Multi-Paradigm Modeling for Design and Operation of Intelligent Cyber-Physical Systems

Keynote Talk, First International Workshop on Multi-Paradigm Modeling of Cyber-Physical Systems (MPM4CPS) Munich, Germany. September 10, 2019

Akshay Rajhans, PhD
arajhans@mathworks.com
https://arajhans.github.io
About me

- ‘CPS’ Practitioner before it was called CPS
  - Embedded controls for diesel engine applications
  - Programmable logic controller for industrial automation

- CPS Research at the intersection of
  - Model-based design and analysis
  - Formal methods
  - Software and system architecture

- CPS Research Scientist at MathWorks
Perspective shaped by my personal career trajectory

Academic Researcher

Tool Developer

Industry Practitioner

Interests span this tradeoff
Outline

- Introduction

- Theoretical aspects of multi-paradigm model-based design for CPS
  - Architecture modeling and structural analysis
  - Semantic analysis and heterogeneous verification
  - Compositional analysis

- Practical aspects of a multi-domain simulation platform
  - Graphical modeling of hybrid dynamics using Simulink and Stateflow

- Recap and conclusions
Cyber-physical systems have societal scale applications

- Smart Manufacturing
- Smart Infrastructure
- Smart Energy
- Smart Transportation
- Smart Health
Traffic accidents are bad

Quick Facts 2017, NHTSA, [https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812747](https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812747)
According to the 2017 American Community Survey from the Census Bureau, an estimated 19 percent of the U.S. population lived in rural areas, and according to FHWA only 30 percent of the total vehicle miles traveled (VMT) in 2017 were in rural areas. However, rural areas accounted for 46 percent of all traffic fatalities in 2017.

Rural/Urban Comparison of Traffic Fatalities, NHTSA [https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812741](https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812741)

Intersection collision avoidance system

Prototypical “heterogeneous” CPS

- Sensing
- Communication
- Computation
- Physics and actuation

Can we use technology (connectivity or autonomy)?
Models are useful in both design and operation

Control design (e.g., block diagrams)

Filtering (block diagrams)

Protocols and policies (State machines)

Physics and mechanics (Acausal modeling)

Algorithms (e.g., procedural code)

Traffic flow (Discrete-event systems)
• No ‘universal’ modeling formalism, modeling everything together intractable

• Different formalisms suited for different aspects of system design

• Each model represents some design aspect well, but not others

• Models make interdependent assumptions about each other and the system

• Analysis tools often specialized for a particular formalism

- Given all of these, how do we design a correct system?
Outline

- Introduction

  - Theoretical aspects of multi-paradigm model-based design for CPS
    - Architecture modeling and structural analysis
    - Semantic analysis and heterogeneous verification
    - Compositional analysis

- Practical aspects of a multi-domain simulation platform
  - Graphical modeling of hybrid dynamics using Simulink and Stateflow

- Recap and conclusions
From software architecture to CPS architecture

- Even though there is no system-level model, there is a system architecture
- Extend software architecture vocabulary with physical elements
- Heterogeneous component models are annotations on the architecture elements

From software architecture to CPS architecture

Heterogeneous component models are annotations on the architecture elements

Implicit assumption: models composed of the same structure as the architecture

Models have their own structure. What gets abstracted away depends on the paradigm.

Architectures extracted from model structure are ‘views’ of the base architecture.

There are relations between the views and the base architecture.

Simulink architecture view
Simulink architecture view
Outline

- Introduction

- Theoretical aspects of multi-paradigm model-based design for CPS
  - Architecture modeling and structural analysis
  - Semantic analysis and heterogeneous verification
  - Compositional analysis

- Practical aspects of a multi-domain simulation platform
  - Graphical modeling of hybrid dynamics using Simulink and Stateflow

- Recap and conclusions
Semantic interpretations of models and specifications

1) “overshoot is no more than 1.3 units and settling time is less than $\tau$”

2) $\Box (x < 1.3) \land \Diamond (x \in [1 \pm \epsilon])$

Abstraction, implication, and satisfaction as behavior set inclusions

- Model $M_1$ abstracts $M_0$ in $B$, written $M_0 \subseteq^B M_1$
  
  if
  
  \[ [M_0]^B \subseteq [M_1]^B \]

- Specification $S_1$ implies $S_0$ in $B$, written $S_1 \Rightarrow^B S_0$
  
  if
  
  \[ [S_1]^B \subseteq [S_0]^B \]

- Model $M$ satisfies specification $S$ in $B$, written $M \models^B S$
  
  if
  
  \[ [M]^B \subseteq [S]^B \]

Homogeneous in $B$: Same $B$ everywhere

often heterogeneous

can be heterogeneous

can be heterogeneous

Mappings between semantic domains via \textit{behavior relations}

- \textbf{Approach}: Create “relations” between behavior domains

\[ R_1 \subseteq B_0 \times B_1 \]

\[ B_0 : 1-d \text{ continuous trajectories in } x \]

\[ B_1 = \{ \alpha, \bar{\alpha} \}^* \cup \{ \alpha, \bar{\alpha} \}^\omega \]

\[ \text{Given } R_1 \subseteq B_0 \times B_1 \text{ set-based inverse map } R_1^{-1} (\alpha') = \{c, d, \ldots\} \]

Heterogeneous abstraction, implication, and satisfaction

Heterogeneous Abstraction
\[ M_0 \sqsubseteq^{R_1} M_1, \text{ if } \]
\[ [M_0]^{B_0} \subseteq R_1^{-1}([M_1]^{B_1}). \]

Heterogeneous Specification Implication
\[ S_1 \Rightarrow^{R_1} S_0, \text{ if } \]
\[ R_1^{-1}([S_1]^{B_1}) \subseteq [S_0]^{B_0}. \]

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 \( M_0 \models^{B_0} S_0 \).

