Models, Metamodels, and Model Transformation for Cyber-Physical Systems

Nathan Jarus, Sahra Sedigh Sarvestani, and Ali Hurson
Department of Electrical and Computer Engineering
Missouri University of Science and Technology, Rolla, MO 65409, USA

September 8, 2016

Outline

Cyber-Physical Systems

- Tight integration between a large-scale physical network and a cyber network that monitors and controls the physical network.
- Can be used to build sustainable infrastructure:
  - Make existing physical networks more dependable.
  - Reduce the physical resources needed to build new infrastructure.
- Examples:
  - Smart Power Grids
  - Intelligent Water Distribution Networks
  - Smart Transportation Systems

Modeling

- Designing cyber-physical systems requires constructing multiple models of each design
  - Performance models
  - Dependability models (reliability, resilience, survivability, etc.)
- Problems with modeling
  - How do you avoid re-making every model every time the system design changes?
  - How do you make sure each model is working off the same assumptions?

Model Transformation with Metamodeling

- Model transformation converts a model of one type (e.g. a performance model) to a model of another type (e.g. a reliability model)
- Can reduce workload and prevent mistakes when modeling complex systems
- Goals of a transformation technique:
  - *Correctness*: the generated model should model the same system as the source model
  - *Specificity*: the generated model should contain as much information from the source model as possible

Hierarchical Model Composition

- Build small models and link them together into complete system models
- Hierarchical models can contain heterogeneous submodels
- Not model transformation per se; no way to start with one model and derive a different one
- Projects: Ptolemy, Möbius

Graph Transformation

- Formulate models as graphs and model transformation as graph rewriting
- Each model type has a meta-model that describes how its graph can be transformed to graphs of other model types
- Projects: AToM^3, CHESS, CONCERTO
- (CHESS and CONCERTO are more focused on modeling multi-core computer systems)

Class Inheritance Transformation

- Each model type corresponds to a class in a class hierarchy
- Models are instances of their type's class
- Transformation occurs by using inheritance prinicples to cast models between different types
- Projects: OsMoSys, SIMTHESys

Coalgebraic Transformation

- Each modeling formalism is described as a coalgebra, a mathematical system useful for describing arbitrary transitions on an arbitrary state
- The coalgebras are placed in a lattice to provide a structure for determining how transformations are 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

Our Contribution

A model transformation technique based on Abstract Interpretation, a program analysis technique

- Models are viewed as syntactic representations of systems
- Properties can be *abstracted* from a model
- Semantically equivalent models can be *concretized* from a set of properties

Our Contribution

A model transformation technique based on Abstract Interpretation, a program analysis technique

- Models are viewed as syntactic representations of systems
- Properties can be *abstracted* from a model
- Semantically equivalent models can be *concretized* from a set of properties

We can show that this technique is both correct and specific.

Order Relationships

Definition

A *partially ordered set* or *poset* $(L, \sqsubseteq)$ is a set L and an order relation $\sqsubseteq: L \times L \mapsto \{\textbf{true}, \textbf{false}\}$ (read 'less than or equal') that is

1. *Reflexive*: $l \sqsubseteq l, \forall l \in L$
2. *Transitive*: If $l_1 \sqsubseteq l_2$ and $l_2 \sqsubseteq l_3$, then $l_1 \sqsubseteq l_3, \forall l_1, l_2, l_3 \in L$
3. *Anti-symmetric*: If $l_1 \sqsubseteq l_2$ and $l_2 \sqsubseteq l_1$, then $l_1 = l_2, \forall l_1, l_2 \in L$

Meet and Join

Definition (Upper Bound)

A set $Y \subseteq L$ has $l \in L$ as an *upper bound* if $y \sqsubseteq l, \forall y \in Y$. l is a *least upper bound* if, for any upper bound l', $l \sqsubseteq l'$. We denote the least upper bound by $\bigsqcup Y$ (sometimes called the *meet* of Y). $\bigsqcup\{l_1, l_2\}$ can also be written as $l_1 \sqcup l_2$.

Definition (Lower Bound)

l is a lower bound of Y if $l \sqsubseteq y, \forall y \in Y$. The *greatest lower bound* is defined analogously to the least upper bound. It is denoted $\bigsqcap Y$ and sometimes called the *join* of Y.

Order-Preserving Functions

We can relate two posets P_1 and P_2 through functions with certain properties:

Definition

A function $f : P_1 \mapsto P_2$ between posets $(P_1, \sqsubseteq_1)$ and $(P_2, \sqsubseteq_2)$ is *monotone* (or *order-preserving*) if

$$p_1 \sqsubseteq_1 p_2 \implies f(p_1) \sqsubseteq_2 f(p_2), \forall p_1, p_2 \in P_1$$

Lattices

Definition

A *complete lattice* L is a partially ordered set where $\bigsqcup Y$ and $\bigsqcap Y$ is defined for all $Y \subseteq L$.

