A Runtime Safety Analysis Concept for Open Adaptive Systems

Sohag Kabir[1], Ioannis Sorokos[1], Koorosh Aslansefat[1], Yiannis Papadopoulos[1], Youcef Gheraibia[1], Jan Reich[2], Merve Saimler[3], and Ran Wei[4]

[1] Department of Computer Science and Technology, University of Hull, Hull, UK
{s.kabir, k.aslansefat-2018, i.sorokos, y.i.papadopoulos, y.gheraibia}@hull.ac.uk
[2] Fraunhofer Institute for Experimental Software Engineering (IESE), Kaiserslautern, Germany
jan.reich@iese.fraunhofer.de
[3] AVL/TR, Istanbul, Turkey
merve.saimler@avl.com
[4] Department of Computer Science, University of York, York, UK
ran.wei@york.ac.uk

Abstract. In the automotive industry, modern cyber-physical systems feature cooperation and autonomy. Such systems share information to enable collaborative functions, allowing dynamic component integration and architecture reconfiguration. Given the safety-critical nature of the applications involved, an approach for addressing safety in the context of reconfiguration impacting functional and non-functional properties at runtime is needed. In this paper, we introduce a concept for runtime safety analysis and decision input for open adaptive systems. We combine static safety analysis and evidence collected during operation to analyse, reason and provide online recommendations to minimize deviation from a systems safe states. We illustrate our concept via an abstract vehicle platooning system use case.

Keywords: platooning · bayesian networks · model-based dependability analysis · runtime assurance

1 Introduction

Autonomous driving has gained significant financial and public interest in recent years. The idea of reducing or even removing the human control factor from driving is inherently quite exciting and promises many direct benefits to drivers, such as increased safety, comfort, improved fuel efficiency and flexible parking in cities [7]. With regards to financial potential, a widely cited market forecast for the UK's government's Centre for Connected and Autonomous Vehicles (CAVs) estimates the market for CAVs to be worth "£28bn in 2035, capturing 3% of the £907bn global market" [4, pg. 3]. The topic also presents a significant research challenge, incorporating multi-disciplinary issues from domains such as

artificial intelligence, cyber-security, sensor fusion, safety and more. A defining characteristic of autonomous driving are the highly dynamic and rapidly changing conditions experienced by CAVs during operation. CAVs must continuously adapt to varying road layouts, infrastructure, neighbouring vehicle composition and driving behaviors. These considerations compound existing dynamics experienced in non-autonomous vehicles such as variable weather conditions and road conditions. In particular, the issue of safety presents a significant concern. Traditional safety assurance is defined as activities performed during development to support the overall claim that the system will be safe to operate. These activities are typically bound to the development, not the runtime phase of the system lifecycle. In safety standards like ISO26262[10] and IEC 61508 [9], safety analysis activities are expected to only be performed during design and not during operation. However, the dynamism that CAVs feature, renders traditional, exhaustive approaches of safety assurance intractable against the potentially infinite combinations of factors and scenarios to consider. To reap the full potential offered by CAVs, assurance must adapt to their open and collaborative nature. Previous work in [14] investigated the use of their proposed Dynamic Safety Contracts (DSCs) and earlier Conditional Safety Certificates (ConSerts) [19] to address the issue of safety assurance for CAVs. The authors describe how evidence collected at runtime from various sources can be processed by predefined DCSs and ConSerts to provide runtime safety guarantees, enabling collaborative services to be negotiated dynamically. However, we note some limitations with regards to this previous work; both DSCs and ConSerts rely on the use of binary conditional variables to evaluate guarantees.

In this paper, we expand upon previous contract-oriented approaches in the following ways:

- We introduce the use of Bayesian Networks (BNs) as an alternate inference mechanism for probabilistic reasoning of guarantees. BNs provide means of expressing uncertainty and accounting for uncertainty in the assurance process.
- Beyond inference, our expanded framework also proposes recommended actions to be applied to minimize system risk during operation; these actions are predefined and linked to a state machine. This element allows the system to address partially unanticipated scenarios by re-evaluating previous assumptions about the state of the system and responding appropriately.

To illustrate our methodology, we apply it towards an abstract vehicle platooning system use case. Notably, we believe the approach is flexible and could be transferred to other domains. The use case discussed is largely based on the ongoing research contributed by AVL/TR[1] and earlier research by Fraunhofer IESE[2], partners of the Dependability Engineering Innovation for cyber-physical Systems (DEIS) research project[3]. The remainder of the paper is structured as

[1] `https://www.avl.com/-/avl-turkey`
[2] `https://www.iese.fraunhofer.de/en.html`
[3] `http://www.deis-project.eu/`

follows; in section 2, an overview of previous research and associated literature is reviewed. In section 3, our proposed framework is presented in detail. Section 4 discusses the application of the framework on the vehicle platoon case study mentioned earlier.

