Combined Online Checking and Control Synthesis: A Study on a Vehicle Platoon Testbed

Jiawan Wang, Lei Bu(✉), Shaopeng Xing, Yuming Wu, and Xuandong Li

State Key Laboratory of Novel Software Technology, Nanjing University, Nanjing, China
bulei@nju.edu.cn

Abstract. Vehicle platoon systems are typical safety-critical cyber-physical systems (CPS), and are designed for safe and efficient transportation. However, vehicles' complex dynamics and uncertain runtime environment make it difficult to apply conventional offline model checking methods to ensure their safety. To address this challenge, we propose an online safety assurance framework for CPS, conducting combined online model checking and control synthesis in well-scheduled cycles. In each cycle, we conduct (1) a quick online formal verification on systems' coarse-grained hybrid automata (HA) models, as a fault prediction mechanism; (2) for potential risks, an accurate optimal control synthesis on systems' fine-grained HA models. Furthermore, we develop a robotic vehicle platoon testbed, and implement our framework on it. We conduct a series of evaluations, and experimental results show that the systems' safety and efficiency are significantly enhanced by our framework.

1 Introduction

Cyber-physical systems (CPS) [1] tightly integrate discrete computational processes and continuous physical components, exhibiting inherently hybrid and dynamic behaviors. Nowadays, CPS can be found in various safety-critical areas, such as automotive, aerospace, healthcare, and infrastructure. Robots, autonomous vehicles, implantable medical devices, and intelligent buildings are all typical CPS, where a failure may cause severe damage to human life and property. Thus, formal safety assurance is important to these systems.

However, it is challenging to ensure the safety of CPS at design time. Large uncertainties exist during CPS operations, and could potentially lead to failures. For one thing, CPS are often deployed in intrinsically unpredictable physical environment. For another, distributed sub-CPS could exchange data online by communication networks. Since these parameters are all unpredictable until runtime, conventional offline model checking [2] is infeasible for many CPS. Therefore, there is a need for online methods to handle uncertainties and prevent failures during CPS operations.

This work is supported in part by the Leading-edge Technology Program of Jiangsu Natural Science Foundation (No. BK20202001).

M. Huisman et al. (Eds.): FM 2021, LNCS 13047, pp. 752–762, 2021.
https://doi.org/10.1007/978-3-030-90870-6_43

To study the online safety of CPS, we set up an indoor robotic vehicle testbed, including sub-systems, such as wheeled robotic cars, wayside sensors, and a wayside control center, connected by wired and wireless communication networks. Then, we construct a vehicle platoon scenario on our testbed.

Automated vehicle platoon requires strings of vehicles to drive together with ideal inter-vehicle distances. It could increase road capacity and reduce fuel consumption, due to decreased inter-vehicle distances [3]. However, a smaller inter-vehicle distance not only leads to higher traffic and fuel efficiency, but also higher risks of collision; and safety is the prerequisite to achieve any potential efficiency benefits. How to maintain a small inter-vehicle distance with formal safety guarantees is key to vehicle platoon. Thus, after our testbed is constructed, we further analyze the main challenges in vehicle platoon into the following two points, and propose our solutions:

1. Vehicles' dynamics are complex, including both continuous and discrete ones. The composition of multiple vehicles further enlarges their state space. Besides, their received wayside data and runtime environment are highly uncertain (e.g., real-time road conditions and front vehicle's behaviors). All these together make the offline model checking of such system inapplicable. Therefore, instead of checking the system at design-time, we conduct online model checking for platooning vehicles: building online models for vehicles, and verifying safety properties for their time-bounded short-run behaviors.
2. Once a potential collision is reported after online checking, the next challenging thing is to synthesize suitable control commands for vehicles. For a vehicle with collision risks, braking with its maximum braking power may avoid collisions, but it usually causes low transport efficiency, un-smooth vehicle behavior, and even potential danger to the onboard passengers. Therefore, instead of braking immediately, we synthesize optimal control commands for risky vehicles in the form of acceleration profiles.

In sum, we propose an online safety assurance framework for CPS and apply it to platooning vehicles. Combined online model checking and control synthesis are conducted in well-scheduled cycles to ensure complete runtime safety. In each cycle, coarse-grained linear hybrid automata (HA) [4] models and fine-grained nonlinear HA models are built online for vehicles. To ensure vehicles' safety in the short-run future, time-bounded verification of the target safety property is conducted on coarse-grained models. For potential risks, we then synthesize safe and optimal control commands on vehicles' fine-grained models.

