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

November 9, 2016

Introduction

- *Cyber-physical systems* (CPSs) are characterized by tight integration between a 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 grids
  - Intelligent water distribution networks
  - Intelligent transportation systems

Motivation

- Designing cyber-physical systems requires constructing multiple models of each design:
  - Performance models
  - Dependability models (reliability, resilience, survivability, etc.)
- Modeling challenges:
  - How do you avoid the need to reconstruct each model every time the system design changes?
  - How do you make sure each model is operating from the same assumptions?

Related Work

Related work in metamodeling falls into two categories:

- Model composition
- Model transformation

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.
- Commonly mischaracterized as model transformation, but does not start with one model and derive a different one.
- Projects: Ptolemy, Möbius

Model Transformation with Metamodeling

- Model transformation converts a model of one type, e.g., a survivability model, to a model of another type, e.g., a reliability model.
- Can facilitate and prevent mistakes when modeling complex systems.
- Typically carried out in three approaches:
  - Graph transformation
  - Class inheritance transformation
  - Coalgebraic transformation

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.
- 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 principles to convert a model from one type to another.
- Projects: OsMoSys, SIMTHESys

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

Related Model Composition and Transformation Tools

Project	Target Field	Applicable To	Approach
Ptolemy	Cyber-physical systems	Anything with a computation model	Composition
Möbius	Complex network systems	Performance and dependability	Composition
AToM^3	General modeling	Anything that can be metamodeled	Graph rewriting
CONCERTO	Model-driven engineering	Functional models	Graph rewriting
OsMoSys	General modeling	Anything that can be metamodeled	Class Inheritance
SIMTHESys	General modeling	Anything that can be metamodeled	Class Inheritance
Rosetta	General modeling	Discrete event systems	Coalgebras

Open Problems

Our work is necessary because:

- Some approaches, such as graph or class inheritance transformation, are difficult to apply to certain model types.
  - In particular, relating discrete- and continuous-time models is a challenge.

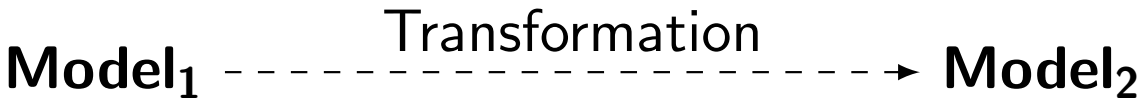
Open Problems

Our work is necessary because:

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

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

Transformation



Transformation

- How do we define transformations between very different models?
  - For example, transforming a model of a nonfunctional attribute into a performance model
- The task is complicated by a potential lack of transitivity:
  - Transforming $\mathbf{Model_1} \rightarrow \mathbf{Model_2}$, then $\mathbf{Model_2} \rightarrow \mathbf{Model_3}$ may not be equivalent to directly transforming $\mathbf{Model_1} \rightarrow \mathbf{Model_3}$.

Original Research Contribution

Creation of a model transformation method based on *abstract interpretation*.