2 Brief background and literature

2.1 Runtime Assurance for Vehicle Platooning

Integration of automated control in vehicles is not a recently proposed idea; for instance, the PATH project[4], as early as 1986, begun pursuing the topic of Intelligent Transportation Systems (ITS). As vehicles are safety-critical systems, providing equal or even more robust guarantees for safety, security and other dependability characteristics is paramount for the successful adoption of ITS. Vehicle platooning systems are an application of ITS also explored in PATH, and in other research projects such as KONVOI, SARTRE and more [23], [2], [17, p. 19].

Vehicle platooning also falls under the definition of self-adaptive systems. In [5], an overview of the topic of assurance using models at runtime for self-adaptive software systems (SASSs) is provided. Platoons are SASSs in the sense that they are formed, modified and dissolved dynamically at runtime. The authors view SASSs as compositions of steady-state programs, managed by controllers or autonomic managers. SASSs require assurance to be performed at runtime as well, as part of their functionality is conditional on their adaptation. Thus, dependability-critical properties of SASSs must be assured before, during and on completion of each adaptation over the course of operation.

Earlier, in [18], the idea of using monitoring for adaptive system runtime verification is explored. The author notes that adaptive systems may feature unanticipated interactions at runtime. Rushby proposes a framework based on assurance cases and monitors generated from runtime verification languages to oversee whether assumptions and other safety properties of the case are not violated during operation. In cases where violations are detected, fault diagnosis can be applied, to trace indirect violations back to their source and identify appropriate options for recovering from failure or mitigating risk e.g. switching to backup systems or degrading services.

In [14], the issue of safety assurance for emergent collaboration of open distributed systems is explored. The authors note, like others have e.g. [18, p.1-2, 5-6][5, p. 1],[17, p. 2], that established practice restricts validation and verification processes to be performed exclusively during development. System behavior is decided and assured before operation, severely restricting the potential of runtime collaborative operations. The established view is rejected by the authors, who claim that available information on the operational context can be exploited to inform and assure collaborative functions at runtime. Their proposed approach is to predefine certain collaboration options during development. As the

[4] `https://path.berkeley.edu/home`

options are predefined, a priori analysis can determine what should the appropriate reaction be to specific conditions during operation. The reaction would ideally reduce or eliminate the risk of collaborative services during operation, such that it is acceptable for the services to be performed. The decision of the appropriate reaction during operation relies on the systems' capacity to monitor the condition of dependability properties. A notable concept that support this view are Conditional Safety Certificates (ConSerts) [19]. ConSerts extend the notion of a modular assurance case with conditional dependability guarantees. Guarantees are provided using user-defined Boolean logic gates (AND, OR) to combine demands imposed on further ConSerts or runtime evidence that will be collected during operation. Further ConSerts satisfy imposed demands by evaluating their own guarantees and so on. The authors of [14] note that their proposed scheme can be expanded to identify potential collaborative services at runtime and negotiate their assurance via ConSerts.

In their current state, ConSerts use binary variables to provide/request guarantees/demands. ConSerts Trees (CSTs) are formed to represent connectivity and hierarchical relationships between multiple ConSerts. To evaluate CSTs during operation, each CST is converted to a Binary Decision Diagram (BDD) [19, p. 13]. In [15], the authors note that ConSerts does not address uncertainty and propose an additional safety manager to be added to the AUTOSAR standard. The introduced manager actively manages policy, not only monitoring safety rules, but also enforcing them via the existing AUTOSAR mode manager. The proposed manager addresses uncertainty by monitoring the data quality and integrity of safety-critical information stored in safety contracts, the latter being a comparable concept to ConSerts. Alternative approaches for addressing uncertainty in CAV platoons include using formal methods [6], Hidden Markov Models [21] and a requirements language [24].

2.2 State Machines and Bayesian Networks

State machines (SMs) are a high-level modelling formalism used to explicitly express the behaviour of systems. In its basic form, a SM could be defined as:

$$SM = (S, \Sigma, \delta, s_0) \tag{1}$$

Where S is the set of all possible states in the SM, Σ is the set of all possible events, δ is the transition function $\delta : S \times \Sigma \rightarrow S$ and s_0 is the initial state. From a system behaviour modelling point of view, at the beginning of operational time, the system is in state s_0. When an event occurs, a state transition defined by the transition function can happen to take the system to another state. As SMs can readily model the effects of different events on different states of a system, they are well-suited for capturing the effects of failure and faults on the behaviour of a system. As a result, SMs have been utilised in a number of model-based safety analysis paradigms such as AADL [8], Altarica [1], xSAP[3], HiP-HOPS [16,12], etc.

Bayesian networks (BNs) as a probabilistic graphical model have flexible architecture, which can make decisions under uncertainty and can provide a global

assessment about different dependability properties such as reliability and availability by combining local level information from different sources. Graphically, BNs represent the relationships between a set of random variables in the form of a directed acyclic graph. These relationships can be interpreted as parent-child relations. In a BN, if an arc originates from a node X and terminates at another node Y, then X is the parent node and Y is the child node. A parent has direct effect on its child nodes. Such effects can be either be deterministic (in the sense they are guaranteed to occur) or probabilistic [13]. The probability distribution of a node X_i conditioned on its parents can be expressed as $Pr\{X_i|Parents(X_i)\}$. A node without a parent and one without children are known as root and leaf nodes, respectively. Using the BNs, the joint probability distribution of a set of random variables $\{V_1, V_2, V_3, \ldots V_{n-1}, V_n\}$ can be obtained by a chain rule as:

$$Pr\{V_1, V_2, V_3, \ldots V_{n-1}, V_n\} = \prod_{i=1}^{n} Pr\{V_i|\ Parent(V_i)\} \tag{2}$$

where $Pr\{x\}$ is the probability of x and $Pr\{x|y\}$ is the conditional probability of x given y.

