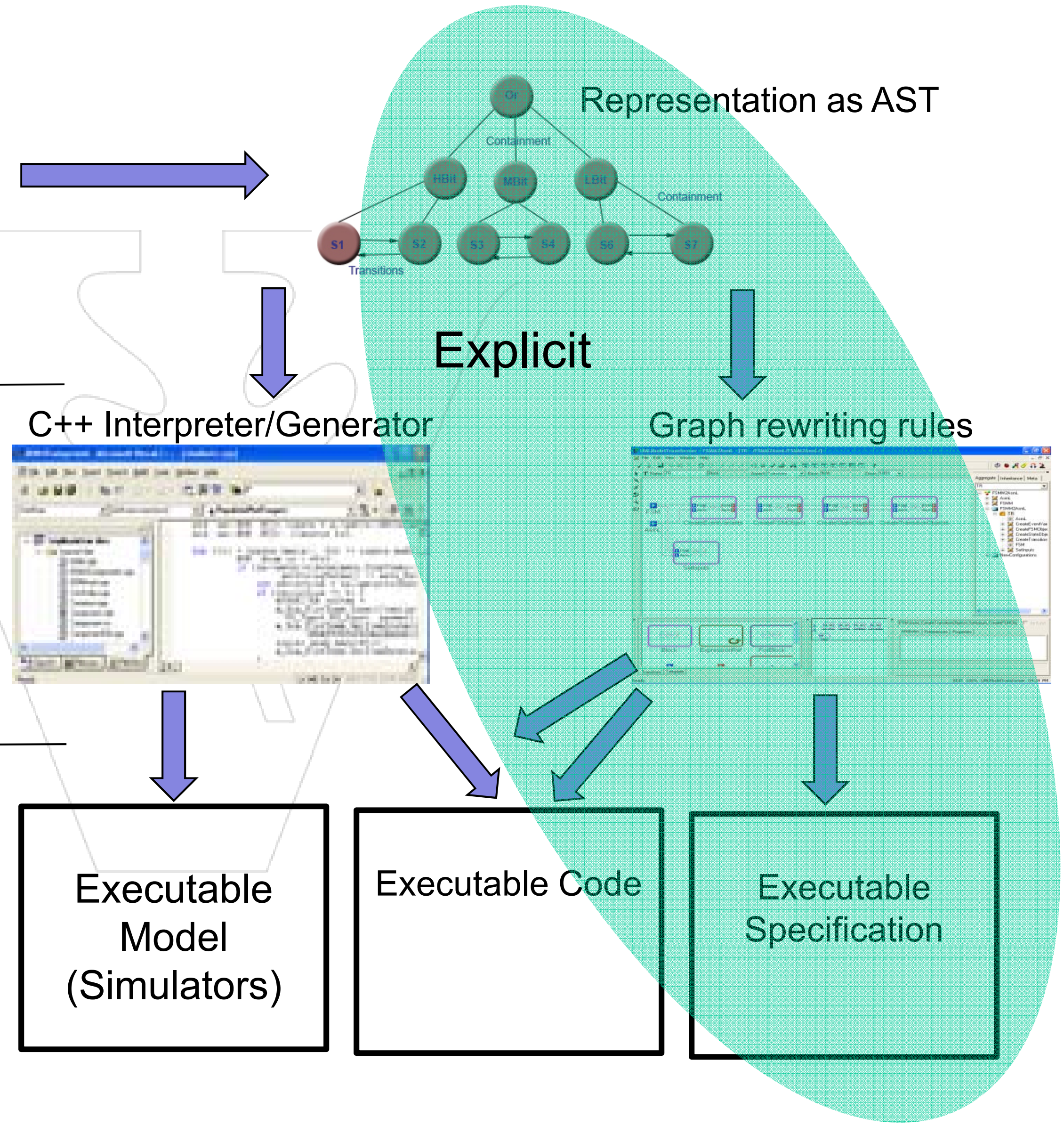
Explicit Methods for Specifying Behavioral Semantics



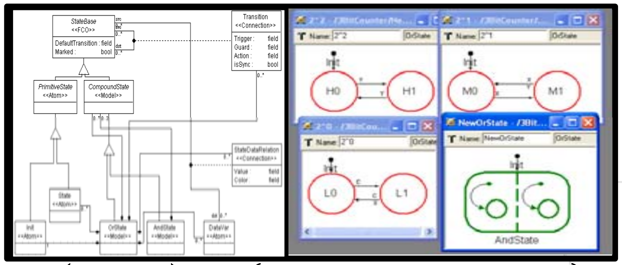
$$D(Y,C)=\{r\in R_Y \mid r\models C\}$$

$$[\;]: R_Y \mapsto R_{Y'}$$

$$D(Y',C')=\{r\in R_{Y'} \mid r\models C'\}$$
$$[\;]: R_{Y'} \mapsto R_{Y''}$$

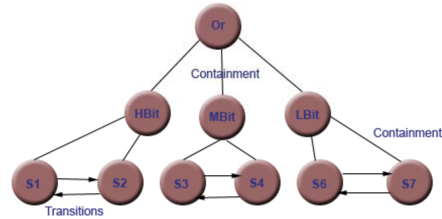


Specifying Behavioral Semantics With Semantic Anchoring



$$D(Y,C)=\{r \in R_Y \mid r \models C\}$$

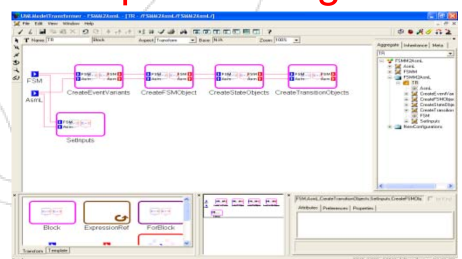
Representation as AST



MIC-UDM
MIC-GME

$$[\;]: R_Y \mapsto R_{Y'}$$

Graph rewriting rules



MIC-GReAT
(Karsai, VU-ISIS)

$$D(Y',C')=\{r \in R_{Y'} \mid r \models C'\}$$

$$[\;]: R_{Y'} \mapsto R_{Y''}$$

Abstract State Machine Formalism

```
structure Event
  eventType as String

class State
  id         as String
  initial    as Boolean
  var active as Boolean = false

class Transition
  id as String

abstract class FSM
  id as String

  abstract property states as Set of State
    get
  abstract property transitions as Set of Transition
    get
  abstract property outTransitions as Map of <State, Set of Transition>
    get
  abstract property dstState as Map of <Transition, State>
    get
  abstract property triggerEventType as Map of <Transition, String>
    get
  abstract property outputEventType as Map of <Transition, String>
```

