




An Semi-formal Co-verification Approach for High-Assurance CPS

Yu Zhang^{1,2}(✉) , Mengxing Huang^{1,3}, and Wenlong Feng^{1,3}

¹ State Key Laboratory of Marine Resource Utilization in South China Sea,
Hainan University, Haikou 570228, China

² School of Computer Science and Cyberspace Security,
Hainan University, Haikou 570228, China
yuzhang_nwpu@163.com

³ School of Information and Communication Engineering,
Hainan University, Haikou 570228, China

Abstract. Cyber-Physical Systems (CPS) are often mission-critical, therefore, they must be high-assurance. High-assurance CPS require extensive formal verification. Formal verification techniques can discover subtle design errors where simulation fails. However, due to the state explosion problem, formal techniques usually cannot handle large designs. This paper introduces a semi-formal verification methodology in which formal co-verification and co-simulation are tightly coupled. We propose an online-capture offline-replay approach to improve the usefulness for formal verification. We analyze these simulation traces, find some critical states and assisted with formal verification under these circumstances. The experiment results show that our approach has major potential in verifying system level properties of complex CPS, therefore improving the high-assurance of CPS.

Keywords: Cyber-Physical Systems · Semi-formal · Co-simulation

1 Introduction

Cyber-Physical Systems(CPS) enable objects to be sensed and controlled remotely across network infrastructure, thereby creating opportunities for more direct integration of the physical world into the cyber world [2,6]. Depending on the applications, their failures could have dire consequences. Therefore, verification of correctness properties is a key step in developing high assurance CPS. Formal verification attracts great attention because of their ability to find subtle design errors where simulation fails. However, due to the state explosion problem, formal verification usually cannot handle complex system.

In this paper, we present a semi-formal cyber/physical co-verification method using co-simulation and formal co-verification to ensure the high-assurance of CPS. We analyze these simulation traces, find some critical states and assisted

with formal verification under these circumstances. This online-capture offline-replay approach combines the benefits of going deeper and explore exhaustively the state space of the system. The semi-formal verification approach can be used to overcome the drawbacks of both co-simulation and formal co-verification.

The major contributions of our approach include the semi-formal co-verification for CPS that unifies control and embedded software, and seamless integration of co-simulation and co-verification into CPS development. It has great potentials in building high-assurance CPS by enabling effective co-verification.

The rest of this paper is organized as follows. Section 2 presents the semi-formal co-verification framework. Section 3 introduce the semi-formal co-Verification execution. Section 4 presents the implementation of this approach. Section 5 elaborates on case studies we have conducted and discusses the experimental results. Section 6 reviews the related research. Section 7 concludes this paper and discusses future work.

2 Semi-formal Co-verification Framework for CPS

The correctness of the safety-critical scenarios plays a very important role in developing high-assurance CPS. On the one hand, due to the increasing complexity of the CPS system, formal verification has become more and more difficult. On the other hand, the simulation has been widely used in engineering practice. But simulation test can only observation incomplete system behavior in the case of a certain input operation system, and unable to test of all possible input and scenarios. Simulation test method uses concrete value to performance test scenarios and expected results, while bounded model checking uses symbolic system model to describe the expected and unexpected system behavior. Bounded model checking depends on the strict mathematical process. Simulation method is complementary to formal verification method which could be used to found unsafe events whether and under what conditions will happen in specific dynamic operation. These information can be used to improve design and the requirements. So integrated simulation and formal verification in the CPS system verification is essential, especially for complex CPS.

Therefore, Semi-Formal co-verification method which combining co-simulation and formal co-verification is proposed in this paper. On the basis of our previous work [11, 12], this semi-formal co-verification approach combine simulation execution and k steps bounded model checking to verify this key scenes. As illustrated in Fig. 1, drive system starting from the initial state into a critical key state through co-simulation, and then combined with bounded model checking from the critical key state in k steps.

In the actual system, engineers will pay more attention to some key scenes or some time instead of the whole process. Such as the key point for satellite launch is whether the satellite can enter the normal orbit. If pay attention to the behavior of the system at this stage, we can first generate the simulation trace in the scenario through co-simulation, and then perform formal verification to the key scenario.