In recent years, BNs have gained popularity in the dependability engineering area and an overview of widespread applications of BNs in safety and reliablity analysis can be found in [11].

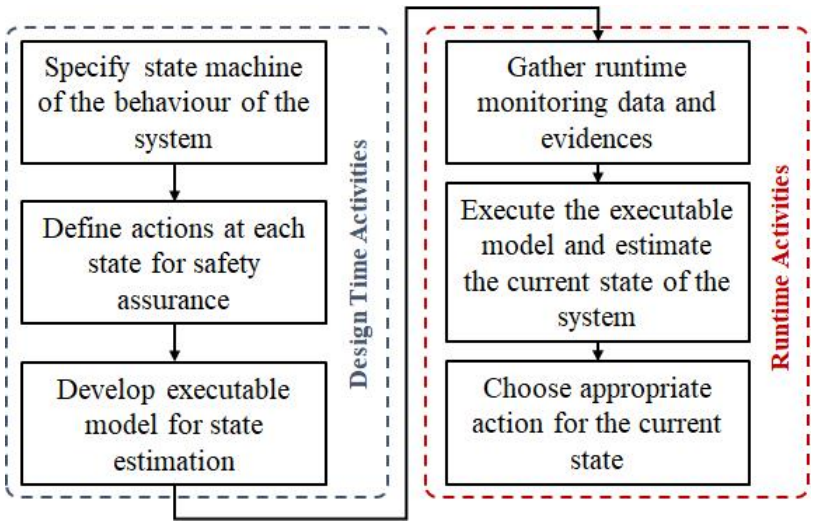

Fig. 1. Framework of the proposed approach

3 Proposed Framework

Our proposed framework relies on certain assumptions, described here. As mentioned earlier, during operation, self-adaptive systems can react to different unex-

pected and/or expected events and reconfigure themselves to continue providing services. That means such systems can operate in different mode of operations and each mode can be considered as a distinct system state. In this paper, we assume that during design stage, it is possible to foresee all the different possible states a system can be in during operation. The only a priori unknown is the actual state of the system at a given point during operation.

Fig. 1 shows the proposed framework for runtime safety assurance of self-adaptive systems. Note that, we assume that the user of this framework have the safety goal(s) defined for the subject system. As seen in the figure, the framework contains six different steps. The first three of these steps are performed during design time and the rest of the steps are performed repeatedly during runtime. The framework steps are performed as follows:

1. In light of assumptions made previously, as the first step of the framework, the behavioural model of the system is developed as a state machine. To do so, analysts require knowledge of the architecture and both nominal and failure behaviour of the studied system. Digital Dependability Identities (DDIs) [20] can be used to encapsulate this behavior into a machine-readable format.
2. Once the state machine is formed, per the second step of Fig. 1, analysts must identify the safety status of the system in each of the previously identified states with respect to the safety goal(s) defined earlier. Based on the state machine, per-state actions are defined to assure safety of the overall system operation. In other words, as a particular state depicts a distinct operational context for the system, actions are defined to reduce or eliminate unacceptable i.e. unsafe risk of operation under that context. How such actions should be processed in general depends on the specific application. For instance, some actions may be provided as recommendations to human operators, others may be input for automated controls.
3. The third step involves developing an executable model used at runtime to estimate the operational state of the system. To develop such an executable model, the first task is to identify necessary conditions that must be verified at runtime to ascertain the system state. Once such conditions are known, these conditions must be formulated in a verifiable format using parameters known at design time and parameters that can be monitored during system operation. The monitoring data based on these parameters includes internal system data, external data received from other collaborating systems, and environmental conditions. In this paper, we utilise the modelling capability of Bayesian Networks (BNs) to formulate such an executable model. In the BN model, leaf nodes are variables representing different parameters necessary to learn about the state of the system. The root node evaluates the current state of the system based on the leaf nodes. Note that, if multiple states could be evaluated from one BN, then multiple root nodes can be included. Alternatively, it is also possible to separate the executable model into several BNs when different state evaluations are needed. In this paper, we assume that each of the smaller participating subsystems to the larger system operate based on the principle of self-safety with group-awareness.

This means that each subsystem aims to ensure its own safety in the context of the whole system's operation, based on the subsystem's own data and data received from others. For this reason, each such subsystem contains executable model(s) (i.e. BN model(s)) to evaluate its own state.

