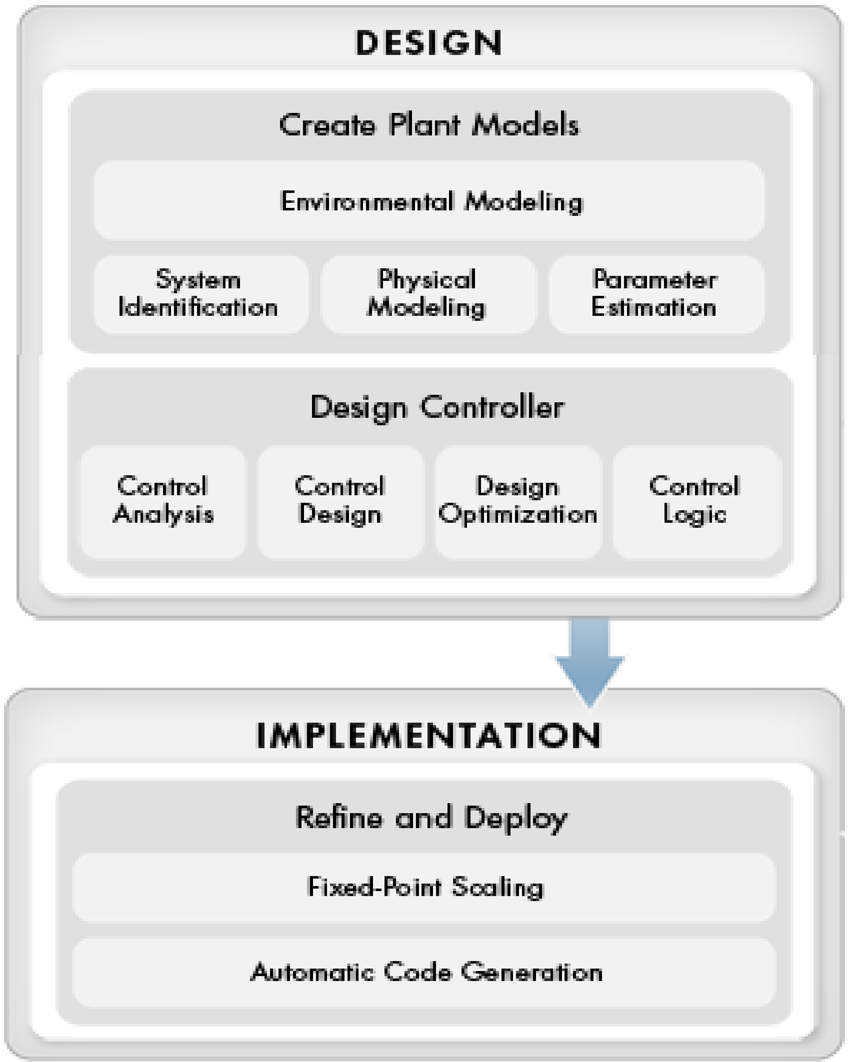
Using Heterogeneous Formal Methods in Model-Based Development

LCCC Workshop on Formal Verification of Embedded Control Systems

Bruce H. Krogh
Carnegie Mellon University in Rwanda
Kigali, Rwanda

Model-Based Development

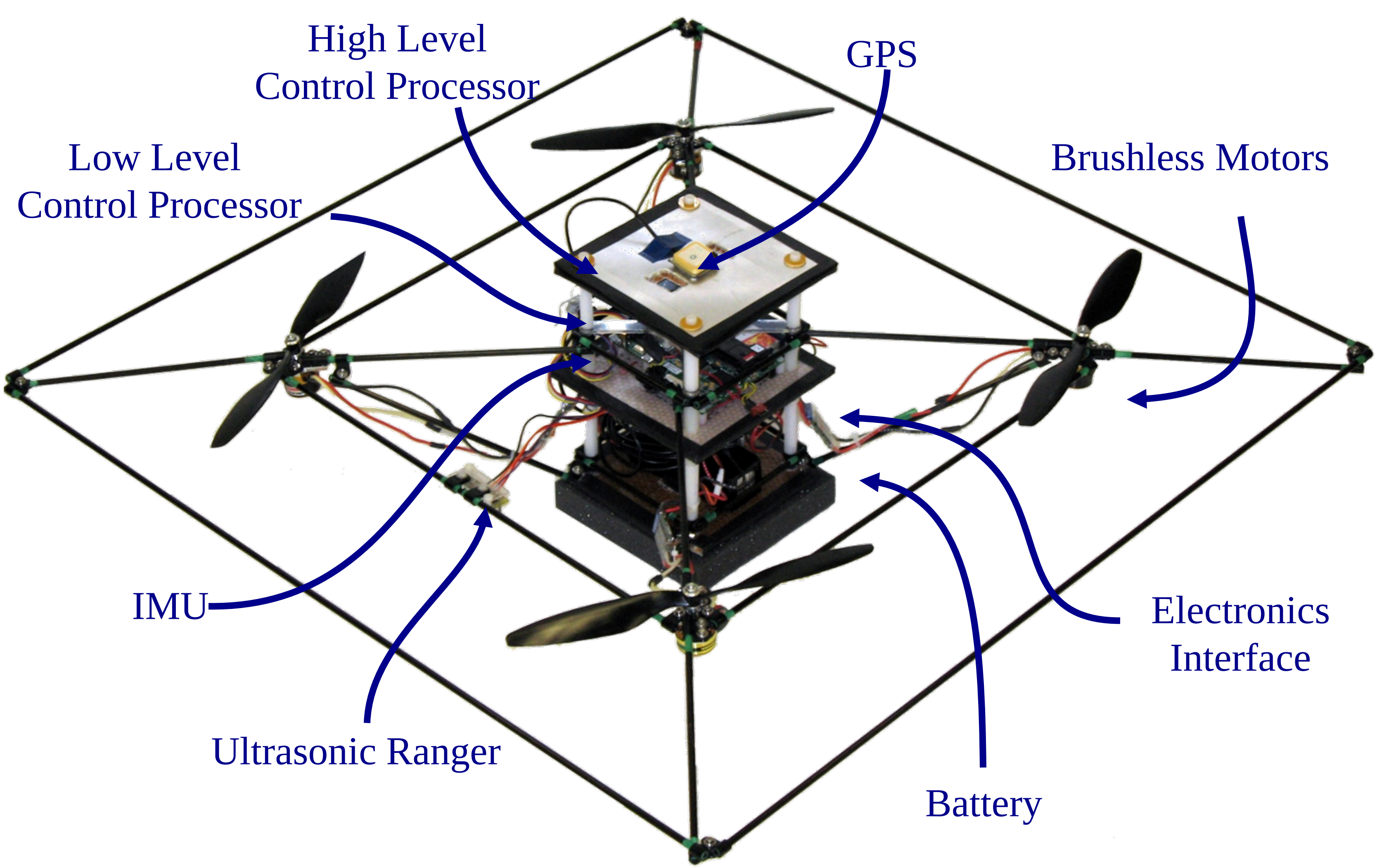


Use models to:

- evaluate alternative designs
- document the design
- generate implementations

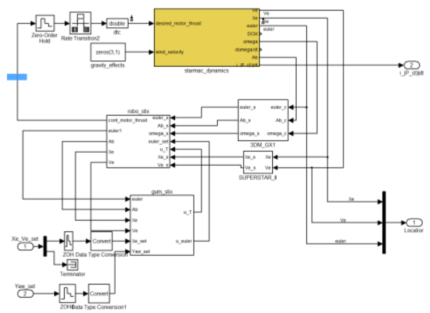
* composition of figures from http://www.mathworks.com/control-systems/

