Towards a Theory for Cyber-Physical Systems Modeling

Gabor Simko
Vanderbilt University
Nashville, TN 37212 USA

Tihamer Levendovszky
Vanderbilt University
Nashville, TN 37212 USA

Miklos Maroti
Vanderbilt University
Nashville, TN 37212 USA

Janos Sztipanovits
Vanderbilt University
Nashville, TN 37212 USA

ABSTRACT

Modeling the heterogeneous composition of physical, computational and communication systems is an important challenge in engineering Cyber-Physical Systems (CPS), where the major sources of heterogeneity are causality, time semantics, and different physical domains. Classical physical laws capture acausal continuous-time dynamics, thus the behavior of physical systems are inherently characterized by acausal continuous-time equations. On the other hand, computational and communication systems are based on the notion of causality and discrete-time semantics. Connecting the two worlds is challenging, and calls for proper formalization of the composition. In this paper, we discuss a formalism that captures both acausal physical laws, unidirectional analog signals, and is capable of describing causal computational systems, as well as the composition of CPS models.

Categories and Subject Descriptors

C.3 [**Special-purpose and Application-based Systems**]: real-time and embedded Systems; F.1.1 [**Computation by Abstract Devices**]: Models of Computation – cyber-physical systems; I.6.5 [**Simulation and Modeling**]: Model Development[models cyber-physical interactions]; J.2 [**Physical Sciences and Engineering**]: Engineering

General Terms

Theory, Design

Keywords

Cyber-Physical Systems, formalization, Model-Based Engineering, heterogeneous composition

1. INTRODUCTION

In component based engineering compositionality means that properties of a system can be inferred from the properties of its constituent components and the interaction model used for connecting the components. Recently, numerous compositional environments were introduced for modeling computational systems [2,11]. Synchronization and scheduling of components are the most important questions concerning the interaction models for computational systems. One of the prominent trends for tackling these problems is based on the hierarchical composition of various interaction models, called Models of Computation (MoC) [11, 12]. An MoC is a model describing the semantics of interactions. Well-known examples are the process network, dataflow, petri-net, finite-state machine, and others.

A different approach is the characterization of different interaction types, such as rendezvous, broadcast, atomic broadcast, priority. Bliudze and Sifakis [5] introduce an algebra for describing these interaction types, that is implemented in the BIP framework [2].

The interaction models of physical systems, however, raise different questions. Here synchronization and scheduling are out of questions, and the main concern is the semantics of interactions: what are the consequences of a connection on the behavior of the components and the system. Unlike in computational systems, the semantics of physical interactions are not freely choosable. Rather the semantics of the interaction model must reflect the physical reality.

Recently acausal physical modeling has gained traction [10, 14, 18, 23], and got accepted as the de facto standard for physical modeling. In this model, the acausal nature of physics is faithfully reflected, and the physical laws are directly represented in their mathematical form. This is a remarkable difference compared to the classical system theory based modeling that was inherently causality based.

While acausal modeling is more appropriate from a physical point of view [23], it also brings new challenges. Most importantly, the semantics of acausal interaction is not local: it is not only dependent on the connected components, but transitively dependent upon a chain of interconnections.

Cyber-Physical Systems (CPS) are the integration of computational and physical systems. In this paper we develop a formalism for discussing CPS systems that are based on energy-flow based acausal physical modeling, but which also contains unidirectional physical signals and computational interactions. CPS systems are heterogeneous, software integrated physical systems, where heterogeneity emerges in several aspects: CPS are heterogeneous in physical domains (multi-domain physics), abstraction levels (physical systems, communication and computational systems), timing semantics (continuous-time, discrete-time and logical time), and in the mathematical apparatus (differential algebraic equations

CyPhy '13 Philadelphia USA

vs transition systems) used to describe them.

Already, there are several modeling environments for CPS that tremendously helps the creation of CPS systems, but it is the best interest of the CPS community to further enhance the experience. In particular, one of the most promising and exciting step is the support for multi-modeling, where the different modeling aspects of a system do not live independent from each other. In multi-modeling, lumped parameter physical modeling, CAD modeling, FEA (finite element analysis) simulations, computational and communication system modeling are integrated in such a way, that reasoning about the overall system becomes possible.

In the spirit of component-based engineering, we should consider such a multi-modeling environment as the composition of heterogeneous components coming from heterogeneous languages and tools. By understanding the cross-interactions between them, the multi-modeling vision could become a reality.