4. During operation, each subsystem monitors its own data and data collected from other systems, and provides runtime input to the BN model(s). In the context of a BN model, this involves setting evidence on the leaf nodes of the appropriate BNs.
5. In the next step of Fig. 1, whenever new evidence is provided to the BN, the model is executed automatically to update the knowledge about the current state of the system's operation.
6. Once the current system state is known, appropriate actions can be selected for the current state from the predefined set of actions. As actions are predefined to assure system safety, proper execution of the actions should guarantee the safe operation of the system from any state, providing development assumptions hold. However, even if recommended actions are not executed properly, the executable model can revise the earlier recommendation by accounting for the new situation. In the worst case scenario, if the subsystem's safe operation cannot be guaranteed in the context of the larger system, then the operation can be suspended.

4 Illustrative Example

In this section, we use an abstract vehicle platooning system (see Fig. 2) example to demonstrate the proposed approach. For illustration, an already formed, stable platoon scenario with two vehicles, a leader and a follower, is assumed. Each vehicle is equipped with sensors and other components to detect frontal obstacles and communicate with other vehicles and roadside infrastructure. For brevity, we consider all communication channels adequately secure, hence, security issues are not considered in this use case. Moreover, we consider that both vehicles can operate in either Cooperative or Adaptive Cruise Control (CACC/ACC) mode. In CACC mode, each vehicle collaborates with other vehicles to provide safety guarantees. In the absence of collaboration (ACC mode), vehicles rely only on their own components to drive safely.

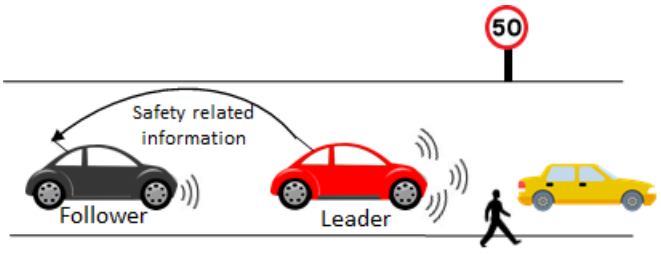

Fig. 2. Platoon with two vehicles

In our illustration, we assume that two vehicles are driving in CACC mode. For effective platooning, the follower vehicle always attempts to maintain the minimum distance from the leader such that inter-vehicle distances should not increase downstream from leader to followers. This concept is also known as 'string stability', [22]. Driving in close proximity to the leader leads offers improved fuel efficiency, at the cost of increased risk of frontal collision. The aim of this use case is to show that the follower vehicle can ensure, through the proposed approach, that the risk of frontal collision with the leader in any of its operational modes is minimal. Towards this end, the follower vehicle must verify specific safety properties during runtime. Similar to the example provided in [14], we consider that to ensure safe and lawful driving the follower vehicle has to ensure the following conditions:

- Condition 1: $d \geq d_s$, where d is the distance from the front vehicle and d_s is the minimum safety distance.
- Condition 2: $Speed_{vehicle} \leq Speed_{limit}$, where $Speed_{vehicle}$ is the current speed of the vehicle and $Speed_{limit}$ is the speed limit of the road.

Therefore, the safety goal considered is **"avoid violation of the safe distance and legal speed limit"**. For the purposes of the study presented here, only a few factors are considered for the determination of d_{min}. In practice, there is a plethora of additional dynamic factors that affect this limit and will need to be accounted for. A more detailed analysis of the involved factors can be found in [17, p. 45].

Across all scenarios, if the above conditions cannot be satisfied, then the follower vehicle will take appropriate action based to achieve safety by satisfying the conditions. However, if any of the above conditions are not verifiable at runtime, then the proposed approach will recommend the vehicle switch to ACC mode until verification can be performed. Potential reasons for non-verifiability include the unavailability of one or more parameters required for verification or poor parameter detection quality by the vehicles.

Fig. 3 presents the block diagram showing how safety is assured during runtime, from the follower vehicle's perspective. In the figure, the follower vehicle collects external runtime evidence from the leader and the roadside infrastructure via its communication interface. This evidence, coupled with the follower's internal parameters, are fed to its calculation unit. Within the calculation unit, the safety distance is calculated by accounting for different factors that may affect the vehicle's reaction capability. The specifics of the calculation are outside the scope of this paper. We consider the outcome of the calculation block available to the executable models created as part of the proposed framework.

The state machine of the behaviour of the follower vehicle in the context of the whole platoon is shown in Fig. 4. The state machine accounts for the different operational system contexts, with the follower vehicle being in six possible operational states. To guarantee safety in each state, table 1 shows the required action in each state. That means a distinct safety guarantee is associated with each state. In state S0, both conditions (distance and legal speed limit) are satisfied, therefore no special action is needed in this state. However, in S2, the first

