The Journal of Systems and Software 230 (2025) 112499

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Compositional code-level safety verification for automated driving controllers☆

Vladislav Nenchev [a,*,1], Calum Imrie [b], Simos Gerasimou [b], Radu Calinescu [b]

[a] Department of Electrical and Computer Engineering, University of the Bundeswehr Munich, Neubiberg, Germany
[b] Department of Computer Science, University of York, York, UK

ARTICLE INFO

Keywords:
Formal verification
Compositional reasoning
Deep neural network verification
Model-based software analysis
Adaptive cruise control
Lane-keeping assist
Automated driving

ABSTRACT

Ensuring the safety of automated driving vehicles is particularly challenging due to the wide range of their operating conditions. This paper introduces CoCoSaFe, a Compositional Code-level formal Safety verification Framework for automated driving controllers. Unlike traditional verification methods, such as model-based analysis, counterexample detection by guided simulation, or runtime verification through online monitoring, our approach verifies controller implementations directly at code level in an offline setting. Compositional contracts and bounded model checking are employed to assess the implementation of subsystem controllers against invariant sets. For neural network-based controllers, we introduce a scalable three-step decomposition method that utilizes a neural network verifier. CoCoSaFe is applied to adaptive cruise and lane-keeping controllers, for which we derive formal specifications and analytical models of the desired longitudinal and lateral behaviors, amenable for decoupled invariant sets. Various types of traditional and neural network controllers are verified in the order of minutes, showcasing its broad applicability and effectiveness in ensuring behavioral safety of software for automated driving and similar cyber–physical systems.

1. Introduction

The engineering of self-driving vehicles is an important research and development area in both academia and industry, as this technology has the potential to improve the safety of all traffic participants. In this context, the Operational Design Domain (ODD) that defines the set of conditions under which an ADAS is intended to operate has grown significantly in recent years (Scholtes et al., 2021). Guaranteeing safe operation of modern automated driving vehicles involves controlling the system through software, often developed by large teams of developers using advanced architectures and algorithms, diverse programming languages, and various software frameworks (Staron, 2021). This complexity is one of the main reasons why verifying safety for all possible driving scenarios in the vehicle's ODD has proven to be very challenging. This is often addressed through a decomposition of requirements into sets corresponding to different subsystems, followed by the verification of each such subsystem with respect to its separate functional requirements (Ma et al., 2022). In ADAS, a common separation is conducted by considering a longitudinal and a lateral subsystem (Rajamani, 2011). Longitudinally, ACC has to keep a suitable distance to relevant target objects, such that the automated driving vehicle maintains a safe distance even in the presence of uncertainties. Laterally, a LKA must keep the vehicle in the lane, again considering relevant uncertainties associated with this task.

Although the states, control inputs and specifications can be formulated separately for these two subsystems, the longitudinal dynamics of the vehicle are not independent of its lateral dynamics and vice versa (Rajamani, 2011). For example, it is not possible for the LKA to guarantee on its own that the vehicle does not cross the lane boundaries on a curved road at all possible speeds. Similarly, without assumptions on the operation of the LKA, the ACC cannot guarantee that a curved road is traversed with adequate lateral acceleration. Automating the validation and verification of ADAS software is crucial for certification and a rapid release cycle. Verifying that each subsystem's controller does not violate its respective assumptions and guarantees is often achieved by a tailored redundant architecture (Behere and Törngren, 2016), as well as exhaustive simulation and testing in practice. Although the last two methods can be performed automatically even when a controller utilizes a Deep Neural Network (DNN) (Tian et al.,

☆ Editor: Raffaela Mirandola.

* Corresponding author.
E-mail addresses: vladislav.nenchev@unibw.de (V. Nenchev), calum.imrie@york.ac.uk (C. Imrie), simos.gerasimou@york.ac.uk (S. Gerasimou), radu.calinescu@york.ac.uk (R. Calinescu).

[1] V. Nenchev was with BMW Group when this work was conducted.

https://doi.org/10.1016/j.jss.2025.112499
Received 18 September 2024; Received in revised form 26 March 2025; Accepted 7 May 2025
Available online 2 June 2025

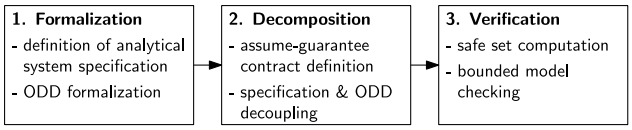

Fig. 1. Code-level automated driving verification process in CoCoSaFe.

2018), they are neither sound (i.e., not every bug report corresponds to a real bug) nor complete (i.e., some bugs may be missed).