Example 1: STARMAC quadrotor

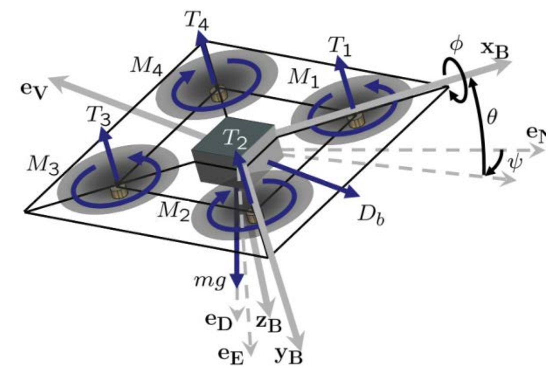


*http://hybrid.eecs.berkeley.edu/starmac/

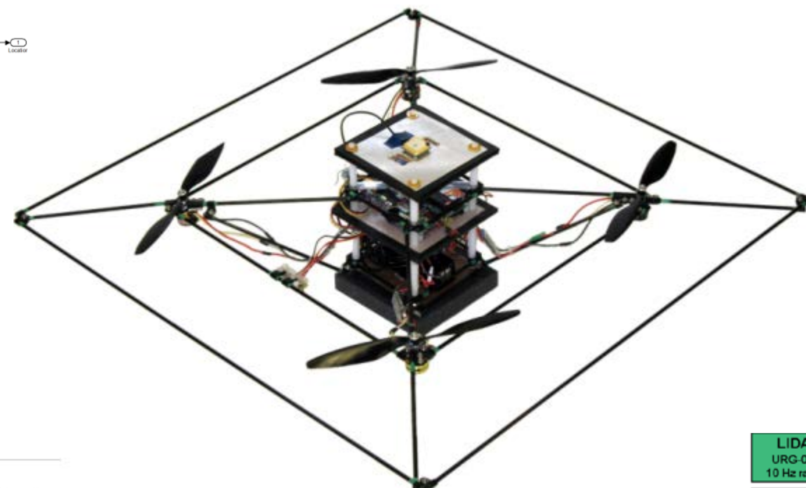
Multiple Models for the STARMAC

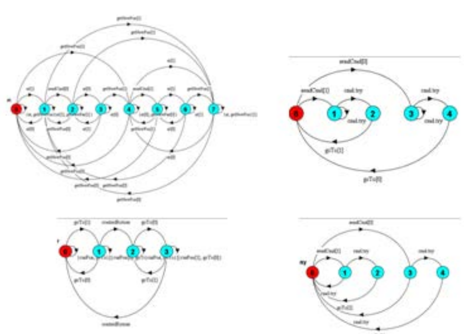


Control Model

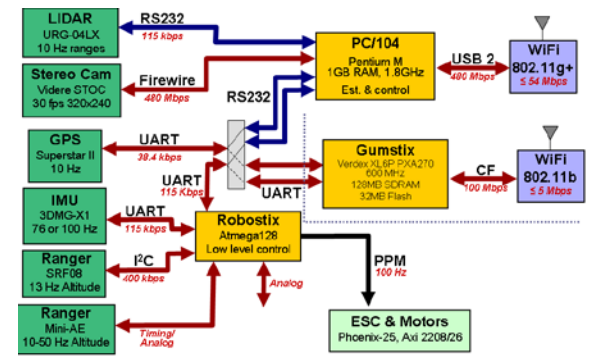


Physical Model



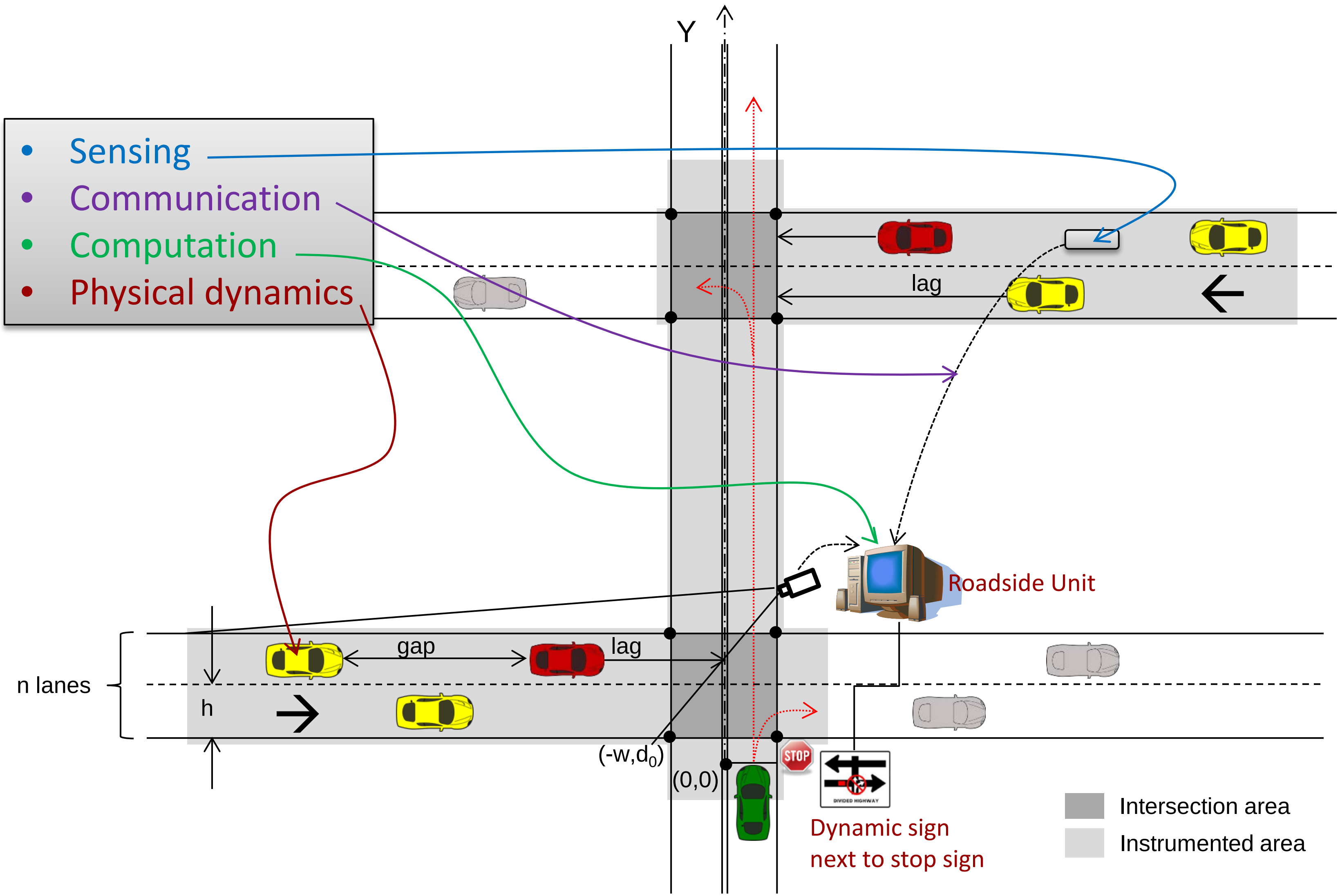


Software Model



Hardware Model

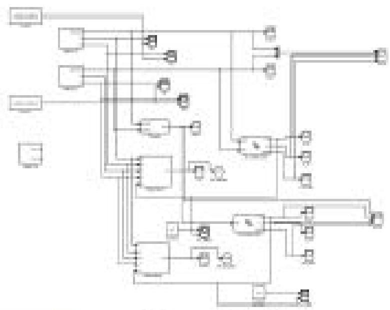
Example 2: Cooperative Intersection Collision Avoidance System – Stop Sign Assist (CICAS-SSA)*



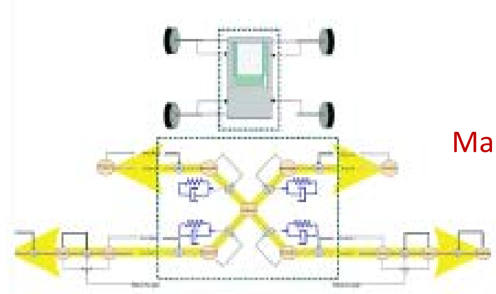
* http://www.dot.state.mn.us/guidestar/2006_2010/cicas/CICAS_SSA_ConOps_FINAL_3_18_08.pdf

Multiple Models for CICAS-SSA

Simulink



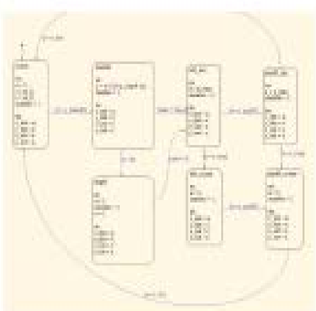
Signal-flow



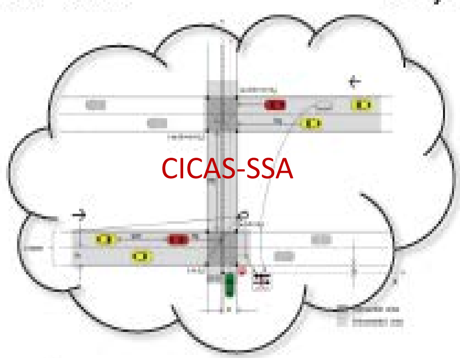
Physics-based

MapleSim, Modelica

Stateflow,
UppAal, …



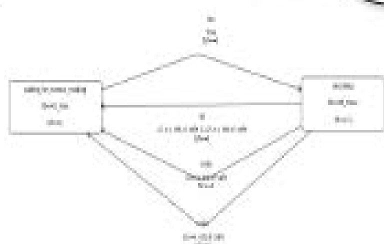
Discrete & timed



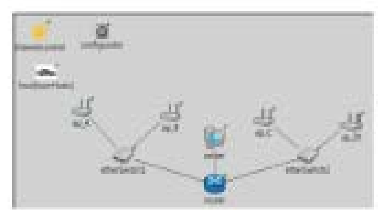
Software

LTSA, SPIN

PHAVer, SpaceEx, STRONG,
KeYmaera, CheckMate, …



Hybrid-dynamic



Network

OMNET++, ns2

Challenges in Multi-Domain MBD

No single model captures everything

- Each model represents some design aspect well, but not the others.
- Models are based on <u>interdependent</u> simplifying assumptions.
- Different tools focus on different properties and work only with particular modeling formalisms.

