MODEL-DRIVEN TIME-ACCURATE DEVS-BASED APPROACHES FOR CPS DESIGN

Abdurrahman Alshareef
Hessam S. Sarjoughian

Arizona Center for Integrative Modeling & Simulation
School of Computing, Informatics and Decision Systems Engineering
Arizona State University
699 S. Mill Avenue
Tempe, AZ, 85281, USA
alshareef@asu.edu, sarjoughian@asu.edu

ABSTRACT

Performance analysis and verification of Cyber-Physical Systems (CPS) is of utmost importance due to the cruciality of the decision making in such systems. Therefore, modeling can be beneficial especially for issues related to the tight coupling between computational and physical parts. In this work, we utilize the extensive research on the simulation and model-checking for designing computational-physical interactions in the context of CPS. We also propose an action-level model-driven activity modeling approach based on DEVS. We employ time intervals (TIs) to govern communication between computational and physical components at the level of actions. We extend the activities metamodel to instantiate activities suitable for time-critical cyber-physical systems. We create a DEVS-Suite generic and polymorphic library to simulate these models conforming to the parallel DEVS formalism. We demonstrate with a smart vehicle intersection model and discuss some verification capabilities.

Keywords: Behavioral Specifications, Cyber-Physical Systems, Model-Driven Engineering, Parallel DEVS.

1 INTRODUCTION

In Cyber-Physical Systems (CPS), the system can be recursively built upon smaller parts that are much simpler to develop, operate, and maintain. This method of incremental construction ultimately is aimed at achieving correctness through restraining behavioral complexity (Sztipanovits et al. 2011, Derler, Lee, and Vincentelli 2012). In the literature, concepts and formalisms are extensively discussed to establish the basis for systems to be designed in such disciplined and accurate manners utilizing modularity. Some formalisms allow models to be specified separately or collectively based on component and composition concepts (Zeigler, Praehofer, and Kim 2000, Giese and Burmester 2003, Alur 2015). These definitions, as well as their other corresponding incarnations, are shown to be key for designing CPSs (for an example see Mosterman and Zander 2016).

Inherent in any CPS is heavy interaction among decision points in some computational world and their interacting components in the physical world. The nature of such a relationship is tight, making the flow of information central in both direction, from computational to physical and the other way around. A direct implication is that coordinated interactions must satisfy both logical and physical rules. The immersion of computational consequences on physical environment denotes one direction of the relationship while the

other direction is denoted by the information observed in the Cyber parts. The relationship, in its broader sense, is considered from multiple perspectives beside the direction. Multiplicity and containment are two important examples of relationship properties by which complexity of such systems can be determined. Within the context of CPS, a specific type of relationship is also considered where one end is the physical entity. The broader properties can be specialized for this specific type in order to ensure the correctness especially for critical interactions.

Timing is a crucial aspect in CPS. Major difficulties are caused in CPS due the physical nature of time (NIST 2016) which is inherent and yet cannot be controlled. Operations of some computational and physical entities are not inclined to an isolation of uncontrollable phenomena as physical time passes. The impact of the overall performance of system components can be affected to a larger degree relative to the variety of conditions that are accompanied with the timing specifications. Any breach of the timing agreement between the heterogeneous entities of the system may reveal threats to the entire system and therefore pose further difficulties.

The seclusion of timing in modern software as well as hardware systems has led to a major deficiency for time-critical CPSs. Much of the processes design is currently performed on the basis of as fast as possible execution with as much time granularity that can be afforded. Therefore, abstracting out the time aspect significantly is evident in order to achieve an optimal or even a satisfying result. However, in the critical stages of system design including validation, some computation may turn out to be not useful and possibly at the expense of some others. A late execution may not be valid and may even cause a serious damage. The validity of taking unsanctioned actions may not hold under some timing constraints.

Several concepts have been introduced in the theory of modeling and simulation in order to inherently account for timing needs. These concepts, such as elapsed time, deadlines and time intervals, are very beneficial and their importance increases symmetrically relative to the cruciality of the CPS. In this work, we attempt to work on the assimilation of the system-theory definitions toward better accommodation of modern system design concepts as manifested by the CPS. An action-level modeling approach is introduced with an emphasis on actions constrained with time invariant for real-time environment. The conformance of the developed models is improved through meta-modeling in which the higher concepts are addressed at a higher level of abstraction when possible. A model-driven approach is then attained for the behavioral specification for model components at the individual and composite levels. The proposed approach makes a clear distinction between actions for the logic associated with the computational entities and their interactions with physical entities.

Thus, it is important to characterize actions in the CPS to appropriately account for their consequences. Overall, actions may be performed at any point in time for different purposes subject to acceptable and possible time granularity. Some of these actions only take place in the computational part of the system, others in the physical part only. An example for the former can involve any pure computations. The latter can involve physical operations and responses. Another set of actions includes the interaction between the computing environment and the physical parts. We characterize these actions to be actuating and sensing actions. This set of actions is crucial in the context of CPS since they deal with coupling between computational and physical parts as shown in Figure 1.

