Formalizing Cyber–Physical System Model Transformation via Abstract Interpretation

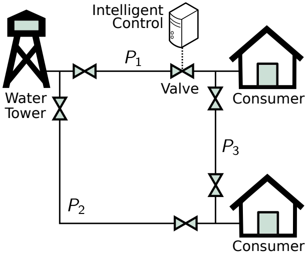
Natasha Jarus, Sahra Sedigh Sarvestani, and Ali Hurson

Department of Electrical and Computer Engineering
Missouri University of Science and Technology
Rolla, USA 65409
Email: {jarus, sedighs, hurson}@mst.edu

4 January 2019

Introduction

- The numerous requirements for *Cyber-physical systems* (CPSs) present challenges to system designers.
- An intelligent water distribution network (IWDN) must
  - be able to *supply* all its customers,
  - be *fault–tolerant* in the face of component failure,
  - be *secure* against physical and cyber attacks, and
  - achieve all these goals with *efficient* infrastructure.
- *Model–based design and evaluation* can facilitate the design process.
  - No single modeling formalism captures all requirements
  - Constructing multiple models is laborious and error-prone



Model Transformation

Model transformation converts a model of one type, e.g., a survivability model, to a model of another type, e.g., a reliability model.

Model transformation techniques:

- Reduce the *effort* required to model complex systems.
- Reduce *errors* in the modeling process.
- Enable *cross-domain application* of modeling approaches.

Research Contribution

Formalize system and model semantics, enabling *sound* system model transformation.

- Relating models to each other ensures that modeling assumptions are consistent across every model of a system.
- Identifying and understanding relationships between system attributes improves our understanding of complex system behavior and improves the accuracy of the models.

This work formalizes the research approach outlined in our previous work:

- Models abstract the semantics of a system—its structure and behavior.
- Defining sound mappings between system and model semantics enables sound model transformation.

Related Work: Model Transformation

- *Graph transformation:* Models are represented by graphs and transformed through graph rewriting.
  - Projects: AToM^3 (McGill, 2002), CHESS (Intecs, Italy, 2016), CONCERTO (Intecs, Italy, 2015)
- *Class inheritance transformation:* Models are instances of classes in an object-oriented class hierarchy.
  - Projects: OsMoSys (University of Napoli, 2007), SIMTHESys (University of Napoli, 2016)
- *Coalgebraic transformation:* Models are coalgebras in a lattice of possible transformations.
  - Projects: Rosetta (University of Kansas, 2012)

Open Problems

Our work is necessary because:

- Existing approaches, such as graph or class inheritance transformation, are difficult to apply or inapplicable to certain model types.
    - In particular, relating discrete- and continuous-time models is a challenge.
    - This challenge is critical to address for CPSs.

Open Problems

Our work is necessary because:

- Existing approaches, such as graph or class inheritance transformation, are difficult to apply or inapplicable to certain model types.
  - In particular, relating discrete- and continuous-time models is a challenge.
  - This challenge is critical to address for CPSs.
- Model transformation techniques must exhibit two attributes:
  - *Correctness:* the inferred model describes the same system as the source model
  - *Specificity:* the inferred model contains at least as much information as possible from the source model

  Of these attributes, correctness is only addressed by Rosetta, and to our knowledge no approaches address specificity.

Properties

- System semantics are represented in terms of properties that hold for the system.
- As models abstract system semantics, our representation of properties must allow models to not capture all aspects of a system.

We require the properties domain, $\mathbb{P}$**roperties**, to be a complete lattice.

- $p_1 \sqsubseteq p_2$ if p_1 is *more specific than* p_2. For example,
  - $p_1 \sqsubseteq p_2$ if p_1 requires a component's reliability to be 0.95 but p_2 constrains it within the range $[0.7, 1.0)$.
  - p_1 and p_2 are not comparable if p_1 knows the cyber components of a system and p_2 knows the physical components.
- $p_1 \sqcap p_2$, or the *meet* of p_1 and p_2, requires the system to meet the constraints of p_1 *and* p_2.
- $p_1 \sqcup p_2$, or the *join* of p_1 and p_2, requires the system to meet the constraints of p_1 *or* p_2.

Models

- As a model may not specify all aspects of a system, so an element of $\mathbb{P}$**roperties** may not specify a single model.
  - We can derive a single reliability model only when each component's reliability is known exactly.
- We can derive a *set of models* containing every possible model consistent with the known system properties.