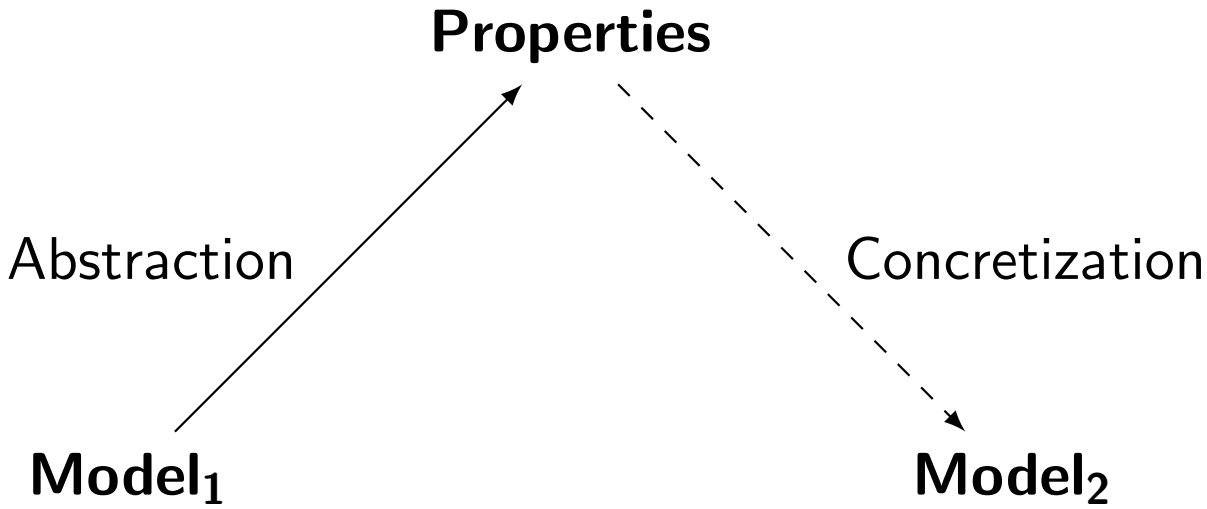
- We view models as *syntactic representations* of a system.
- We *abstract* properties from a model.
- We *concretize* semantically equivalent models from a set of properties.
- We show that this technique is both correct and specific.

Original Research Contribution

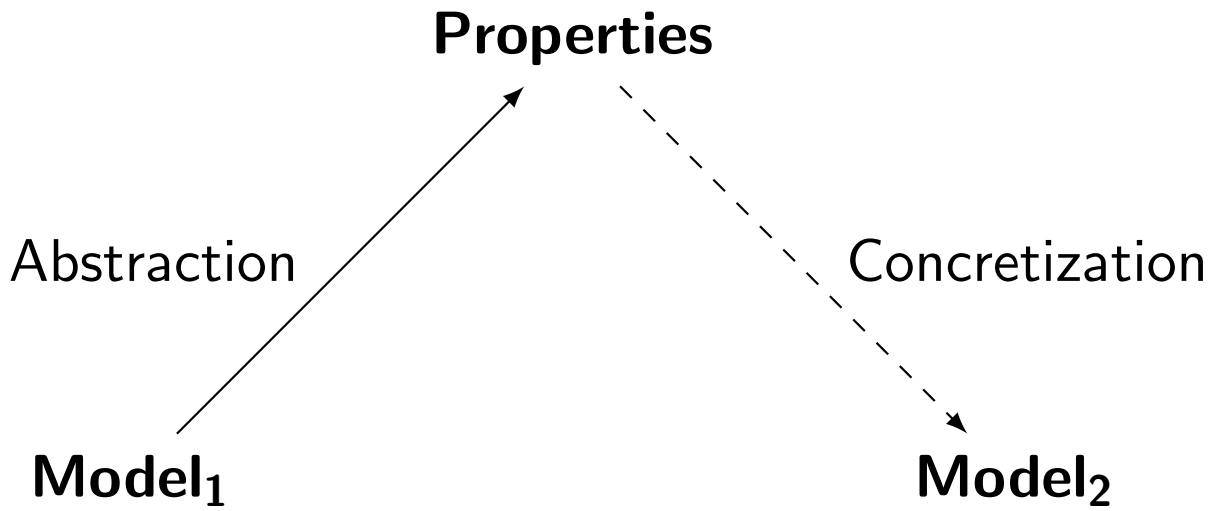
Creation of a model transformation method based on *abstract interpretation*.

- We view models as *syntactic representations* of a system.
- We *abstract* properties from a model.
- We *concretize* semantically equivalent models from a set of properties.
- We show that this technique is both correct and specific.