Our main goal is to facilitate the correct-by-construction modeling of CPS systems. This means that the modeling environment provides mechanisms for ensuring that structurally well-formed models of the language are structurally feasible: port connections are semantically matched (e.g. only compatible ports are connected), different models have the same view of the parameters (e.g. for a parameterized pipe the length parameter of the pipe is the same for the lumped-parameter and the FEA simulations). Further, as a future work we plan to extend the formalism to make deductions on the solvability (existentiality and uniqueness) of models based on their structure. On the other hand, we do not discuss model fidelity, i.e. how close the behavior of the model is to the behavior of the modeled system.

Our formalism addresses a subset of the multi-modeling environment, the interaction of lumped parameter physical modeling and computational systems modeling. Even though this is a small subset, it is an important cornerstone for CPS modeling.

There is a strong reason for being formal: avoiding ambiguities. Integration of domain-specific CPS languages calls for the unambiguous understanding and specification of these languages. Misunderstanding can result in faulty or incorrect designs, which do not meet the design requirements. Especially in CPS, which is often safety- and mission-critical, we cannot afford such design flaws.

Our contribution and the primary focus of the paper is the generalization of CPS models' behavior, and the development of a theory for composing CPS models from acausal and causal components. Currently, we focus on the acausal and causal physical connections, and leave the details of the interconnection of physical and computational systems for the future.

The structure of the paper is the following: Section 2 introduces the related work. In Section 3 we introduce our formalism and define the semantics for physical interconnections. In Section 5 the formalism is discussed with respect to some of the state of the art physical modeling environments. Finally, in Section 6 we conclude.

2. RELATED WORK

For physical system modeling, Willems [23] provides an excellent introduction showing the necessity of acausal modeling. A well-known and widely-used multi-domain acausal modeling paradigm is the theory of bond graphs [14], which supports the composition of physical models from the mechanical, electrical, thermal, hydraulic and magnetic domains. Modelica [10] is a standardized language for describing multi-domain physics and computational systems. While there are industrial strength tools supporting the language, the complete formalization of Modelica is lacking. Nonetheless, there are several attempts for the partial formalization of Modelica.

Kågedal and Fritzson [13] describes a formal semantics for the Modelica language using natural semantics. They use the RML compiler generator to generate executable implementation based on the semantic specifications. While the RML specification also contains specifications for instance creation, Mauss [19] argues that for a human reader a procedural view of Modelica instant creation is more helpful. Broman [7] discusses the type system of Modelica, and concludes that the Modelica type system is too informal, and needs proper formalization in the future.

A different direction for equation-based modeling is the usage of functional languages, that naturally lends itself to higher-order and structurally dynamic modeling. Broman and Nilsson [8] introduce a node-based connection semantics for equation based object-oriented (EOO) modeling languages, and they introduce a prototype implementation called the Modeling Kernel Language (MKL). They argue that the node-based approach solves the problem of connection semantics in functional EOO languages. Another functional paradigm is the Functional Hybrid Modeling (FHM) [22] that is the foundation for Hydra, a declarative non-causal modeling language supporting dynamic structures. Nilsson [21] introduced a type-based structural analysis for FHM, but the idea is applicable to other equation-based languages as well. Such a type system can be used for arguing about the solvability of the equations.

For computational system, BIP (Behavior, Interaction and Priority) [2] is a framework developed at VERIMAG, which supports the composition of computational systems by abstracting interactions. Synchronous and asynchronous, rendezvous and broadcast interactions are provided by the BIP language, allowing the expression of all kinds of computational interactions. Bliudze and Sifakis [5] introduce the algebra for BIP connectors and interactions, and they define [6] the SOS style formalization of glue operators as well.

Ptolemy [11, 12] is a hybrid framework built upon the notion of actors and directors. Actors are components, whose interactions is controlled by the directors. Directors implement different Model of Computations (MoC), e.g. finite-state machines, synchronous and dynamic data flows, process networks, discrete events, continuous-time and synchronous-reactive systems. The semantics of Ptolemy is explained by the actor abstract semantics [16]. Being a hybrid framework Ptolemy provides a continuous-time director, however acausal modeling is not supported.

3. CYBER-PHYSICAL SYSTEMS

In this section we introduce a formalism that can be used for modeling acausal CPS languages. Our model builds on several well-known formalism such as the interface theory [1] and the tagged signal model [15]. The formalism provides support for acausal physical modeling, analog signal interactions and computational interactions.