(in words)

(in pictures)

Outline

▪ Introduction

▪ Theoretical aspects of multi-paradigm model-based design for CPS
  – Architecture modeling and structural analysis
  – Semantic analysis and heterogeneous verification
  ➢ Compositional analysis

▪ Practical aspects of a multi-domain simulation platform
  – Graphical modeling of hybrid dynamics using Simulink and Stateflow

▪ Recap and conclusions
Compositional heterogeneous abstraction

Heterogeneous Verification

Compositional Heterogeneous Verification

If $M_0 = P_0 \parallel Q_0$ and $M_1 = P_1 \parallel Q_1$, can we analyze $P$s and $Q$s independently?

Objective: Conclude heterogeneous abstraction of the composition by establishing that of the components

Rationale: Component’s local semantics defined in a behavior domain of smaller dimension

“Models as composition of components”
Analysis: Compositional Abstraction

Leveraging compositionality for heterogeneous abstraction

**Objective:** Conclude heterogeneous abstraction of the composition by establishing that of the components

**Rationale:** Component’s local semantics defined in a behavior domain of smaller dimension

**Need**
- Behavior abstraction functions $\mathcal{A}$: behavior relations that are also functions
- Mappings between local/global behavior domains of the same type
- Mappings between local/global abstraction functions

\[
\begin{align*}
M_0 \subseteq^A M_1 & \quad \text{Abstract composition behavior domain } B_1 \\
\llbracket M_0 \rrbracket_{B_0} & \subseteq A^{-1}(\llbracket M_1 \rrbracket_{B_1}) \\
\text{Detailed composition behavior domain } B_0
\end{align*}
\]

\[
\begin{align*}
P_0 \subseteq^P P_1 & \quad \text{Abstract component behavior domain } B_1^P \\
\llbracket P_0 \rrbracket_{B_0^P} & \subseteq A_P^{-1}(\llbracket P_1 \rrbracket_{B_1^P}) \\
\text{Detailed component behavior domain } B_0^P
\end{align*}
\]

\[
\begin{align*}
Q_0 \subseteq^Q Q_1 & \quad \text{Abstract component behavior domain } B_1^Q \\
\llbracket Q_0 \rrbracket_{B_0^Q} & \subseteq A_Q^{-1}(\llbracket Q_1 \rrbracket_{B_1^Q}) \\
\text{Detailed component behavior domain } B_0^Q
\end{align*}
\]

Compositionality conditions

conclude \[ [M_0]^{B_0} \subseteq A^{-1}([M_1]^{B_1}) \]

using \[ [P_0]^{B_0} \subseteq A^{-1}(P_1^{B_1}) \] and \[ [Q_0]^{P_0} \subseteq A^{-1}(Q_1^{B_0}) \]

Behavior localization (projections)
\[ B_0 \downarrow P = B_0^P \quad B_1 \downarrow P = B_1^P \]

Abstraction function localization (projections)
\[ A \downarrow P = A^P \]

Commutative diagram

Centralized Development

Start with \( A \), localize to get \( A^P, A^Q \)

If localizations of \( A \) are \( A^P \) and \( A^Q \), then compositional heterogeneous abstraction via \( A \) holds

Decentralized Development

Start with \( A^P, A^Q \), globalize to get \( A \)

If globalizations of \( A^P \), \( A^Q \) are consistent (call it \( \bar{A} \)), then compositional heterogeneous abstraction via \( \bar{A} \) holds

Semantic assumptions as parameter constraints

Problem
• Semantic interdependencies across formalisms
• Consistency

Challenge
• Formal representation that is universal to all modeling formalisms

Approach
• interdependencies as an auxiliary constraint on parameters
• Find effective constraint on given model/spec. parameters (existential quantification)
• Use SMT solvers or theorem provers to prove consistency

References
Completing the picture: Semantic and structural hierarchies

Outline

- Introduction

- Theoretical aspects of multi-paradigm model-based design for CPS
  - Architecture modeling and structural analysis
  - Semantic analysis and heterogeneous verification
  - Compositional analysis

- Practical aspects of a multi-domain simulation platform
  - Graphical modeling of hybrid dynamics using Simulink and Stateflow

- Recap and conclusions
Modeling hybrid (discrete + continuous) dynamics graphically using Simulink and Stateflow
Hybrid dynamics arise in CPS models quite often