We begin by presenting some of the necessary background with respect to the underlying formalisms of this work, that is, brief background is given about P-DEVS, RT-DEVS and ALRT-DEVS, all of which are targeted for simulation. To support verification, instead of validation, the Finite-Deterministic DEVS (Hwang and Zeigler 2006) is developed. This is a DEVS variant for model-checking. More recently, the Constrained DEVS is developed. It targets the underpinning non-deterministic and stochastic aspects of CPS (Gholami and Sarjoughian 2017). Background is also given for timed automata formalism. We follow that with a discussion of the existing works with a focus on model-driven DEVS-based methodologies for

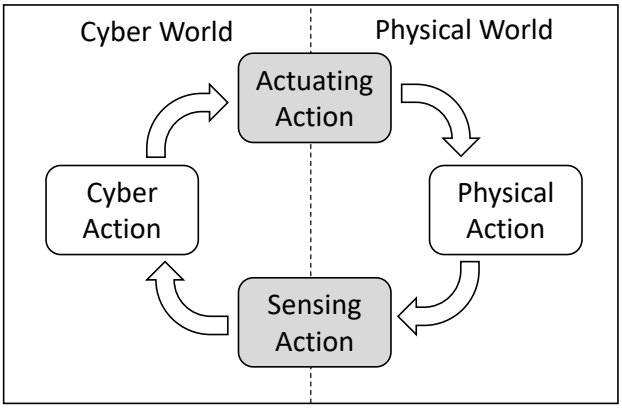

Figure 1: Actions in the CPS are characterized into four types. The types in grey are crucial from a CPS standpoint since they are akin to the tight coupling between cyber and physical parts. Actuating actions, for example, can impact the physical environment directly and therefore their consequences are critical.

addressing similar problems that can be applied to CPS. Then, we present the action-level specification for modeling CPS. Before concluding, we demonstrate the approach by modeling the dynamics of a smart intersection with a multiple relays and a controller. A discussion on the verification of these models is also presented.

2 BACKGROUND

There are many extensions of classic DEVS. In every one, the extensions and variants maintain some of the key concepts and properties while extending or replacing them for certain needs. A prime example is parallel-DEVS where atomic and coupled models can execute simultaneously as compared with classic DEVS. Several extensions are conducted with respect to the time advance function to provide capabilities such as real-time Real-Time DEVS (RT-DEVS) which uses time-window (aka Time Interval (TI)) (Wang and Cellier 1990) and actions. This work relies on Action-Level Real-Time DEVS which introduces real-time statecharts for modeling actions. We will describe these briefly as well as other related formalisms in the following sub-sections.

2.1 Parallel DEVS

The set-theoretic specification of the atomic model is an abstract representation of a system component. The formal specification can be defined independent of any specific platform, language, and simulator. The Parallel DEVS (P-DEVS) was proposed by (Chow and Zeigler 1994) to provide both conceptual and execution benefits for the modelers. The basic formalism of P-DEVS model is an algebraic structure – atomic model = $(X, Y, S, \delta_{ext}, \delta_{int}, \delta_{con}, \lambda, ta)$. X is the set of input events. S is the tuple of sequential states with at least two variables which are *sigma* (σ) and *phase*. Y is the set of output events. δ_{int} and δ_{ext} are the internal and external transition functions, respectively. δ_{con} is the confluent transition function which can be specified to handle the collision between external and internal events. λ is the output function which transforms S into Y at specific time instances. ta is the time advance function which maps the internal state into a positive real number using elapsed time since last state transition.

2.2 Real-Time DEVS (RT-DEVS)

This extension is developed to allow for real-time simulation of the RT-DEVS models (Zeigler, Praehofer, and Kim 2000). The simulator of the parent formalism is extended in such a way to account for the interaction with the real environment (Stankovic 1988). The definition of the atomic model in RT-DEVS is extended with interval time and activity mapping functions. A set of activities with timing constraints is defined. In the external transition function, the total state set is defined where the elapsed time is bounded by 0 and the time interval for the state, $ti(s)$ where $s \in S$, inclusive. Models of this nature need to be properly created to account for the real-time definitions. The execution time is constrained by lower and upper bound that are imposed on the activity to map the state into a Cartesian product of two sets of non-negative real numbers and infinity inclusive, $ti : s \rightarrow \mathbb{R}^{+}_{0,\infty} \times \mathbb{R}^{+}_{0,\infty}$. It is important to note that the execution time is non-deterministic due to the limited control imposed by the physical environment. It is also important that the evaluation of the computational time along with their associated variables has to be proper to insure the validity of the model.