3.1 Variables

Given a set of tags T and set of values V, an event $e \in T \times V$ is defined as a pair of tag and value. Let $\perp \notin V$ be a special value denoting the absence of value, and $V^{\perp} = V \cup \{\perp\}$.

A variable x is a named entity with tag set T_x and value set V_x. A trajectory $\nu\colon T_x \to V_x$ of variable x is a (total) functional relation between tags and values. The set of all trajectories Γ_x is then

$$\Gamma_x = (V_x^{\perp})^{T_x}$$

Further, we write $T(\nu)$ to denote the set $T(\nu) = \{t \mid \nu(t) \neq \perp\}$, i.e. the tags at which the trajectory has a defined (non-absent) value.

Given a set of variables $X = \{x_1, ..., x_n\}$, $W_X = \prod_{x \in X} \Gamma_x$ is the tuple of all the possible trajectories for its variables. Clearly, we can decompose any trajectory $w \in W_X$ to its constituents: let $w_x \in \Gamma_x$ denote the trajectory of variable $x \in X$ in w.

Given a trajectory $w \in W_X$ and a set of variables $Y \subseteq X$, we define $\pi_Y(w)$ to be the projection of w onto Y. Formally, $\pi_Y(w) = \prod_{y \in Y} w_y$.

We can distinguish different types of variables based on their tag set, value set and their physical interpretation. For simplicity, in the following we assume that the value set V_x for all variables is the set of real numbers, and we distinguish the following variable types:

- X_p is a set of physical variables denoting physical quantities. Their tag set $T = \mathbb{R}$ is the real numbers, their trajectory are everywhere defined $T(\nu) = \mathbb{R}$ for any $x \in X_p$ and $\nu \in \Gamma_x$, and they are always valued over the reals $V = \mathbb{R}$. Further, in a more detailed model we would distinguish electrical, mechanical, and other physical variables.
- X_c is a set of computational (discrete-event) variables. The tag set for these variables is arbitrary (e.g. $T = \mathbb{N}$ for synchronous/reactive systems or $T = R \times N$ for discrete-event systems), as long as $T(\nu)$ for any $x \in X_c$ and $\nu \in \Gamma_x$ is order-isomorphic to a subset of integers [15].

Note that computational variables that are interpreted over the physical time may take several values at any given physical time instant. In order to represent the causality of these values, the tag set is often enriched to represent steps. For example, super-dense time [16,17] defines the tag set for discrete-events as $\mathbb{R} \times \mathbb{N}$, and non-standard real time ${}^*R$ [3]) enriches the physical real time by infinitesimal time steps.

3.2 Ports

A port $p = \{x_1, ..., x_n\}$ is a finite set of variables. Ports are typed according to the variables they consists of, and port types have well-formedness rules that describe the variables that can live on that port. In the following, we distinguish the following primitive port types:

- power port $p = \langle x, y \rangle$ is a pair of physical variables $x \in X_p$ and $y \in X_p$. We call the first component x the effort variable, and the second component y the flow variable (in some environments they are called across and through variables). The physical interpretation of these variables are modeling environment and physical domain dependent.
- physical signal port $p = \{x\}$ is a singleton set of physical variable $x \in X_p$, that we call a signal variable.
- computational port $p = \{x\}$ is a singleton set of computational variable $x \in X_c$.

There are no other primitive port types. Note that based on these port types we could define compound ports, however, we leave this as a future work. In the following, we write P_P for power ports, P_S for physical signal ports and P_C for computational ports.

In the following, we assume an infinite pool of variables X^* that denotes all the possible variables in the system, and an infinite pool of ports P^* that are partitions over X^*. I.e. $p \in P^*$, $q \in P^*$ and $p \neq q$ implies $p \cap q = \emptyset$.

3.3 Components

A CPS process is characterized by a set of physical and computational variables and their possible trajectories over time. A CPS component is a structure for CPS processes, that defines an interface (a set of ports) through which the component interacts with its environment. Thus, a CPS component is a set of variables and ports, and their behavior over time.

DEFINITION 1. *A CPS component $\mathcal{C} = \langle X, P, Beh \rangle$ consists of the following:*

- *a finite set of variables $X = \{X_1, \ldots, X_n\}$,*
- *a finite set of non-overlapping ports $P \subseteq 2^X$, where port $p \in P$ denotes a set of variables that can be observed through it,*
- *a set of behaviors $Beh \subseteq W_X$.*

Intuitively, the behaviors Beh of a CPS component is a set of trajectories that contains all the allowed evolutions of its variables. Equivalently, the behavior of a component can be represented as a finite or infinite set of constraints Θ.