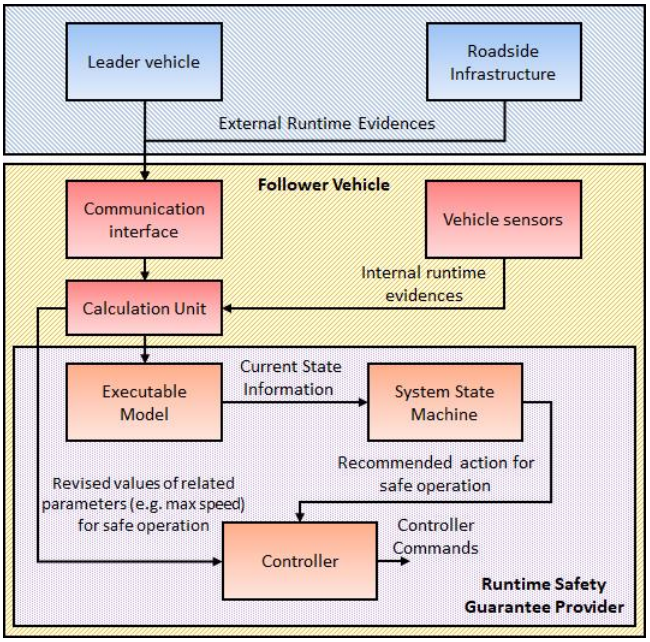

Fig. 3. Runtime safety assurance concept for the following vehicle

condition is not satisfied, therefore the follower should decelerate to increase distance from the leader. We should note that the state machine's role is advisory rather than prescriptive or descriptive. This means that it is used as a guide for identifying recommended actions to transition to safe/safer states, rather than being actively executed.

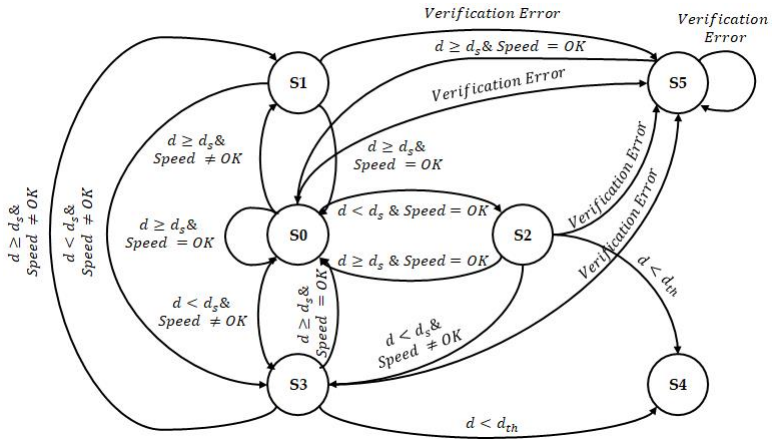

Fig. 4. State machine of the platoon system

Table 1. State with their description and associated actions

State	Description	Actions
S_0	The safety constraint is fulfilled and the vehicle is driving within the speed limit of the road.	The state is safe, therefore continue driving.
S_1	The safety constraint is fulfilled but the vehicle is driving outside the speed limit of the road.	Decelerate to fall within the speed limit.
S_2	The safety constraint is not fulfilled and the vehicle is driving within the speed limit of the road.	Decelerate to increase distance with the front vehicle until safety constraint is fulfilled.
S_3	The safety constraint is not fulfilled and the vehicle is driving outside the speed limit of the road.	Decelerate to achieve safety distance and fall within speed limit.
S_4	The safety constraint is not fulfilled, the vehicle is driving outside the speed limit of the road, and it is driving too closely.	Brake to stop driving.
S_5	Safety constraint and/or speed limit cannot be verified.	Switch to ACC mode.