2.3 Action-Level Real-Time DEVS (ALRT-DEVS)

This extension has been proposed by (Sarjoughian and Gholami 2015) to support defining real-time constraints at the action level from both modeling and simulation point of views and supported with the concept of locations defined for real-time statecharts. That is, the modeler will be able to develop a real-time model and, under certain conformance condition, this model will be simulatable by the provided real-time simulator in which the abstract simulator has been extended for real-time software system. It fundamentally lies on the basis of the parallel and real-time DEVS as well as real-time statecharts.

The notion of time in ALRT-DEVS is defined based on both theoretical and pragmatic perspectives. A unified concept with formalized specification for logical-time, real-time and physical-time underlie ALRT-DEVS formalism. The time in models as well as their simulators is concertized according to a physical clock where the distinction between physical-time, real-time, and logical-time, is well-established. The physical time denotes the time in the actual (physical) environment in which infinite accuracy and precision is encountered. All other timing schemes include a physical signal (NIST 2016). Therefore, real-time is an approximation of physical-time but not equal to it due to uncontrollable factors in computing platforms. The logical-time is an abstract computable quantity to provide the basis by which the software logical clock can proceed in an increasing manner. It only ideally has the properties of the physical clock.

First, state variables are characterized to be primary and secondary variables. The primary state variables are the *phase* and the sigma σ by which the next state is determined in P-DEVS models generally. The secondary state variables are defined as needed to denote for specific system dynamics. The notion of location is therefore defined to enable different kind of transition on the basis to the state change whether it is associated with primary or secondary state variable, or possibly both. Transitioning between different locations is usually associated with which guards are specified in terms of secondary state variable. Actions, which are the fundamental units in the scope of the current work, can be then specified for a location.

2.4 Timed Automata

The theory of timed automata (Alur 1999) explicitly admits the notion of time by which it becomes suitable for the modeling and analysis of real-time systems as opposed to basic logical model checking. The behavior of such systems can be modeled in state-transition formal notations. The notation is then annotated with timing constraints using clock variables. Further restrictions can be therefore imposed on the state space in order to allow for the verification of some system properties under the given timing constraints. A transition

system is defined by a set of states, a set of initial states, a set of labels or events, and a set of transitions. The timing constraints are then introduced with a finite set of real valued clocks for finite graph where the vertices are called locations and the edges are called switches. The locations are associated with some time invariant to constrain the elapsing of time in that location. The switches are instantaneous.

3 MODEL-DRIVEN DEVS-BASED METHODOLOGIES FOR CPS

There have been several DEVS-based approaches that are suitable for use in CPS. In (Sarjoughian, Gholami, and Jackson 2013), a new model for interacting an ALRT-DEVS with a physical system is proposed. The simulation is composed with a computational-physical system. This kind of systems can be considered a cyber-physical system if it offers the designated capabilities for CPS. The DEVS-Suite simulator is extended to support the capability of communication between the computational and the physical parts of the system. The experiment is devised with a tight coupling connection to ensure the validity of the proposed model under hard real-time constraints. The connection between a controller and multiple relay phidget is thoroughly conducted under different setting to examine the turnaround time for the switching actions. The role of examining such a hard constraint is complementary to the logical-time constraints. And therefore, together they form a stronger basis for carrying out different experiments about the CPS under study.

There are many other works that do not directly address CPS, however, they can be utilized toward that direction since the problems, in which they try to address, are akin to their counterparts within CPS context. In (Risco-Martín et al. 2016), although the work is not directly targeting CPS, the authors proposes a model-driven hardware-in-the-loop method to incrementally obtain embedded hardware starting with their software representations. The methodology is based on the DEVS formalism. Instead of extending the simulator, as proposed in (Hong et al. 1997), the hard real-time constraints are specified in the parent formalism through star models. Star models are an interface for atomic models to allow for building an abstract models for the concrete hardware ones based on the concept of transparent simulation environment. The elevator circuit real-time model (Zeigler, Praehofer, and Kim 2000) is designed and implemented with the adder as an HIL component.

Nonetheless major problem still persists in modeling CPS notwithstanding the ongoing efforts in languages, notations, and tools (Derler, Lee, and Vincentelli 2012). The CPS sensitivity to timing poses challenges. Time-accurate approaches are significant in the modeling of CPS. Regardless of the usefulness of modeling languages such as (OMG 2012, OMG 2016), the missing semantics and the weaker notion of time are two major causes for not using them in time-critical system design.

4 ACTION-LEVEL DEVS SPECIFICATION USING ACTIVITY MODELING

In a previous work (Alshareef and Sarjoughian 2017), we proposed establishing a DEVS foundation for the activity modeling and simulation. However, the time notion was limited to support a basic simulation step to somehow correspond to the debugging step. We extend this notion of time to be supported at the action level. That is, the action can be defined with time constraints to elevate the activity modeling further toward time-accurate activities.