DEFINITION 2. *The behavioral constraints Θ of a CPS component is a finite or infinite set of first-order sentences (i.e. they contain no free variables), for which $Beh = \{x \in W_X \mid x \models \Theta\}$, where $\models$ is a relation that captures that trajectory x satisfies each formula $\theta \in \Theta$.*

3.4 Composition

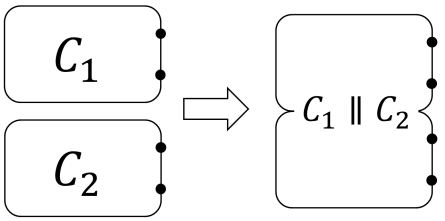
We assume a set of atomic CPS components that are predefined. Compounds CPS components are then composed from atomic components and other compound components by means of two operators, composition and connection.

DEFINITION 3. *Two components are compatible if and only if their variables are non-overlapping. Formally, component A and B are compatible iff*

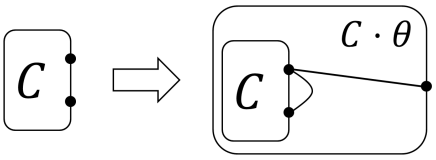
$$X_A \cap X_B = \emptyset$$

DEFINITION 4. *Parallel composition $A \parallel B$ of compatible components A and B is the interaction-free composition of A and B defined as the set-theoretical union of their structures, and its behavior is the cross-product of the behaviors of A and B:*

$$A \parallel B = \langle X_A \cup X_B, P_A \cup P_B, Beh_A \times Beh_B \rangle$$



(a) Parallel composition of $C_1 \parallel C_2$



(b) Connection and port hiding by $C \cdot \theta$

Figure 1: Operators of component composition.

The parallel composition operator is shown in Fig. 1(a).

Alternatively, the behavioral constraints of their parallel composition $A \parallel B$ is the conjunction of their behavioral constraints:

$$Beh_{A\parallel B} = \{x \in W_{X_A \cup X_B} \mid x \models \Theta_A \wedge \Theta_B\} \qquad (1)$$

LEMMA 1. *Composition is associative and commutative, thus* $A \parallel B = B \parallel A$ *and* $(A \parallel B) \parallel C = A \parallel (B \parallel C)$.

DEFINITION 5. *An interconnection* $\theta \subseteq \mathcal{P}^* \times \mathcal{P}^*$ *is a set of port pairs, where* $\mathcal{P}^*$ *denotes all the ports of the system.*

However, not all interconnections are well-formed. Therefore, we need a definition for well-formed interconnections.

DEFINITION 6. *A well-formed interconnection* $\theta \subseteq \mathcal{P}^* \times \mathcal{P}^*$ *contains only compatible connections, i.e. physical ports to physical ports, signal ports to signal ports and computational ports to computational ports, and* θ *does not connect any port to itself.*

DEFINITION 7. *The external ports* $E_{C\theta}$ *of a component* C *and interconnection* θ *is a set of ports that are elements of the interconnection but not the component. Formally,*

$$E_{C\theta} = \{x \in P^* \mid x \notin P_C \wedge \exists y \in P_C . \langle x, y\rangle \in \theta \text{ or } \langle y, x\rangle \in \theta\}))$$

DEFINITION 8. *The connection* $C \cdot \theta$ *of a component* C *by a well-formed interconnection* θ *is defined as follows:*

$$C \cdot \theta = \langle X \cup E, E_{C\theta}, Beh_{C\cdot\theta}\rangle$$

where $E = \{x \mid x \in E_{C\theta}\}$ *is the set of external variables, and the behavior is defined as follows:*

$$Beh_{C\cdot\theta} = \{x \in W_E \mid \exists y \in W_{X\cup E}.\ \pi_E(y) = x \text{ and } y \models \Theta_C \wedge \theta^I\}$$

where θ^I *is a set of formula describing the interpretation for the interconnections, and* π *denotes projection as defined above.*

The behavior $Beh_{C\cdot\theta}$ of compound component contains exactly those trajectories whose projection to C are trajectories of Beh_C, and that fulfill the interpretations of connections. Note that the trajectories of variables belonging to unconnected ports are unconstrained from outside (except for the flow variables which are zero according to the interpretation of power connections). The connection operator is shown in Fig. 1(b).