Challenges in Multi-Domain MBD

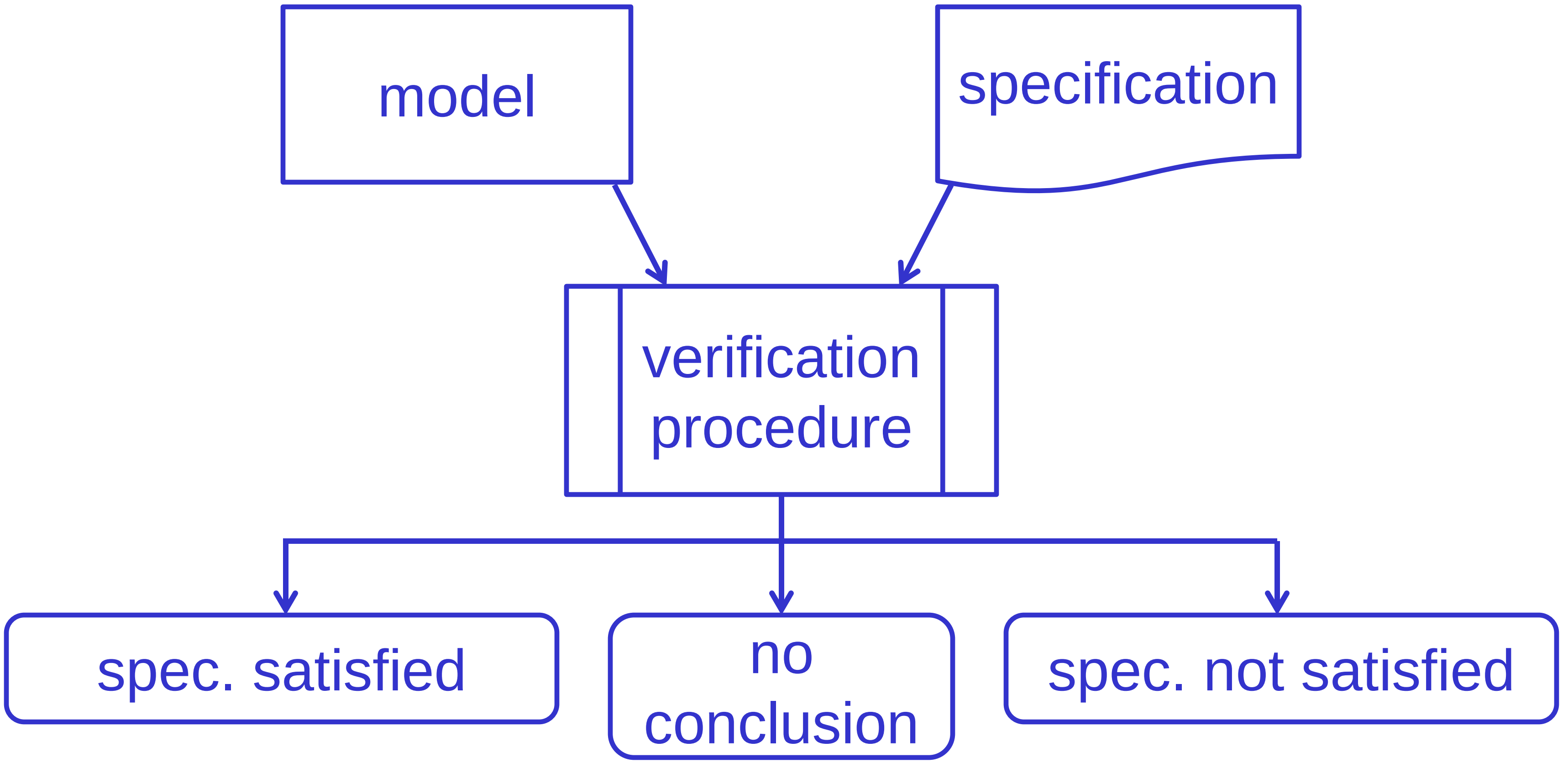
No single model captures everything:

- Each model represents some design aspect well, but not the others.
- Models are based on interdependent simplifying assumptions.
- Different tools focus on different properties and work only with particular modeling formalisms.

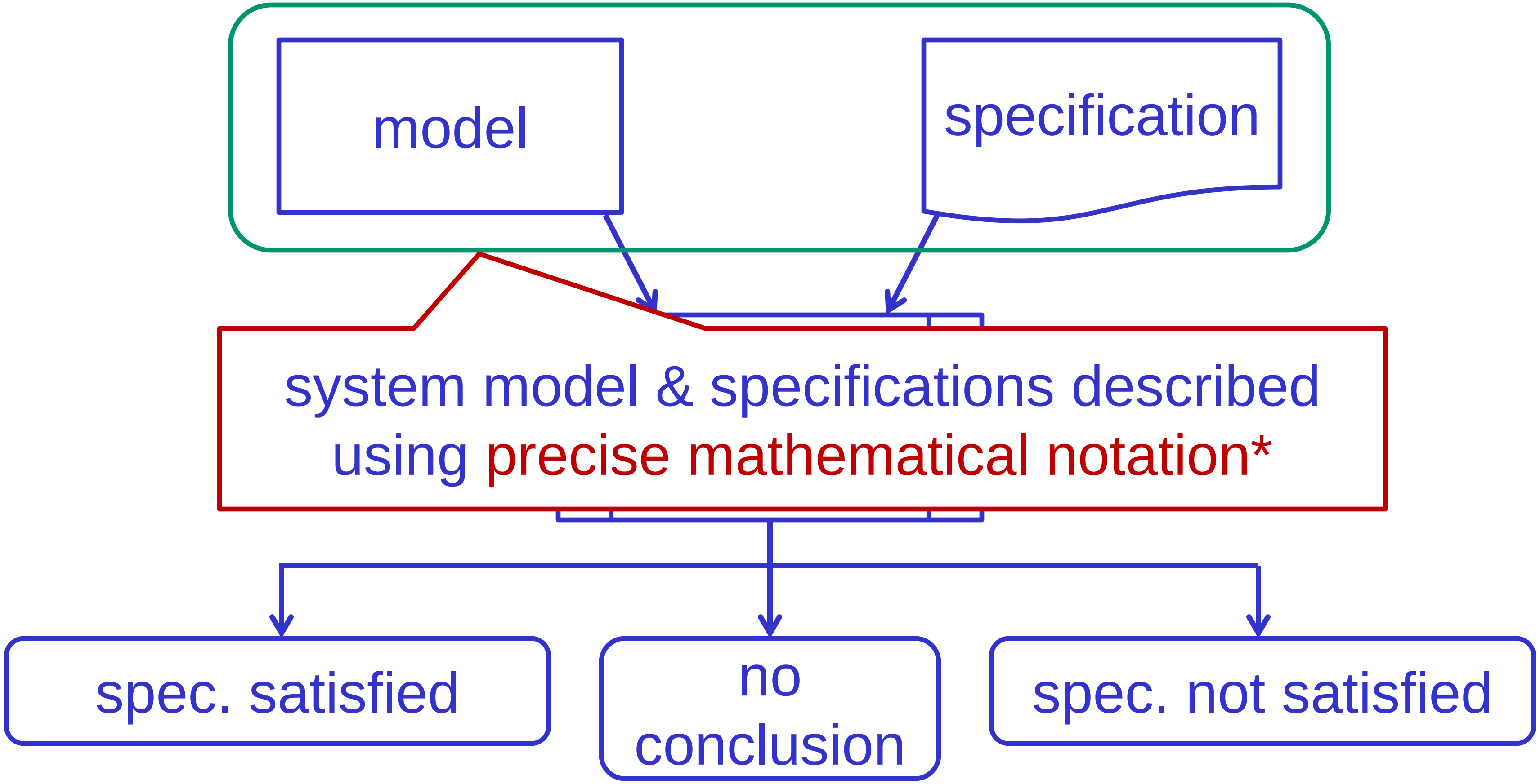
How can we:

1. Guarantee models are consistent with each other?
2. Infer system-level properties from heterogeneous analyses of heterogeneous models?