Abstract Data Model

```
React (e as Event) as Event?
   step
     let CS as State = GetCurrentState ()
   step
     let enabledTs as Set of Transition = {t | t in outTransitions (CS) where
e.eventType = triggerEventType(t)}
   step
     if Size (enabledTs) = 1 then
       choose t in enabledTs
         step
          // WriteLine ("Execute transition: " + t.id)
           CS.active := false
         step
           dstState(t).active := true
         step
           if t in me.outputEventType then
             return Event(outputEventType(t))
           else
             return null
     else
       if Size(enabledTs) > 1 then
         error ("NON-DETERMINISM ERROR!")
       else
         return null
```

Model Interpreter

Example Specification : FSM

Abstract Data Model

```
structure Event
  eventType as String
class State
  initial    as Boolean
  var active as Boolean = false
class Transition
abstract class FSM
  abstract property states           as Set of State
    get
  abstract property transitions      as Set of Transition
    get
  abstract property outTransitions   as Map of
    <State, Set of Transition>
    get
  abstract property dstState as Map of <Transition, State>
    get
  abstract property triggerEventType as Map of
    <Transition, String>
    get
  abstract property outputEventType  as Map of
    <Transition, String>
    get
```

Interpreter

```
abstract class FSM
  Run (e as Event) as Event?
    step
      let CS as State = GetCurrentState ()
    step
      let enabledTs as Set of Transition = {t | t in
        outTransitions (CS) where e.eventType =
        triggerEventType(t)}
    step
      if Size (enabledTs) >= 1 then
        choose t in enabledTs
          step
            CS.active := false
          step
            dstState(t).active := true
          step
            if t in me.outputEventType then
              return Event(outputEventType(t))
            else
              return null
      else
        return null
```

Underlying abstract machine - ASM Language: AsmL
Yuri Gurevich, MSR

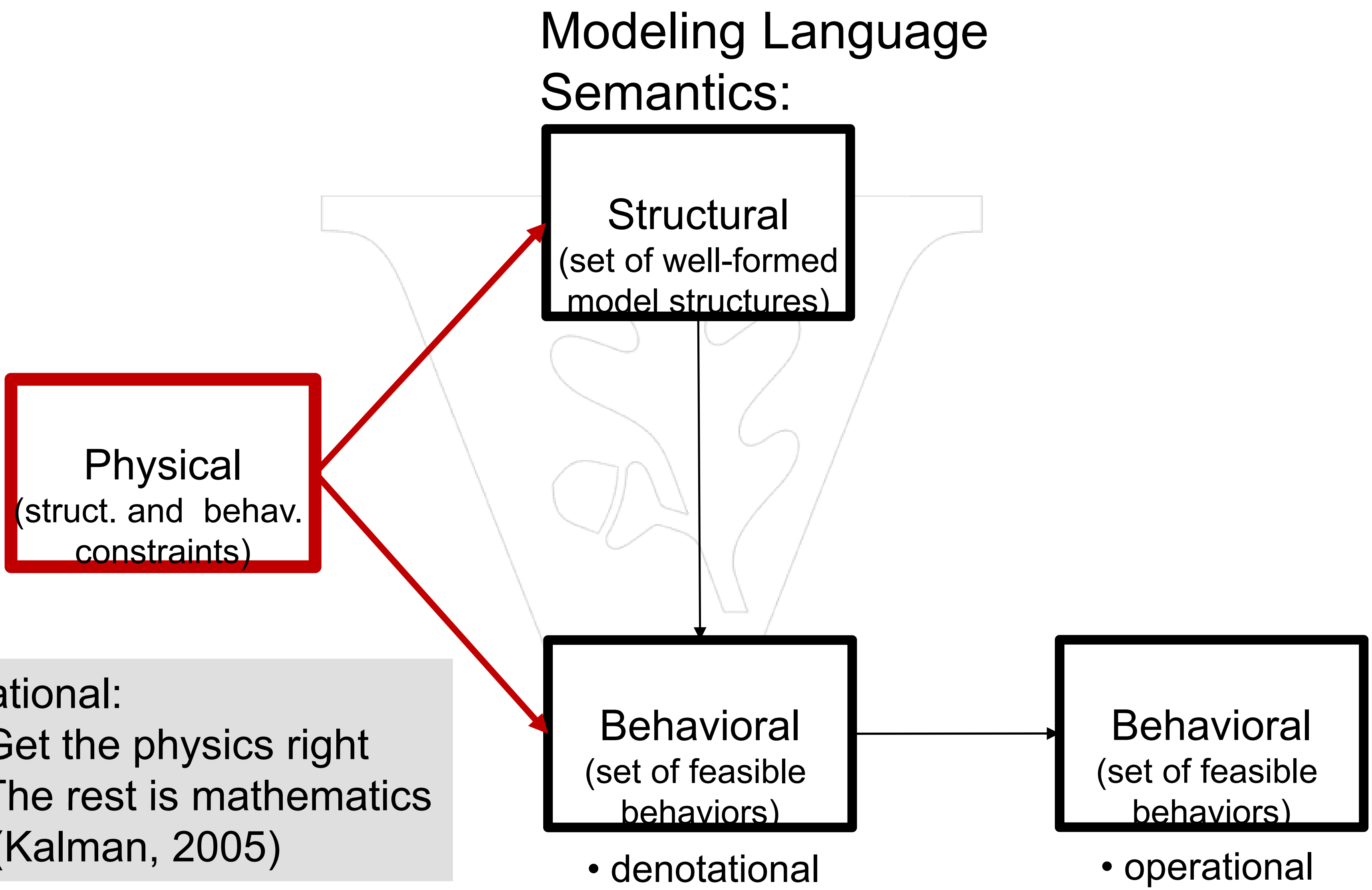
Ongoing Work

- Semantic anchoring of DSMLs using "semantic units"
- Compositional specification of semantics for heterogeneous modeling languages
- Investigating alternative frameworks (e.g. based on FORMULA)