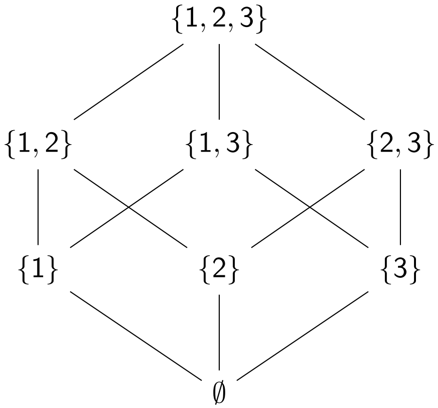
A consequence of 3.5 is that every complete lattice has a least element, $\bot = \bigsqcap L$, and a greatest element, $\top = \bigsqcup L$.

Powerset Lattices

A common complete lattice is the set of all subsets of a set S, called the powerset and denoted $\mathcal{P}(S)$. $(\mathcal{P}(S), \subseteq)$ forms a complete lattice with $\bigsqcup = \bigcup$ and $\bigsqcap = \bigcap$.

$\top = S$ and $\bot = \emptyset$

$\mathcal{P}(\{1,2,3\}) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$



Galois Connections

Definition

A *Galois connection* (P, α, γ, Q) between two posets $(P, \sqsubseteq_P)$ and $(Q, \sqsubseteq_Q)$, is a pair of monotone functions $\alpha : P \mapsto Q$ and $\gamma : Q \mapsto P$ such that

$$p \sqsubseteq_P (\gamma \circ \alpha)(p), \forall p \in P$$

$$(\alpha \circ \gamma)(q) \sqsubseteq_Q q, \forall q \in Q$$

- P is referred to as the *concrete domain* and Q as the *abstract domain*.
- α is called the *abstraction operator* and γ the *concretization operator*.

Galois Connections

Definition

A *Galois connection* (P, α, γ, Q) between two posets $(P, \sqsubseteq_P)$ and $(Q, \sqsubseteq_Q)$, is a pair of monotone functions $\alpha : P \mapsto Q$ and $\gamma : Q \mapsto P$ such that

$$p \sqsubseteq_P (\gamma \circ \alpha)(p), \forall p \in P$$

$$(\alpha \circ \gamma)(q) \sqsubseteq_Q q, \forall q \in Q$$

- P is referred to as the *concrete domain* and Q as the *abstract domain*.
- α is called the *abstraction operator* and γ the *concretization operator*.
- We will use Galois Connections to abstract properties from models and concretize models from properties.

Properties of Galois Connections

- If you have one side of a Galois Connection, you can construct the other:
  - α uniquely determines γ by $\gamma(q) = \bigsqcup\{p : \alpha(p) \sqsubseteq_Q q\}$
  - γ uniquely determines α by $\alpha(p) = \bigsqcap\{q : p \sqsubseteq_P \gamma(q)\}$

Properties of Galois Connections

- If you have one side of a Galois Connection, you can construct the other:
  - α uniquely determines γ by $\gamma(q) = \bigsqcup\{p : \alpha(p) \sqsubseteq_Q q\}$
  - γ uniquely determines α by $\alpha(p) = \bigsqcap\{q : p \sqsubseteq_P \gamma(q)\}$
- Repeated abstraction and concretization is 'free'; i.e. it does not lose precision: $\alpha \circ \gamma \circ \alpha = \alpha$ and $\gamma \circ \alpha \circ \gamma = \alpha$

Systems, models, and properties

Let's consider a system **S**. We write

- $\mathbf{S} \vdash m$ if the model $m \in M$ describes **S**
- $\mathbf{S} \vdash m_1 \rightsquigarrow m_2$ if m_1 can be transformed to m_2 (while still describing **S**)
- We don't require $\rightsquigarrow$ to be easily calculable or even a function in the mathematical sense

Systems, models, and properties

Let's consider a system $\mathbf{S}$. We write

- $\mathbf{S} \vdash m$ if the model $m \in M$ describes $\mathbf{S}$
- $\mathbf{S} \vdash m_1 \rightsquigarrow m_2$ if m_1 can be transformed to m_2 (while still describing $\mathbf{S}$)
- We don't require $\rightsquigarrow$ to be easily calculable or even a function in the mathematical sense
- $\mathbf{S} \vdash p$ if the set of properties $p \in P$ hold for $\mathbf{S}$
- $\mathbf{S} \vdash p_1 \rhd p_2$ if the properties p_1 can be transformed into properties p_2 while still describing $\mathbf{S}$
- We **do** require $\rhd$ to be deterministic!