Formal Methods

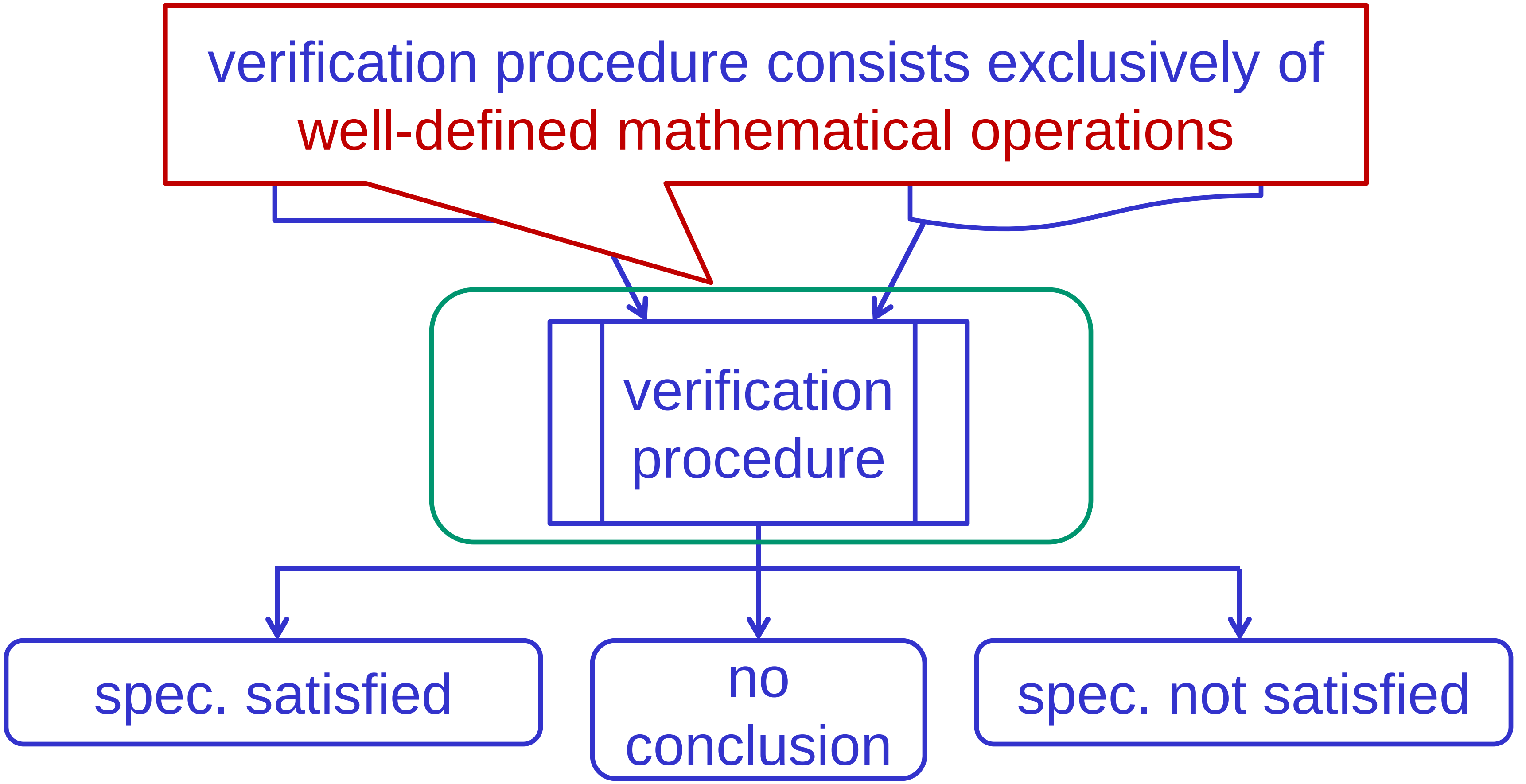


Formal Methods

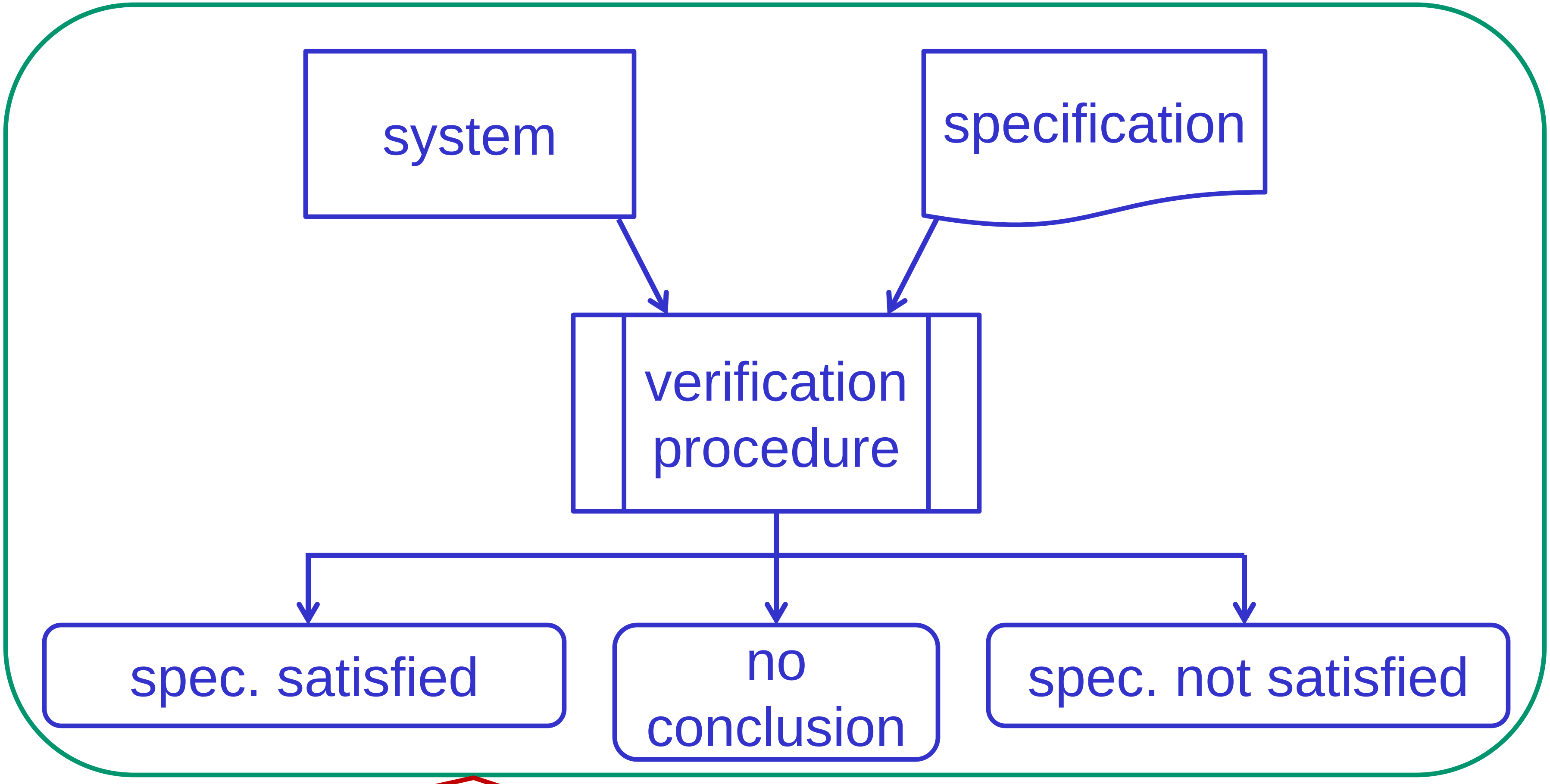


* unambiguous syntax and semantics

Formal Methods

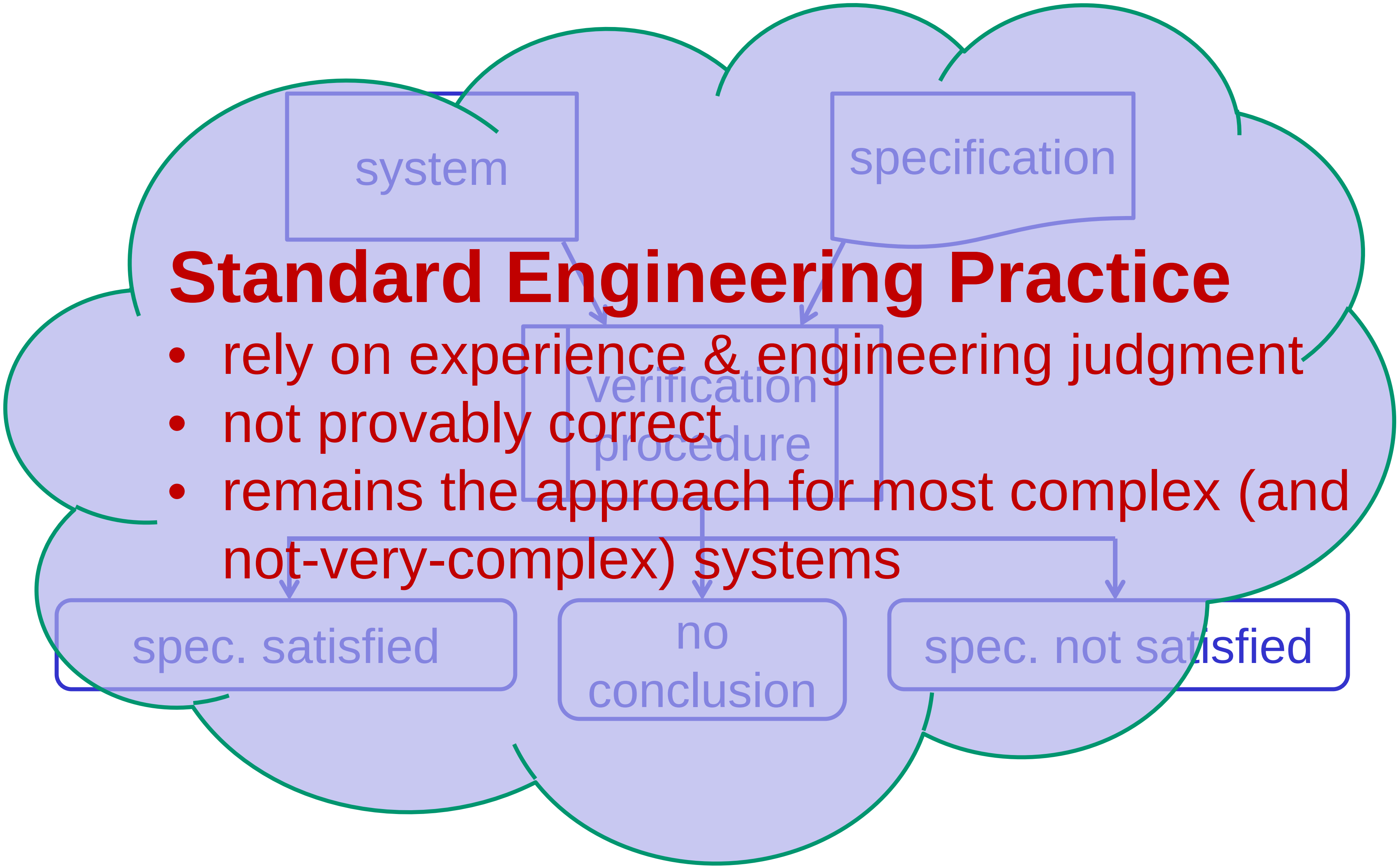


Formal Verification



a rigorous framework for proving specs. are satisfied

Informal Methods

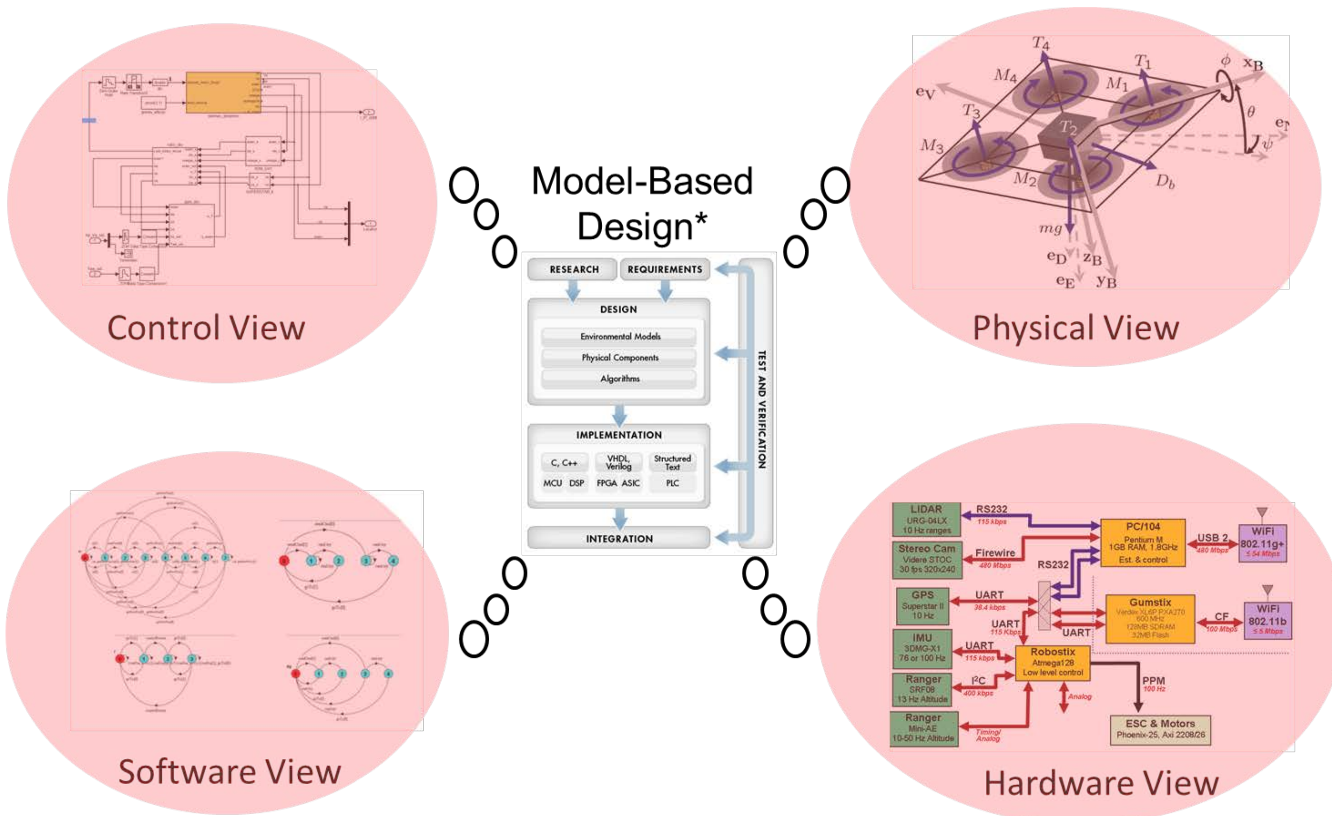


Standard Engineering Practice

- rely on experience & engineering judgment
- not provably correct
- remains the approach for most complex (and not-very-complex) systems

Approaches to Multi-Domain MBD:

1. Create a universal modeling language encompassing *everything* that needs to be modeled.



– UML/SysML (actually multiple views)
– MATLAB Simulink+Toolboxes

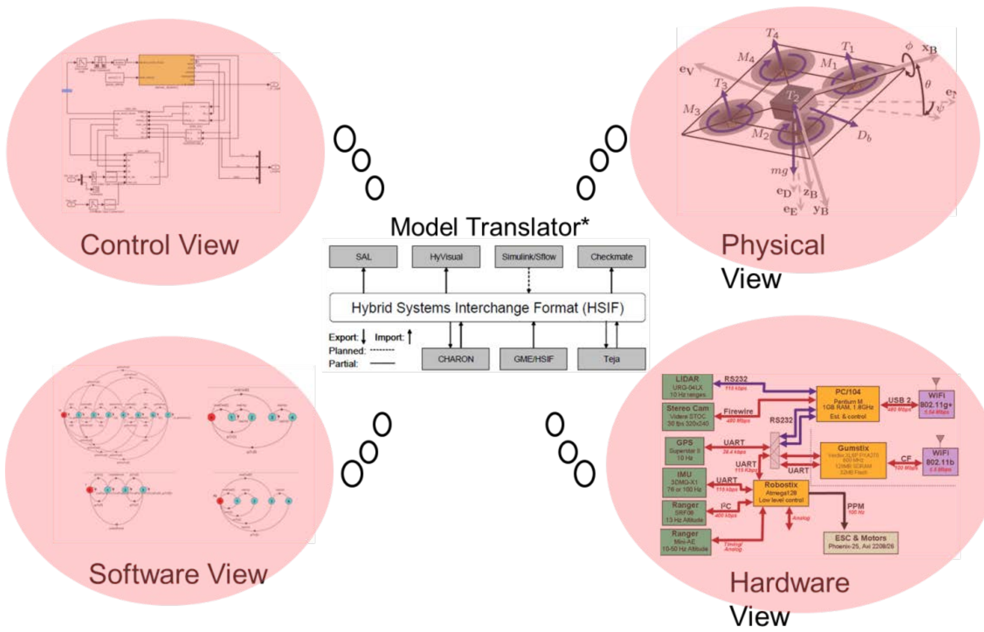
* http://www.mathworks.com/model-based-design/

Problems with Universal Models

- Comprehensive models representing *everything* become intractable
- Multi-domain MDB is based on separation of concerns: no one wants or needs the universal model
- Existing tools operate on specific types of models, not universal models

Approaches to Multi-Domain MBD:

2. Create tools that perform model translation between modeling formalisms.



- ARIES (Automatic Integration of Reusable Embedded Software) http://kabru.eecs.umich.edu/bin/view/Main/AIRES
- HSIF (Hybrid Systems Interchange Format) http://ptolemy.eecs.berkeley.edu/projects/mobies/

* J. Sprinkle, Generative components for hybrid systems tools, Journal of Object Technology, Mar-Apr 2003.

Problems with Model Translation

- Tool-specific translation isn't scalable
- Universal translation essentially requires a universal modeling language (Approach 1)
- Translators are difficult to maintain because modeling languages and tools continually evolve

Two Proposals for Multi-Domain MBD

1. How can we guarantee models are consistent with each other?
   – Formalize consistency at the architectural level
2. How can we infer system-level properties from heterogeneous analyses of heterogeneous models?
   – Formalize heterogeneity as mappings between behavioral semantic domains

(See Nikos Arechiga for point 3 in the abstract: using theorem proving to establish control design constraints)

An Architectural Framework for Multi-Domain MDB

Goal: Unify heterogeneous models through *light-weight* representations of their structure and semantics using architecture description languages (ADLs)

Architectural analysis:

- Does each model adhere to the base system structure & constraints (consistency)?
- Are all system elements represented in at least one model (completeness)?