4.1 CPS Activities Metamodel

The metamodel of the UML activities (OMG 2012, Eclipse Foundation 2016) is circumscribed and then extended with the basic necessary definitions to elevate the support for the concept of state by providing a basis for their conformance to the DEVS formalism at a higher level. The metamodel consists of three major elements. The first element is the action in its broader sense to support modeling at the action level.

The second major element is the control node to support defining control logic. The last but not least is the activity edges where they can also be specialized to be control and object flows. Their mapping to DEVS has been discussed thoroughly in the previous work (Alshareef and Sarjoughian 2017). It should be noted that the performed process is not merely transformation from one form to another, but rather, we advocate grounding activity modeling with the rigorous formal specification where we argue that the DEVS formalism is suitable candidate for this purpose. On the one hand, the definition of state with the strong notion of time can be considered as a fundamental basis for the proposed modeling approach. On the other hand, the modeler can also benefit from the behavioral modeling constructs that are provided within the activity metamodel. Therefore, the action is considered as an abstraction of its corresponding atomic model.

The action is specialized from activity node which also defines the super-type for the control nodes. The control and object flows are both specialized from the activity edge. The edges are instantaneous. For the action, we define time boundaries based on DEVS temporal structure. The ongoing action can have an elapsed time. Therefore, it can be interrupted at any point of time upon receiving some external input event. The elapsed time also indicates the completion of the action. These boundaries are defined in terms of the time advance function. Their corresponding values at the time base may differ without violating their time invariant. This is crucial especially in the modeling of CPS since actions may not be always completed and therefore further considerations should be taken during the modeling process. With the existing DEVS metamodel (i.e., Sarjoughian and Markid 2012), we can incorporate these definitions with the DEVS metamodel at the higher level (see Figure 2).

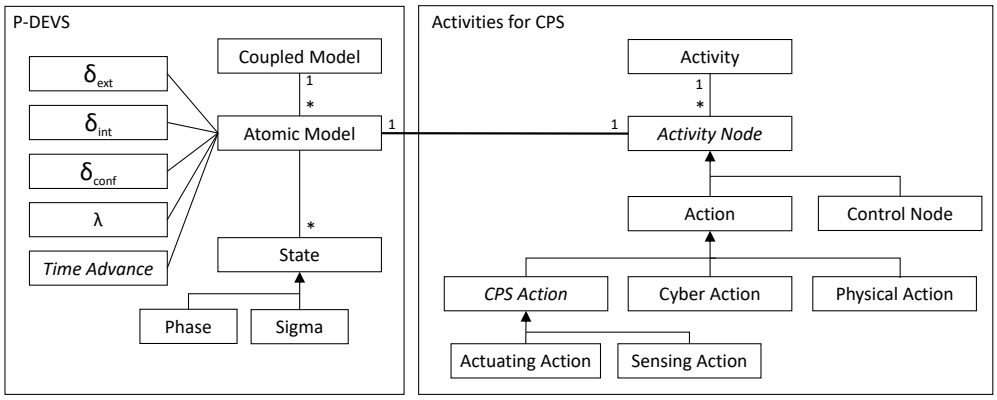

Figure 2: The activities metamodel is circumscribed and extended with CPS action. The DEVS metamodel is also linked with the activities metamodel at a high level to establish the grounding for the P-DEVS modeling and simulation of the CPS activity. Some cardinalities are visually hidden for simplicity. The elements with italic are abstract super-type elements.

4.2 The modeling and Simulation of Smart Intersection

The process of interest is a smart intersection where multiple cars may approach the intersection from multiple directions. There are many crucial time-sensitive requirements that have to be accounted for in this model in various modeling environments. From a physical point of view, cars can approach the intersection from one and only one direction. There are also temporal logic requirements as well where the car has to approach the intersection before it enters it. Such hard real-time constraints would go beyond the logical constraints toward the physical environments and the encountered limitations when interacting with physical aspects such as time latency from dispatching events to a physical relay until getting the acknowledgment back. Each of these aspects is crucial at the design stage of the CPS.

We have covered the logical aspects at the activity level in the previous work. We discuss the temporal aspects to some extent in this work. And we postpone the discussion on a real-time extension of the approach for the subsequent Section 5.

In Figure 3, we simplify the process of the smart intersection by creating a yet another abstraction of it at the action-level in the collective activity. The actions are timed in such a way to ensure the safety by accounting for ramifications. The actions of approaching an intersection can be in some active state in parallel. However, we assume that the intersection must allow the flow for one direction only. Otherwise, accident happens and gets reported to the monitoring model thereafter. Parallel entrance to the intersection can happen only for vehicles approaching from the same direction. These constraints are mere examples and yet further elaborations can be made by the modelers throughout the model development life cycle. The goal is to establish the basis for modelers by taking these models and interpreting them by the simulator to conduct the necessary analysis and verification for the models based on their specification. The state space of the coupled model consists of all permutations for all the possible states of the actions thereof. We will discuss some possibilities of verifying such models in Section 6. For the simulation, after parsing the previous activities, they get interpreted as an activity digraph in the DEVS-Suite simulator (see Listing1). This code snippet shows reading activity nodes and instantiating the atomic models thereafter.