We develop connections between $\mathbb{P}$**roperties** and a complete lattice of sets of models, $\mathbb{M}$**odel**, for each modeling formalism:

- $M_1 \subseteq M_2$ if M_1 is a *subset* of M_2 and therefore *more specific*—it has fewer possible models and therefore more system constraints.
- The *meet* $M_1 \cap M_2$ corresponds to the logical *and* of the possible system models.
- The *join* $M_1 \cup M_2$ corresponds to the logical *or* of the possible system models.
- The empty set $\emptyset$ corresponds to a modeling contradiction: no models of this system are possible!

Correctness

How do we know that an element of $\mathbb{P}$**roperties** or a set of models from some $\mathbb{M}$**odel** corresponds to the system we are modeling, s?

We suppose that we know the *correctness relation* $R_{\mathbb{P}}$ where $s \, R_{\mathbb{P}} \, p$ if $p \in \mathbb{P}$**roperties** correctly describes s.

A correctness relation for a lattice $\mathbb{L}$ has two properties:

- If some properties correctly describe a system, all properties less specific than them also describe that system.
  - If $s \, R_{\mathbb{L}} \, l_1$ and $l_1 \sqsubseteq l_2$, then $s \, R_{\mathbb{L}} \, l_2$.
- If several properties describe a system, the conjunction of them also describes that system.
  - If $s \, R_{\mathbb{L}} \, l$ for all $l \in \mathbf{L} \subseteq \mathbb{L}$, then $s \, R_{\mathbb{L}} \bigsqcap \mathbf{L}$.

The connections we build between $\mathbb{P}$**roperties** and our $\mathbb{M}$**odel** domains *induce* a correctness relation for each modeling formalism.

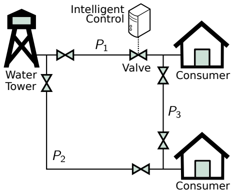
Abstraction and Concretization

We derive, or *concretize*, system properties from a model with the concretization operator $\gamma : \mathbb{M}\textbf{odel} \to \mathbb{P}\textbf{roperties}$.

$$\begin{tikzcd}
\begin{bmatrix} p & 0 & q & 0 \\ 0 & p & 0 & q \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \arrow[d, "\text{Concretization}"] \\
\textbf{Dependencies}
\end{tikzcd}$$

We *abstract* models from system properties with the abstraction operator $\alpha : \mathbb{P}\textbf{roperties} \to \mathbb{M}\textbf{odel}$.

$$\begin{tikzcd}
\textbf{Components} \arrow[d, "\text{Abstraction}"] \\
{}
\end{tikzcd}$$



Correctness of Abstraction and Concretization

- Abstraction may lose some properties of the system—those not captured by the modeling formalism—but cannot add any new properties:
  - $p \sqsubseteq (\alpha \circ \gamma)(p)$
- Concretization must completely capture the system properties given by a model:
  - $(\gamma \circ \alpha)(M) \sqsubseteq M$
  - (in practice, the $\sqsubseteq$ above is often strict equality.)

This relationship allows us to *induce a correctness relation* on $\mathbb{M}$**odel** by $s \, R_{\mathbb{M}} \, M \iff s \, R_{\mathbb{P}} \, \gamma(M)$.

The operators α and γ can be used to build model transformation operators and the relationship $R_{\mathbb{M}}$ allows us to prove those transformations are sound.

Model Transformation

The goal of model transformation is to define a *sound mapping* from one modeling formalism to another: $\tau^{\mathbb{M}_2}_{\mathbb{M}_1} : \mathbb{M}\textbf{odel}_\mathbf{1} \to \mathbb{M}\textbf{odel}_\mathbf{2}$.

For example, given a correct set of reliability models R, we can produce a correct set of topology models $T = \tau^{\mathbb{T}\textbf{op}}_{\mathbb{R}\textbf{el}}$.

We define $\tau^{\mathbb{M}_2}_{\mathbb{M}_1}(M_1)$ by $(\alpha_{\mathbb{M}_2} \circ \gamma_{\mathbb{M}_1})(M_1)$:

- Concretize system properties from the given set of models.
- Abstract a set of models of the desired formalism.

$$\begin{tikzcd}
s & s & s \\
\vdots & \vdots & \vdots \\
R_{\mathbb{M}_1} \arrow[r, phantom, "\Longrightarrow"] & R_{\mathbb{P}} \arrow[r, phantom, "\Longrightarrow"] & R_{\mathbb{M}_2} \\
\vdots & \vdots & \vdots \\
\mathbb{M}\textbf{odel}_\mathbf{1} \arrow[r, "\gamma_{\mathbb{M}_1}"'] \arrow[rr, bend right, "\tau^{\mathbb{M}_2}_{\mathbb{M}_1}"'] & \mathbb{P}\textbf{roperties} \arrow[r, "\alpha_{\mathbb{M}_2}"'] & \mathbb{M}\textbf{odel}_\mathbf{2}
\end{tikzcd}$$

Selection

Up to this point, we have been focused on relating and transforming sets of models.