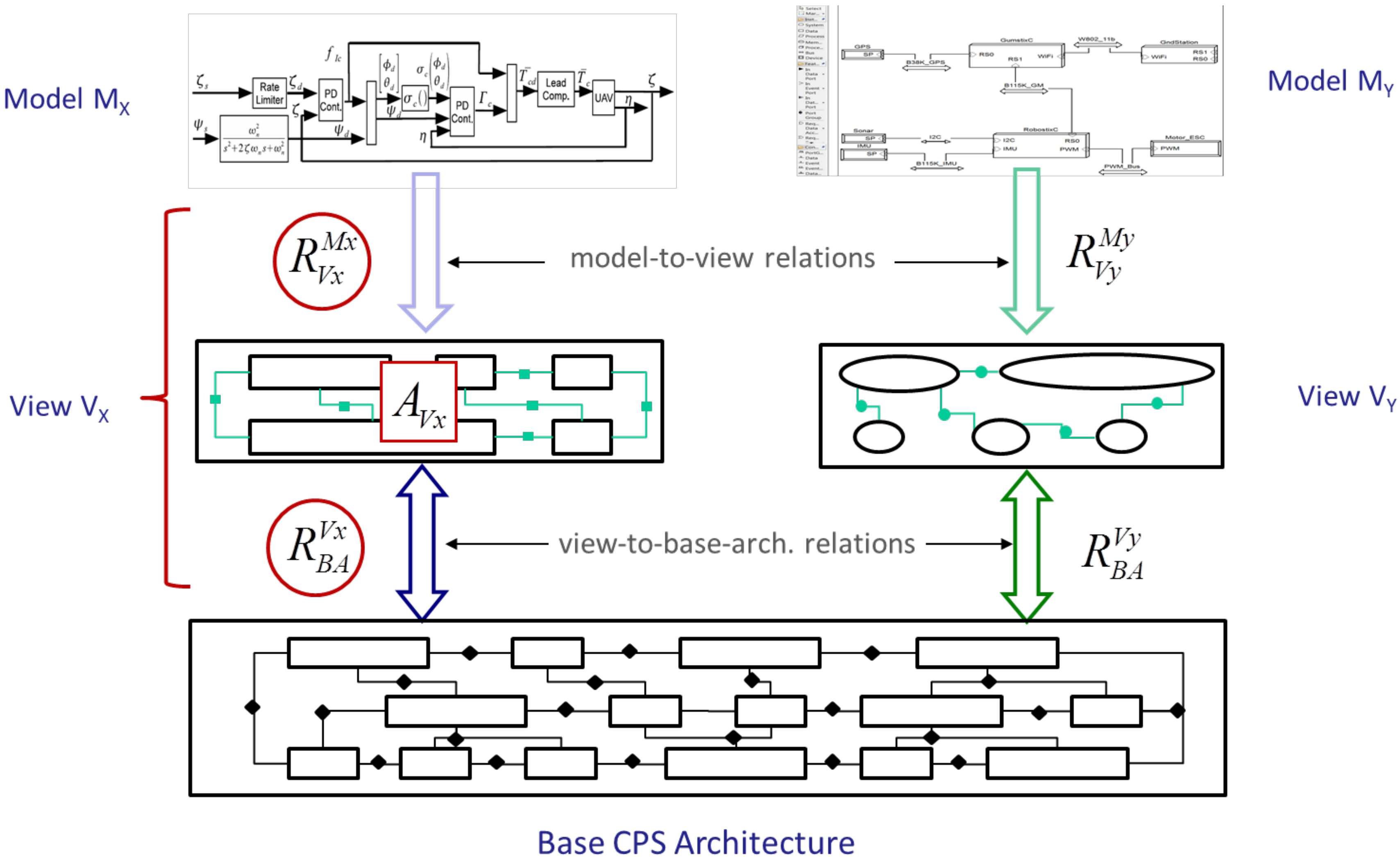
Architectures for CPS

Architecture: The set of structures needed to reason about the system, which comprise functional elements, relations among them and properties of both.*

- CPS *base architecture* defines
  - component connectivity & physical coupling
  - data, control, & physical signal flows
- Model architectures define
  - components and connectors exposing the model structure for evaluation vis-à-vis the base architecture

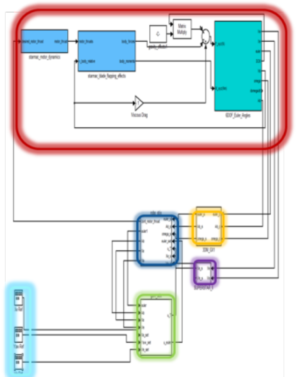
* *Documenting Software Architecture: Views and Beyond*, 2nd Ed. Clements et al. 2010.

Models as Architectural Views

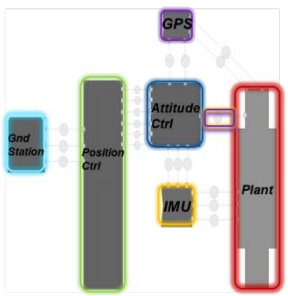


Example: Consistency Analysis

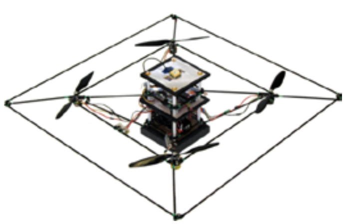
- Typed-graph morphisms expose inconsistencies between model architectures and the base architecture.



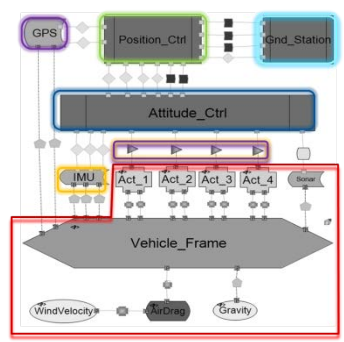
Simulink Model



Control View



STARMAC quadrotor



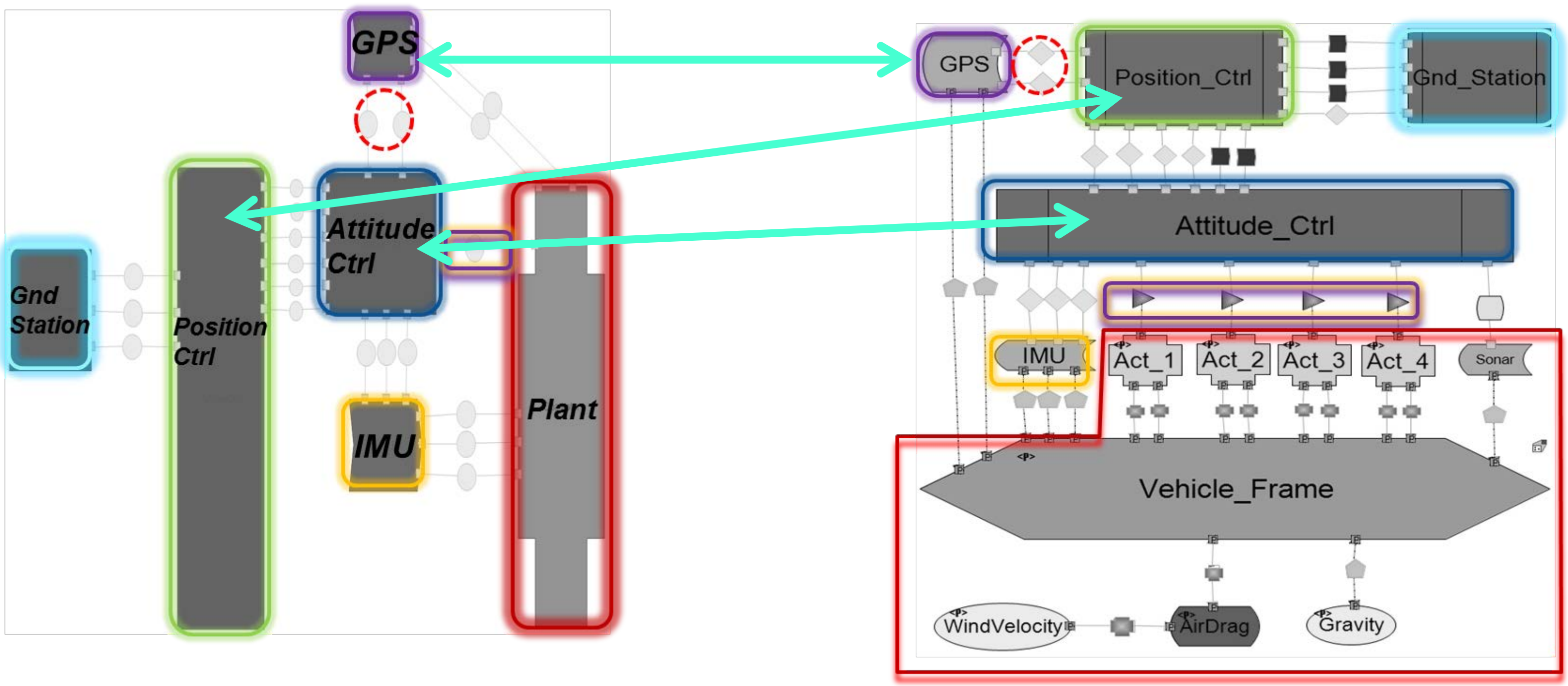
Base Architecture

Component and connector relations defined by architecture types and semantics.