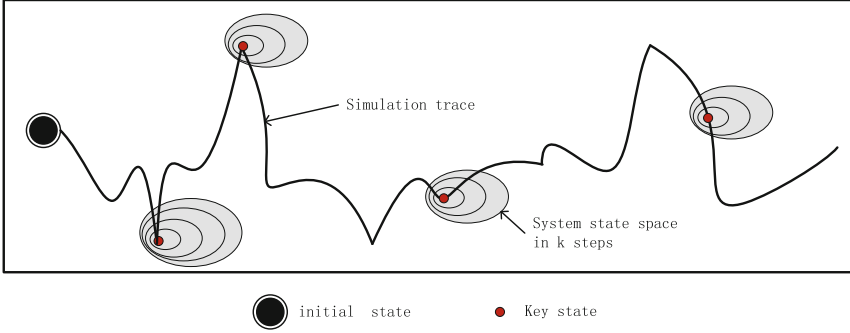


Fig. 1. Semi-formal co-verification for CPS.

3 Semi-formal Co-verification Execution

In our previous work [11,12], we define three types of primitive components in the co-verification: cyber component, physical component and cyber/physical interface component model. Cyber model is essentially a discrete event model, and its operational semantics refers to the execution sequence with a time stamp; Physical model is essentially a continuous time model, the model is formulated for differential equation. For integration of discrete event model and continuous time model, its execution sequence in the operational semantics is interaction protocols between the two heterogeneous models.

Below we characterize the dynamic of the main components in detail and discuss how their integration is handled.

Definition 1. A CPS model is denoted as a tuple $S = (S_{cps}, HA_{cps})$, where $S_{cps} = \bigcup_{k=1}^n S_{cyber} + \bigcup_{k=1}^k S_{interface} + \bigcup_{k=1}^m S_{physical}$ is static structure of CPS model, $HA_{cps} = TA_1 || TA_2 || \dots || TA_n || HA_1 || HA_2 || \dots || HA_m$ is a cartesian product of automata.

Definition 2. A state of CPS model is denoted as a tuple $S = (S_{cyber}, S_{interface}, S_{physical})$, where S_{cyber} is a set of cyber model, $S_{physical}$ is a set of physical model, $S_{interface}$ is a set of interface model,

Definition 3. Timing parameters of a control task is denoted as a tuple (t_k, t_i, t_o) , where $t_k = T * k$, T is the fixed sampling interval of the controller, k is the number of controller iterations, t_i and t_o represent the A/D and D/A conversion periodical instants, respectively. Due to preemption and blocking from other tasks in the OS, the actual start of the task may be delayed for some time L_s . This is called the sampling latency of the controller. After some computation time and possible further preemption from other tasks, the controller will actuate the control signal. The delay of the actuation is called the input-output latency, denoted L_{io} .

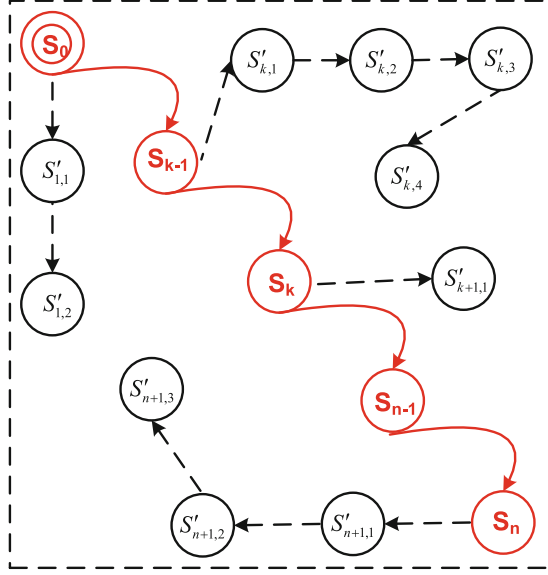


Fig. 2. Semi-formal co-verification for CPS.