Example: clutch

- Need to model and orchestrate
  1. Continuous dynamics
  2. Discrete modes
  3. Mode switching
     - Guard conditions
     - State handoff

https://www.mathworks.com/help/simulink/slref/building-a-clutch-lock-up-model.html
Modeling hybrid dynamics [Option 1]: Entirely in Stateflow

Continuous-time Stateflow chart

sf_bounce
Can get cumbersome for complex ODE dynamics

+ Intuitive for discrete dynamics
- ‘hand-coding’, difficult to debug
Modeling hybrid dynamics [Option 2]: Entirely in Simulink

Explicit mode switching examples

Implicit mode switching examples
Modeling hybrid dynamics [Option 2]: Entirely in Simulink
State handoff considerations

\[
\begin{align*}
\text{Falling du:} & \\
\% \text{ derivatives} & \\
p = 10; & \\
v = 15; & \\
p \rightarrow \text{p_out} = p; & \\
v \rightarrow \text{v_out} = v; & \\
\end{align*}
\]

\[
\begin{align*}
[p \leq 0 \land v < 0] & \\
p = 0; & \\
v = -0.8 \times v; & \\
\end{align*}
\]

\text{sf_bounce}

\[
\begin{align*}
y &= x \\
\dot{x} &= u
\end{align*}
\]
Modeling hybrid dynamics [Option 2]: Entirely in Simulink

```
Lock Unlock Lock_ Locked
0 0 0 0
0 0 1 1
0 1 0 0
0 1 1 0
1 0 0 1
1 0 1 1
1 1 0 1
1 1 1 0
```

Truth table:

```
[0;1;0;0;1;1;1;0]
```

Parameters

- **Tin**
- **Engine Torque**
- **Tin**
- **Tin**
- **Tina**
- **Tina**
- **Veh_Speed**
- **Veh_Speed**
- **Veh_Speed**

- **Friction Model**
- **Friction Model**
- **Friction Model**
- **Friction Model**
- **Friction Model**

- **sl democlutch_if**
Modeling hybrid dynamics [Option 3]: Stateflow drives Simulink

+ Intuitive for continuous dynamics
+ Intuitive for discrete dynamics
- Intuitive for hybrid dynamics? Can do better
  - Signal lines between Simulink and Stateflow
  - State handoff

(previously) sf_clutch

(now) sf_clutch_enabled_subsystems
State handoff considerations

(previously) sf_clutch
(now) sf_clutch_enabled_subsystems
Simulink-based states in Stateflow

Simulink-based states in Stateflow

Choosing a regular state:

\[ \omega_i = \omega_e \]

Choosing a Simulink-based state:

\[ a_i = a_{in} \]

\[ a_{in} = a_{in} \]

\[ \omega_e = \omega_e \]

\[ T_{in} = T_{in} \]

\[ T_{in} = T_{in} \]

\[ T_{in} = T_{in} \]
Simulink-based states in Stateflow
Graphical remote state access

State Reader

State Writer

\[ y = x \]
\[ \dot{x} = u \]
Graphical and textual remote state access
Easy copy-paste workflow
Outline

- Introduction

- Theoretical aspects of multi-paradigm model-based design for CPS
  - Architecture modeling and structural analysis
  - Semantic analysis and heterogeneous verification
  - Compositional analysis

- Practical aspects of a multi-domain simulation platform
  - Graphical modeling of hybrid dynamics using Simulink and Stateflow

- Recap
Recap

- CPS have a global societal scale impact – challenges and opportunities
- Models are used in design and operation of complex CPS

- Heterogeneity due to multiple paradigms presents a research challenge
  - Architecture presents an anchoring framework and enables structural analysis
  - Behavior domain associations enable semantic analysis

- Particulars of bridging the gap across formalisms in a simulation platform
  - Discussed one specific connection between two specific formalisms
  - Many other interesting details across other formalisms
Simulink Architecture ➔ Simulink Model: Manual Step in 2010

Fix propagation

Behavior construction

Structure extraction
System Composer NEW PRODUCT

Model Browser

Editing Canvas

Property Inspector

Interface Editor

Simulink to architecture
Architecture to Simulink
Interesting connections across other formalisms

- **Messages**
  - Simulink (drives)
  - Stateflow (drives)
  - SimEvents,

- **Function calls**
  - SimEvents (calls)
  - Stateflow (calls)
  - Simulink,

- **MATLAB Function**
  - Stateflow (calls)
  - Simulink (uses)
  - MATLAB,

- **System Objects**
  - MATLAB (calls)
  - Simulink

- **Stateflow for MATLAB**
  - MATLAB (calls)
  - Stateflow

- **MATLAB DES Block**
  - SimEvents (uses)
  - MATLAB

- **DES Chart**
  - SimEvents (uses)
  - Stateflow
Acknowledgments

- Architectures and multi-model heterogeneous design and analysis
  - Ajinkya Bhave, Bruce Krogh, David Garlan, Ivan Ruchkin, Bradley Schmerl

- Graphical hybrid automata using Simulink and Stateflow
  - Srinath Avadhanula, Alongkrit Chutinan, Pieter Mosterman, Fu Zhang
References


https://arajhans.github.io
send email for unpublished papers: arajhans@mathworks.com