

Using Parameters in Architectural Views to Support Heterogeneous Design and Verification

Akshay Rajhans[†], Ajinkya Bhavne[†], Sarah Loos[‡], Bruce H. Krogh[†], André Platzer[‡], David Garlan[‡]

[†]Department of Electrical and Computer Engineering {arajhans|jinx|krogh}@ece.cmu.edu

[‡]Department of Computer Science {sloos|aplatzer|garlan}@cs.cmu.edu
Carnegie Mellon University, Pittsburgh, PA 15213-3890

Abstract—Current methods for designing cyber-physical systems lack a unifying framework due to the heterogeneous nature of the constituent models and their respective analysis and verification tools. There is a need for a formal representation of the relationships between the different models. Our approach is to define these relationships at the architectural level, associating with each model a particular view of the overall system base architecture. This architectural framework captures critical structural and semantic information without including all the details of the various modeling formalisms. This paper introduces the use of logical constraints over parameters in the architectural views to represent the conditions under which the specifications verified for each model are true and imply the system-level specification. Interdependencies and connections between the constraints in the architectural views are managed in the base architecture using first-order logic of real arithmetic to ensure consistency and correct reasoning. The approach is illustrated in the context of heterogeneous verification of a leader-follower vehicle scenario.

I. INTRODUCTION

Designing a complex cyber-physical system involves analyses of multiple models that capture different aspects of the system using heterogeneous formalisms and tools. To ensure correct system design, there is a need to guarantee that the models are consistent with the system and that system-level properties are correct from the verification of properties of the individual models. In practice, consistency between heterogeneous models and their relationship to system-level specifications is established informally at best. This paper presents a new approach to formal integration of heterogeneous design and verification activities based on parametric relations between architectural views.

Several efforts have focused on supporting multi-view, model-based system development. Some frameworks such as Ptolemy II support simulation of heterogeneous models by hierarchically integrating multiple “models of computation” into a single simulation model in an actor-oriented formalism [2]. Lattice-based ontologies can then be used to infer semantic relationships between elements of heterogeneous models [11]. Kumar et al. also propose an ontology-based approach, but in the context of managing knowledge gained from heterogeneous verification activities [9]. Other directions for enabling heterogeneous model analysis focus on either the creation of meta-models for each modeling formalism [7], [10], [1], [19] and the use of toolchains [16] to create an integrated analysis environment; or on the creation of

a common interchange format [14], [8] and the use of translation schemes [5] to translate models between different formalisms.

Rather than attempting to integrate heterogeneous models into a single framework or tool, or translating between modeling formalisms, we have proposed the use of architectures as a lightweight representation of relationships and interdependencies between the models [3]. The objective is to represent critical features of the structural and semantic relationships between models to support some level of reasoning about model consistency and system-level specifications, while not attempting to deal with all of the details that are best represented and analyzed in particular modeling frameworks. We aim to support the traditional separation of concerns with a more principled and formal way to deal with critical inter-model dependencies early in, and throughout, the development process. In this approach, models are represented as architectural views of a base architecture for the complete system, providing a unifying framework for defining and analyzing structural consistency between models [4]. This paper introduces the first steps toward the specification and analysis of inter-model consistency at the semantic level.

II. AN ARCHITECTURAL FRAMEWORK FOR HETEROGENEOUS DESIGN AND VERIFICATION

Architectures are annotated structural representations that describe systems at a high level of abstraction, allowing system designers to determine appropriate assignment of functionality to elements, evaluate the compatibility of the parts, and make trade-offs between different quality attributes such as performance, reliability, and maintainability [6]. We have developed the CPS architectural style [17] for a unified representation of the system structure. A system’s *base architecture* is an annotated graph of components and connectors in which the components represent the principal computational and physical elements of the complete system’s run-time structure, the connectors represent pathways of communication and physical couplings between components, and annotations represent properties of the elements.

An *architectural view* is another component-connector graph that represents a particular abstraction and refinement of various elements of the base architecture. Each model used in the system development is associated with an architectural

view, based on the particular design perspective of the model. For example, a Simulink model created for feedback control analysis will be associated with a view defined in the control design perspective, while a process algebra model created to analyze system deadlock behavior will be associated with a software design perspective. In this context, well-defined mappings between a view and the base architecture can be used as the basis for identifying and managing the structural and semantic dependencies among the various verification models, and to evaluate mutually constraining design choices.