Abstraction

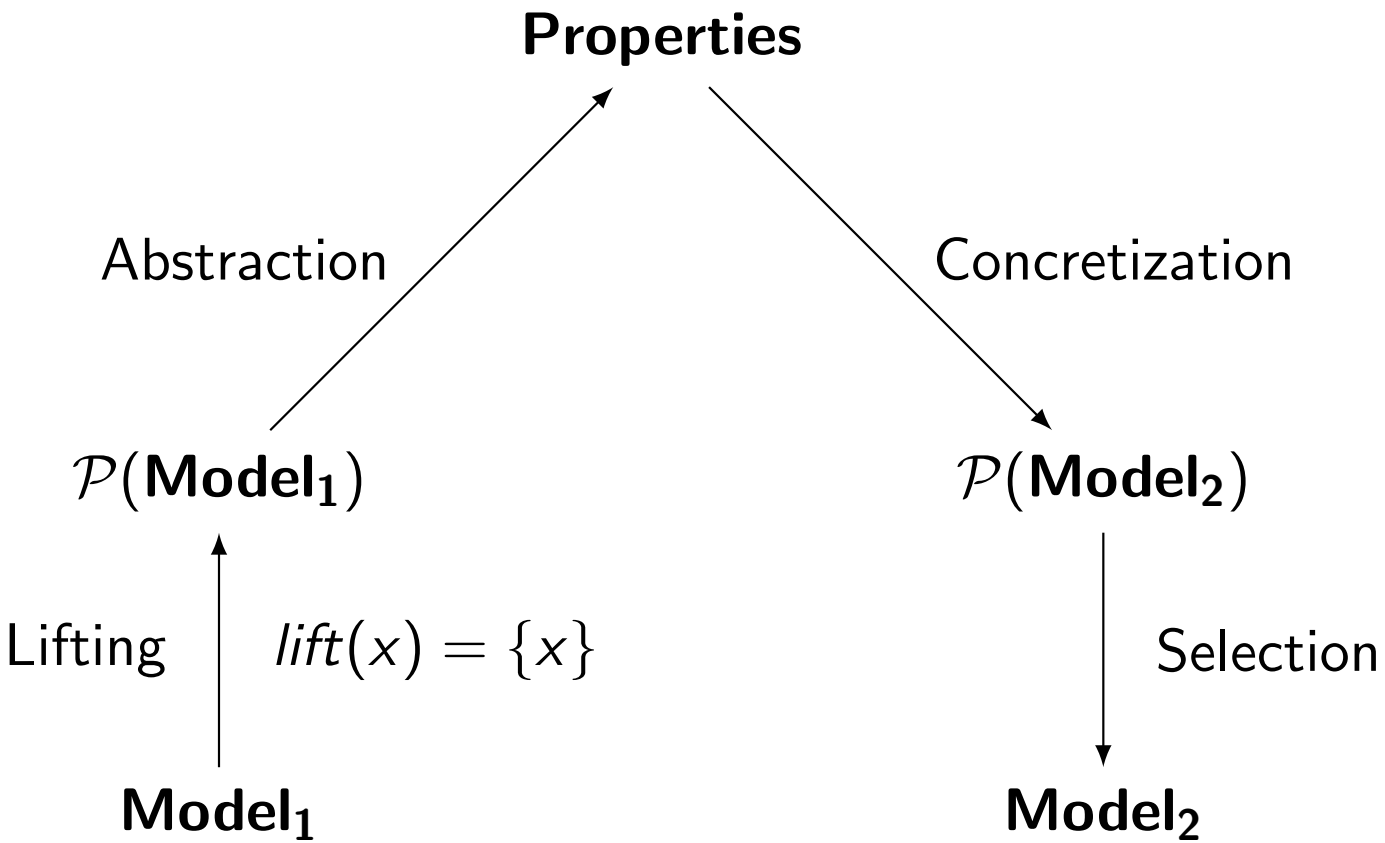


Abstraction

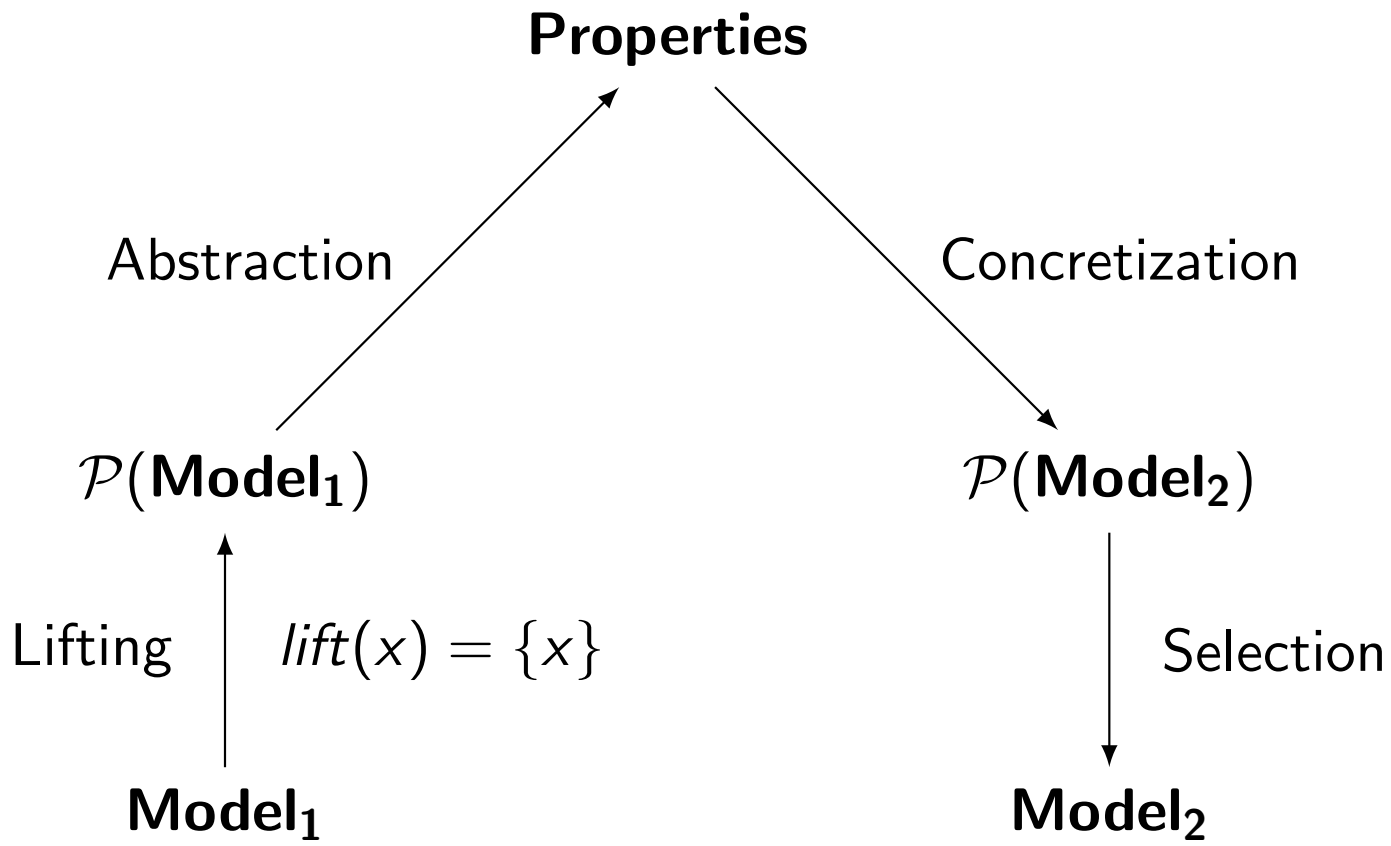


- It is always feasible to abstract system properties from a given model.
- However, it may be infeasible to construct a model of a different type from these properties.

Abstraction



Abstraction



To generate a model of type 2 from a given model of type 1:

- We first extract the properties of the type 1 model and concretize these properties to generate a set of type 2 models.
- A single type 2 model is selected from the resulting set.

Correctness and Specificity

Correctness:

- Need to show that the transformation process does not produce results that do not follow from the input model.
- If the transformation is from one model type to the same model type, abstracting properties from a model and then concretizing those properties will produce a set of models containing the input model.

Specificity:

- Smaller sets of models are more specific.
- Larger sets of properties are more specific.
- The selection process requires consideration of information not present in the initial model.

Order Relationships

- Ordering sets of models and properties based on their specificity is essential to formalizing model transformation.
- We write $M_1 \sqsubseteq_M M_2$ if the set of models M_1 is more specific, i.e., smaller, than M_2.
- Likewise, $P_1 \sqsubseteq_P P_2$ implies that the properties in P_1 include detail beyond what is present in P_2.