Observe that the ports of the original components are hidden in the new component. The reason for this is found in the semantics of physical interconnections. In order to formulate the equations for a physical port, we need all its interconnections. If we do not hide the ports of the original component, additional interconnections could be established to them, and therefore the interpretation for the connections cannot be encapsulated in the component. If we hide the original ports, the external port variables are still dependent on the environment, but they create a barrier between the internal variables and the environment.

4. INTERPRETATION OF CONNECTIONS

So far we defined operators that allow the hierarchical composition of components, however we deliberately left out the details of the interpretation of connections. The reason being that for different combinations of ports/variables, the interconnections have different semantics as discussed below.

In the following, we only deal with well-formed connections, where only compatible ports are interconnected. Note that heterogeneous connections can be modelled by interconnecting heterogeneous ports through auxiliary components that perform the semantic matching between them.

Based on the definition, a well-formed connection θ is the disjoint union of θ_P, θ_S and θ_C, where $\theta_P \subseteq P_P \times P_P$ is the set of power connections, $\theta_S \subseteq P_S \times P_S$ is the set of physical signal connections, $\theta_C \subseteq P_C \times P_C$ is the set of computational connections.

Power connections

For physical connections, the semantics is defined by variable sharing across the connector. A physical connector is a set of ports, whose members are defined by the transitive closure of their connections. The set of connectors θ_P^* is the transitive closure of port connections θ_P, given as the least fixed point solution for the following equations:

$$\begin{array}{ll} p \in P_P \implies (p,p) \in \theta_P^* & \text{(reflexivity)} \\ (p,p') \in \theta_P \implies (p,p') \in \theta_P^* \wedge (p',p) \in \theta_P^* & \text{(symmetry)} \\ (p,p') \in \theta_P^* \wedge (p',p'') \in \theta_P^* \implies (p,p'') \in \theta_P^* & \text{(transitivity)} \end{array}$$

Therefore, $\{p' \mid (p,p') \in \theta_P^*\}$ denotes the physical connector containing port p. Next, define a *sgn* function as follows:

$$sgn(p) = \begin{cases} -1 & \text{if } p \text{ is external port} \\ 1 & \text{otherwise} \end{cases} \qquad (2)$$

The interpretation for physical connections are then:

$$\theta_P^I = \bigwedge_{(p,p')\in\theta_P} p_1(t) = p_1'(t) \qquad (3)$$

and

$$\theta_P^I = \bigwedge_{p\in P_P} \left(\sum_{\langle p,p'\rangle \in \theta_P^*} sgn(p') \cdot p_2' \right)(t) = 0 \qquad (4)$$

for any $t \in \mathbb{R}$, where p_1 and p_2 denote the first and second component of the power port, and function $(\sum f)$ is defined by $(\sum f)(t) = \sum f(t)$.

Note that Eq. 3 describes an acausal constraint (i.e. p_1 determines p_1' and p_1' determines p_1). Such acausality is an important property of physical systems [14].

Physical Signal and Computational Connections

The semantics of physical signal and computational connection is variable assignment, i.e. the value of the source port is assigned to the destination port at every time they are defined. Given the set of connections $SC = \theta_S \cup \theta_C$, their interpretation is defined by the following equation:

$$SC^I = \bigwedge_{\langle p,p' \rangle \in SC} \langle t, v \rangle \in p \text{ and } v \neq \perp \implies \langle \varphi(t), v \rangle \in p' \tag{5}$$

for all $t \in T$ (where T is the tag set for the connected variables), where φ is a function for expressing causality (in this paper we do not develop the idea, however it can be used for defining an ordering of events). In this work, we can assume that $\varphi(x) = x$ is the identity function. Note that here we assume that only ports with the same tag sets are interconnected, it remains a further work to examine heterogeneous connections.

Unlike the acausal equation for power connections, Eq. 5 represents a causal dependency. This allows p' to take any values when p is absent, which enables signal multiplexing (however note that multiplexing needs heterogeneous tag sets).

An algebraic loop is a directed cycle that does not contain any integrators or delays. The semantics of such loops needs further elaboration, since algebraic loops may have no solution or multiple solutions, furthermore they lead to situations when a variable has multiple values at a given physical time instant. There are three ways to tackle the problem: (i) prohibit algebraic loops (e.g. LUSTRE [4]), (ii) define the non-deterministic semantics by allowing any values that satisfy the loop equations (e.g. the SIGNAL language [4]), (iii) develop the Scott semantics of the language, and define its semantics as the least fixed point solution for the loop (e.g. ESTEREL [4], Ptolemy).