In this work, we deploy this framework on our testbed, applying a formal time-bounded HA reachability verification tool BACH [5] and an optimal HA control synthesis tool CDH [6] in it. Evaluations show significant enhancement in system safety and efficiency after the deployment of our framework.

2 Robotic Vehicle Testbed and Platoon Scenario

2.1 Robotic Vehicle Testbed

As shown in Fig. 1, instead of software-based vehicle simulators running in virtual driving environment, we construct a robotic vehicle testbed. It consists of a rounded rectangle magnetic track with three wayside ultra-wideband (UWB) anchors, four-wheeled CVTECH A8 robotic cars, and a laptop (Intel Core i7 2.20 GHz and 16 GB RAM) with a ZigBee module as a wayside control center.

Each robotic car is equipped with motors, magnetic sensors, a UWB tag for position measurement, a grating sensor for speed measurement, a ZigBee module for wireless communication, and a Samsung board for control. Cars' top speed is limited to 55 cm/s. Wayside UWB anchors receive pulses emitted by UWB tags on cars, and measure their distances with cars. Thus, three UWB anchors enable real-time locating for cars running in three-dimensional space.

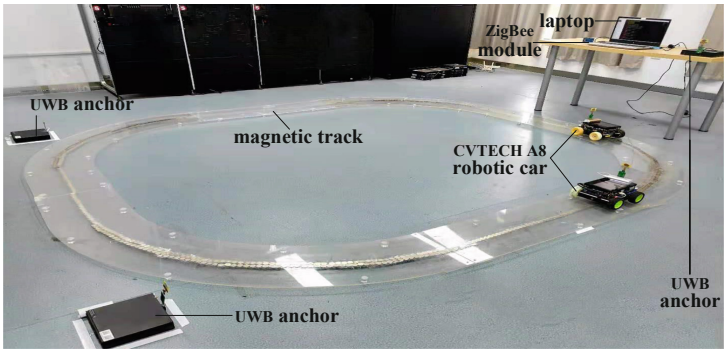


Fig. 1. A picture of the wheeled robotic vehicle testbed

2.2 Vehicle Platoon Scenario

In the vehicle platoon scenario on our testbed, two robotic cars are programmed to run along the magnetic track according to sensed magnetic information.

Vehicle Dynamics. The car ahead keeps cruising at 10 cm/s. The following car plans its movements by its current mode, emergency brake intervention speed (*vebi*), and movement authority (MA)[1]. *vebi* is calculated by function $f_{\mathrm{MA}}(x)$:

$$vebi = f_{\mathrm{MA}}(x) = \sqrt{(2 * a_{mbd} * (\mathrm{MA} - x))} \tag{1}$$

where x denotes its position and a_{mbd} denotes its maximum braking deceleration, which is 10 cm/s^2 here. The conditions and dynamics inside all three modes are given below:

[1] MA indicates the allowable travel distance for a vehicle by specifying its end-of-authority point [7]. Once a vehicle moves beyond its MA, a collision may occur.

- **Acceleration (AC)**: If its current speed is lower than ($vebi - 20$ cm/s), it enters acceleration mode, accelerating at 5 cm/s^2.
- **Emergency Braking (EB)**: If its current speed exceeds $vebi$, it enters EB mode, braking with a_{mbd}.
- **Cruise Control (CC)**: Otherwise, it stays in CC mode with random accelerations within $[-5, 5]$ cm/s^2.

Communication Topology. As shown in Fig. 2, there is wireless bi-directional communication between robotic cars and the wayside control center to exchange car speed and MA data, supported by onboard and wayside ZigBee modules. There is also wired communication between an UWB anchor and the wayside control center. Besides, all UWB anchors could receive pulses from each other.

Safety Property. In order to avoid collision in this highly interconnected platoon system with dynamic robotic cars, the following car's position should never reach its MA, formally written as "$x <$ MA". Once we can prove that this safety property holds, the safety of the platoon can be formally guaranteed.