Idea: properties are easier to reason about than models.

Relating Models and Properties

- If properties p hold for a model m, we write $m \mathrel{R} p$
- $\rhd$ preserves R: if $\mathbf{S} \vdash m_1 \rightsquigarrow m_2$, $\mathbf{S} \vdash c_1 \rhd p_2$, and $m_1 \mathrel{R} p_1$, then $m_2 \mathrel{R} p_2$

$$\begin{array}{ccccc}
\mathbf{S} & \vdash & m_1 & \rightsquigarrow & m_2 \\
& & \| & & \| \\
& & R & \Longrightarrow & R \\
& & \| & & \| \\
\mathbf{S} & \vdash & p_1 & \rhd & p_2
\end{array}$$

Relating Models and Properties

- Even if we know that there exists an m_2 such that $m_2 \, R \, p_2$, it may be hard to find such a m_2
- We can use lattices to provide structure that helps us approximate m_2
- Let's view $(P, \sqsubseteq)$ is a lattice and R is constrained by two conditions:
  - If $m \, R \, p_1$ and $p_1 \sqsubseteq p_2$, then $m \, R \, p_2$
  - If $m \, R \, p$ for all $p \in P' \sqsubseteq P$, then $m \, R \, \bigsqcap P'$
- In other words, the more specific p is, the better; and there exists a best p to describe any model

Representation Functions

- We define a representation function $\beta : M \mapsto P$ that maps $m \in M$ to the most specific $p \in P$ describing it
- β is also preserved by $\rhd$: if $\mathbf{S} \vdash m_1 \leadsto m_2$, $\mathbf{S} \vdash p_1 \rhd p_2$, and $\beta(m_1) \sqsubseteq p_1$, then $\beta(m_2) \sqsubseteq p_2$

$$\begin{array}{ccccc} \mathbf{S} & \vdash & m_1 & \leadsto & m_2 \\ & & \beta\big\downarrow & \Longrightarrow & \beta\big\downarrow \\ & & \sqsubseteq & & \sqsubseteq \\ \mathbf{S} & \vdash & p_1 & \rhd & p_2 \end{array}$$

Introducing Correctness

- We can show β is correct by constructing a Galois Connection between $\mathcal{P}(M)$ and P
- $\gamma(p) = \{m \in M : \beta(m) \sqsubseteq p\}$
- $\alpha(M') = \bigsqcup\{\beta(m) : m \in M'\}$
- Intuitively, concretizing properties gives you the models those properties hold for
- Abstracting a set of models gives the most specific set of properties that hold for all those models

$$
\begin{tikzcd}
\mathcal{P}(M) \arrow[rr, shift left, "\alpha"] & & P \arrow[ll, shift left, "\gamma"] \\
M \arrow[u, "\{\cdot\}"] \arrow[urr, "\beta"] & &
\end{tikzcd}
$$

Applying Abstract Interpretation to Model Transformation

- Let's suppose we have two model types, M_1 and M_2, and associated Galois connections $(\mathcal{P}(M_1), \alpha_1, \gamma_1, P)$ and $(\mathcal{P}(M_2), \alpha_2, \gamma_2, P)$
- If we have $\mathbf{S} \vdash m_1$ for $m_1 \in M_1$, we can abstract properties from m_1 by taking $p_1 = \beta_1(m_1)$

Applying Abstract Interpretation to Model Transformation

- Let's suppose we have two model types, M_1 and M_2, and associated Galois connections $(\mathcal{P}(M_1), \alpha_1, \gamma_1, P)$ and $(\mathcal{P}(M_2), \alpha_2, \gamma_2, P)$
- If we have $\mathbf{S} \vdash m_1$ for $m_1 \in M_1$, we can abstract properties from m_1 by taking $p_1 = \beta_1(m_1)$
- We can concretize p_1 into models in M_2 by letting $M_2' = \gamma_2(p_1)$
- This gives us a set of M_2 models where the properties p_1 hold

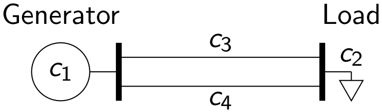
Specificity

- Ideally, concretizing any set of properties will result in a single model
- This is not usually the case, since a destination model may require an assumption about the system not specified in the source model
- The function $\sigma : \mathcal{P}(M) \mapsto M$ allows the user to select one model from a set of transformed models
- E.g. $\sigma(M_2') = m_2$ introduces additional assumptions needed to get a unique model as the result of a transformation

Specificity

- Ideally, concretizing any set of properties will result in a single model
- This is not usually the case, since a destination model may require an assumption about the system not specified in the source model
- The function $\sigma : \mathcal{P}(M) \mapsto M$ allows the user to select one model from a set of transformed models
- E.g. $\sigma(M_2') = m_2$ introduces additional assumptions needed to get a unique model as the result of a transformation
- We can capture the properties of these additional assumptions by taking $p_2 = \beta_2(m_2)$
- These properties can be combined with the properties from m_1 by taking $p = p_1 \sqcup p_2$