The transaction condition of CPS is denoted as $r = \varphi \cup t$, where φ is a set of events, t is a set of clock. r can be expressed either event trigger or time trigger. A trace $S_0, S_1, S_2, \dots, S_{k-1}, S_k, \dots, S_n$ can be denoted as $\pi = S_0 \xrightarrow{r_0} S_1 \xrightarrow{r_1} S_2 \dots \xrightarrow{r_{k-2}} S_{k-1} \xrightarrow{r_{k-1}} S_k \dots \xrightarrow{r_{n-1}} S_n$. From the view of the CPS system, the CPS model is consisted of a series of discrete states, and each discrete state itself may be a continuous time model.

As shown in Fig. 2, we analyze a simulation trace $\pi = S_0, S_1, S_2, \dots, S_{k-1}, S_k, \dots, S_n$. For example, take S_{k-1} as a key state. Combined with the r_{k-1} condition, we conduct four steps bounded model checking from start state S_{k-1} .

We will construct semi-formal execution by Algorithm 1. i is loop iteration, S_n is a simulation trace in simulation trace set ST , num is the amount of states which need to be verified. Firstly, get the i -th state S_i from simulation trace S_n by the function $Get_State(S_n, i)$. Secondly, initial the state $S_{i,0}$ by the function $Reset_State(S_i)$. $S_{i,0}$ is the initial state in the model checking. And then perform bounded model checking within deadline until the timeout for the next state which is needed to be verified.

4 Semi-formal Co-verification Environment

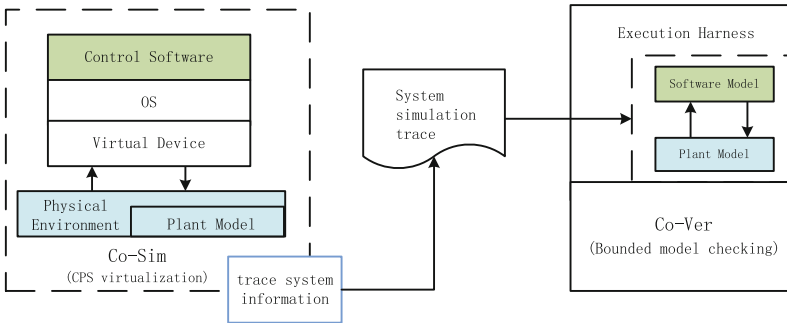
As a proof of concept, the Semi-Formal co-verification environments are developed. Below introduces these environments. As shown in Fig. 3, the co-verification algorithm has been realized in the environment which integrates corresponding prototype tools: formal collaborative verification tools Co-VerCPS

Algorithm 1. SEMI-FORMAL_EXECUTION

```

1 Input: Simulation trace  $S_n \in ST$ , Deadline  $Time\_Bound$ ;
2 Output: Verification result  $Result$ 
3  $i \leftarrow 0$ ; //loop iteration;
4  $S_n \leftarrow Load\_Simulation\_Trace(ST)$ ;
5  $num \leftarrow number\_of\_state\_in\_checking\_states$ ;
6 forall the  $ij=num$  do
7    $S_i \leftarrow Get\_State(S_n, i)$ ;
8    $S_{i,0} \leftarrow Reset\_State(S_i)$ ;
9   forall the  $tjnum$  do
10     $S_{i,k+1} \leftarrow Compute\_Next\_State(S_{i,k})$ ;
11     $Result \leftarrow Check\_State(S_{i,k+1})$ ;
12     $k \leftarrow k + 1$ ;

```

**Fig. 3.** Implementation of Semi-formal Co-verification.

and collaborative simulation tools Co-Sim [13]. Co-Ver is primarily for abstract model of C program (Labeled Pushdown System) and abstract model of a physical system (hybrid automata). Co-Sim focuses on the application, the physical simulation model (Simulink model), and virtual device platform. These models are widely used in related fields.

First, execute the test cases and record system information in the Co-Sim. Second, export simulation data into system simulation trace. Finally, execute bounded model checking and output verification results in the Co-Ver. The Execution Harness is a set which consists of system model and simulation data set (these data has been configured). Execution Harness makes a system run under different conditions, and monitor its behavior and output. It has three main parts: execution engine Co-Ver, system model and simulation state sequence set.