Order Relationships

- Ordering sets of models and properties based on their specificity is essential to formalizing model transformation.
- We write $M_1 \sqsubseteq_M M_2$ if the set of models M_1 is more specific, i.e., smaller, than M_2.
- Likewise, $P_1 \sqsubseteq_P P_2$ implies that the properties in P_1 include detail beyond what is present in P_2.
- These relationships form a *partial order* – we do not require each pair of sets of models or properties to be comparable.
- For instance, the sets of models $\{m_1\}$ and $\{m_2\}$ – each of which has a single member – are equally specific, but not equal to each other.

Galois Connections

Correctness and specificity can be mathematically formalized using *Galois connections*.

A *Galois connection* $(\mathcal{P}(M), \alpha, \gamma, P)$ between two sets with partial orders $(\mathcal{P}(M), \sqsubseteq_M)$ and $(P, \sqsubseteq_P)$, is a pair of order-preserving functions $\alpha : M \mapsto P$ and $\gamma : P \mapsto M$ such that

$$m \sqsubseteq_M (\gamma \circ \alpha)(m), \forall m \in \mathcal{P}(M)$$

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

- $\mathcal{P}(M)$ is referred to as the *concrete domain* and P as the *abstract domain*.
- α is called the *abstraction operator* and γ the *concretization operator*.

Properties of Galois Connections

- The abstraction operator can be inferred from the concretization operator, or vice versa:
  - α uniquely determines γ by $\gamma(p) = \bigsqcup\{m : \alpha(m) \sqsubseteq_P p\}$
  - γ uniquely determines α by $\alpha(m) = \bigsqcap\{p : m \sqsubseteq_M \gamma(p)\}$
- Repeated abstraction and concretization cannot introduce additional models or properties:
  - $\alpha \circ \gamma \circ \alpha = \alpha$
  - $\gamma \circ \alpha \circ \gamma = \alpha$
- Therefore, formalizing model transformation with Galois connections clearly defines the relationship between models and properties.

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.

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 require $\rhd$ to be deterministic.

Idea: properties are easier to reason about than models.

Abstract Interpretation

- 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 $\mathbf{S} \vdash m_1 \rightsquigarrow m_2$, $\mathbf{S} \vdash p_1 \rhd p_2$, and $\beta(m_1) \sqsubseteq p_1$, then $\beta(m_2) \sqsubseteq p_2$.

$$\begin{array}{ccccc} \mathbf{S} & \vdash & m_1 & \rightsquigarrow & m_2 \\ & & \beta \downarrow & \Longrightarrow & \beta \downarrow \\ & & \sqcap\mid & & \sqcap\mid \\ \mathbf{S} & \vdash & p_1 & \rhd & p_2 \end{array}$$