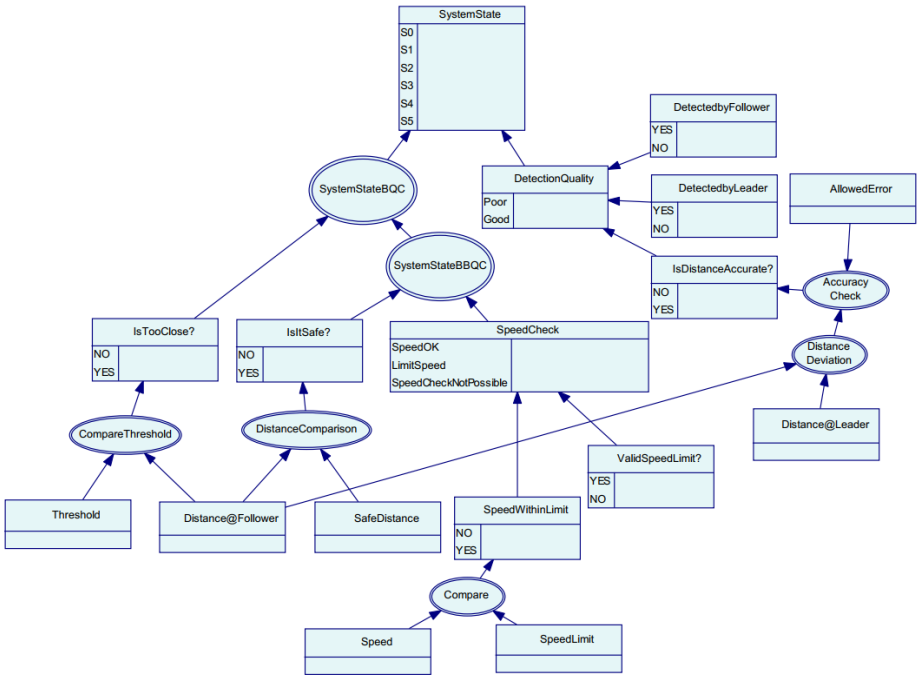

Fig. 5. BN model for deterministic estimation of system state

To identify the current operational situation for the follower vehicle, we consider internal safety-related data from the vehicle itself and external data from the leader and the environment together in a unified BN model as shown in Fig. 5. Note that this model is defined at design time and evaluated at runtime. In this model, we combine both quantitative and qualitative safety parameters for runtime inference about system state. For instance, Speed, SpeedLimit, Distance@Follower, SafetyDistance etc. are quantitative parameters. On the other hand, DetectedbyFollower, DetectedbyLeader, 'ValidSpeedLimit?' are qualitative binary parameters. In the model, different nodes are responsible for guaranteeing different conditions. For instance, the SpeedCheck node guarantees compliance of the vehicle's speed with the legal speed limit. The SpeedCheck node receives input from two child nodes. The child node 'ValidSpeedLimit?' represents a certificate about the validity of the speed limit, which is shared either by other vehicles or by roadside infrastructure. As vehicles may drive very closely in a platoon, street signs could be missed due to the view being obstructed by nearby vehicles. Moreover, the speed limit varies based on location, therefore it is also necessary to have a guarantee about the liveness of the monitored speed limit. Another child node, SpeedWithinLimit, monitors the legality of the vehicle's current speed by comparing with the current speed limit. In the absence of a certificate on the validity of the speed limit, no guarantee is provided regarding legality. However, if the speed limit is validated, the internal safety property SpeedOK is guaranteed if the current speed is within the speed limit. If the speed limit is valid and is exceeded, the safety property LimitSpeed is set; this is equivalent to $Speed \neq OK$ in Fig. 4. Similarly, the node 'IsItSafe?' provides a guarantee about whether the vehicle is maintaining safe distance from the leader. The DetectionQuality provides a guarantee about the detection capacity of the two vehicles. A guarantee about the detection quality is provided if the follower and leader vehicle detect each other and the distances measured by them do not deviate by a value larger than a predefined threshold.

In order to test whether the approach can detect different scenarios based on the runtime inputs, provide appropriate level of safety guarantees and recommend proper actions for ensuring safety, we randomly generated several test cases and tested the executable model of Fig. 5. Out of these test cases, Table 2 shows 6 different test cases ($C1$ to $C6$) which lead the executable model to provide six different guarantees, i.e., the follower vehicle was detected to be in six different states. Fig. 6, shows the first test case where the system was detected to be in $S0$, meaning a guarantee is provided about 'complete' safety i.e. safe distance and legal speed. On the other hand, in case **C2**, the system state is estimated as **S2**, meaning the system is violating the first safety constraint, thus **LimitSpeed** guarantee is provided in this case.

Note that, in the above test cases it was assumed that the values of the parameters are deterministic and their values are known with certainty, as a result, using the model in Fig. 5, the system states were estimated deterministically. However, in practice, we may be uncertain about the parameter values. In such cases, to address parameter uncertainty, we propose to use a probabilistic ver-

Table 2. Results of runtime verification of safety guarantees

Parameters	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**
Distance@Follower (m)	6.0	4.0	6.0	1.8	4.2	5.5
Distance@Leader (m)	6.2	3.9	5.9	1.9	4.4	6.2
Safe distance (m)	5.0	5.0	5.0	5.0	5.0	5.0
Too close distance (m)	2.0	2.0	2.0	2.0	2.0	2.0
Allowed error in distances (m)	0.5	0.5	0.5	0.5	0.5	0.5
Speed (miles/h)	48	47	37	37	77	48
Speed limit (miles/h)	50	50	30	30	70	50
Validity of speed limit	Yes	Yes	Yes	Yes	Yes	No
Leader detected by follower	Yes	Yes	Yes	Yes	Yes	Yes
Follower detected by leader	Yes	Yes	Yes	Yes	Yes	Yes
State estimated	**S0**	**S2**	**S1**	**S4**	**S3**	**S5**