5 Evaluation

To evaluate the proposed co-verification approach, we have applied the approach to real-world control systems. In all experiments, we want to check whether the system meet the constrains or not with slight perturbations in the inputs and outputs to the system. All experiments were performed on a machine with 3.40 GHz Intel(R) Core(TM) and 16 G memory. As shown in Fig. 4, a co-simulation environment is developed for TableSat. A X86 processor model is utilized to emulate the Athena II SBC in QEMU. The embedded control program is written in C language. The plant components are modeled mathematically according to respective physical characteristics in Matlab/Simulink.

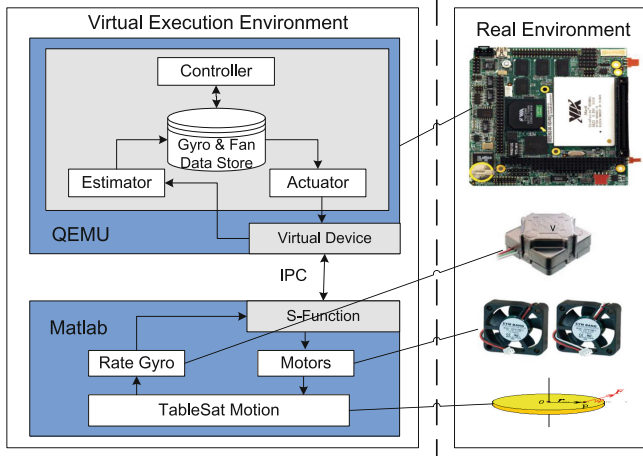


Fig. 4. Co-simulation environment for TableSat.

In these experiments, we select two simulation test scenarios to verify system constrains. The first experiment is in the case of a single control target rotary velocity, the second one is in the case of the multi control target rotary velocity. Single control target rotary velocity means the expected target rotary velocity value remains the same during the simulation. In each experiment, we select a set of key state to support formal verification and certified the validity of semi-formal verification.

5.1 Single Control Target Rotary Velocity

In the simulation, the initial angular velocity of TableSat is concrete value. In a sense, the start instantaneous response of system is a key scenario. First, we constructed the program and plant model based on the cyber/physical interface. Then we formulated the constrains of the system with LTL, and conducted bounded model checking. We chose the fixed execution time type of

cyber/physical interface in this experiment. The following initial set of parameters is used in the experiment: the sampling interval ($T=2$ s), the A/D conversion instant ($t_i=0.4$ s), and the D/A conversion instant ($t_o=1.6$ s). We specified a target rotary velocity ($TargetVelocity=30^\circ/s$) as TableSat input. We set the initial value of angular velocity ($[0, 40]$) as a symbolic variable. Figure 8 summarised the results. The verification result shows that the TableSat satisfies the last six LTL constrains. For the first LTL property, bounded model checker revealed a simple bug of the controller that: If the initial value of angular velocity is $39.960621^\circ/s$, then the rotary velocity reaches ($63.649414^\circ/s$) above a threshold ($VelocityUpBound=60^\circ/s$) at 2.324336 s. The verification run took 7015.26 s. The running time largely depends on the backend SMT solver (Fig. 5).

5.2 Multi Control Target Rotary Velocity

In the period of $[0s, 40s)$, the control target rotary velocity is $30^\circ/s$; in the period of $[40s, 80s)$, the control target rotary velocity is $50^\circ/s$; in the period of $[80s, 120s)$, the control target rotary velocity is $70^\circ/s$; in the period of $[120s, 160s)$, the control target rotary velocity is $20^\circ/s$; in the period of $[160s, 200s)$, the control target rotary velocity is $50^\circ/s$; in the period of $[200s, 240s)$, the control target rotary velocity is $30^\circ/s$. The following initial set of parameters is used in the experiment: the sampling interval ($T=0.4$ s), the A/D conversion instant ($t_i^k \in [0, 0.1]$), and the D/A conversion instant ($t_o^k \in [0, 0.1]$). We record measurements in every 0.01 s. The simulation result are shown in Fig. 6.