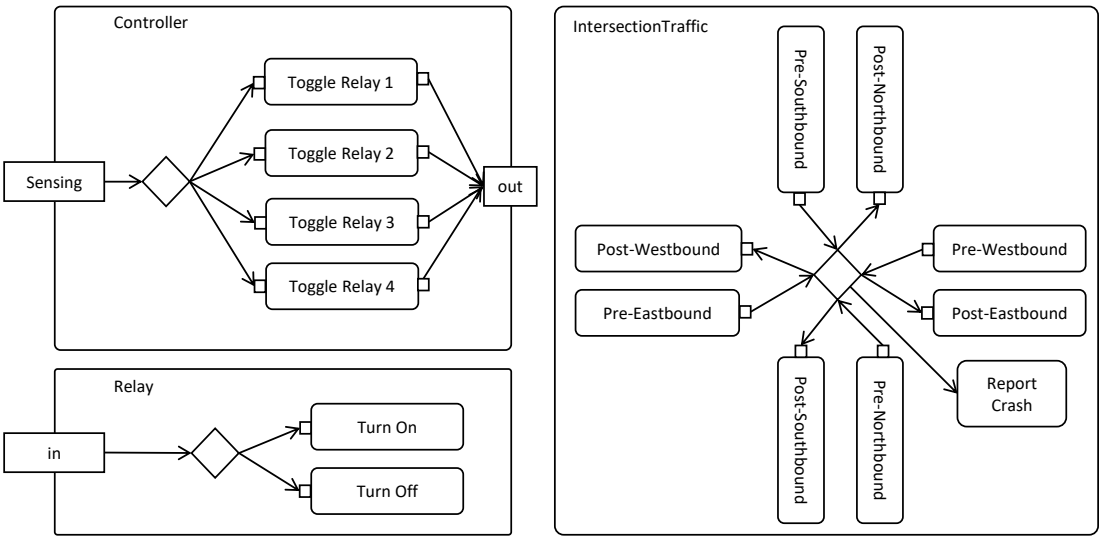

Figure 3: The activities for modeling the smart intersection including the intersection, the relay, and the control.

The external transition function for the atomic model that represents the decision node is shown in Listing 2. The semantics of the decision node has been specified in its general form with respect to the UML. Therefore, it is a domain-specific abstraction in the P-DEVS which yet generally represents the corresponding semantics of the decision node in handling incoming flows. First, the atomic model is initialized in a passive state for unbounded time. Then, upon receiving input events, the model reacts to these input based on the condition associated with the incoming flow which is mapped into a coupling and input port in the corresponding atomic model. After checking the input and the condition, possibly along with the other state variable, the next state is determined. The output is sent out thereafter when applicable and then the internal transition is performed.

This kind of modeling and simulation falls into the realm of methods for analyzing and designing CPS. We consider the modeling and simulation in (Damodaran and Mittal 2017, Alur 1999) to be relatively part of this taxonomy. These approaches focus on the logical aspects and the temporal logical ones whether it is

based on the DEVS formalism as in (Damodaran and Mittal 2017) or on the notion of clock variables with real values such as in (Alur 1999).

Listing 1: The activity interpretation in DEVS-Suite

```
for(ActivityNode node:nodes)
  if(node.getType().equals(NodeType.ACTION)){
    ViewableAtomic atomic = new ActionAtomic(node.getName(), step);
    atomics.add(atomic);
    map.put(node, atomic);
  }
  else if(node.getType().equals(NodeType.DECISION)){
    ViewableAtomic atomic = new DecisionAtomic(node.getName(), step);
    atomics.add(atomic);
    map.put(node, atomic);
  }
```

Listing 2: The external transition function for the decision node

```
public void deltext(double e,message x)
{
  Continue(e);

  if (phaseIs("passive"))
    for (int i=0; i< x.getLength();i++)
      for (String inPort : inPorts)
        if (messageOnPort(x,inPort,i))
        {
          job = x.getValOnPort(inPort,i);
          input = inPort;
          holdIn("executing",processing_time);
        }
}
```