Example: Consistency Analysis

GPS sensor connected to:

attitude controller position controller



Control View

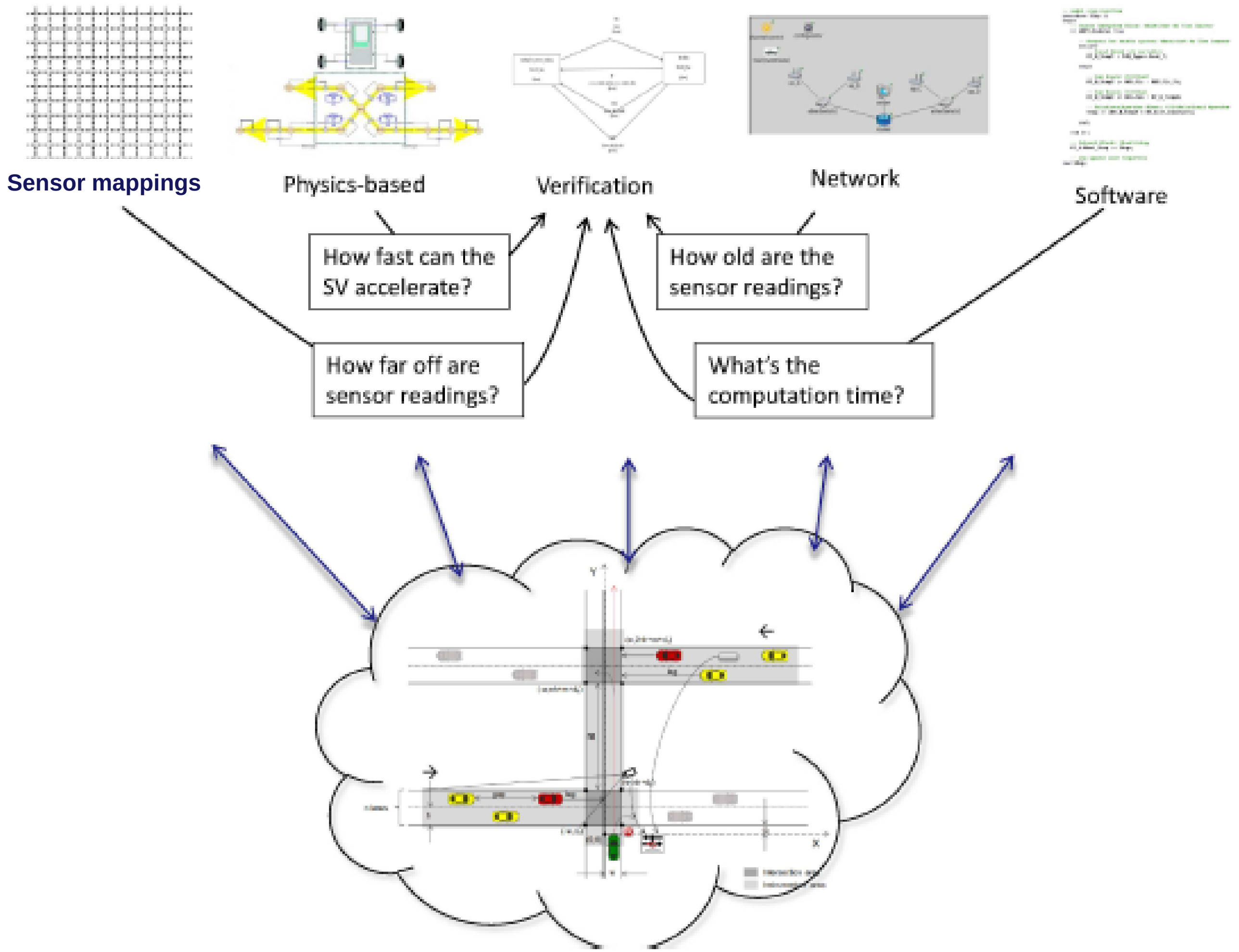
Base Architecture

Architectures for Multi-Domain MBP

- CPS architectures: extension of software/hardware to include physical domains
- consistency and completeness analysis
- parametric consistency analysis
- implementation in ADL tool (ACME)

papers: google "Bhave Garlan Krogh Architecture"

Heterogeneous Verification



Using Behavioral Semantics for Heterogeneous Verification

Goal: Provide a formal framework to perform verification using a heterogeneous set of modeling formalisms and analysis tools.

Heterogeneous Verification:

- abstraction using different formalisms
- implication using different formalisms
- composition using different formalisms

Using Behavioral Semantics for Heterogeneous Verification

Goal: Provide a formal framework to perform verification using a heterogeneous set of modeling formalisms and analysis

to

Just presented last week
at HSCC 2013 by
Akshay Rajhans

- abst
- implication ... different formalisms
- composition using different formalisms

Heterogeneous Verification via Behavioral Semantics

Goal: Provide a formal framework to

n … eneous

… is

too…

Today's presentation.

Heterogeneous Verification:

- abstraction using different formalisms
- implication using different formalisms
- composition using different formalisms

Models, Specifications and Behaviors

Models: $M \in \mathcal{M}$ means the model M is constructed using a modeling formalism $\mathcal{M}$ (e.g., state equations, Petri nets, block diagrams).

Specifications: $S \in \mathcal{S}$ means the specification S is constructed using a specification formalism $\mathcal{S}$ (e.g., inequalities, logical expressions, automata, differential inclusions).

Behaviors: $B \subseteq \mathcal{B}$ means the behavior domain B is in the class of behaviors $\mathcal{B}$ (e.g., traces, piecewise continuous functions, real numbers).

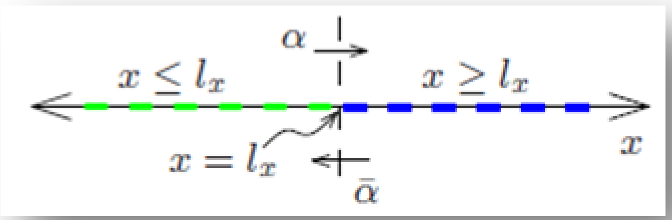
Homogeneous Behavioral Semantics

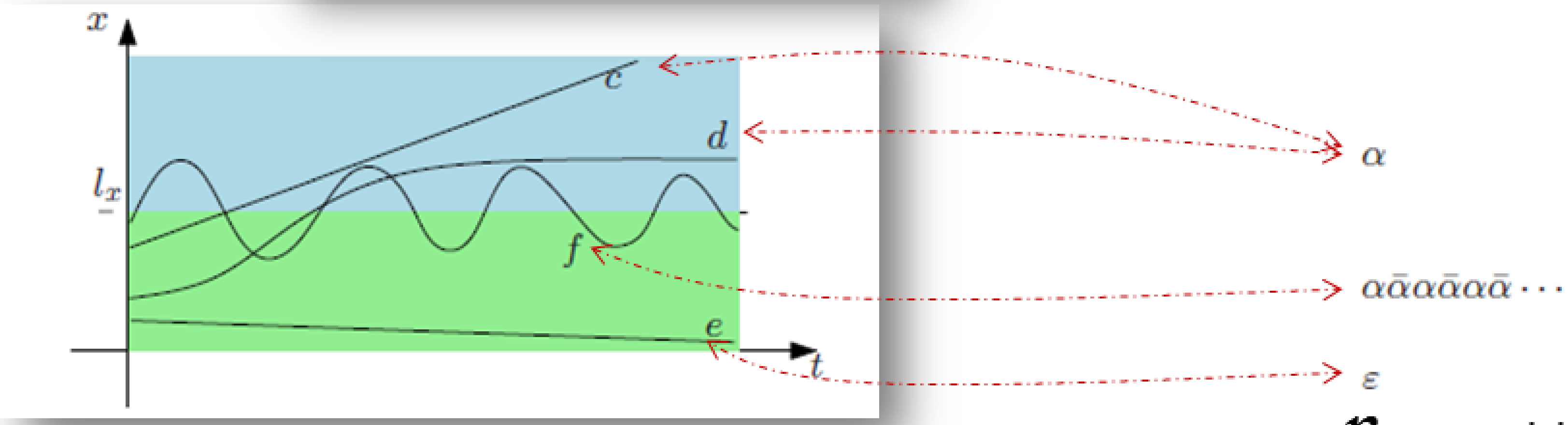
- Given a model M and a behavior domain B, the behavioral semantics for M is a set of behaviors in B, denoted $[[M]]^B$.
- Given a model S and a behavior domain B, the behavioral semantics for S is a set of behaviors in B, denoted $[[S]]^B$
  - M_1 an abstraction M_0 : $[[M_0]]^B \subseteq [[M_1]]^B$.
  - S_1 implies S_0 : $[[S_1]]^B \subseteq [[S_0]]^B$
  - S is true for M (entailment) : $[[M]]^B \subseteq [[S]]^B$

Heterogeneity: Relations on Pairs of Behavioral Domains

$$R_1 \subseteq B_0 \times B_1$$

Example





$\mathcal{B}_0$: continuous trajectories

B_0:1-d continuous trajectories in x

$\mathcal{B}_1$: event traces

$B_1 = \{\alpha, \overline{\alpha}\}^* \cup \{\alpha b, \overline{\alpha}\}^\omega$

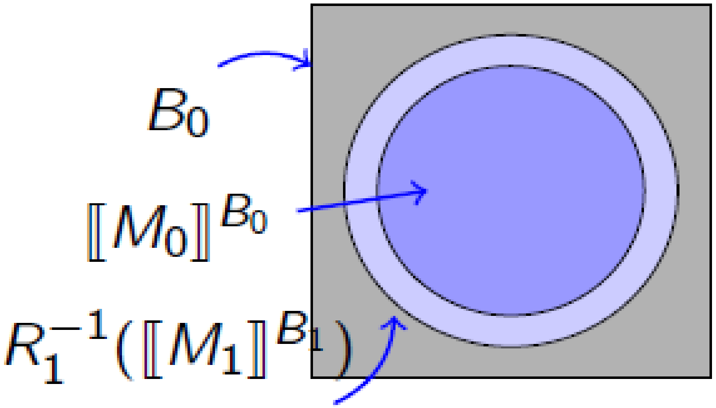
Heterogeneous Abstraction

$$R_1 \subseteq B_0 \times B_1$$

Heterogeneous Abstraction

$M_0 \sqsubseteq^{R_1} M_1$, if

$$[\![M_0]\!]^{B_0} \subseteq R_1^{-1}([\![M_1]\!]^{B_1}).$$



M_1 abstracts M_0
(via R_1)