5. DIFFERENCES IN MODELING ENVIRONMENTS

Most of the physical and CPS modeling environments are based on graphical network models that use hierarchical composition of components. Such a hierarchy can be represented using our formalism, even though a lot of details must be abstracted away. What concerns us more is that there are major differences between different modeling environments in the representation of the behavior and the interpretation of their physical variables. In this section, we discuss some of the variants for these differences.

5.1 Behavioral Model

The behaviors of physical lumped parameter models are often represented as a set of differential algebraic equations. Such paradigm is followed by Modelica [10] and Simscape [18], where the behavior of the components are described using a textual language for differential algebraic equations (DAEs).

A different path is followed in bond graphs modeling [14], where the behaviors of the basic components are implicit, and are determined by the type of the component. In the most basic bond graph language, nine components are distinguished, seven of which represent different behavioral equations, and two are topological configurations.

On the computational part, Simscape is integrated with Matlab Simulink, which means that its computational behavior is described by stateflow, Matlab scripts, and external imperative languages such as C and Java. Modelica has a simple built-in imperative language that can be used for describing simple controllers, but it also provides an interface for interacting with external languages. Bond graph does not define any computational models, but a bond graph variant, the hybrid bond graphs [20] were defined together with a simple automata language.

5.2 Interpretation for Physical Variables

Modeling environments use different interpretations for the physical variables. In Modelica *connectors* are used for connecting Modelica models. For physical connections, a connector is a set of continuous-time variables, some of which are marked as flow. Unmarked continuous-time variables are assumed to be potential variables. Since in our formalism we did not define how aggregated (non-primitive) ports can be built, we consider only the simplest connectors in Modelica: connectors that are composed of a potential and a flow variable. The semantics of such connections are similar to the model we formalized in the paper. However, it remains a future work to prove this, as well as to discuss the behavior of input and output ports of Modelica.

The semantics for Simscape is very similar, however, the physical variables are called across-variables and through-variables. In mechanical domains, through-variables are effort type, and the across-variables are flow type variables. In non-mechanical domains, the across-variables are effort type, and the through-variables are flow type variables.

Bond graphs define effort and flow variables for representing physical quantities. However, the interpretation of these variables are different from other modeling environments. For example, Modelica uses flanges for describing translational mechanical systems, where flanges are connectors built from a force and a position variable. In contrast, the quantities of the translational mechanical domain in bond graphs is force and velocity. Further, bond graphs use voltage-difference and current in electrical domains, while Modelica is based on electrical pins that use voltage and current. Such semantic differences must be accounted for when composing different modeling languages.

In bond graphs each bond defines a connection between exactly two components with the meaning the efforts are equal on both ends of the bond, and whatever flows in on one end of the bond must flow out on the other end. For representing topology, bond graphs define two types of junctions, one-junction and zero-junction, whose behavior define the topological constraints for the effort and flow variables.

We can explain the semantics of bond graphs with the model we presented in this paper, we only need to model the junctions as components with a predefined behavior (of course this is not a new idea, e.g. [9]):

- A one-junction with n bonds is a component which has $2n$ physical variables partitioned into n physical power ports. The behaviors of the component is all

the trajectories over the variables where all the effort variables are equal, and the flow variables sum up to zero.

- A zero-junction with n bonds is a component which has $2n$ physical variables partitioned into n physical power ports. The behaviors of the component is all the trajectories over the variables where all the flow variables are equal, and the effort variables sum up to zero.

Notice that in this model each bond graph node is modelled with a component, each bond is modelled with an interconnection and each port of the components are connected with exactly one another port.

6. CONCLUSION

In this paper we have defined a formalism for discussing the generic CPS modeling concepts used for system integration. Our intention is not to define a framework that can be used for describing every aspect of CPS modeling languages. Rather, we consider it as a language that can be used for discussing a small but important portion of the behavioral semantics of CPS languages. Further, we consider it as a tool for comparing different variants, and a model that can be used for further investigations in CPS design. As shown in the related work, there are different alternatives to the component-based approach that do not fit into the world of our model. It needs further work to determine the applicability and usability of our approach.