5 INTERACTING WITH REACTIVE COMPUTATIONAL-PHYSICAL SYSTEM

One of the major challenges in modeling CPS is the integration of the notion of time with the current frameworks and tools (Derler, Lee, and Vincentelli 2012). ALRT-DEVS (Sarjoughian and Gholami 2015) uses the notion of time windows to recognize the uncontrollable factors in which imperfect physical environment operates. In the following work (Sarjoughian, Gholami, and Jackson 2013), the focus is on the actuating actions within the context of a CPS action. The experiment was conducted to closely investigate the turnaround time of the performed action on the physical entity which is a phidget with four relays. The experiment has been conducted within multiple settings to investigate the property in the real-time. We note that this taxonomy of modeling is distinct from the previous one since it focuses on dissecting the real-time properties during the run-time of the simulation with real-time constraints. That is, TIs are introduced at the action-level of the modeling and enforced thereafter by the extended simulator where the synchronization between the cyber and physical actions takes place. We consider it to be a cyber-physical modeling and simulation for CPS which can be suitable for simulation with hardware-in-the-loop. It is not restricted to computational aspect of CPS. The time constraints are strictly enforced on the actuating and sensing actions. For example, an actuating action has to take place within a certain time window, otherwise it is considered to be invalid. Missing the execution of actuating actions, and also the sensing ones, may result in loss of

the fidelity of the model depending on the critical situation of the system. Some systems may have less or more tolerance than the others based on the domain and potentially other variables in which the time can play a critical role. The important aspect from a modeling perspective is that such properties are rigorously accounted for in a formal specification and well-formed models.

Since we look closely into actions, this taxonomy applies to the CPS actions, that is, the actuating and sensing actions where interactions between the computational and physical worlds take place. The time windows are enforced on these actions given some time restrictions (i.e., time invariants are specified for these actions). We note that a connection can be established with the taxonomy discussed in the previous section (see 4.2) to enable the actuating and sensing actions to take place in real-time. For example, the actuating actions can be employed for the intersection to control the relay of the traffic flow or signals by sending an actuating actions. Also the sensing actions can take place by detecting the moving vehicles toward the intersection in order to perform the necessary computations and send the corresponding actuating actions thereafter. Both type of actions can only take place under hard real-time constraints.

6 VERIFICATION OF THE CPS ACTIVITIES MODELS

Verification of CPS activities is achieved to some degree by significantly bounding the state space that defines a model's dynamics. A key goal is to sacrifice, as little as possible, both the rigor and expressiveness of the model as measured. Although this places restrictions on the degree to which a model can be simulated, this leads to achieving formal verification through model checking (Gholami and Sarjoughian 2017).

In such a simulation-based verification approach, the constructed model can be helpful for ensuring various temporal properties particularly considering that cyber-physical systems are non-deterministic and stochastic. The goal is to use modeling of activities given a set of a-priori defined atomic models behaving as expected. It also enables the use of other verification techniques. Many scenarios are drawn to verify the behavior of the activity after being interpreted by the DEVS simulator. For example, in the smart intersection model (see Section 4.2), we can check if the model corresponding to decision node reports a crash when two cars are injected approaching from the same direction. This means that model is not behaving correctly according the problem specification. We also can check if the car gets directed to the correct destination after passing through the intersection. We can also check if the crash gets reported by simply injecting two cars into the intersection at the same time. These scenarios are checked, for example, by thorough disciplined experimentation. The DEVS-Suite simulator uses and expands the notion of experimental frame (Rozenblit 1991) where defined temporal properties can be verified.

6.1 Reasoning about temporal behavior

We aim to build an integrative environment to allow the development of activities as standalone behavioral models and also enable further reasoning capabilities about them. The reasoning can take different forms. However, it is quite common to be used in the form of model checking to facilitate various forms of knowledge representation. From this point of view, the knowledge is represented in the form of a DEVS model. On the other hand, the reasoning capabilities is also used to perform the model checking after imposing restrictions on the state, timing, ports, and external/internal events.

Thus, history of the causal effects is computed using an Answer Set Programming (ASP) tool (Bartholomew and Lee 2014) for a fixed integer m that represent the length of the history. The tool itself is aimed at the first-order logic; however the introduction of the history makes it possible to deduce about some temporal properties. The formulation of the ASP program takes into consideration bounded ranges and sends them out to the model as well as the transducer. An example of that could be the *phase* state variable of the intersection and some direction toward it. The rule that describes the causal effects considering the temporal aspect can

be formulated as $i : phase(pre-eastbound) = occupied \rightarrow i+t : phase(Intersection) = occupied$ where i is the timestamp and t is determined time advance for the phase *occupied* in the pre-eastbound atomic model. This rule can be examined within some defined finite m under the imposed restrictions on the state space and specifically the time advance. Other properties can be also checked in a similar manner. Outputs of the ASP program are generated and fed to the model and then checked against by the transducer for the verification. The following rules describe the effects approaching from the pre-eastbout (PE) and pre-northbound (PN) actions and their rules with respect to the intersection node (I):

$i : phase(PE) = occupied \rightarrow i+t : phase(I) = occupied$
$i : phase(PN) = occupied \rightarrow i+t : phase(I) = occupied$
$i : location(x) = PE \rightarrow i+t : location(x) = I$
$i : location(x) = PN \rightarrow i+t : location(x) = I$
$i : location(x_1) = I \wedge location(x_2) = I \wedge x_1 \neq x_2 \rightarrow i+t : phase(ReportCrash) = active$

Based on the previous rules, we can formulate questions about some basic facts such as:

$$\left[1 : phase(ReportCrash) = active \wedge \left(0 : \bigvee_{l} location(x) = n\right)\right] \rightarrow 0 : location(x) = I$$

The rules and the entailment question can be both represented in the language of F2LP (Lee and Palla 2009). A future work is considered on providing some action-level verification capabilities for behavioral models that are specified while recognizing the notion of action.