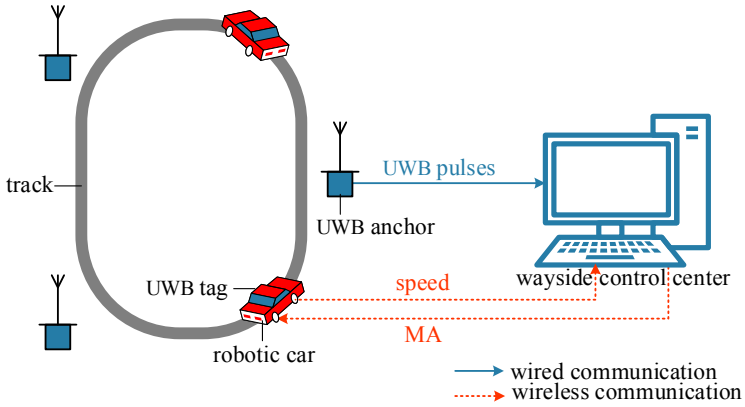


Fig. 2. Communication topology of the robotic vehicle platoon scenario

3 Periodically Online Safety Assurance Framework

To assure the safety of this scenario, we give our online safety assurance framework in Fig. 3. It performs online model checking and control synthesis for the running vehicle in scheduled cycles. In each cycle, we (1) concretize vehicle's online HA model by runtime parameters, (2) check whether any unsafe behavior will occur in the short-run future by time-bounded online reachability verification, (3) and once finding any potential risk, synthesize control commands online with safety guarantees, and feedback such commands to the vehicle.

3.1 Online Modeling

In this framework, we build online HA models periodically to handle runtime uncertainties. According to the vehicle dynamics introduced in Sect. 2.2 and monitored real-time parameters, we build a coarse-grained HA model for efficient online verification, and a fine-grained HA model for accurate control synthesis.

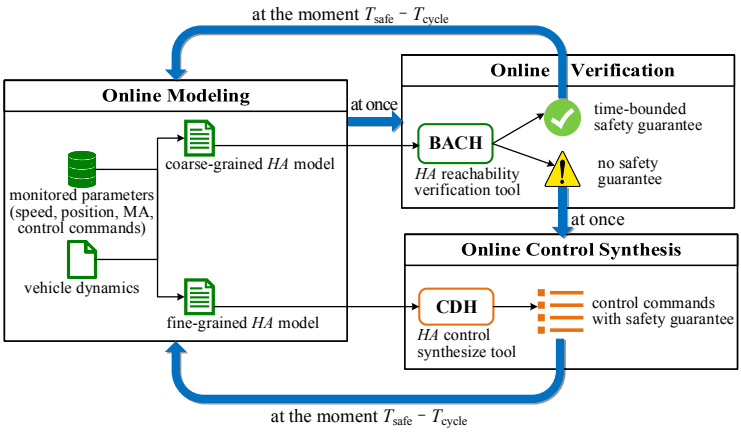


Fig. 3. Online and periodical safety assurance framework

For the coarse-grained model, as shown in Fig. 4(a), there is only one mode called RUN, modeling the vehicle's continuous dynamics. Variables 'x', 'v', and 't' are used to denote the vehicle's current position, speed, and execution time, and are initialized by vehicle's monitored real-time position, speed, and 0. There is also a timer variable, denoted as '*clock*', such that speed can be discretely updated every '$\triangle t$' time by jumping edges labeled AC, CC, and EB. Jumping conditions on edges are set by the vehicle's mode conditions described in Sect. 2.2, comparing its current speed 'v' with its emergency brake intervention speed computed by function $f_{\mathrm{MA}}(x)$ in Eq. 1.

For the fine-grained model in Fig. 4(b), there are additional variables '*vebi*' and 'a' to denote real-time emergency brake intervention speed and acceleration speed. Except the INIT mode to initialize variables, there are three modes, AC, CC, and EB, as described in Sect. 2.2, denoting that the vehicle's speed should be accelerated, approximately maintained, and decelerate respectively.