Overview

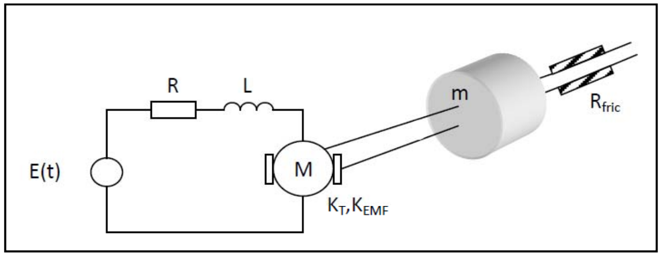
- Cyber-Physical Systems (CPS)
  - CPS and Domain Specific Modeling Languages
  - Model Integration Challenge
- Formal Semantics of DSMLs
  - Structural Semantics
  - Behavioral Semantics
- Practical Use of Formal Semantics
  - Addressing Horizontal Heterogeneity
  - Addressing Vertical Heterogeneity
- Summary

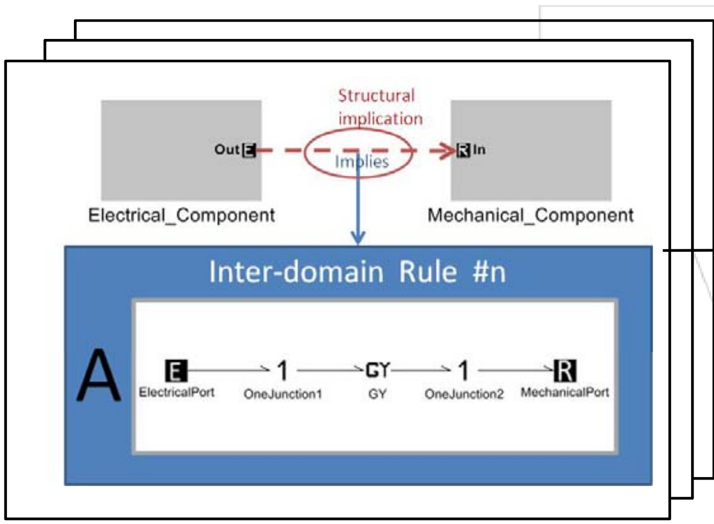
Capturing Physical Semantics

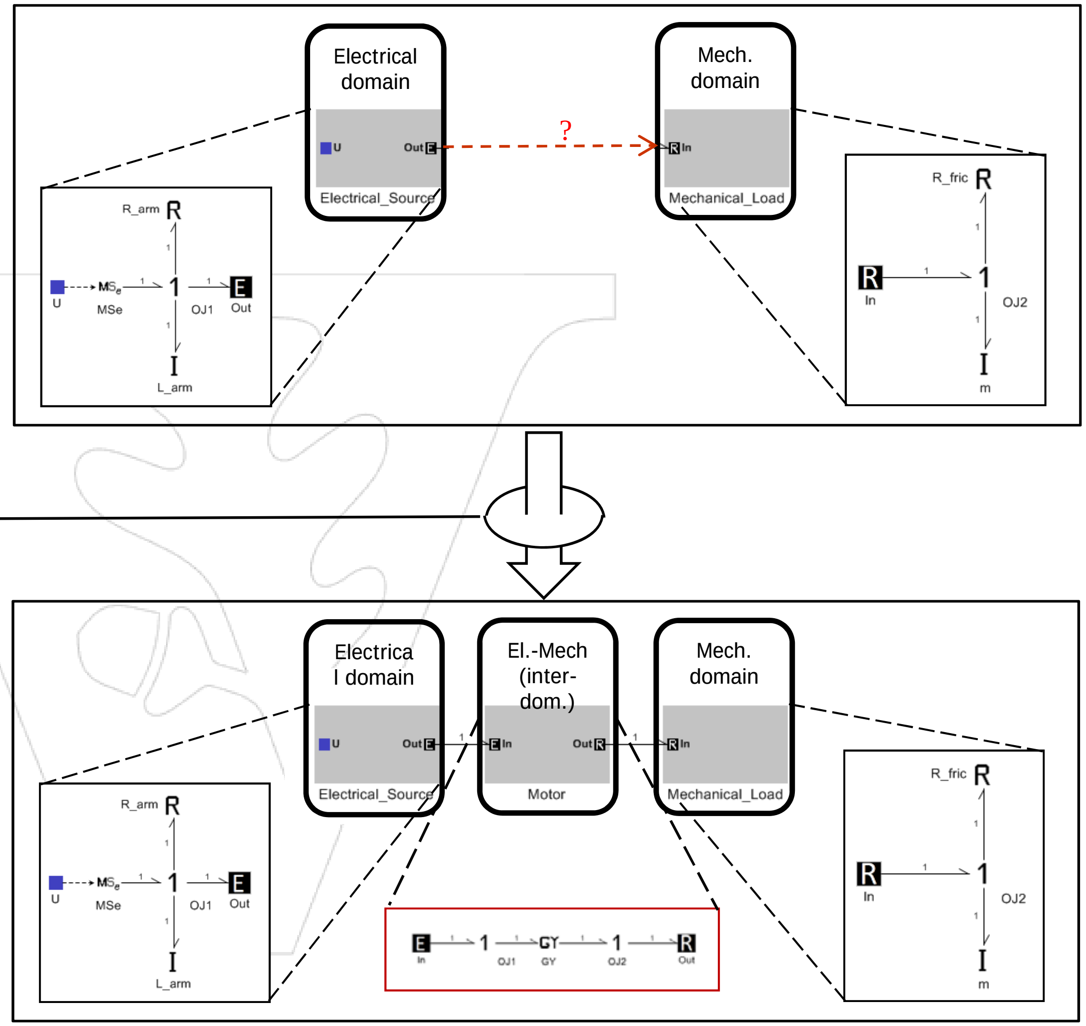


Rational:
- Get the physics right
- The rest is mathematics (Kalman, 2005)