As shown in Fig. 6, the simulation result indicates that system satisfy the bounded input bounded output stability. In the simulation, the delay is generated by the random function on an interval and is actually a constant during the simulation. So the simulation test is not complete since it only observe limited system behavior. The initial state in the system is zero, there is no guaranteed that the controller can correct control plant when the sampling jitter j_s and input-output jitter J_{io} under uncertainty condition. Combined with the simulation data, we will verify the seven LTL constrains used in the model checking. We select eight key states on a simulation trace. The Timeout is set to 30000 s. The experimental results showed that each key state can perform 3 step bounded model checking. The followings are two scenarios to illustrate our approach (Table 1).

Scenario i. Recorded the system information when the system response curve across the steady-state and reach a peak point moment for the first time. And then based on this state execute bounded model checking. According to the simulation data, at the moment $t=3.5$ s, the system response curve across the steady-state value and reach the peak point($\omega=36.7298^\circ/s$). Set j_s and J_{io} as symbolic variable. As shown in Fig. 7, the verification result shows that the TableSat satisfies all LTL constrains.

Scenario ii. Recorded the system information when the control target rotary velocity change from $20^\circ/s$ to $50^\circ/s$. And then based on this state execute bounded model checking. According to the simulation data, at the

Table 1. Design constraints for TableSat

No	LTL constraint	Result (within 4 s)
1	$\mathbf{G}(\text{Rotary.Velocity} \leq \text{VelocityUpBound})$: The controller never accelerates the TableSat over the rotary velocity limit VelocityUpBound	\perp
2	$\mathbf{G}(\text{Time} \geq \text{TimeBound}) \rightarrow (\text{Rotary.Velocity} \geq \text{VelocityDownBound})$: When running more than a threshold TimeBound , the controller will always accelerate the TableSat over the rotary velocity limit VelocityDownBound	\top^P
3	$\mathbf{G}((\text{Rotary.Velocity} < 0.4 * \text{TargetVelocity}) \rightarrow (\text{Actuator.FanVoltage} \equiv 12))$: When the rotary velocity below 0.4 times of its expected value TargetVelocity , the controller will set the fans to 12 V	\top^P
4	$\mathbf{G}(\text{Rotary.Velocity} > 1.5 * \text{TargetVelocity}) \rightarrow (\text{Actuator.FanVoltage} \equiv 0)$: When the rotary velocity exceeds 1.5 times of its expected value TargetVelocity , the controller will set the fans to 0V	\top^P
5	$\mathbf{G}(\text{Time} \geq \text{TimeBound}) \rightarrow \text{Rotary.Velocity} - \text{TargetVelocity} \leq \text{SteadyStateError}$: When running more than a threshold TimeBound , the rotary velocity must be stable at TargetVelocity within the error SteadyStateError	\top^P
6	$\mathbf{G}(\text{Rotary.Velocity} < T) \rightarrow \mathbf{F}(\text{Actuator.FanVoltage} = \text{full})$: after the Rotary velocity belows its bound, Actuator will set the motors to the full voltage	\top^P
7	$\mathbf{G}(\text{Rotary.Velocity} > T) \rightarrow \mathbf{F}(\text{Actuator.FanVoltage} = \text{fullNeg})$: after the Rotary velocity exceeds its bound, Actuator will set the motors to the full negative voltage	\top^P

NO.	K.	TIME(S).	RESULT.
1.	1.	14.615.	\perp .
	2.	N/A.	N/A.
	3.	N/A.	N/A.
	4.	N/A.	N/A.
2.	1.	14.615.	\top^P .
3.	2.	164.288.	\top^P .
4.	3.	713.443.	\top^P .
5.	4.	2396.97.	\top^P .

Fig. 5. Test results.

moment $t = 160$ s, $\omega = 25.06078^\circ/\text{s}$), the speed of fan is 1789 RPM. Set $j_s \in [0, 0.1]$ and $J_{io} \in [0, 0.1]$ as symbolic variable. As shown in Fig. 8, the verification result shows that the TableSat satisfies all LTL constrains.