Comparing the coarse-grained model with the fine-grained model, there exist two major differences. (1) In the coarse-grained one, the computation of speed 'v' and emergency brake intervention speed '*vebi*' are conducted every '$\triangle t$' time by jumping edges; but conducted continuously in the fine-grained one as vehicle's original dynamics. So is the comparison between v and *vebi* for checking jumping conditions, which is also discretely conducted in the coarse-grained one, but continuously in the fine-grained one. For example, once its real-time speed reaches real-time *vebi* in the fine-grained model, it jumps to the mode EB at once. Besides, the dynamics are nonlinear in the fine-grained model, but, as a

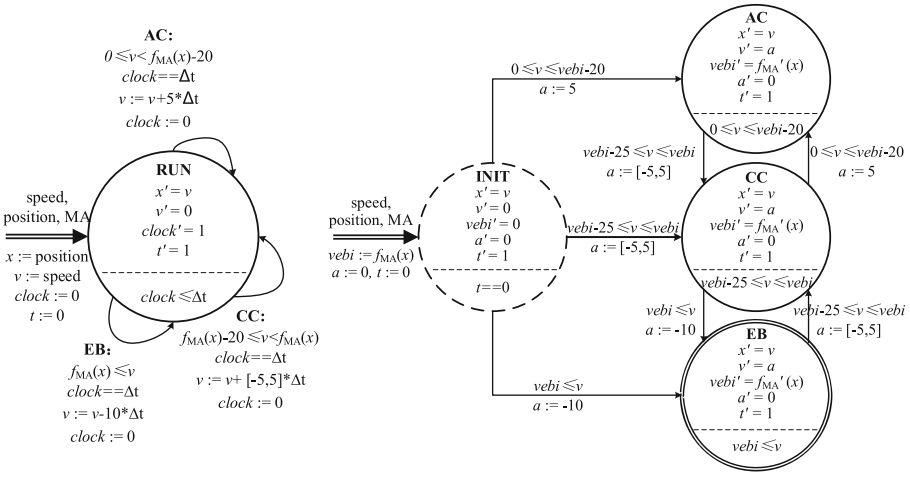
benefit of discretization abstraction, linear in the coarse-grained one. (2) The fine-grained model covers the vehicle's original behavior, but includes additional nondeterminism. The condition of CC mode is enlarged from $v \in [vebi-20, vebi]$ to $v \in [vebi-25, vebi]$, partial overlapping with the condition of AC mode, in order to provide additional space for optimal control synthesis.

3.2 Combined Online Verification and Control Synthesis

As shown in Fig. 3, in each cycle, we conduct time-bounded online verification for short-run fault prediction, and optimal control synthesis as a timely remedial measure for predicted faults.

We apply BACH, a time-bounded reachability checker for HA [5,8], to coarse-grained HA models, checking whether the target safety property, "$x <$ MA", always holds in the following bounded time. The tool BACH conducts path-oriented formal reachability verification by linear programming, and is chosen here for its efficiency in linear HA. If BACH returns safe, it means that in the following bounded time, this safety property always holds and no collision happens. If BACH returns unsafe, potential collisions might happen in the near future; thus, we conduct online control synthesis at once.

We apply CDH, a HA optimal control synthesis tool [6], to fine-grained HA models, synthesizing control commands in the form of acceleration profiles for the vehicle. CDH supports the generation of feasible and piecewise optimal control inputs for given control tasks in arbitrary nonlinear and nondeterministic HA. We apply CDH with a safety task that must be satisfied, which is "$x <$ MA" always hold before the end of the next verification and control cycle. Meanwhile, we can apply CDH with optional optimization tasks to realize other goals such as lower fuel cost and smoother vehicle behavior.



(a) coarse-grained HA model (b) fine-grained HA model

Fig. 4. Online vehicle models with different granularity

3.3 Assignment Scheduling

The assignment of verification and control cycles are well-scheduled in our framework, such that the runtime safety for vehicles' complete execution can be ensured. We give an illustrative example for our assignment scheduling in Fig. 5.

T_{safe} denotes the valid scope of the safety guarantee obtained in each cycle, i.e., the bounded time given to BACH. T_{cycle} denotes the maximum time required for one online model checking cycle, i.e., the maximum execution time for BACH and CDH. A cycle starts if there is only T_{cycle} time left before the expiration of last cycle's safety guarantee.