The notion of *structural view consistency* guarantees that the models adhere to the component-connector structure of the base architecture, thereby ensuring consistent functional decomposition of the system as represented by the base architecture. We use the concept of graph morphisms to check the types of structural relationships that naturally arise in the construction of the verification models [4]. The focus of this work is to develop the notion of *semantic view consistency*, which checks whether each model has correct and consistent assumptions about the behavior of the rest of the system.

As a first step towards addressing semantic consistency, we introduce constraints over parameters to capture the semantic information shared between different views. In addition to extracting the structure of a model M_i , each architectural view of the system also maintains a set of parameters P_i associated either with specific elements of the view or with the view itself. The base architecture maintains the set of system-level parameters P_0 , and the auxiliary constraint C_{aux} that represents the relation between P_0 and each P_i , as well as interdependencies between the various P_i 's. Parameters that are assigned to specific elements in a view can be checked for syntactic correctness, based on the structural mappings between the view and the base architecture. As illustrated in Fig. 1, the semantic interdependencies between the views and the base architecture are represented by the *external constraints* C_i^{ext} on the parameters P_i arising from the rest of the system.

The remainder of the paper develops the formulation of conditions for consistency and correctness of verification activities based on the relationships between the view-level and base-architecture-level parameter constraints.

III. VERIFICATION OF PARAMETERIZED MODELS

We begin by introducing definitions and notation for describing the use of parameterized models for verifying system properties. A *parameter* p of a system is a real-valued static variable that affects the system behavior. The *valuation* of a set of parameters P is a function $v : P \rightarrow \mathbb{R}$ that associates each parameter with a value. $V(P)$ denotes the set of all possible valuations of the parameters in P .

A *constraint* $C(P)$ over a set of parameters P is an expression written in a constraint formalism \mathcal{C} . For a given $v \in V(P)$, $\llbracket C(P) \rrbracket_v \in \{\top, \perp\}$ denotes the evaluation of the constraint $C(P)$ at v , and $\llbracket C(P) \rrbracket$ denotes the set of all valuations v over P for which $\llbracket C(P) \rrbracket_v = \top$. In this paper,

we assume \mathcal{C} is the language of first-order real arithmetic ($FOL_{\mathbb{R}}$). Thus, validity and satisfiability of parameter constraints is decidable by Tarski's quantifier elimination [18].

Conjunction of constraints $C_1(P)$ and $C_2(P)$, written $C_1(P) \wedge C_2(P)$, is also a constraint whose corresponding parameter valuations are the intersection of the parameter valuations of the original constraints, i.e., $\llbracket C_1(P) \wedge C_2(P) \rrbracket = \llbracket C_1(P) \rrbracket \cap \llbracket C_2(P) \rrbracket$. Similarly, disjunction of constraints is a constraint whose corresponding parameter valuations are the union of the parameter valuations of the original constraints. We write $C'(P) \Rightarrow C(P)$ when $\llbracket C'(P) \rrbracket \subseteq \llbracket C(P) \rrbracket$ and $C'(P) \equiv C(P)$ when $\llbracket C'(P) \rrbracket = \llbracket C(P) \rrbracket$.

Given two sets of parameters P and P' , the *projection* of a constraint $C(P)$ onto P' , written as $C(P) \downarrow_{P'}$, is the constraint over P' defined by existential quantification of the parameters in $P \setminus P'$. Its valuations $\llbracket C(P) \downarrow_{P'} \rrbracket$ are

$$\{v' \in V(P') \mid \exists v \in \llbracket C(P) \rrbracket : v'(p') = v(p') \forall p' \in P' \cap P\}.$$

The constraint resulting from this existential quantification can be computed effectively for $FOL_{\mathbb{R}}$. Given a collection of constraints $C_1(P), \dots, C_n(P)$, it is straightforward to show that

$$\left(\bigwedge_{i=1}^n C_i(P) \right) \downarrow_{P'} \Rightarrow \bigwedge_{i=1}^n (C_i(P) \downarrow_{P'}). \quad (1)$$

Let M be a *parameterized model* with a set of parameters P that is constructed in a particular modeling formalism \mathcal{M} . Let \mathcal{B} denote a *behavioral domain*, that is, \mathcal{B} is the set of all possible behaviors selected for defining the behavioral semantics of models in \mathcal{M} . Depending on the context, the behavioral domain could be, for example, event traces, pairs of continuous input-output signals, or hybrid trajectories. We define the semantics of a particular model by a mapping $\rho_{\mathcal{M}} : \mathcal{C} \times \mathcal{M} \rightarrow 2^{\mathcal{B}}$. Given a constraint $C(P)$ and model M , $\rho_{\mathcal{M}}(C(P), M)$ denotes the set of all possible behaviors in \mathcal{B} associated with the model M for all parameter valuations in $\llbracket C(P) \rrbracket$.