7 CONCLUSION

Substantial effort is required to bring definitions from the DEVS formalism and its variants along with their underlying simulators and model-checkers for analyzing and designing cyber-physical systems. We expect that the process can be advanced further by employing concepts from model-driven approaches where they have been proven useful in multiple occasions for DEVS modeling within different variants (Cetinkaya, Verbraeck, and Seck 2011, Moallemi and Wainer 2010). The promising capabilities of the employment of Model-Driven Engineering (MDE) concepts can be achieved with a proper conformance to the Model Driven Architecture (MDA) meta-layers and the DEVS formalism. Combination of MDA and DEVS stands to benefit the process of model development by accounting for some domain-specific knowledge added to the general-purpose DEVS model abstraction (Sarjoughian, Alshareef, and Lei 2015). In this work, our attempt can be viewed as a contribution to the effort required to concretize behavioral abstractions. Therefore, further effort is required to account for activity-based behavioral specification relative to the MDA modeling layers.

We have demonstrated how design of CPS can be approached with the help of activity-based modeling incorporated into system-theory and DEVS in particular. The use of the P-DEVS formalism and its underlying simulators as a platform for CPS is effective due to the inherent timing and modularity enabled by benefiting UML behavioral activity modeling and grounded with ALRT-DEVS modeling formalism. In contrast, many existing approaches do not account for the notion of time inherently and therefore leading to possible inconsistency and conflicts unless these issues are resolved at the implementation level. Furthermore, modularity can also serve as a means for the scalability at structural and behavioral of cyber-physical systems that have strong non-determinism and stochastic traits.

Another major advantage of this work is overcoming the issue of ending with a large fUML model that are hard to develop. This problem has been realized in the (OMG 2016). It is the reason for not creating the models of the execution model in activities. Instead, they have been specified in their corresponding Java code because significant activities quickly become too large to handle. A recent work has been proposed to address it (Bedini et al. 2017). Richer fUML models tends to become large which may lead to other issues with respect to scale. As discussed in the smart intersection example, the activity models have been made

significantly richer to handle complex time-critical dynamics in the smart intersection model with relatively fewer elements due to the DEVS formal grounding.

REFERENCES

Alshareef, A., and H. S. Sarjoughian. 2017. "DEVS specification for modeling and simulation of the UML activities". In *Proceedings of the Symposium on Model-Driven Approaches for Simulation Engineering*. Society for Computer Simulation International.

Alur, R. 1999. "Timed automata". In *International Conference on Computer Aided Verification*, pp. 8–22. Springer.

Alur, R. 2015. *Principles of cyber-physical systems*. MIT Press.

Bartholomew, M., and J. Lee. 2014. "Stable Models of Multi-Valued Formulas: Partial versus Total Functions.". In *KR*.

Bedini, F., A. Wichmann, R. Maschotta, and A. Zimmermann. 2017. "An fUML Extension Simplifying Executable UML Models Implemented For A C++ Execution Engine". In *Proceedings of the Symposium on Model-Driven Approaches for Simulation Engineering*. Society for Computer Simulation International.

Cetinkaya, D., A. Verbraeck, and M. D. Seck. 2011. "MDD4MS: a model driven development framework for modeling and simulation". In *Proceedings of the 2011 summer computer simulation conference*, pp. 113–121. Society for Modeling & Simulation International.

Chow, A. C. H., and B. P. Zeigler. 1994. "Parallel DEVS: A parallel, hierarchical, modular modeling formalism". In *Simulation Conference Proceedings, 1994. Winter*, pp. 716–722. IEEE.

Damodaran, S., and S. Mittal. 2017. "Modeling Cyber Effects in Cyber-Physical Systems with DEVS". In *Proceedings of the Symposium on Theory of Modeling and Simulation-DEVS Integrative M&S Symposium*. Society for Computer Simulation International.

Derler, P., E. A. Lee, and A. S. Vincentelli. 2012. "Modeling cyber–physical systems". *Proceedings of the IEEE* vol. 100 (1), pp. 13–28.

Eclipse Foundation 2016. "Model Development Tools (MDT)". http://www.eclipse.org/modeling/mdt/.

Gholami, S., and H. S. Sarjoughian. 2017. "Modeling and Verification of Network-on-Chip using Constrained-DEVS". In *Proceedings of the Symposium on Theory of Modeling and Simulation-DEVS Integrative M&S Symposium*. Society for Computer Simulation International.