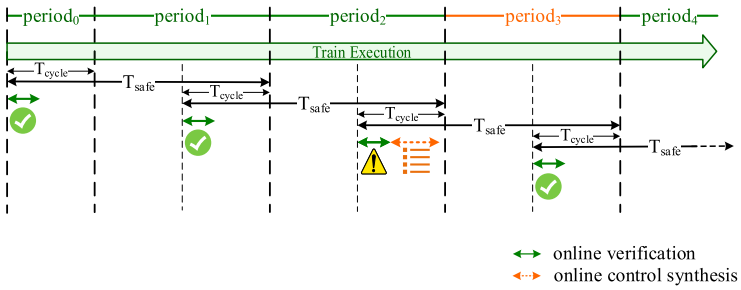


Fig. 5. An example of online model checking cycles

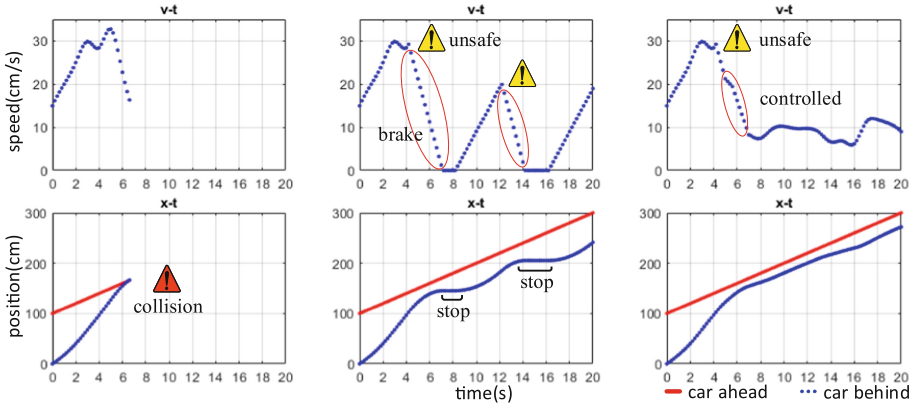
As shown in Fig. 5, we divide the vehicle's complete execution into multiple periods. Except for the first one denoted as period_0, which is a window period with length T_{cycle}, all other periods have the same length of $T_{\text{safe}} - T_{\text{cycle}}$. For period_{i+1}, its safety guarantee is obtained in period_i, either by the verification process or the control synthesis process. For example, the online verification process in period_1 returns safe, so we are assured that the whole period_2 is safe. While in period_2, the online verification process warns that the vehicle might be unsafe in the following T_{safe} scope. Thus, the control synthesis process is then performed, generating safe commands for the vehicle to execute during period_3.

4 Deployment and Evaluations

4.1 Framework Deployment

We deployed our framework on the vehicle platoon testbed, implementing periodical online model checking and control synthesis in the wayside control center. In our evaluation scene, when the car behind starts to move at an initial speed of 15 cm/s, the front one is 100 cm ahead of it.

In our experience, the control center typically accomplishes the online modeling and verification processes in 0.14–0.16 s by BACH, and the control synthesis process in 0.5–0.7 s by CDH in such scenarios. Considering the data transfer rate



(a) Scenario 1: no online checking and control synthesis

(b) Scenario 2: online checking only

(c) Scenario 3: online checking and control synthesis

Fig. 6. Vehicle's behavior in different scenarios w.r.t. our framework deployment (Color figure online)

and the message size in our ZigBee network, the delay of wireless communication is very short, typically 0.015–0.03 s. Thus, we set the maximum execution time for an online model checking cycle (i.e. T_{cycle}) as 1 s, and the valid scope for each cycle's model checking (i.e. T_{safe}) as 3 s. Besides, except for the requisite safety task, an optimization task to maximize the car's moving distance was also given to CDH.

4.2 Framework Evaluation

During the evaluation, we aim to study whether the car's safety and the transport efficiency are improved by our online checking and control synthesis framework. Our evaluation is designed with three phases, applying none of, part of, and the full of our framework respectively. We plot the speed-time (v–t) graph and position-time (x–t) graph for all three phases in Fig. 6(a)–(c). The behavior of the car ahead is given in red lines and the following car in blue dots.

Scenario 1. In this scenario, neither the online checking nor the control synthesis module of our framework was deployed. As shown in Fig. 6(a), a dangerous rear-end collision happened after 6.6 s when the car behind running at a speed higher than 15 cm/s. Although the car behind started to conduct emergent braking since $t = 5.2$ s, it was already too late to prevent the collision.

Scenario 2. In this scenario, the modeling and verification module of our framework was applied, but the control module was not. As shown in Fig. 6(b), BACH fired the collision alarm in the third cycle of checking (conducted during $t = [4, 4.2)$ s)[2]. The car behind braked immediately and stopped at $t = 7.2$ s. As shown in its x-t graph, the collision that happened in scenario 1 was avoided.

[2] As $\mathrm{T}_{cycle} = 1$, the length of one period is 2 s accordingly.