Physical Semantics: Structural Implications 1/2

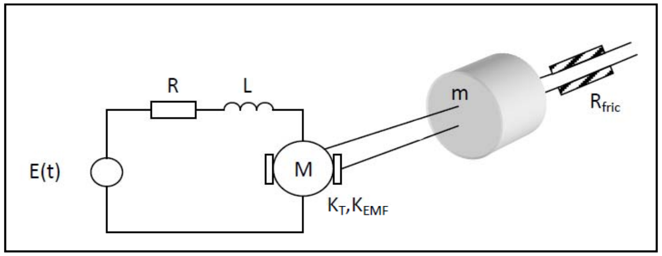




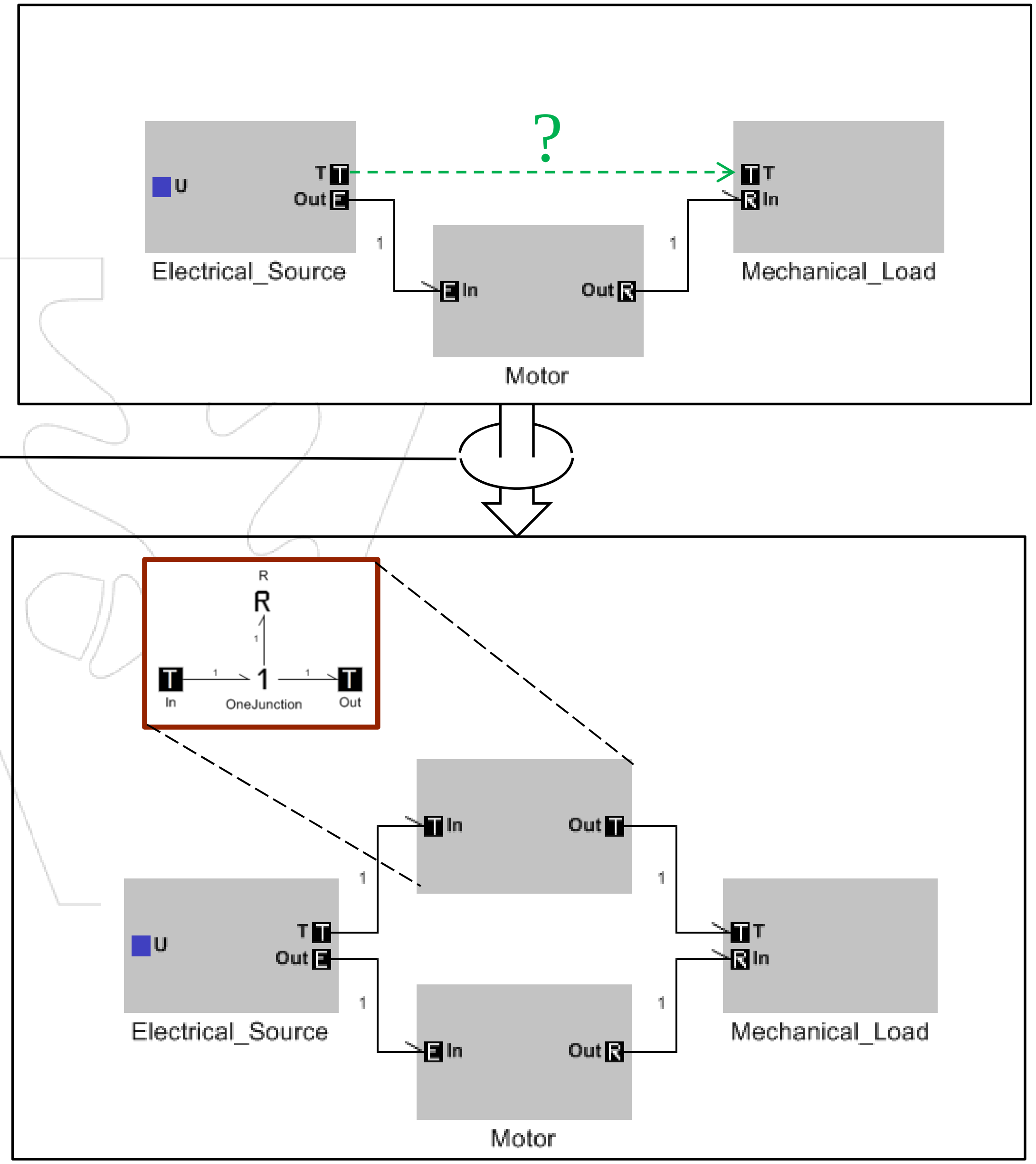


Energy is conserved at couplings between domains

Physical Semantics: Structural Implications 2/2

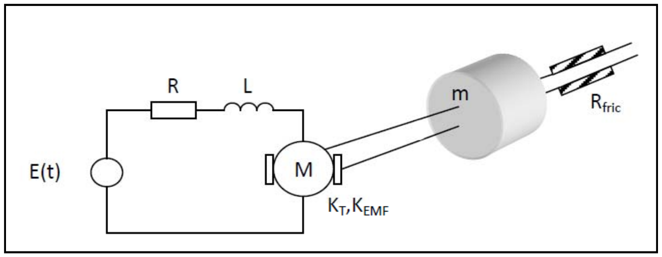


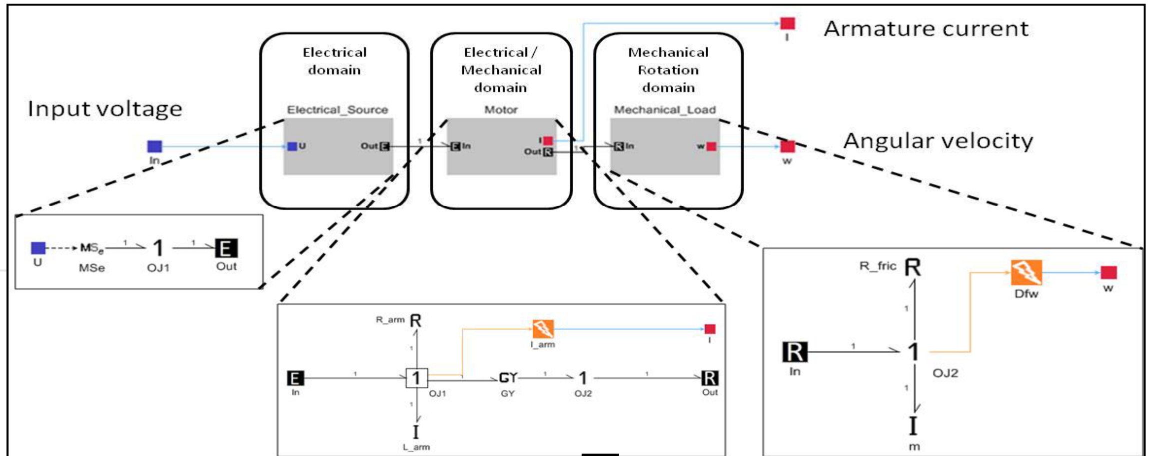
Collateral energy flow
…other rules…



Heat energy generated on dissipative elements: creates additional energy coupling