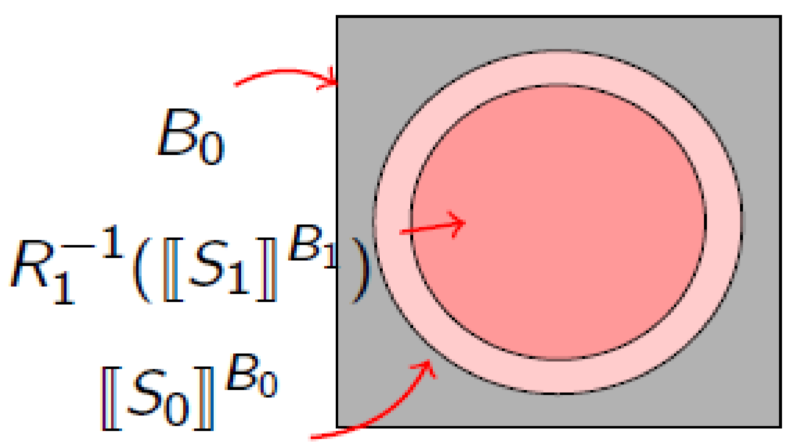
Heterogeneous Specification Implication

$$R_1 \subseteq B_0 \times B_1$$

Heterogeneous Specification Implication

$S_1 \Rightarrow^{R_1} S_0$, if

$$R_1^{-1}([\![S_1]\!]^{B_1}) \subseteq [\![S_0]\!]^{B_0}.$$



S_1 stronger than S_0
(via R_1)

Heterogeneous Verification

$$R_1 \subseteq B_0 \times B_1$$

Heterogeneous Verification

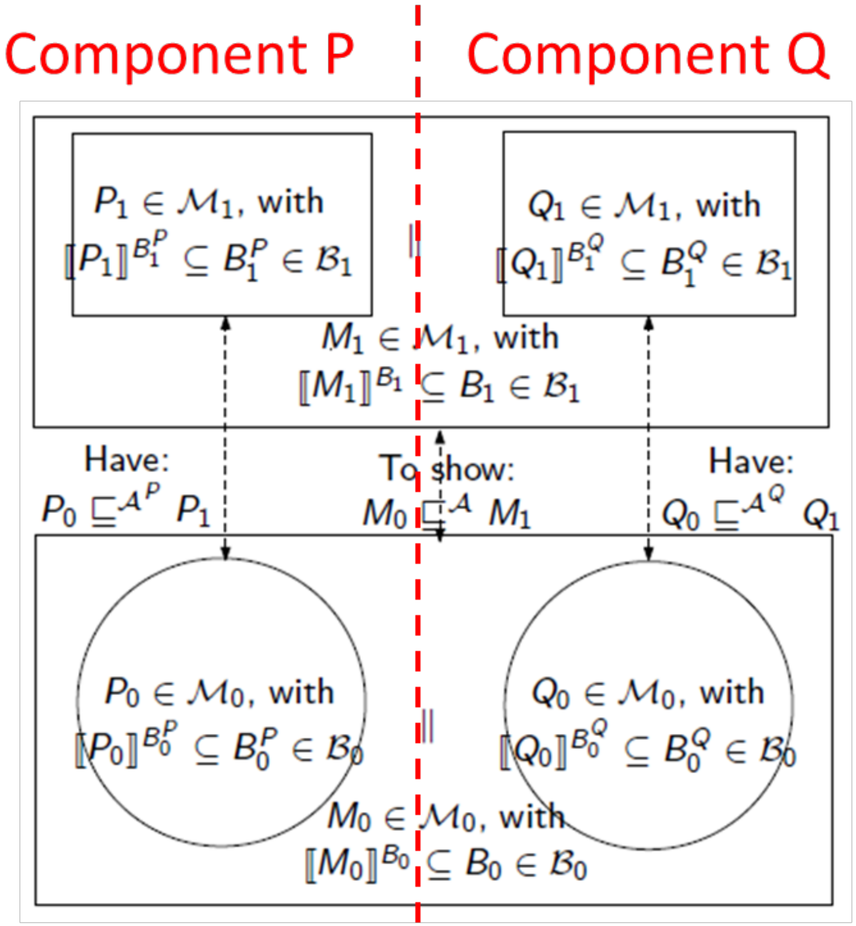
If $M_0 \sqsubseteq^{R_1} M_1$, $M_1 \models^{B_1} S_1$ and $S_1 \Rightarrow^{R_1} S_0$, then $M_0 \models^{B_0} S_0$.

heterogeneous abstraction

$$\begin{array}{c|c|c} M_1 & \models^{B_1} & S_1 \\ \hline \rotatebox{90}{$\sqsubseteq^{R_1}$} & \Downarrow & \Downarrow^{R_1} \\ \hline M_0 & \models^{B_0} & S_0 \end{array}$$

heterogeneous implication

Compositional Heterogeneous Abstraction



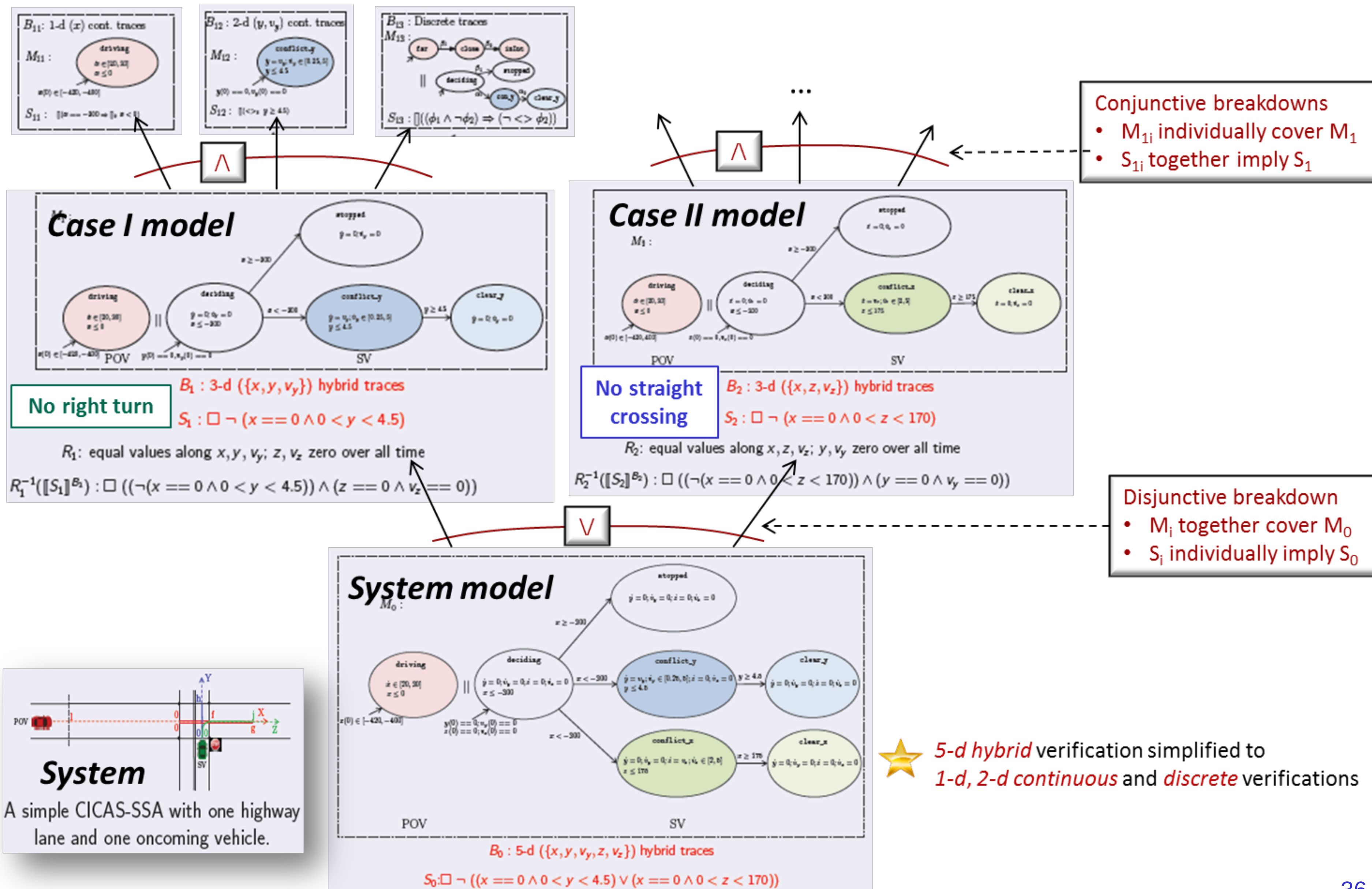
When is the composition of component abstractions an abstraction of the composition of components?

<u>Sufficient Condition</u>

Abstraction relations $\mathcal{A}^P$, $\mathcal{A}^Q$ have a common globalization $\mathcal{A}$.

Note: A single class of behaviors is used at each level.

Application to CICAS-SSA



Heterogeneous Verification via Behavioral Semantics

- Behavior domain relations support heterogeneous
  - abstraction
  - implication
  - compositional abstraction
- Applications to CICAS-SSA
- Formalizes specific cases in the literature

papers: google “Rajhans Krogh heterogeneous”

Using Architecture to Manage Formal Analysis