We observe that the set of possible behaviors grows or shrinks *monotonically* with increasing or decreasing sets of parameter valuations. I.e., if $C'(P) \Rightarrow C(P)$, then $\rho_{\mathcal{M}}(C'(P), M) \subseteq \rho_{\mathcal{M}}(C(P), M)$.

A *specification* S is a condition written in a specification formalism \mathcal{S} . S can be, for example, various temporal logics, finite automata, Kripke structures, or a set of states to be avoided. The semantics for specifications is given by a mapping $\rho_{\mathcal{S}} : \mathcal{S} \rightarrow 2^{\mathcal{B}}$ that associates a specification S with a subset of the behaviors in \mathcal{B} for which the specification S is satisfied. In this paper we consider only *safety specifications*. More specifically, we assume the specifications satisfy the following property.

Definition 1 A *specification formalism* \mathcal{S} is said to be *conjunctive with respect to a given semantic mapping* $\rho_{\mathcal{S}}$ if

$$\forall S', S'' \in \mathcal{S}, \rho_{\mathcal{S}}(S' \wedge S'') = \rho_{\mathcal{S}}(S') \cap \rho_{\mathcal{S}}(S'').$$

Safety specifications are conjunctive, because they concern the absence of bad behavior, which can be checked separately

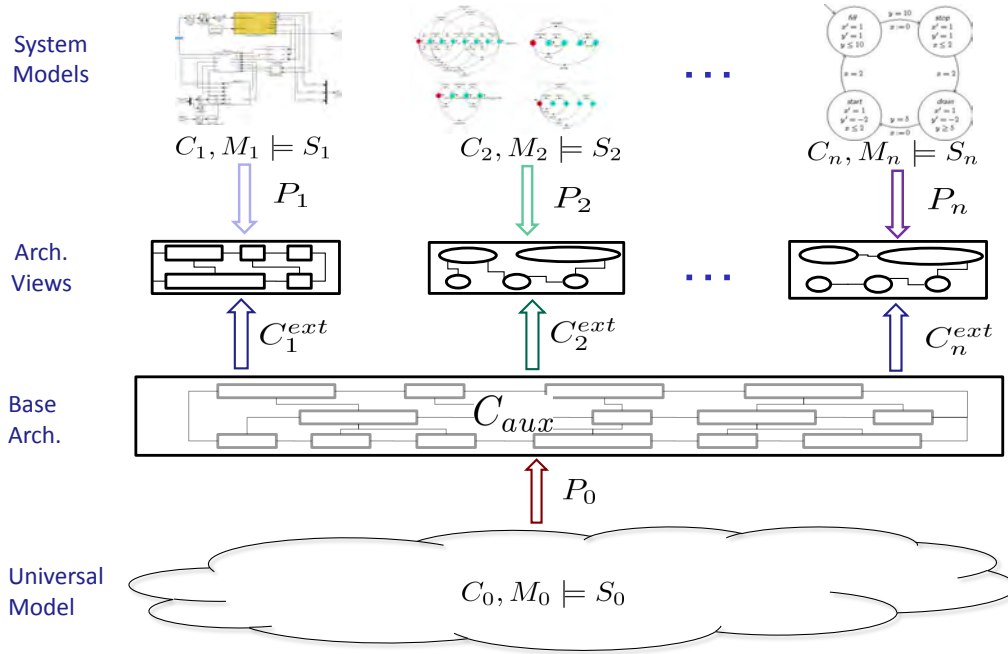


Fig. 1. Parameterized architectural views for heterogeneous system models.

for separate properties. Given two specifications S and S' from a conjunctive formalism, we write $S' \Rightarrow S$ when $\rho_S(S') \subseteq \rho_S(S)$.