Physical Semantics: Behavioral Implications





One Junction Rule

$$\sum_i e_i = 0$$

$$f_i = f_k; i,k \in N$$

Rate of power transfer between components is balanced

$OJ1: e_1 - e_2 - e_3 - e_4 = 0$
$OJ1: f_1 = f_2 = f_3 = f_4$

$Se: e_1 = E(t)$

$R_{arm}: e_2 = R_{arm} * f_2$

$L_{arm}: e_3 = L_{arm} * \dot{f}_3$

$GY: e_5 = f_1 * K_T$
$GY: e_4 = f_2 * K_{EMF}$

$OJ2: e_5 - e_6 - e_7 = 0$
$OJ2: f_5 = f_6 = f_7$

$R_{fric}: e_6 = R_{fric} * f_6$

$m: e_7 = m * \dot{f}_2$

Denotational behavioral semantics

Physical Semantics: Ongoing Work

- Extend metamodeling language and metaprogrammable modeling tool (GME) with *generative constructs*
- Make specification of generative modeling constructs integrated with metamodeling
- Extend structural semantics and tools with dynamic constructs
- Develop rule libraries for relevant cross-physical domains **(in progress**)

Overview

- Cyber-Physical Systems (CPS)
  - CPS and Domain Specific Modeling Languages
  - Model Integration Challenge
- Formal Semantics of DSMLs
  - Structural Semantics
  - Behavioral Semantics
- Practical Use of Formal Semantics
  - Addressing Horizontal Heterogeneity
  - ⇨ Addressing Vertical Heterogeneity
- Summary

Integration Inside Abstraction Layers: Composition

Plant Dynamics Models ⟺ **Controller Models**

Physical design

Dynamics: $B(t) = \kappa_p(B_1(t),\dots,B_j(t))$

- *Properties*: stability, safety, performance
- *Abstractions*: continuous time, functions, signals, flows,…

Software Architecture Models ⟺ **Software Component Code**

Software design

Software : $B(i) = \kappa_c(B_1(i),\dots,B_k(i))$

- *Properties*: deadlock, invariants, security,…
- *Abstractions*: logical-time, concurrency, atomicity, ideal communication,..

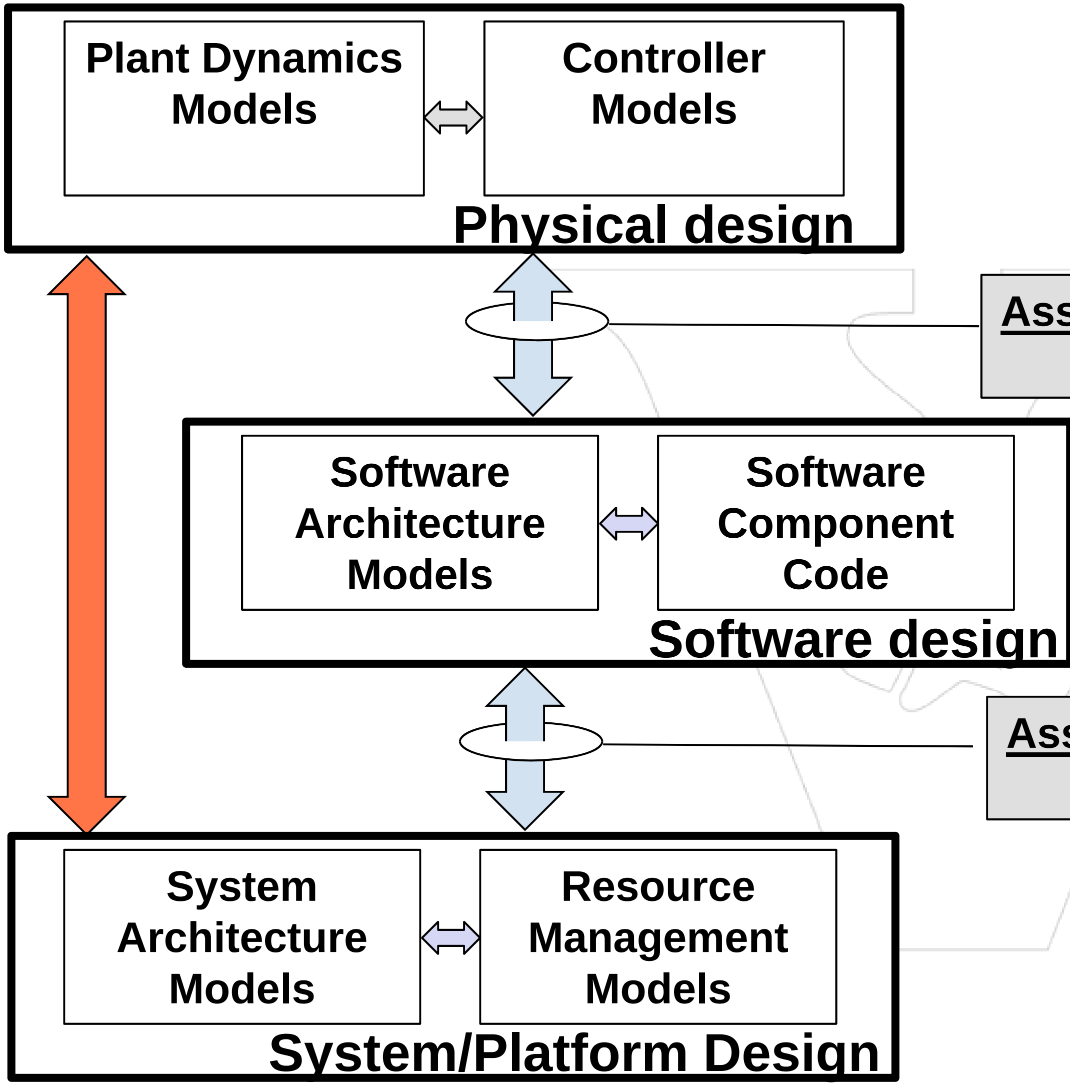
System Architecture Models ⟺ **Resource Management Models**

System/Platform Design

Systems : $B(t_j) = \kappa_p(B_1(t_i),\dots,B_k(t_i))$

- *Properties*: timing, power, security, fault tolerance
- *Abstractions*: discrete-time, delays, resources, scheduling,

Integration Across Abstraction Layers: Much Unsolved Problems



Controller dynamics is developed without considering implementation uncertainties (e.g. word length, clock accuracy) optimizing performance.