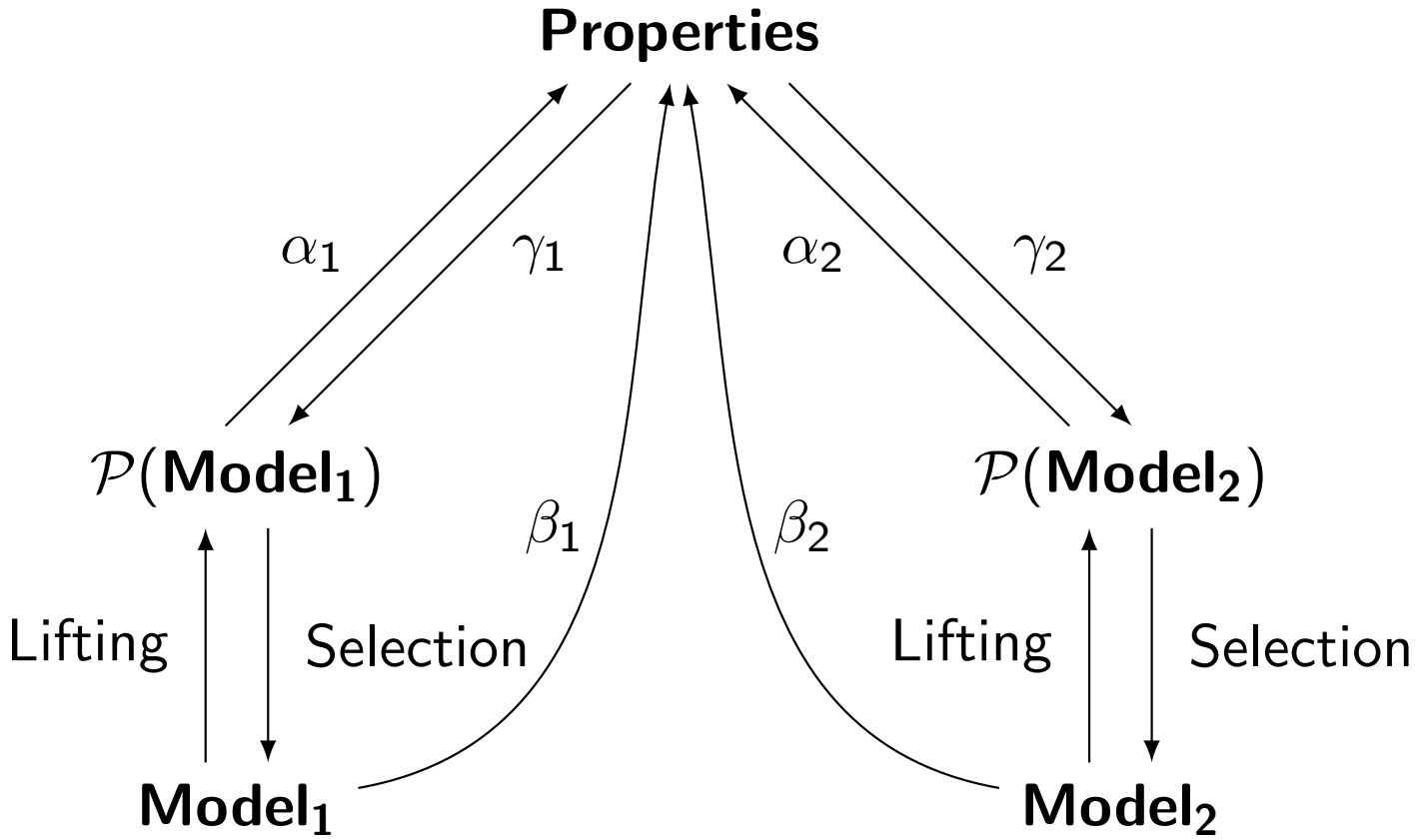
Abstract Interpretation

- 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 $\mathbf{S} \vdash m_1 \rightsquigarrow m_2$, $\mathbf{S} \vdash p_1 \rhd p_2$, and $\beta(m_1) \sqsubseteq p_1$, then $\beta(m_2) \sqsubseteq p_2$.

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

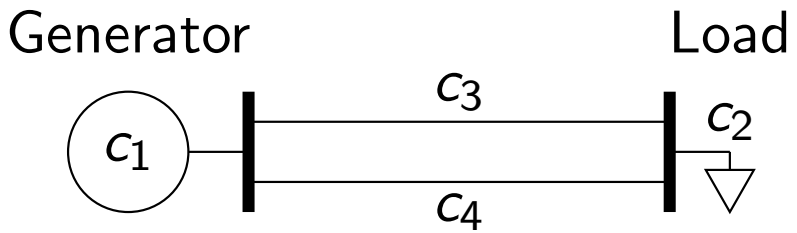
- We can show β is correct by constructing a Galois Connection between $\mathcal{P}(M)$ and P.
- Intuitively, concretizing properties gives you the set of models for which those properties hold.
- Abstracting a set of models gives the most specific set of properties that hold for all models in that set.

Formal Model Transformation



- We can transform between model types given that each model has a Galois connection with **Properties**.
- Following a transformation, we can follow a model-aware selection process to introduce new modeling assumptions.

Example: Topological Model



- This model, M_T, shows the topology of a power grid.
- We seek to derive other models from this topological model.