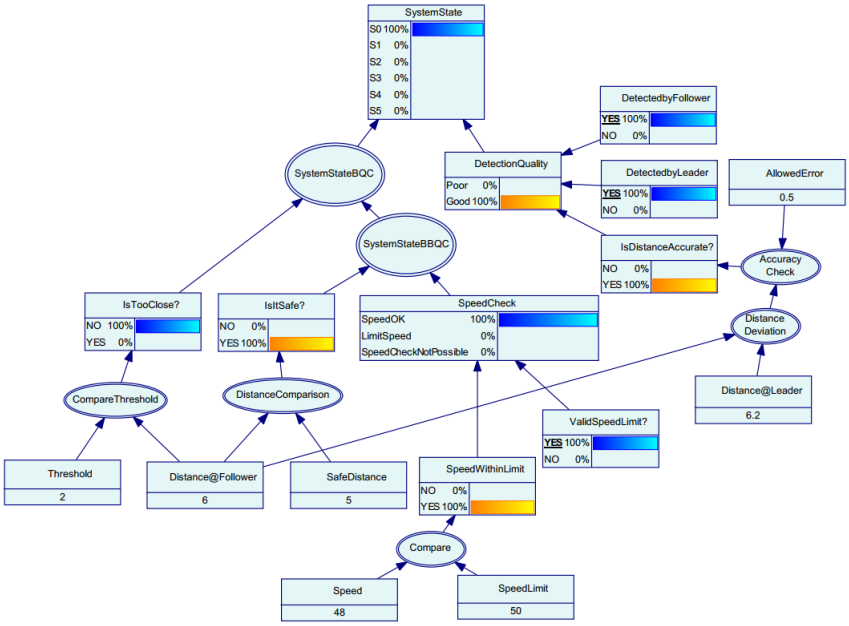

Fig. 6. Case 1 (C1) as shown in Table 2

sion of the executable model. As an example, we present a probabilistic version of system estimation in Fig. 7. In this example, the inputs to the BN model are probabilities instead of deterministic binary values. For instance, the node SpeedWithinLimit represents that there is an 80% chance that the speed of the vehicle is within limit and 20% of chance of exceeding it. Similarly, all other root nodes of the BN model represent probabilistic values for different parameters.

As a result of using such probabilistic values for the inference process, unlike the deterministic model, the system states are estimated with probabilistic rather than absolute guarantees. For instance, in the case of Fig. 7, the system was estimated to be in states S0 to S5 with 53%, 13%, 6%, 1%,0%, and 26% probability, respectively. As S0 state has the highest probability, it could be said that the system is most likely in state S0. Thereby, a (probabilistic) safety guarantee for this state can be provided and actions for this state can be executed. The simplistic rule applied is that the state with the highest probability is selected. However, there may be cases where two states both have (approximately) the highest probability. To resolve such cases, predefined rules can be applied for choosing state. For instance, the more safety-critical state can be chosen in the case of ties.

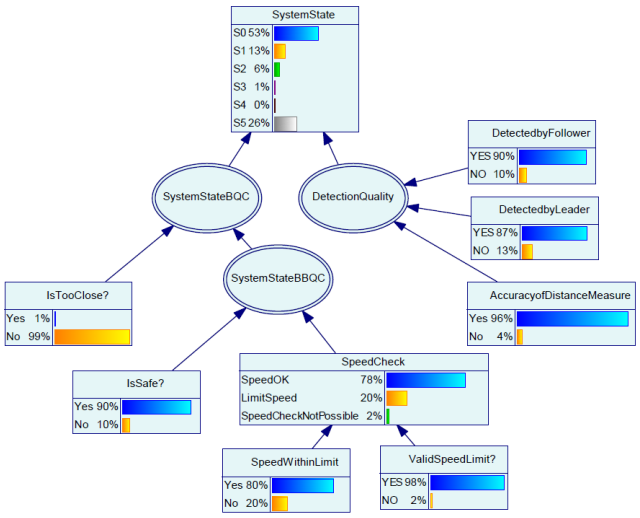

Fig. 7. An example of probabilistic system state estimation

5 Conclusion

In this paper, we present a conceptual framework for addressing the issue of safety under uncertainty in open adaptive systems. Our approach builds upon previous work on runtime certification, through the use of design-time dependability artifacts such as safety contracts, state machines and Bayesian Networks. Artifacts are deployed at runtime alongside a monitoring framework for observing system and environmental state. Thus, runtime knowledge is utilised to maintain safety properties or recover from unsafe situations.

The work presented here is part of a larger effort, the DEIS research project. As part of our ongoing research, we aim to integrate our proposed framework

with the concept of the DDI. DDIs aim to support modularity, composition, seamless exchange and evaluation of the associated dependability artifacts at runtime. DDI integration offers an avenue for implementing the approach in a larger systematic, top-down, traceable development framework. Such a framework provides justified confidence in the assurance of dependability-critical properties of CPS. Further avenues of investigation include linking the proposed approach with ConSerts. By combining modular and conditional certification with probabilistic reasoning and runtime monitoring, a larger section of the development lifecycle could be supported via relevant model-based techniques. Further, an assumption of our current approach is that the actions for mitigating safety risk defined from each system state have deterministic outcomes. A more robust framework would ideally be capable of deciding on actions with uncertain effects as well.

Acknowledgements

This work was supported by the DEIS H2020 Project under Grant 732242.