Assumption: Effects of digital implementation can be neglected

Software architecture models are developed without explicitly considering systems platform characteristics, even though key behavioral properties depend on it.

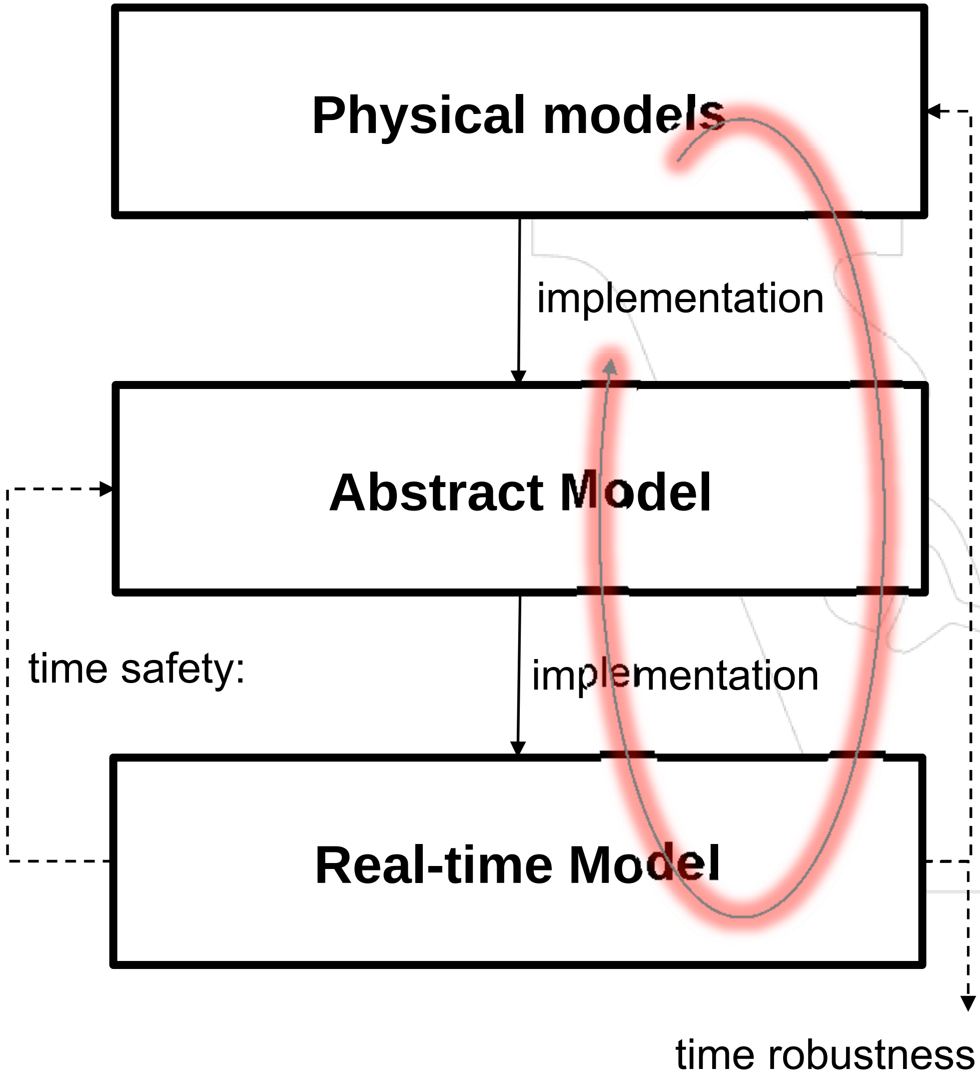
Assumption: Effects of platform properties can be neglected

System-level architecture defines implementation platform configuration. Scheduling, network uncertainties, etc. are introduce time variant delays that may require re-verification of key properties on all levels.

Dealing With Leaky Abstractions

- Leaky abstractions are caused by lack of composability across system layers. Consequences:
  - intractable interactions
  - unpredictable system level behavior
  - full-system verification does not scale
- Solution: simplification strategies
  - ***Decoupling:*** **Use design concepts that decouple systems layers for selected properties**
  - ***Cross-layer Abstractions:*** **Develop methods that can handle effects of cross-layer interactions**