We consider design and verification procedures that aim to demonstrate that the behaviors of a model satisfy a given specification over a range of parameter valuations. We denote satisfaction of a given specification by entailment, namely, $C(P), M \models S$ when $\rho_{\mathcal{M}}(C(P), M) \subseteq \rho_S(S)$. Entailment specifies the relation between the model and a parameter constraint and the specification. For a given set of parameters P and a specification S , building a model M and constructing $C(P)$ such that $C(P), M \models S$ is a *design* task. On the other hand, checking whether $C(P), M \models S$ holds for given $C(P)$ and M using a suitable analysis procedure is a *verification* task. The next section develops the concept of design and verification using *heterogeneous* models.

IV. HETEROGENEOUS DESIGN AND VERIFICATION

We return to the multi-model setting described in Sec. II using the formal framework for modeling and verification introduced in the previous section. Suppose one wants to design a system with parameters P_0 that satisfies the specification S_0 for the range of parameter valuations represented by a constraint $C_0(P_0)$. If there existed a universal modeling formalism \mathcal{M}_0 that could model everything needed for the system design, a model M_0 of the whole system could be constructed and an appropriate verification procedure could be applied to demonstrate $C_0(P_0), M_0 \models S_0$. In reality, there is no such modeling formalism that can capture all the aspects of a complex cyber-physical system, including the physical dynamics, software, hardware, control, scheduling, communication protocols, etc. Even if there were a such a

modeling formalism, building a detailed universal model M_0 would be impractical and doing any kind of verification with the complexity of this grand universal model would be even more hopelessly impractical, if not impossible.

Alternatively, rather than designing a complete system at once, a separation of concerns approach is used to design and analyze different aspects of the system using a variety of models, each suited to a particular type of problem. To describe this approach formally, let $\mathcal{M}_i, i = 1, \dots, n$ denote the various modeling formalisms used for the heterogeneous design, and let $M_i \in \mathcal{M}_i$ be the specific models that are constructed. In the process of decomposing the system design task into several smaller ones pertaining to the particular design tasks, model-level constraints $C_i(P_i)$ are introduced using sets of parameters P_i . To simplify the notation, we will use C_i to denote $C_i(P_i)$ in the following. These model-specific parameters, such as controller gains in the control perspective, are not necessarily in the set of original system-level parameters P_0 , but there are relationships between the various sets of parameters, and the original system constraints impose constraints on the parameters in the various models. We let $C_{aux}(P)$ denote the auxiliary constraint that captures these dependencies over the parameters $P = \bigcup_{j=0}^n P_j$, which is the set of all parameters being used, including the original parameters P_0 . Without loss of generality we assume the sets $P_j, j = 0, 1, \dots, n$ are disjoint. We also assume the auxiliary constraint does not impose a restriction on the original constraint, that is,

$$(C_0 \wedge C_{aux}) \downarrow_{P_0} = C_0. \quad (2)$$

Specifications S_i are also introduced for each of the models. Given the individual models M_i , the model-level

constraints C_i and the specifications S_i for $i = 1, \dots, n$, we then invoke analysis procedures to determine whether $C_i, M_i \models S_i$. Figure 1 illustrates the overall decomposition of the design and verification problem.

The hope is that the demonstration of entailment for each of the individual models will somehow guarantee that $C_0, M_0 \models S_0$. In the following, we develop conditions under which this will be true.

Definition 2 A set of specifications S_1, \dots, S_n is said to satisfy specification coverage with respect to the original system (safety) specification S_0 if

$$\left(\bigwedge_{i=1}^n S_i \right) \Rightarrow S_0.$$

This definition of coverage for the specifications is similar to the notion of requirements traceability: it is necessary to show that the requirements that are actually verified (S_i , $i = 1, \dots, n$) are in fact sufficient to imply the original requirements. It is also necessary to demonstrate that the design has been verified for C_0 , which specifies the full range of parameters P_0 for which the specification S_0 must hold. This is captured in the following definition that relates the parameter constraints in the models to the constraints on the original parameters.

Definition 3 A set of constraints C_1, \dots, C_n is said to satisfy constraint coverage with respect to the original system constraint C_0 if

$$C_0 \Rightarrow \left(C_{aux} \wedge \bigwedge_{i=1}^n C_i \right) \downarrow_{P_0}.$$

The notion of constraint coverage defined above requires that the set of constraints for the models overapproximates the original constraint when they are projected into the original parameter set through the auxiliary constraint.