References

1. Arnold, A., Point, G., Griffault, A., Rauzy, A.: The AltaRica formalism for describing concurrent systems. Fundamenta Informaticae **40**(2), 109–124 (2000)
2. Bergenhem, C., Shladover, S., Coelingh, E., Englund, C., Tsugawa, S.: Overview of platooning systems. In: Proceedings of the 19th ITS World Congress, Oct 22-26, Vienna, Austria (2012) (2012)
3. Bittner, B., Bozzano, M., Cavada, R., Cimatti, A., Gario, M., Griggio, A., Mattarei, C., Micheli, A., Zampedri, G.: The xSAP safety analysis platform. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 533–539 (2016)
4. CCAV: Connected and autonomous vehicles: Market forecast. Tech. rep., Centre for Connected and Autonomous Vehicles; UK Department of Transport (2017), `https://www.gov.uk/government/publications/connected-and-autonomous-vehicles-market-forcecast`
5. Cheng, B.H.C., Eder, K.I., Gogolla, M., Grunske, L., Litoiu, M., Müller, H.A., Pelliccione, P., Perini, A., Qureshi, N.A., Rumpe, B., Schneider, D., Trollmann, F., Villegas, N.M.: Using Models at Runtime to Address Assurance for Self-Adaptive Systems, pp. 101–136. Springer International Publishing, Cham (2014). https://doi.org/10.1007/978-3-319-08915-7_4
6. Dolginova, E., Lynch, N.: Safety verification for automated platoon maneuvers: A case study. In: International Workshop on Hybrid and Real-Time Systems. pp. 154–170. Springer (1997)
7. Fagnant, D.J., Kockelman, K.: Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations. Transportation Research Part A: Policy and Practice **77**, 167–181 (2015)
8. Feiler, P., Rugina, A.: Dependability Modeling with the Architecture Analysis & Design Language (AADL). Tech. Rep. July, Software Engineering Institute, Carnegie Mellon University (2007)

9. International Electrotechnical Commission: IEC 61508: Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems, ed. Geneva: Switzerland. Tech. rep., International Electrotechnical Commission (1997)
10. ISO: ISO 26262: Road vehicles - functional safety. Tech. rep., International Organization for Standardization, Geneva, Switzerland (2011)
11. Kabir, S., Papadopoulos, Y.: Applications of Bayesian networks and Petri nets in safety, reliability, and risk assessments: a review. Safety science **115**, 154–175 (2019)
12. Kabir, S., Papadopoulos, Y., Walker, M., Parker, D., Aizpurua, J.I., Lampe, J., Rüde, E.: A model-based extension to HiP-HOPS for dynamic fault propagation studies. In: 5th International Symposium on Model-Based Safety and Assessment. pp. 163–178 (2017). https://doi.org/10.1007/978-3-319-64119-5_11
13. Kabir, S., Walker, M., Papadopoulos, Y.: Dynamic system safety analysis in HiP-HOPS with Petri nets and Bayesian networks. Safety science **105**, 55–70 (2018)
14. Mueller, S., Liggesmeyer, P.: Safety assurance for emergent collaboration of open distributed systems. In: IEEE International Symposium on Software Reliability Engineering Workshops. pp. 249–256. IEEE (2016)
15. Östberg, K., Bengtsson, M.: Run time safety analysis for automotive systems in an open and adaptive environment. In: SAFECOMP 2013-Workshop ASCoMS (Architecting Safety in Collaborative Mobile Systems) of the 32nd International Conference on Computer Safety, Reliability and Security. p. NA (2013)
16. Papadopoulos, Y., Walker, M., Parker, D., Sharvia, S., Bottaci, L., Kabir, S., Azevedo, L., Sorokos, I.: A synthesis of logic and bio-inspired techniques in the design of dependable systems. Annual Reviews in Control **41**, 170–182 (2016)
17. Reich, J.: Systematic engineering of safe open adaptive systems shown for truck platooning. MSc thesis, Technical University of Kaiserslautern, Kaiserslautern, Germany (2016). https://doi.org/10.13140/RG.2.2.27809.61283
18. Rushby, J.: Runtime certification. In: International Workshop on Runtime Verification. pp. 21–35. Springer (2008)
19. Schneider, D., Trapp, M.: Conditional safety certification of open adaptive systems. ACM Transactions on Autonomous and Adaptive Systems (TAAS) **8**(2), 1–20 (2013)
20. Schneider, D., Trapp, M., Papadopoulos, Y., Armengaud, E., Zeller, M., Höfig, K.: Wap: digital dependability identities. In: 2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE). pp. 324–329. IEEE (2015)
21. Stoller, S.D., Bartocci, E., Seyster, J., Grosu, R., Havelund, K., Smolka, S.A., Zadok, E.: Runtime verification with state estimation. In: Khurshid, S., Sen, K. (eds.) Runtime Verification. pp. 193–207. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
22. Swaroop, D., Hedrick, J.K.: String stability of interconnected systems. IEEE transactions on automatic control **41**(3), 349–357 (1996)
23. Tsugawa, S., Jeschke, S., Shladover, S.E.: A review of truck platooning projects for energy savings. IEEE Transactions on Intelligent Vehicles **1**(1), 68–77 (2016)
24. Whittle, J., Sawyer, P., Bencomo, N., Cheng, B.H.C., Bruel, J.: RELAX: Incorporating Uncertainty into the Specification of Self-Adaptive Systems. In: 17th IEEE International Requirements Engineering Conference. pp. 79–88 (Aug 2009)