Example for Decoupling: Passive Dynamics

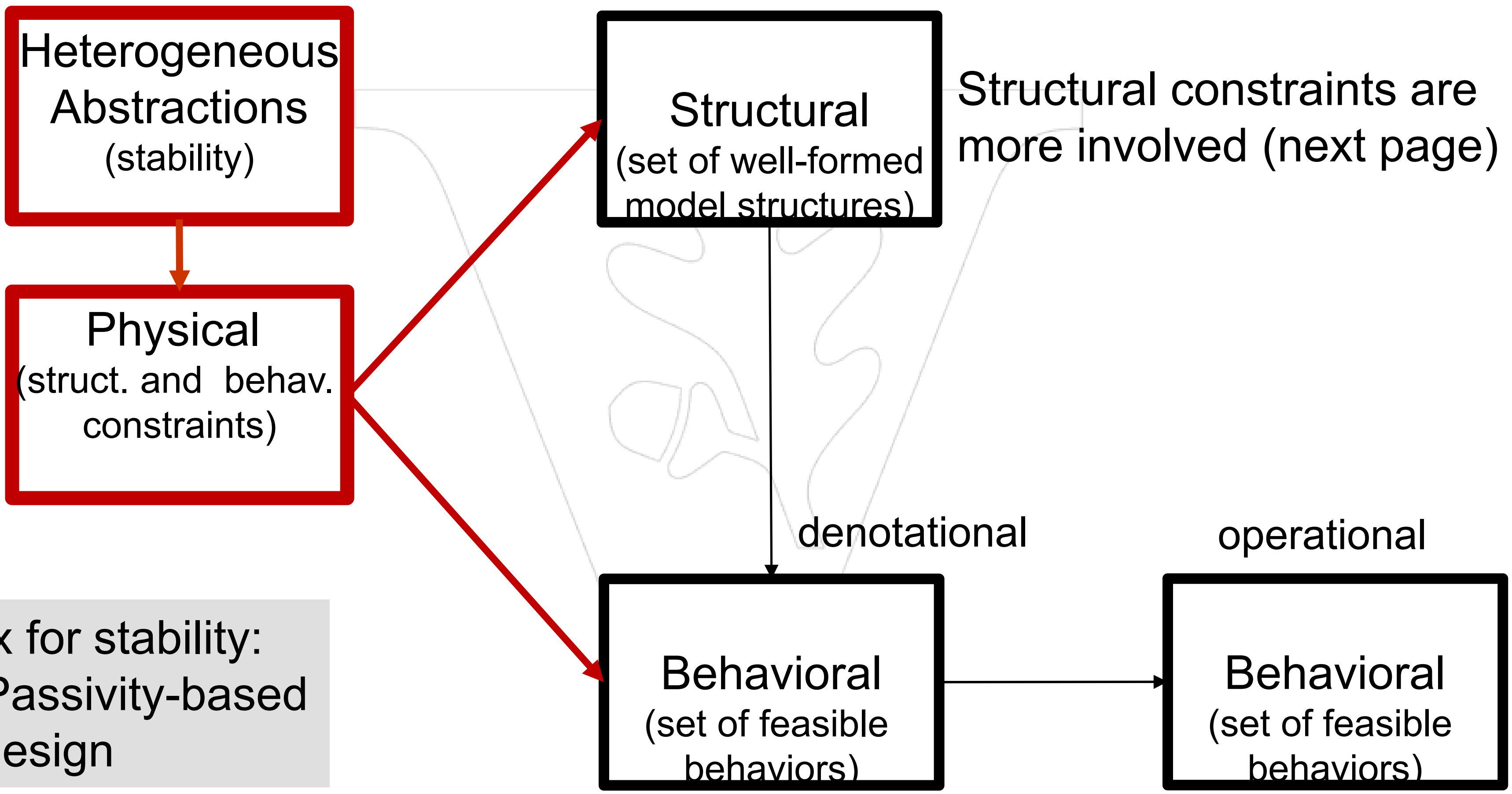


Goals:

- **Effect of "leaky abstraction": loss of stability due to implementation-induced time delays (networks, schedulers)**
- ***Passivity* of dynamics decouples stability from time varying delays**
- ***Compositional* verification of essential dynamic properties**
  - stability
  - safety
- **Hugely decreased verification complexity**
- **Hugely increased flexibility**

Passivity-based Design and Modeling Languages 1 / 4

Modeling Language Semantics:



Fix for stability:
- Passivity-based design

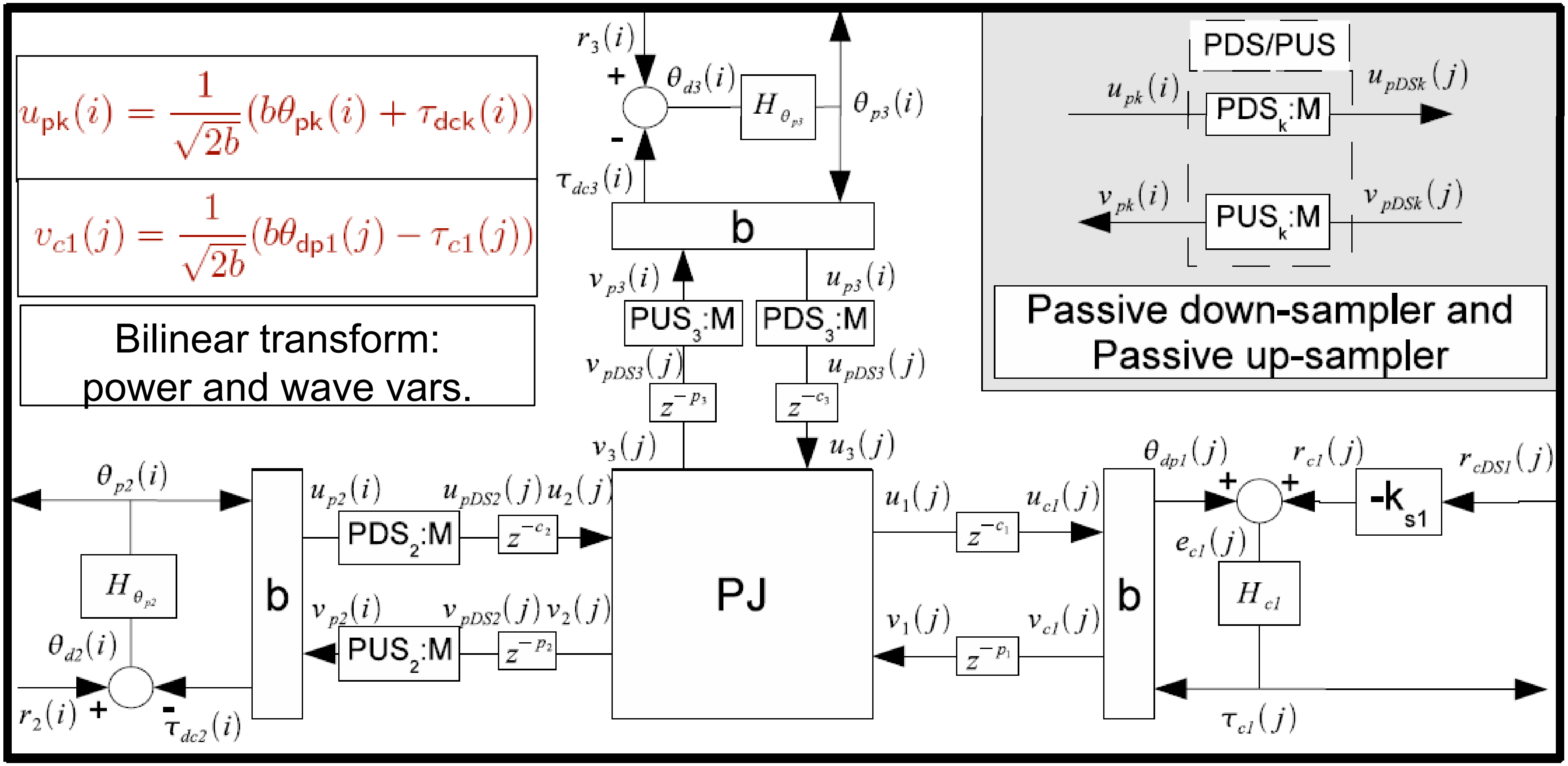
$$\dot{x} = f(x,u) \qquad y = h(x,u).$$

$$\int_{t_1}^{t_2} u^T(t)y(t)dt + V(x(t_1)) \geq V(x(t_2))$$

for all $t_2 \geq t_1$ and the input $u(t) \in U$ [Antsaklis '2008]

Passivity-based Design and Modeling Languages 2/4

Constrain modeling language with constructs below:

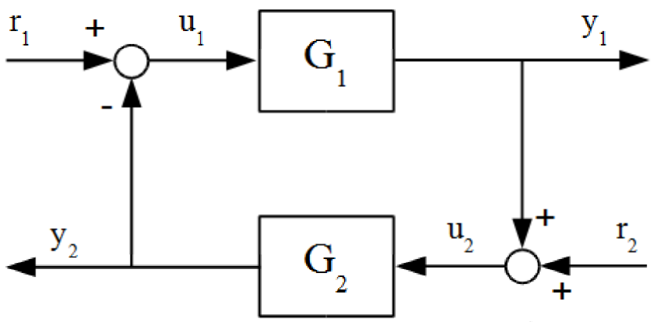


- Bilinear transform (b)
- Power and Wave variables
- Passive down- and up-sampler (PUS, PDS)
- Delays
- Power junction
- Passive dynamical system

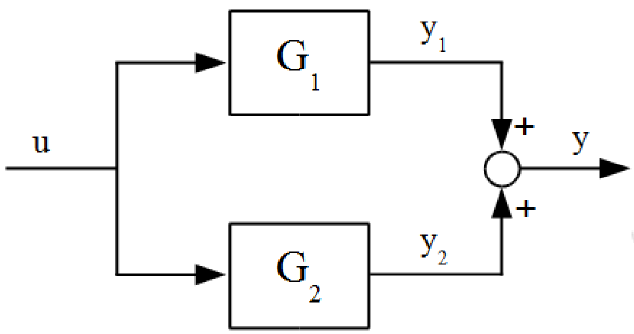
[Kottenstette'2011]

Passivity-based Design and Modeling Languages 3 / 4

Constrain modeling language with composition constraints below:



negative feedback interconnection of two passive systems is passive



parallel interconnection of two passive systems is still passive

Extensive research in the VU/ND/UMD NSF project toward correct-by-construction design environments (where *correct-by-construction means what the term suggest)*

Passivity-based Design and Modeling Languages 4 / 4

Constrain modeling language behavior with these constraints (for LTI)

- For LTI passive systems, we can always assume quadratic storage function

$$V(x) = \frac{1}{2} x^T P x \quad \text{where} \quad P = P^T > 0.$$

- For continuous-time system this leads to the following LMI

$$\begin{bmatrix} A^T P + PA & PB - C^T \\ B^T P - C & -D - D^T \end{bmatrix} \leq 0$$

- In discrete-time the LMI becomes the following

$$\begin{bmatrix} A^T PA - P & A^T PB - C^T \\ B^T PA - C & B^T PB - D - D^T \end{bmatrix} \leq 0$$

[Antsaklis ‘2008]

Summary

- Penetration of networking and computing in engineered systems forces a grand convergence across engineering disciplines.
- Signs of this convergence presents new opportunities and challenges for formal methods research:
  - New foundation for model integration – emergence of metaprogrammable tool suites and multi-modeling
  - Embedding physical semantics in modeling languages
- Model-based design facilitates a necessary convergence among software, system, control and network engineering

References

- Jackson, E., Sztipanovits, J.: 'Formalizing the Structural Semantics of Domain-Specific Modeling Languages," *Journal of Software and Systems Modeling* pp. 451-478, September 2009
- Jackson, Thibodeaux, Porter, Sztipanovits: "Semantics of Domain-Specific Modeling Languages," in P. Mosterman, G. Nicolescu: Model-Based Design of Heterogeneous Embedded Systems. Pp. 437-486, CRC Press, November 24, 2009
- Ethan K. Jackson, Wolfram Schulte, and Janos Sztipanovits: The Power of Rich Syntax for Model-based Development, MSR Technical Report, 2009
- Kai Chen, Janos Sztipanovits, Sandeep Neema: "Compositional Specification of Behavioral Semantics," in *Design, Automation, and Test in Europe:* The *Most Influential Papers of 10 Years DATE,* Rudy Lauwereins and Jan Madsen (Eds), Springer 2008
- Nicholas Kottenstette, Joe Hall, Xenofon Koutsoukos, Panos Antsaklis, and Janos Sztipanovits, "Digital Control of Multiple Discrete Passive Plants Over Networks", International Journal of Systems, Control and Communications (IJSCC), Special Issue on Progress in Networked Control Systems. (Accepted for publication)
- Xenofon Koutsoukos, Nicholas Kottenstette, Joe Hall, Emeka Eiysi, Heath Leblanc, Joseph Porter and Janos Sztipanovits, "A Passivity Approach for Model-Based Compositional Design of Networked Control Systems", ACM Transactions on Computational Logic . (Accepted for publication)
- Heath LeBlanc, Emeka Eyisi, Nicholas Kottenstette, Xenofon Koutsoukos, and Janos Sztipanovits. "A Passivity-Based Approach to Deployment in Multi-Agent Networks", *7th International Conference on Informatics in Control, Automation, and Robotics (ICINCO 2010)*. Funchal, Madeira, June 15-18, 2010. **(**Best Student Paper Award)

About the CPS Name…

“What’s in a name?
That which we call a rose by any other name would smell as sweet”

– Shakespeare