Finally, it is necessary to assure that the behaviors modeled by the individual models indeed cover the behaviors of interest in the original model. This requirement needs to be satisfied by the way in which the models are constructed. In particular, for the purposes of verification, it is necessary that the models overapproximate the behaviors of the actual system characterized by the system model M_0 in the universal modeling framework \mathcal{M}_0 . This is the concept of abstraction for heterogeneous models.

Definition 4 A model M_i in a modeling formalism \mathcal{M}_i is said to be an abstraction of a model M_0 in the modeling framework \mathcal{M}_0 if for any constraints C_i

$$\rho_{\mathcal{M}_0}([C_{aux} \wedge C_i] \downarrow_{P_0}, M_0) \subseteq \rho_{\mathcal{M}_i}(C_i, M_i).$$

Note that the definition of abstraction maps the constraints over P_i into the constraints over P_0 using the auxiliary constraint. In practice, the modeling framework \mathcal{M}_0 and model M_0 are not actually specified. Nevertheless, this describes the intuition behind the current informal practice of heterogeneous design and verification. The definitions above

provide conditions under which verification of the specifications for the individual models will imply satisfaction of the original specifications.

Proposition 1 Given a model M_0 in a modeling formalism \mathcal{M}_0 with constraint C_0 over parameters P_0 , and (safety) specification S_0 , and a set of models M_i constructed using modeling formalisms \mathcal{M}_i with constraints C_i over parameters P_i and (safety) specifications S_i for $i = 1, \dots, n$. If

- i. constraints C_i ($i = 1, \dots, n$) cover C_0 ,
 - ii. model M_i is an abstraction of M_0 for each $i = 1, \dots, n$
 - iii. specifications S_i ($i = 1, \dots, n$) cover S_0 , and
 - iv. $C_i, M_i \models S_i$ for each $i = 1, \dots, n$
- then $C_0, M_0 \models S_0$.

Proof: In the following all conjunctions and intersections are over $i = 1, \dots, n$. We first note that constraint coverage implies

$$C_0 \Rightarrow \left(\bigwedge_i C_i \wedge C_{aux} \right) \downarrow_{P_0} \Rightarrow \bigwedge_i (C_{aux} \wedge C_i) \downarrow_{P_0}.$$

Therefore, monotonicity implies

$$\rho_{\mathcal{M}_0}(C_0, M_0) \subseteq \rho_{\mathcal{M}_0}\left(\bigwedge_i (C_{aux} \wedge C_i) \downarrow_{P_0}, M_0\right)$$

$$\text{(monotonicity)} \subseteq \bigcap_i \rho_{\mathcal{M}_0}((C_{aux} \wedge C_i) \downarrow_{P_0}, M_0)$$

$$\text{(abstraction)} \subseteq \bigcap_i \rho_{\mathcal{M}_i}(C_i, M_i)$$

$$\text{(since } C_i, M_i \models S_i) \subseteq \bigcap_i \rho_S(S_i) \stackrel{\text{Def. 1}}{=} \rho_S\left(\bigwedge_i S_i\right)$$

$$\text{(specification coverage)} \subseteq \rho_S(S_0).$$

Therefore, $C_0, M_0 \models S_0$. ■

The conditions introduced in Proposition 1 pertain to how the heterogeneous models and specifications relate to the original constraints in the original parameter space P_0 . To perform analysis and design in the individual modeling frameworks, conditions are needed in the parameter spaces for each of the models. The constraint C_i for model i should be chosen so that it is not more restrictive than the original system constraints projected into the parameters P_i . This condition is stated in the following definition.

Definition 5 Constraint C_i for model i is said to be original-constraint consistent if

$$C_i^{ext} := (C_0 \wedge C_{aux}) \downarrow_{P_i} \Rightarrow C_i.$$

Lemma 1 If C_i is original-constraint consistent, then

$$(C_0 \wedge C_{aux}) \downarrow_{P_i} \Rightarrow (C_i \wedge C_{aux}) \downarrow_{P_i}.$$

Proof: Consider any $v'_i \in \llbracket (C_0 \wedge C_{aux}) \downarrow_{P_i} \rrbracket$. Then $v'_i \in \llbracket C_{aux} \downarrow_{P_i} \rrbracket$ (note: C_0 constrains only parameters P_0 , which are disjoint from P_i). From the definition of original-constraint consistent, $v'_i \in \llbracket (C_0 \wedge C_{aux}) \downarrow_{P_i} \rrbracket$ also implies $v'_i \in \llbracket C_i \rrbracket$, which in turn implies $v'_i \in \llbracket (C_i \wedge C_{aux}) \downarrow_{P_i} \rrbracket$. ■

The following proposition states the conditions for the constraints in the parameter spaces for each modeling framework that will guarantee that the original specifications are satisfied.