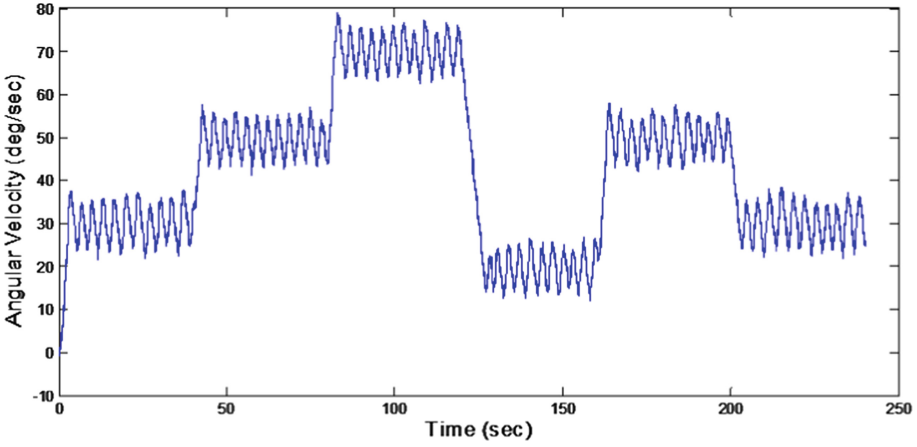


Fig. 6. Simulation Test results.

NO.	$k = 1.$		$k = 2.$		$k = 3.$	
	TIME (S)	RESULT	TIME (S)	RESULT	TIME (S)	RESULT
1.	271.57.	T ^p .	1746.37.	T ^p .	28422.	T ^p .
2.	271.57.	T ^p .	1746.37.	T ^p .	28422.	T ^p .
3.	271.57.	T ^p .	1746.37.	T ^p .	28422.	T ^p .
4.	271.57.	T ^p .	1746.37.	T ^p .	28422.	T ^p .
5.	271.57.	T ^p .	1746.37.	T ^p .	28422.	T ^p .
6.	271.57.	T ^p .	1746.37.	T ^p .	28422.	T ^p .
7.	271.57.	T ^p .	1746.37.	T ^p .	28422.	T ^p .

Fig. 7. Test results.

The above experimental results indicate that directly using formal verification to realize the CPS has become more and more difficult. In TableSat, we can only just check three system steps form the initial state. So formal verification and simulation should effectively complement each other. The semi-formal verification method is essential to establishing the correctness of a complete system, therefore improving the high-assurance of CPS.

NO.	$k = 1$		$k = 2$		$k = 3$	
	TIME (S)	RESULT	TIME (S)	RESULT	TIME (S)	RESULT
1	315.027	T ^p	1841.2	T ^p	27056	T ^p
2	315.027	T ^p	1841.2	T ^p	27056	T ^p
3	315.027	T ^p	1841.2	T ^p	27056	T ^p
4	315.027	T ^p	1841.2	T ^p	27056	T ^p
5	315.027	T ^p	1841.2	T ^p	27056	T ^p
6	315.027	T ^p	1841.2	T ^p	27056	T ^p
7	315.027	T ^p	1841.2	T ^p	27056	T ^p

Fig. 8. Test results.

6 Related Work

Many scholars have did much work and gained their research results on CPS verification [7–10].

Various formal verification methods have been proposed for Cyber-Physical Systems. Well-known tools for verifying such systems include HyTech and Uppaal. In [3] and [5] they propose an approach to formally analyzing such control software using model checking of UPPAAL. In [4], they propose a delta-complete algorithm for solving satisfiability of nonlinear SMT over real numbers. This approach has a key drawback: The focus has been on control logic design on high level with simplifying assumptions. Therefor they can't handle complex systems.

Due to the scalability of formal verification is not high, simulation is a low-cost and efficient method in detecting shallow bugs. The most closely related work is presented [1], a comprehensive co-simulation platform for CPS and examples showing the capabilities of the platform were presented. The simulation platform is built on Modelica and ns-2 tools.