How do we transform a single model into a single model in a different formalism?

We can transform a single input model m_1 into a set of result models $M_2 = \tau_{\mathbb{M}_1}^{\mathbb{M}_2}(\{m_1\})$.
To *select* a single model from M_2:

- If $M_2 = \emptyset$, the input data contained a modeling contradiction and no result is produced.
- If M_2 contains a single model m_2, that is the result of the transformation.
- Otherwise, the designer must provide information about the system not present in the input model to *select* a single model from the set of possible models.
  - For instance, a topology model does not specify component reliability.

Specificity

- *More comprehensive* sets of properties yield *greater specificity*.
- *Greater specificity* leads to *fewer possible* inferred models.
- The selection process requires incorporation of information not present in the initial model.
- This additional information can be incorporated into a more specific element of $\mathbb{P}\mathbf{roperties}$ for use in future transformations.

$$\begin{tikzcd}
\mathbf{Model_1} \arrow[d, swap, "m \mapsto \{m\}"] & & \mathbf{Model_2} \\
\mathbb{M}\mathbf{odel_1} \arrow[dr, swap, "\text{Concretization}"] & & \mathbb{M}\mathbf{odel_2} \arrow[u, swap, "\text{Selection}"] \\
& \mathbb{P}\mathbf{roperties} \arrow[ur, swap, "\text{Abstraction}"] &
\end{tikzcd}$$

Tag–Options Lattices

When defining $\mathbb{P}$**roperties**, we often need to associate some parts of the system (*tags*) with sets of possible properties (*options*).

For example, system *components* may have an associated *reliability*.

We formalize this relationship with a *tag–options lattice*. The elements of these lattices are pairs (T, f):

- T is the set of known tags
  - e.g., component names
- f is a function mapping each known tag to a corresponding set of known options
  - e.g., $f(\text{pipe } 1) = [0.7, 1.0)$ constrains the reliability of pipe 1 to the range $[0.7, 1.0)$.

These elements are ordered by specificity: the more tags and the fewer options associated with each tag, the more specific the properties.

Conclusions

- We presented an approach to relating models of systems to properties of systems.
- This approach formalizes correctness as a relationship between a system and properties of that system.
- The relationship between models and properties preserves this correctness relationship.
- Such relationships also define sound model transformations.
- This research can accelerate advances in design and analysis of complex systems by enabling cross-domain transfer of knowledge.

Future work will take several directions:

- We are developing a properties domain and formalizing a reliability model formalism and a topology model formalism.
- We plan to extend this work to other modeling formalisms, including continuous– and discrete–time models.

Graph Transformation

- Formulate models as graphs and model transformation as rewriting of the graphs.
- Each model type has a meta-model that describes how its graph can be transformed to graphs of other model types.
- Applies to many formalisms, including Petri nets and Markov chains, but not all.
- Projects: AToM^3, CHESS, CONCERTO
  - CHESS and CONCERTO are more focused on modeling multi-core computer systems.

back

Class Inheritance Transformation

- Each model type corresponds to a class in an object-oriented class hierarchy.
- Models are instances of their type's class.
- Transformation occurs by using inheritance principles to convert a model from one type to another.
- Easy to travel 'up' the class hierarchy; hard to travel back 'down'.
- Projects: OsMoSys, SIMTHESys

back

Coalgebraic Transformation

- Each modeling formalism is described as a *coalgebra* – a mathematical system useful for describing transitions among states.
- The coalgebras are placed in a lattice to provide a structure for determining which transformations can be performed.
- Can relate different types of models of the same system, such as a model of system functionality and a model of system power consumption.
- Projects: Rosetta

back

Related Work: Model Composition

- A hierarchical approach is most commonly taken.
- Build subsystem-level models and link them together into system-level models.
  - Subsystem models may be of different types
  - Example: composing a discrete-time model of control software and a continuous-time model of a water valve
- Projects: Ptolemy (Berkeley, 2018), Möbius (University of Illinois, 2015)
- Both projects typically involve models of functional attributes and are created to facilitate system simulation.