Proposition 2 *Under the assumptions of Prop. 1 except that condition i. on the constraint coverage is replaced with*

i. C_i is original-constraint consistent,

we can still conclude $C_0, M_0 \models S_0$.

Proof: By Prop. 1, it is sufficient to show that constraints C_i cover C_0 . That is, we want to show

$$C_0 \Rightarrow \left(\bigwedge_i C_i \wedge C_{aux} \right) \downarrow_{P_0}. \quad (3)$$

Suppose (3) is not true, i.e., suppose there exists some $v'_0 \in V(P_0)$ such that $\llbracket C_0 \rrbracket_{v'_0} = \top$ but $\llbracket (\bigwedge_i C_i \wedge C_{aux}) \downarrow_{P_0} \rrbracket_{v'_0} = \perp$. This implies

$$\llbracket \bigwedge_i C_i \wedge C_{aux} \rrbracket_{v'_0, \vec{v}} = \perp \quad \forall \vec{v} \in \prod_i V(P_i). \quad (4)$$

From Lemma 1 and (4),

$$\llbracket C_0 \wedge C_{aux} \rrbracket_{v'_0, \vec{v}} = \perp \quad \forall \vec{v} \in \prod_i V(P_i),$$

which in turn implies

$$\llbracket (C_0 \wedge C_{aux}) \downarrow_{P_0} \rrbracket_{v'_0} = \perp.$$

This implies from (2) that $\llbracket C_0 \rrbracket_{v'_0} = \perp$, which contradicts the initial assumption. ■

V. EXAMPLE

We consider a platoon with two cars driving in a single lane, and communicating over an 802.11p inter-vehicular wireless network. The leader car l sends its position and velocity to the follower car f over the network. The position, velocity, and acceleration of l are x_l, v_l , and a_l respectively (and similarly for follower f). The follower has a controller that is required to maintain a specified distance of d_{set} from the leader. The communication delay of the network, T_c , is a function of the inter-vehicle distance d . If d goes beyond d_{max} , the network connectivity is lost, and the two-car platoon can no longer be maintained.

Let \underline{x} and \bar{x} denote the lower and upper bounds, respectively, on a parameter x . The system-level parameters and constraint are

$$\begin{aligned} P_0 : & \quad \bar{a}_l, \underline{a}_l, \bar{a}_f, \underline{a}_f, \bar{v}_l, \underline{v}_l, \bar{v}_f, \underline{v}_f \\ C_0 : & \quad \text{specific values for all parameters in } P_0 \end{aligned}$$

A hypothetical M_0 for this system would be a hybrid system that models the dynamics of the cars, in addition to the behavior of the communication network. The system-level specification S_0 for a safe platoon formation is given as

$$S_0 := (x_l - x_f > 0) \wedge (x_l - x_f \leq d_{max}),$$

which asserts the two cars never collide and are never more than d_{max} meters apart.

We consider three heterogeneous views for the two-car platoon. View 1 is a formal verification view associated with model M_1 created in the KeYmaera tool [15]. M_1 is a very general model that identifies an envelope of behaviors for any sampled-data control algorithm for car f that will ensure that S_0 is valid. This model assumes the acceleration and braking of both cars are bounded by A and B , respectively. There is also an assumption that the minimum velocity of each car is non-negative, and that the total delay ε in the system (communication+sampling+reaction delay) is bounded. The representation of the controller in M_1 is simple. If the controller determines that it is safe for the car to accelerate, any acceleration or braking within the bounded limits is allowed; however, if it determines that the system state is on the boundary of the set of safe states, the car must brake with its full braking force. The controller uses the following constraint to determine whether the car can accelerate safely up to time ε , the maximum delay until the next sample occurs: $x_l > x_f + \frac{v_f^2}{2B} - \frac{v_l^2}{2B} + \left(\frac{A}{B} + 1\right) \left(\frac{A}{2}\varepsilon^2 + v_f\varepsilon\right)$.