Example: Properties of the Topological Model

For example, we could extract properties of the following types:

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

For our model M_T, the corresponding properties P_T would be:

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

Example: Direct Construction of a Specific MIS Reliability Model

- The Markov Imbedded Structure technique can be used to derive system reliability from individual component reliability.
- Each state in the Markov chain ($S_0 - S_3$) corresponds to a combination of functional and failed system components.
  - In this model, we only consider the failure of lines.
- State transitions resulting from the behavior of c_3 are captured by P_{c_3}, where $p_L = 1 - q_L$ is the reliability of the line.

	Components	
States	c_3	c_4
S_0	1	1
S_1	1	0
S_2	0	1
S_3	0	0

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

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

Example: Direct Construction of a Specific MIS Reliability Model

- We identify the probability of the system initially being in each state with the vector Π_0.
- The vector u identifies which states are considered functional; here, both lines must fail for the system to fail.
- System reliability is then given by R.
  - We assume the behavior of each component is independent.

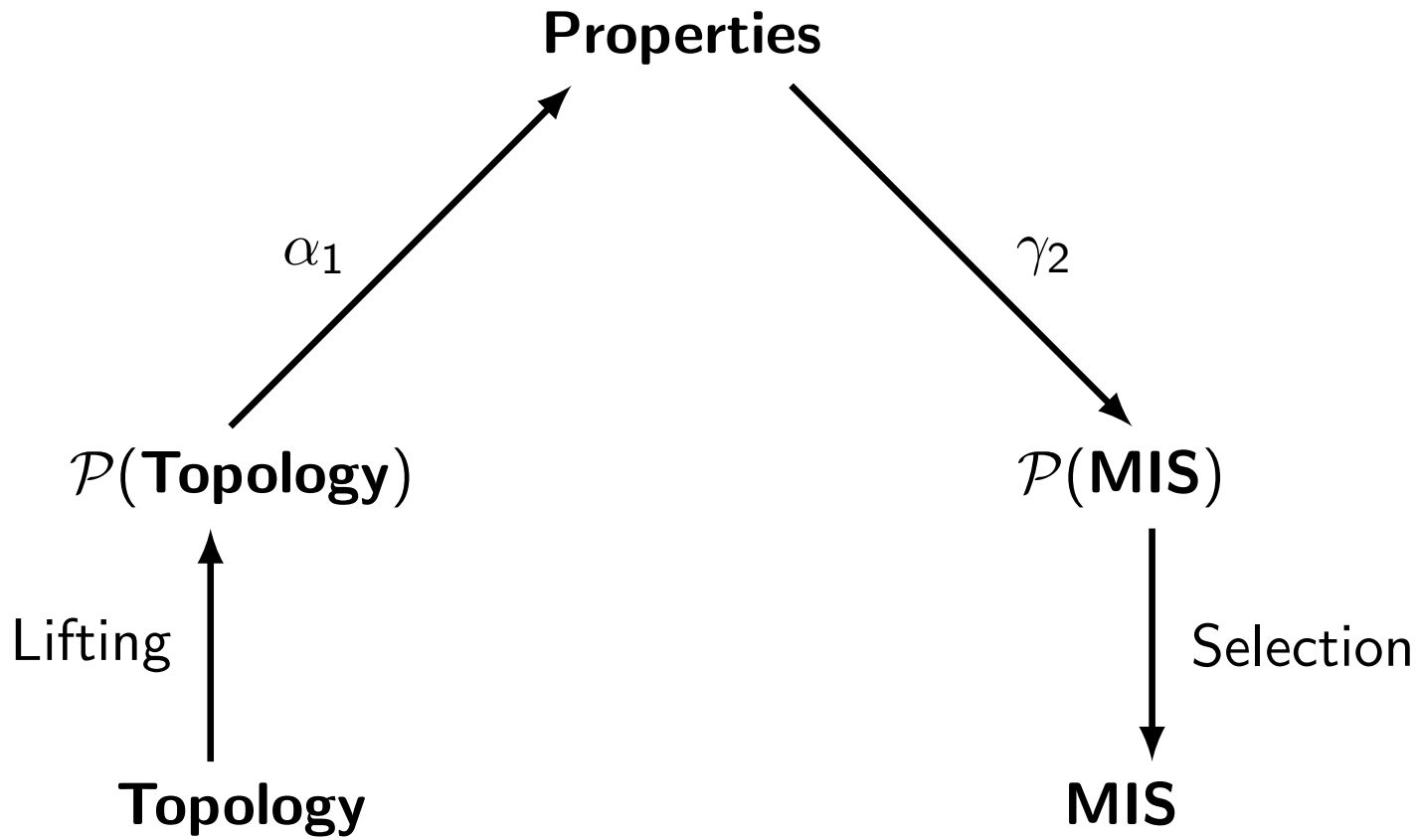
$$\Pi_0 = [1, 0, 0, 0]$$
$$u = [1, 1, 1, 0]$$
$$R = \Pi_0^T * P_{c_3} * P_{c_4} * u = p_L^2 + 2 p_L q_L$$