Topological Model



$$
\begin{aligned}
\text{components} &\subseteq \textbf{component} \\
\text{attributes} &\subseteq \textbf{component} \times \{\texttt{generator}, \texttt{load}, \texttt{line}\} \\
\text{links} &\subseteq \textbf{component} \times \textbf{component} \times \textbf{component} \\
\text{neighbors} &\subseteq \textbf{component} \times \mathcal{P}(\textbf{component})
\end{aligned}
$$

$$
\begin{aligned}
\text{components}(T) &= \{c_1, c_2, c_3, c_4\} \\
\text{attributes}(T) &= \{(c_1, \texttt{generator}), (c_2, \texttt{load}), \\
&\qquad (c_3, \texttt{line}), (c_4, \texttt{line})\} \\
\text{links}(T) &= \{(c_1, c_3, c_2), (c_1, c_4, c_2)\} \\
\text{neighbors}(T) &= \{(c_1, \{c_3, c_4\}), (c_2, \{c_3, c_4\})\}
\end{aligned}
$$

MIS model

	Components	
States	c_3	c_4
S_1	1	1
S_2	1	0
S_3	0	1
S_4	0	0

$$\Pi_0 = [1, 0, 0, 0]$$

$$u = [1, 1, 1, 0]$$

$$P_{c_3} = \begin{bmatrix} p_L & 0 & q_L & 0 \\ 0 & p_L & 0 & q_L \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P_{c_4} = \begin{bmatrix} p_L & q_L & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & p_L & q_L \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R = \Pi_0^T * P_{c_3} * P_{c_4} * u = p_L^2 + 2p_L q_L$$

Properties of MIS model

$$
\begin{aligned}
\text{components} &\subseteq \textbf{component} \\
\text{attributes} &\subseteq \textbf{component} \times [0,1] \\
\text{functional\_states} &\subseteq \mathcal{P}(\textbf{component}) \\
\text{initial\_probability} &\subseteq \mathcal{P}(\textbf{component}) \times [0,1]
\end{aligned}
$$

$$
\begin{aligned}
\text{components}(M) &= \{c_3, c_4\} \\
\text{attributes}(M) &= \{(c_3, p_L), (c_4, p_L)\} \\
\text{functional\_states} &= \{\{c_3, c_4\}, \{c_3\}, \{c_4\}\} \\
\text{initial\_probability} &= \{(\{c_3, c_4\}, 1)\}
\end{aligned}
$$

MIS model generated from Topology model properties

$$\Pi_0 = [\mathfrak{s}_1, \cdots, \mathfrak{s}_{15}]$$
$$u = [\mathfrak{u}_1, \cdots, \mathfrak{u}_{15}]$$
$$P_{c_1} = \begin{bmatrix} \mathfrak{p}_{\mathfrak{c}_1} & 0 & \cdots & \mathfrak{q}_{\mathfrak{c}_1} & 0 & \cdots \\ \vdots & \ddots & & \vdots & \ddots & \\ 0 & \cdots & \mathfrak{p}_{\mathfrak{c}_1} & 0 & \cdots & \mathfrak{q}_{\mathfrak{c}_1} \\ 0 & \cdots & 0 & 1 & 0 & \cdots \\ \vdots & & \vdots & \vdots & \ddots & \\ 0 & \cdots & 0 & 0 & \cdots & 1 \end{bmatrix}$$
$$R = \Pi_0^T * P_{c_1} * P_{c_2} * P_{c_3} * P_{c_4} * u$$

- If σ_{MIS} sets $\mathfrak{p}_{\mathfrak{c}_1} = \mathfrak{p}_{\mathfrak{c}_2} = 1$, then $\mathfrak{u}_5 = \cdots = \mathfrak{u}_{16} = 0$ and $\mathfrak{s}_5 = \cdots = \mathfrak{s}_{16} = 0$
- Binding $\mathfrak{p}_{\mathfrak{c}_3} = \mathfrak{p}_{\mathfrak{c}_4} = p_L$, $\mathfrak{s}_1 = 1$, $\mathfrak{u}_1 = \mathfrak{u}_2 = \mathfrak{u}_3 = 1$, and $\mathfrak{u}_4 = 0$ suffice to generate an MIS model equivalent to our first MIS model.

Conclusions

- We present a system for transforming models based on Abstract Interpretation
- This approach is both correct and specific
- It can be used to generate cross-domain models and nonfunctional models from functional models or vice-versa
- This power makes correctly modeling sustainable infrastructure easier