If this constraint holds for current sensor readings of position and velocity, then car l will still be safely in front of f until the controller receives sensor data and can react again (i.e., after they drive for up to ε time), no matter how l and f accelerate or brake. For a more detailed description of this model and its verification, see [12].

Verifying the correctness of this generic model makes it possible to verify a wide range of controllers by simply assuring that specific controllers have an input-output map that is assured to apply maximum braking on the boundary of the region of safe states. In this example, this condition is used in View 1 to verify the correctness of a sampled-data PID controller design in View 2. The procedure described above could also be used to verify that the distance between the cars does not exceed d_{max} .

View 2 is a control design view that is associated with model M_2 created in Simulink. M_2 is a sampled-data PID controller for the follower car designed to keep f around d_{set} meters of the leader. The control design takes the network communication delay T_c into account. The controller has a sampling period T_s and the set K of PID gains is given by $\{K_p, K_i, K_d\}$. Additionally, we assume bounds on the velocity and acceleration of the two cars in M_2 . The constraints on v_l and v_f indicate the operating regime over which we want to verify the correctness of our controller. The idea is that the real car controller would switch into the PID mode only when the platoon is moving at some nominal speed, and there are other things that are done outside of this regime. Our interest is verifying that if the system remains in this regime where the PID controller operates indefinitely, there will be no collisions. Since the Simulink view can only be used to run simulations, the parameters from the control design are passed to View 1 through C_{aux} for formal verification. This includes the distance set point, d_{set} , which is a design parameter selected in the control design view.

View 3 is a communication perspective. The associated model M_3 is created in the ns-2 network simulator [13] to

calculate T_c for the specified d_{max} . The T_c derived from the communication view is a design constraint that must be adhered to in V_2 , and is also used in View 1 via C_{aux} as one of the elements in the composition of ε .

To summarize, the parameters for the three views are given by

$$\begin{aligned} P_1 &:= P_0 \cup \{\varepsilon, K, d_{set}\} \\ P_2 &:= P_0 \cup \{T_c, T_s, K, d_{set}\} \\ P_3 &:= \{T_c\} \end{aligned}$$

The auxiliary constraint C_{aux} contains the set of equality constraints between the same parameters in the three views, as well as the constraint $\varepsilon = T_s + T_c$ that bounds the total system delay ε . The specifications for the three views are given by $S_1 \equiv S_0$, and $S_2 = S_3 = \top$. In other words, the verification of the system specifications is performed in View 1 using values determined by the designs in Views 2 and 3.

We have verified $C_1, M_1 \models S_1$ in KeYmaera. It is straightforward to show that we have constraint and specification coverage and that $C_2, M_2 \models S_2$ and $C_3, M_3 \models S_3$ trivially hold. If the designer views the heterogeneous models M_1 , M_2 and M_3 to be abstractions of sampled-data PID controllers, it can be concluded that $C_0, M_0 \models S_0$. Note that this last statement is a decision that needs to be made in the context of the actual application. As noted in Sec. IV, the model underlying the base architecture is not actually constructed. Rather, engineering judgment is used to determine whether or not the set of models is acceptable as a collection of abstractions for the real system. This simple example illustrates the application of this approach for only a small part of the specifications that would be addressed for a real system. The value of the approach lies in the discipline of identifying the relationships between the models, parameters and constraints formally.

VI. DISCUSSION

This paper presents a new approach to using heterogeneous models for design and verification of complex cyber-physical systems based on the concept of architectural views where syntactical and semantical consistency relationships between the models are identified explicitly. Heterogeneous design and verification activities are standard engineering practice, because different questions typically need different models, modeling formalisms, and tools. In current practice, the relationships among the heterogeneous models remains informal at best. We have introduced parameter specification and constraint consistency as a first step in developing lightweight tools for assuring some degree of correctness in the application of multiple modeling formalisms. We have shown how heterogeneous models may be used together to reason about the safety of cyber-physical systems.

We are currently implementing support for analyzing structural consistency between architectural views in the AcmeStudio architectural design environment [4]. Next steps will be to add support for the specification and evaluation of parameter consistency, and to extend the approach developed

in this paper to allow dynamic assumptions, e.g., temporal logic formulas, to capture the behavioral assumptions and inter-model dependencies in models used in multi-domain model-based design and verification.