After BACH obtained a short-run safety guarantee in the fifth checking cycle during $[8.0, 8.2)$ s, the following car began to move again by its control logic. Unfortunately, during $[12.0, 12.2)$ s, BACH predicted another risk, and the following car braked again. After that, it did not move until $t = 16.2$ s. Two potential collisions were successfully avoided, due to well-scheduled checking cycles and BACH's efficient performance on linear HA. However, two emergency brakes are applied accordingly, and the car behind stopped completely on the track twice, which is neither comfortable for passengers nor efficient for transportation.

Scenario 3. Different from scenario 2, in scenario 3, control synthesis was conducted after BACH's warnings. As shown in Fig. 6(c), after BACH fired the alarm during $[4, 4.2)$ s, the car behind slowed down and CDH synthesized control commands for the car to execute in the next period $t = [5.0, 7.0)$ s. During the next period, the car's speed first decreased by 1.1 cm/s^2 for 0.8 s, then 1.3 cm/s^2 for 0.6 s, and finally 1.5 cm/s^2 for another 0.8 s. After that, it was reported safe by BACH and continued to move by its control logic.

We can see that, instead of braking urgently and stopping completely, with the help of efficient online control synthesis, the car behind never stopped. It kept moving on the track and also kept a safe distance from the car ahead. Thus, both the safety and efficiency of the running system are enhanced substantially.

5 Related Work

Online Verification and Control for CPS. Online reachability model checking has been recently proposed as a formal CPS fault prediction tool. Study [8] performs online reachability analysis for CPS by path-oriented bounded model checking (BMC). Several other works conduct reachable sets computation online. For linear systems, online reachability computation is conducted by flowpipe construction in [9], and by instantiating a pre-computed offline reachable set with a concrete recent state in [10]. For nonlinear systems, online reachability computation can be performed after decomposition of original system dynamics [11].

In terms of CPS control, conventional gradient-based methods are efficient [12,13], but require differentiable system dynamics. For non-differentiable CPS, sampling-based methods have achieved considerable success [14], but with no optimality guarantee. A robust model predictive control approach based on Monte Carlo simulation and rejection-sampling is proposed in [15], but with limited ability in complex control missions. An optimal control approach based on derivative-free optimization is proposed in [6], where complex control problems can be efficiently solved in a divide-and-conquer manner.

This work proposes a combined framework to ensure CPS runtime safety in scheduled cycles, combining both online verification and control synthesis. Periodically online verification works as a short-run fault prediction tool and control synthesis works as a remedial measure for predicted faults. Online verification solution in [8] and control synthesis solution in [6] are applied in this work.

Formal Verification and Control for Vehicle Platoon. Several works try to verify safety properties for platooning vehicles formally. Study [16] decomposes

platooning vehicles into small components and verifies safety properties by SAL toolkit tool. Studies [17,18] model vehicle dynamics by timed automata and verify safety properties by model checker UPPAAL [19]. It is hard for these offline methods to build accurate vehicle models at design time and conduct precise verification.

There are extensive works on vehicle platoon control, involving both lateral and longitudinal control. Since cars move along a single track in our testbed, only longitudinal control is considered in this work. In general, existing longitudinal control methods mainly include proportional integral derivative (PID) based ones [20], sliding mode control (SMC) based ones [21], model predictive control (MPC) based ones [22], and consensus control based ones [3].

Different from the works above, the combined safety assurance framework proposed in this work is an online one to handle runtime uncertainties. Accurate parameters, like vehicle speed, position, and MA, are updated precisely in each online checking cycle. However, it is worth mentioning that our vehicle platoon testbed is a simplified one. Many interesting elements, such as multi-lane vehicle platoon and vehicle-to-vehicle (V2V) communication, have not been considered yet in this work. How to construct a more complex vehicle platoon testbed with these elements and implement our framework on it, will be our future work.

6 Conclusion

In this work, we proposed an online safety assurance framework for CPS, conducting combined online model checking and control synthesis in cycles. These cycles are well-scheduled, such that we can ensure runtime safety for systems' complete execution. In each cycle, efficient reachability verification on coarse-grained models is conducted for short-run fault prediction, and optimal control synthesis on fine-grained models is conducted for potential faults. We built an indoor robotic vehicle platoon testbed and deployed our framework on it. Evaluations showed a significant enhancement in traffic safety and efficiency.