Giese, H., and S. Burmester. 2003. "Real-time statechart semantics". *TechReport tr-ri-03-239, University of Paderborn*.

Hong, J. S., H.-S. Song, T. G. Kim, and K. H. Park. 1997. "A real-time discrete event system specification formalism for seamless real-time software development". *Discrete Event Dynamic Systems* vol. 7 (4), pp. 355–375.

Hwang, M. H., and B. P. Zeigler. 2006. "A reachable graph of finite and deterministic DEVS networks". *SIMULATION SERIES* vol. 38 (1), pp. 48.

Lee, J., and R. Palla. 2009. "System F2LP–computing answer sets of first-order formulas". In *International Conference on Logic Programming and Nonmonotonic Reasoning*, pp. 515–521. Springer.

Moallemi, M., and G. Wainer. 2010. "Designing an interface for real-time and embedded DEVS". In *Proceedings of the 2010 Spring Simulation Multiconference*, pp. 137. Society for Computer Simulation International.

Mosterman, P. J., and J. Zander. 2016. "Cyber-physical systems challenges: a needs analysis for collaborating embedded software systems". *Software & Systems Modeling* vol. 15 (1), pp. 5–16.

NIST 2016. "Framework for Cyber-Physical Systems". https://www.nist.gov/el/cyber-physical-systems.

OMG 2012. "Unified Modeling Language version 2.5".

OMG 2016. "Semantics of a Foundational Subset for Executable UML Models (fUML) version 1.2.1".

Risco-Martín, J. L., S. Mittal, J. C. Fabero, P. Malagón, and J. L. Ayala. 2016. "Real-time hardware/software co-design using DEVS-based transparent M&S framework". In *Proceedings of the Summer Computer Simulation Conference*, pp. 45. Society for Computer Simulation International.

Rozenblit, J. W. 1991. "Experimental frame specification methodology for hierarchical simulation modeling". *International Journal Of General System* vol. 19 (3), pp. 317–336.

Sarjoughian, H. S., A. Alshareef, and Y. Lei. 2015. "Behavioral DEVS metamodeling". In *Proceedings of the 2015 Winter Simulation Conference*, pp. 2788–2799. IEEE Press.

Sarjoughian, H. S., and S. Gholami. 2015. "Action-level real-time DEVS modeling and simulation". *Simulation* vol. 91 (10), pp. 869–887.

Sarjoughian, H. S., S. Gholami, and T. Jackson. 2013. "Interacting real-time simulation models and reactive computational-physical systems". In *Proceedings of the 2013 Winter Simulation Conference*, pp. 1120–1131. IEEE Press.

Sarjoughian, H. S., and A. M. Markid. 2012. "EMF-DEVS modeling". In *Proceedings of the 2012 Symposium on Theory of Modeling and Simulation-DEVS Integrative M&S Symposium*, pp. 19. Society for Computer Simulation International.

Stankovic, J. A. 1988. "Misconceptions about real-time computing: A serious problem for next-generation systems". *Computer* vol. 21 (10), pp. 10–19.

Sztipanovits, J., X. Koutsoukos, G. Karsai, N. Kottenstette, P. Antsaklis, V. Gupta, B. Goodwine, J. Baras, and S. Wang. 2011. "Toward a science of cyber–physical system integration". *Proceedings of the IEEE* vol. 100 (1), pp. 29–44.

Wang, Q., and F. E. Cellier. 1990. "Time windows: An approach to automated abstraction of continuous-time models into discrete-event models". In *AI, Simulation and Planning in High Autonomy Systems, 1990., Proceedings.*, pp. 204–211. IEEE.

Zeigler, B. P., H. Praehofer, and T. G. Kim. 2000. *Theory of modeling and simulation: integrating discrete event and continuous complex dynamic systems*. Academic press.

AUTHOR BIOGRAPHIES

ABDURRAHMAN ALSHAREEF is a Lecturer in the College of Computer and Information Sciences at King Saud University, Riyadh, SA. He is currently pursuing his Ph.D. in Computer Science program in the School of Computing, Informatics and Decision Systems Engineering at Arizona State University, Tempe, AZ. His research interests lie in the discrete event system models development and simulation, model-driven engineering, software architecture and metamodeling. He can be contacted at alshareef@asu.edu.

HESSAM S. SARJOUGHIAN is an Associate Professor of Computer Science and Computer Engineering in the School of Computing, Informatics, and Decision Systems Engineering (CIDSE) at Arizona State University (ASU), Tempe, AZ, and co-director of the Arizona Center for Integrative Modeling & Simulation (ACIMS). His research interests include model theory, poly-formalism modeling, collaborative modeling, simulation for complexity science, and M&S frameworks/tools. He is the director of the ASU Online Masters of Engineering in Modeling & Simulation program. He can be contacted at hss@asu.edu.