Example: Properties of the Directly Constructed MIS Model

$$\begin{aligned}
\text{components} &\subseteq \textbf{component} \\
\text{attributes} &\subseteq \textbf{component} \times \textbf{probability} \\
\text{functional_states} &\subseteq \mathcal{P}(\textbf{component}) \\
\text{initial_probability} &\subseteq \mathcal{P}(\textbf{component}) \times \textbf{probability} \\
\textbf{probability} &= \{x \in \mathbb{R} : 0 \leq x \leq 1\}
\end{aligned}$$

$$\begin{aligned}
\text{components}(M) &= \{c_3, c_4\} \\
\text{attributes}(M) &= \{(c_3, p_L), (c_4, p_L)\} \\
\text{functional_states}(M) &= \{\{c_3, c_4\}, \{c_3\}, \{c_4\}\} \\
\text{initial_probability}(M) &= \{(\{c_3, c_4\}, 1)\}
\end{aligned}$$

Example: Transforming the Topology Model to an MIS Model



Example: Construction of an MIS Model via Concretization

$$\Pi_0 = [\mathfrak{s}_0, \cdots, \mathfrak{s}_{15}]$$

$$u = [\mathfrak{u}_0, \cdots, \mathfrak{u}_{15}]$$

$$P_{c_1} = \begin{bmatrix} \mathfrak{p}_{c_1} & 0 & \cdots & \mathfrak{q}_{c_1} & 0 & \cdots \\ \vdots & \ddots & & \vdots & \ddots & \\ 0 & \cdots & \mathfrak{p}_{c_1} & 0 & \cdots & \mathfrak{q}_{c_1} \\ 0 & \cdots & 0 & 1 & 0 & \cdots \\ \vdots & & \vdots & \vdots & \ddots & \\ 0 & \cdots & 0 & 0 & \cdots & 1 \end{bmatrix}$$

$$R = \Pi_0^T * P_{c_1} * P_{c_2} * P_{c_3} * P_{c_4} * u$$

- If we select $\mathfrak{p}_{c_1}, \mathfrak{p}_{c_2} = 1$, then $\mathfrak{u}_4, \cdots, \mathfrak{u}_{15} = 0$ and $\mathfrak{s}_4, \cdots, \mathfrak{s}_{15} = 0$ since the failed states for these components are unreachable.
- Binding $\mathfrak{p}_{c_3}, \mathfrak{p}_{c_4} = p_L$, $\mathfrak{s}_0 = 1$, $\mathfrak{u}_0, \mathfrak{u}_1, \mathfrak{u}_2 = 1$, and $\mathfrak{u}_3 = 0$ suffice to generate an MIS model equal to our first MIS model.

Conclusions

- We presented a provably correct and specific approach for model transformation based on Abstract Interpretation.
- It can be used to transform models across domains.
- It can also facilitate transformation between models of nonfunctional and functional attributes.
- Using an example, we have demonstrated both the correctness and specificity of our approach: we concretize a specific model from the properties of another model.
- This research can accelerate advances in design and analysis of sustainable infrastructure by enabling cross-domain transfer of knowledge.