7 Conclusions

An approach has been presented to semi-formal Cyber/Physical co-verification based on the integration of co-simulation and formal co-verification. We analyze these simulation traces, find some critical states and assisted with formal verification under these circumstances. This online-capture offline-replay approach combines the benefits of going deeper and expore exhaustively the state space of the system. The semi-formal verification approach can be used to overcome

the drawbacks of both co-simulation and formal co-verification. We have validated our approach by applying it to TableSat. It combines advantages of formal verification and simulation. The experiment results show that our approach has major potential in verifying system level properties of complex CPS, therefore improving the high-assurance of CPS.

Acknowledgments. This research received financial support from the Key R&D Project of Hainan province (Grant #: ZDYD2019020), the National Key R&D Program of China (Grant #:2018YFB1404401 and 2018YFB1404403), the National Natural Science Foundation of China (Grant #: 61662019 and 61862020), the Education Department of Hainan Province (Grant #: Hnky2019-22), the Higher Education Reform Key Project of Hainan province (Hnjg2017ZD-1) and Academician Workstation in Hainan Intelligent Healthcare Technologies.

References

1. Al-Hammouri, A.T.: A comprehensive co-simulation platform for cyber-physical systems. *Comput. Commun.* **36**(1), 8–19 (2012). <https://doi.org/10.1016/j.comcom.2012.01.003>
2. Chen, D., Chang, G., Sun, D., Li, J., Jia, J., Wang, X.: TRM-IoT: a trust management model based on fuzzy reputation for internet of things. *Comput. Sci. Inf. Syst.* **8**(4), 1207–1228 (2011)
3. Herrmann, P., Blech, J.O., Han, F., Schmidt, H.: A model-based toolchain to verify spatial behavior of cyber-physical systems. *Int. J. Web Serv. Res.* **13**(1), 40–52 (2016)
4. Kong, S., Solar-Lezama, A., Gao, S.: Delta-decision procedures for exists-forall problems over the reals. *CoRR abs/1807.08137* (2018). [arxiv:1807.08137](https://arxiv.org/abs/1807.08137)
5. Li-Jun, S., et al.: Statistical model checking of cyber-physical systems control software. *J. Softw.* **26**(2), 380–389 (2015)
6. Munir, A., Kansakar, P., Khan, S.U.: IFCIoT: Integrated fog cloud IoT: a novel architectural paradigm for the future internet of things. *IEEE Consum. Electron. Mag.* **6**(3), 74–82 (2017). <https://doi.org/10.1109/MCE.2017.2684981>
7. Wang, H., Maccaull, W.: An efficient explicit-time description method for timed model checking. vol. 14, pp. 77–91 (2009). <https://doi.org/10.4204/EPTCS.14.6>
8. Wang, X., Yang, L., Xie, X., Jin, J., Deen, M.: A cloud-edge computing framework for cyber-physical-social services. *IEEE Commun. Mag.* **55**, 80–85 (2017). <https://doi.org/10.1109/MCOM.2017.1700360>
9. Wassyngh, A., et al.: Can product-specific assurance case templates be used as medical device standards? *IEEE Des. Test* **32**, 1–11 (2015). <https://doi.org/10.1109/MDAT.2015.2462720>
10. Yang, L.T., et al.: A multi-order distributed hosvd with its incremental computing for big services in cyber-physical-social systems. *IEEE Trans. Big Data 1* (2018). <https://doi.org/10.1109/TBDATA.2018.2824303>
11. Zhang, Y., Dong, Y., Xie, F.: Bounded model checking of hybrid automata push-down system. In: *Quality Software (QSIC), 2014 14th International Conference on Quality Software*, pp. 190–195. IEEE (2014)

12. Zhang, Y., Huang, M., Wang, H., Feng, W., Cheng, J., Zhou, H.: A co-verification interface design for high-assurance cps. *Comput. Mater. Continua* **58**, 287–306 (2019). <https://doi.org/10.32604/cmc.2019.03736>
13. Zhang, Y., Xie, F., Dong, Y., Yang, G., Zhou, X.: High fidelity virtualization of cyber-physical systems. *Int. J. Model. Simul. Sci. Comput.* **04**(02), 1340005 (2013). <https://doi.org/10.1142/S1793962313400059>. <http://www.worldscientific.com/doi/abs/10.1142/S1793962313400059>