References

1. Lee, E.A., Seshia, S.A.: Introduction to Embedded Systems: A Cyber-Physical Systems Approach. MIT Press (2017)
2. Clarke, E.M., Grumberg, O., et al.: Model Checking. MIT Press (2018)
3. Di Bernardo, M., Salvi, A., Santini, S.: Distributed consensus strategy for platooning of vehicles in the presence of time-varying heterogeneous communication delays. IEEE Trans. Intell. Transp. Syst. **16**(1), 102–112 (2014)
4. Henzinger, T.A.: The theory of hybrid automata. In: Inan, M.K., Kurshan, R.P. (eds.) Verification of Digital and Hybrid Systems, pp. 265–292. Springer, Heidelberg (2000). https://doi.org/10.1007/978-3-642-59615-5_13d
5. Bu, L., Li, Y., Wang, L., Li, X.: Bach: bounded reachability checker for linear hybrid automata. In: FMCAD, pp. 1–4 (2008)
6. Xing, S., Wang, J., Bu, L., et al.: Approximate optimal hybrid control synthesis by classification-based derivative-free optimization. In: HSCC, pp. 1–11 (2021)

7. Pascoe, R.D., Eichorn, T.N.: What is communication-based train control? IEEE Veh. Technol. Mag. **4**(4), 16–21 (2009)
8. Bu, L., Wang, Q., Ren, X., et al.: Scenario-based online reachability validation for cps fault prediction. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **39**(10), 2081–2094 (2019)
9. Johnson, T.T., Bak, S., Caccamo, M., Sha, L.: Real-time reachability for verified simplex design. ACM Trans. Embed. Comput. Syst. **15**(2), 1–27 (2016)
10. Chen, X., Sankaranarayanan, S.: Model predictive real-time monitoring of linear systems. In: RTSS, pp. 297–306. IEEE (2017)
11. Chen, X., Sankaranarayanan, S.: Decomposed reachability analysis for nonlinear systems. In: RTSS, pp. 13–24. IEEE (2016)
12. Axelsson, H., Wardi, Y., Egerstedt, M., Verriest, E.: Gradient descent approach to optimal mode scheduling in hybrid dynamical systems. J. Optim. Theory Appl. **136**(2), 167–186 (2008)
13. Gonzalez, H., Vasudevan, R., Kamgarpour, M., et al.: A descent algorithm for the optimal control of constrained nonlinear switched dynamical systems. In: HSCC, pp. 51–60. ACM (2010)
14. Branicky, M.S., Curtiss, M.M., Levine, J.A., et al.: RRTs for nonlinear, discrete, and hybrid planning and control. In: CDC, pp. 657–663. IEEE (2003)
15. Farahani, S.S., Raman, V., Murray, R.M.: Robust model predictive control for signal temporal logic synthesis. IFAC **48**(27), 323–328 (2015)
16. El-Zaher, M., Contet, J.-M., Gruer, P., et al.: Compositional verification for reactive multi-agent systems applied to platoon non collision verification. Stud. Inform. Univ. **10**(3), 119–141 (2012)
17. Mallozzi, P., Sciancalepore, M., Pelliccione, P.: Formal verification of the on-the-fly vehicle platooning protocol. In: Crnkovic, I., Troubitsyna, E. (eds.) SERENE 2016. LNCS, vol. 9823, pp. 62–75. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45892-2_5
18. Peng, C., Bonsangue, M.M., Xu, Z.: Model checking longitudinal control in vehicle platoon systems. IEEE Access **7**, 112 015–112 (2019)
19. Behrmann, G., David, A., Larsen, K.G.: A tutorial on Uppaal. In: Bernardo, M., Corradini, F. (eds.) SFM-RT 2004. LNCS, vol. 3185, pp. 200–236. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30080-9_7
20. Knights, V.A., Gacovski, Z., Deskovski, S., Petrovska, O.: Guidance and control system for platoon of autonomous mobile robots. J. Electr. Eng. **6**, 281–288 (2018)
21. Xiao, L., Gao, F.: Practical string stability of platoon of adaptive cruise control vehicles. IEEE Trans. Intell. Transp. Syst. **12**(4), 1184–1194 (2011)
22. Graffione, S., Bersani, C., Sacile, R., Zero, E.: Model predictive control of a vehicle platoon. In: SoSE, pp. 513–518. IEEE (2020)