REFERENCES

- [1] F. Balarin, Y. Watanabe, H. Hsieh, L. Lavagno, C. Passerone, and A. Sangiovanni-Vincentelli. Metropolis: an integrated electronic system design environment. *Computer*, 36(4):45–52, April 2003.
- [2] S. S. Bhattacharyya, E. Cheong, and I. Davis. Ptolemy II heterogeneous concurrent modeling and design in Java. Technical report, University of California, Berkeley, 2003.
- [3] A. Bhave, D. Garlan, B. Krogh, A. Rajhans, and B. Schmerl. Augmenting software architectures with physical components. In *Proc. of the Embedded Real Time Software and Systems Conf. (ERTS² 2010)*, 19–21 May 2010.
- [4] A. Bhave, B. H. Krogh, D. Garlan, and B. Schmerl. View consistency in architectures for cyber-physical systems. In *Proc. of Second International Conference on Cyber-Physical Systems, ICCPS*, 2011.
- [5] K. Chen, J. Sztipanovits, and S. Abdelwahed. Toward a semantic anchoring infrastructure for domain-specific modeling languages. In *5th ACM International Conference on Embedded Software*, volume September, 2005.
- [6] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, and J. Stafford. *Documenting Software Architectures: Views and Beyond*. Addison-Wesley, 2002.
- [7] J. Davis. GME: The Generic Modeling Environment. In *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, OOPSLA '03*, New York, NY, USA, 2003. ACM.
- [8] Z. Gu, S. Kodase, S. Wang, and K. Shin. A model-based approach to system-level dependency and real-time analysis of embedded software. In *Proceedings of the 9th IEEE Real-Time and Embedded Technology and Applications Symposium, 2003.*, pages 78 – 85, May 2003.
- [9] R. Kumar, B. H. Krogh, and P. Feiler. An ontology-based approach to heterogeneous verification of embedded control systems. In M. Morari and L. Thiele, editors, *Hybrid Systems: Computation and Control*, volume 3414 of *Lecture Notes in Computer Science*, pages 370–385. Springer Berlin / Heidelberg, 2005.
- [10] A. Ledeczi, J. Davis, S. Neema, and A. Agrawal. Modeling methodology for integrated simulation of embedded systems. *ACM Trans. Model. Comput. Simul.*, 13:82–103, January 2003.
- [11] J. Leung, T. Mandl, E. Lee, E. Latronico, C. Shelton, S. Tripakis, and B. Lickly. Scalable semantic annotation using lattice-based ontologies. In *12th International Conference on Model Driven Engineering Languages and Systems*, pages 393–407. ACM/IEEE, October 2009.
- [12] S. M. Loos, A. Platzer, and L. Nistor. Adaptive cruise control: Hybrid, distributed, and now formally verified. In M. Butler and W. Schulte, editors, *FM, LNCS*. Springer, 2011.
- [13] T. Murray, M. Cojocari, and H. Fu. Measuring the performance of IEEE 802.11p using ns-2 simulator for vehicular networks. In *IEEE Conf. on Electro/Information Technology*, pages 498–503, 2008.
- [14] A. Pinto, L. P. Carloni, R. Passerone, and A. Sangiovanni-Vincentelli. Interchange semantics for hybrid system models. In *Proc. of 5th MATHMOD*, 2006.
- [15] A. Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, Heidelberg, 2010.
- [16] J. Porter, P. Volgyesi, N. Kottenstette, H. Nine, G. Karsai, and J. Sztipanovits. An experimental model-based rapid prototyping environment for high-confidence embedded software. In *RSP '09: Proceedings of the 2009 IEEE/IFIP International Symposium on Rapid System Prototyping*, pages 3–10, Washington, DC, USA, 2009. IEEE Computer Society.
- [17] A. Rajhans, S.-W. Cheng, B. Schmerl, D. Garlan, B. H. Krogh, C. Agbi, and A. Bhave. An architectural approach to the design and analysis of cyber-physical systems. In *Third International Workshop on Multi-Paradigm Modeling*, Denver, Oct 2009.
- [18] A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, Berkeley, 2nd edition, 1951.
- [19] H. L. M. Vangheluwe. DEVS as a common denominator for multi-formalism hybrid systems modeling. In *Proceedings of the 2000 IEEE International Symposium on Computer-Aided Control System Design*, Anchorage, Alaska, USA, 2007.