Clearly, our model needs a great deal of work to understand the continuous-time and discrete-time cross interactions. As a next step we plan to investigate how the discrete-time world fits into the framework. Furthermore, we can extend the model in several ways. For example, by attaching dependencies to the variables of the components, we plan to perform dependency analysis, that can be used for causality analysis and fault propagation analysis. Also, we plan to examine how the model can be extended so that we can structurally ensure the existence of a model behavior, as well as the uniqueness of its behavior.

ACKNOWLEDGEMENT

We are very grateful for the useful suggestions of the anonymous reviewers. This work was supported by the National Science Foundation under grant number CNS-1035655 and by the European Union and the European Social Fund through project FuturICT.hu (grant no.: TAMOP-4.2.2.C-11/1/KONV-2012-0013).

7. REFERENCES

[1] L. D. Alfaro and T. Henzinger. Interface theories for component-based design. *EMSOFT*, 2001.

[2] A. Basu, M. Bozga, and J. Sifakis. Modeling heterogeneous real-time components in BIP. In *SEFM*, pages 3–12, 2006.

[3] A. Benveniste, T. Bourke, B. Caillaud, and M. Pouzet. Non-standard semantics of hybrid systems modelers. *Journal of Computer and System Sciences*, 2011.

[4] A. Benveniste, P. Caspi, S. A. Edwards, N. Halbwachs, P. Le Guernic, and R. De Simone. The synchronous languages 12 years later. *Proceedings of the IEEE*, 91(1):64–83, 2003.

[5] S. Bliudze and J. Sifakis. The algebra of connectors – structuring interaction in BIP. *IEEE Trans. Computers*, 57:1315–1330, 2008.

[6] S. Bliudze and J. Sifakis. A notion of glue expressiveness for component-based systems. CONCUR, pages 508–522, 2008.

[7] D. Broman, P. Fritzson, and S. Furic. Types in the modelica language. In *International Modelica Conference*, pages 303–315, 2006.

[8] D. Broman and H. Nilsson. Node-based connection semantics for equation-based object-oriented modeling languages. *Practical Aspects of Declarative Languages*, pages 258–272, 2012.

[9] F. E. Cellier and À. Nebot. The modelica bond graph library. In *International Modelica Conference*, 2005.

[10] P. Fritzson and V. Engelson. Modelica—A unified object-oriented language for system modeling and simulation. *ECOOP*, page 67–90, 1998.

[11] A. Goderis, C. Brooks, I. Altintas, E. Lee, and C. Goble. Composing different models of computation in kepler and ptolemy II. In *ICCS*, volume 4489, pages 182–190. Springer Berlin / Heidelberg, 2007.

[12] A. Goderis, C. Brooks, I. Altintas, E. A. Lee, and C. Goble. Heterogeneous composition of models of computation. *Future Generation Computer Systems*, 25(5):552–560, May 2009.

[13] D. Kågedal and P. Fritzson. Generating a modelica compiler from natural semantics specifications. In *SCSC*, pages 299–307, 1998.

[14] D. Karnopp, D. Margolis, and R. Rosenberg. *System dynamics: modeling and simulation of mechatronic systems*. Wiley, New York, 1997.

[15] E. A. Lee and A. Sangiovanni-Vincentelli. A framework for comparing models of computation. *IEEE Trans. Computer-Aided Design*, 17:1217–1229, 1998.

[16] E. A. Lee and H. Zheng. Leveraging synchronous language principles for heterogeneous modeling and design of embedded systems. In *EMSOFT*, pages 114–123, 2007.

[17] O. Maler, Z. Manna, and A. Pnueli. From timed to hybrid systems. In *Real-Time: Theory in Practice*, pages 447–484. Springer, 1992.

[18] MathWorks. Simscape, Nov. 2012.

[19] J. Mauss. Modelica instance creation. In *International Modelica Conference*, 2005.

[20] P. Mosterman and G. Biswas. A theory of discontinuities in physical system models. *Journal of the Franklin Institute*, 335(3):401–439, 1998.

[21] H. Nilsson. Type-based structural analysis for modular systems of equations. In *EOOLT*, pages 71–81, 2008.

[22] H. Nilsson, J. Peterson, and P. Hudak. Functional hybrid modeling. *Practical Aspects of Declarative Languages*, pages 376–390, 2003.

[23] J. Willems. The behavioral approach to open and interconnected systems. *IEEE Control Systems*, 27(6):46 –99, Dec. 2007.