To address this limitation of current verification approaches, we propose a Compositional Code-level Safety verification Framework (CoCoSaFe) for the development-time safety verification of automated driving controller implementations. As shown in Fig. 1, CoCoSaFe employs a three-stage safety analysis process based on the well-established phases of formalization, decomposition, and verification that we specialized for automated driving. In the first stage, the overall specification and the ODD of the system are formalized such that they capture automated driving requirements, desired behavior, and safety constraints. Further, a suitable model of the environment for the vehicle controller is derived. Then, in the second CoCoSaFe stage, we define suitable compositional contracts to decompose the analytical specification and the ODD into subsystems. This allows synthesizing a separate Control Invariant Set (CIS) for each subsystem, allowing the computation of a corresponding safe operation set. In the final CoCoSaFe stage, the safe sets are used to verify (or falsify) the closed-loop operation individually at code-level. For traditional controllers, Bounded Model Checking (BMC) is utilized to prove safety. For DNN-based controllers, we propose a new hybrid verification approach based on decomposition: we verify the deployment code (loading the DNN, DNN-based inference, etc.) using a BMC, and the actual DNN with a dedicated neural network verification tool (e.g., Marabou Katz et al., 2019). Thus, satisfying the general safety requirements of the entire system is accomplished by assume-guarantee reasoning and contract-based design and verification of each subsystem (Benvenuti et al., 2008). Taking the example of ACC and LKA of an automated driving vehicle, we specify the safety requirements for each of the two subsystems in terms of their ODD and specification. Using the proposed framework, the safety of both traditional and DNN-based ACC and LKA implementations of various types can be verified (or falsified) within minutes on a standard workstation. The considered controllers are commonly used in contemporary industrial and research approaches for ADAS (Garrido and Resende, 2022). Under certain assumptions on the formal model, the ODD and the employed tools, CoCoSaFe is complete with respect to the safe set. Additionally, it complements data-based testing, which generally cannot encompass the entire safe set but is capable of checking more complex behaviors and certain edge cases that are not represented in the formal model.

A preliminary variant of CoCoSaFe tailored for a specific ACC case study was presented in Nenchev et al. (2024). In this paper, we significantly extend our preliminary work from Nenchev et al. (2024) by:

1. Introducing a generalized verification framework for automated driving subsystem controller implementations.
2. Utilizing maximal safe sets instead of more conservative low complexity safe sets to cover larger parts of the respective ODD's of the controllers.
3. Proposing a possible handling of internal controller states.
4. Deriving a tailored compositional verification solution for adaptive cruise control and lane keeping assist by explicitly considering the interdependences in automated driving behavior.
5. Showcasing the verification of implementations of seven distinct controller types for adaptive cruise control and lane keeping assist.

The primary goal of our paper is to demonstrate how compositional reasoning, pre-computed safe sets, and model checking can be effectively used for the code-level safety verification of automated driving high level controllers such as ACC and LKA. We anticipate that CoCoSaFe can be effectively applied beyond the automated and autonomous driving domain, since (i) many Cyber-Physical Systems (CPS) incorporate controllers with analytical specifications that facilitate the computation of invariants, (ii) an Assume-Guarantee Contract (AGC) naturally reflects their interconnected component structure consisting, e.g. of sensors, controllers, and actuators, allowing verification of each component independently, and (iii): the Stage 3 in Fig. 1 is completely application agnostic.

The remainder of the paper is organized as follows. After summarizing related work (Section 2) and preliminaries (Section 3), Section 4 presents CoCoSaFe for ADAS controllers. Experimental results for various classes of ACC and LKA controllers are provided in Section 5, followed by a discussion (Section 6) and a conclusion (Section 7).

2. Related work

While formal methods have been widely applied in the context of aerospace design (Chrszon et al., 2023) and automated driving (Mehdipour et al., 2023; DeCastro et al., 2020), the main focus so far has been on the overall behavior of an automated vehicle, rather than on the actually deployed code in the vehicle. In turn, there is substantial work of verifying code on the vehicles (Durand et al., 2021), as strongly advised by ISO 26262, which does not capture the behavior of the vehicle. Formal verification of traditional control approaches has been addressed, e.g., by using model checking (Lygeros et al., 1996), counterexample-guided searching (Stursberg et al., 2004), set invariance (Kianfar et al., 2013), or reachability analysis (Alam et al., 2014). Since determining invariant sets is computationally demanding, runtime verification (Camilli et al., 2023) and online monitoring approaches have also been proposed as more scalable options (Kojchev et al., 2020; Pek et al., 2020; Jacumet et al., 2023). Compositional verification has been used for proving the correctness at an architectural subsystem level, possibly including learning-based components (Păsăreanu et al., 2019). A compositional framework, where tasks were a-priori decomposed into modes, separate controllers were learned for each mode, and their joint correctness was verified, was used for an autonomous car (Ivanov et al., 2021). However, none of these methods is directly applicable for functional safety verification of automated driving controllers at code-level.

Many methods for input–output robustness certification of DNNs have also been proposed in recent years, including feed-forward multilayer (Huang et al., 2017), deep feed-forward (Katz et al., 2019), and convolutional neural networks (Gehr et al., 2018). Formal proofs of closed-loop safety have also been obtained for DNN-based controllers and various system types, e.g. Calinescu et al. (2024), Dawson et al. (2023), Lopez et al. (2023), Paterson et al. (2021), Ruoss et al. (2021), Santa Cruz and Shoukry (2022), Sun et al. (2019). However, these methods rely on either approximations or abstractions of the to-be-verified controller or the system, and thus tend to scale poorly with the growing complexity of the system. Satisfiability Modulo Theory (SMT) solvers have also been used for the automatic verification of DNNs with respect to safety properties in CPS by using a dedicated interval constraint propagation (Grundt et al., 2022) or by translating the closed-loop system into an SMT formula (Scheibler et al., 2015). However, also in the context of DNN-based controllers, safety verification has not been addressed for the deployed code.

Current methods for identifying, correcting, and managing vulnerabilities in Embedded, Cyber–Physical, and Internet-of-Things systems and software are primarily concentrated on detecting software vulnerabilities in low-level software, e.g., by utilizing customized formal real-time operating system models (Adelt et al., 2024), or applying dynamic analysis on binary and executable files of the software (Marchetto and

Scanniello, 2024). Safety verification at code level has been addressed for an automated driving supervisor by automatically obtaining a finite discrete abstraction by black-box simulation (Nenchev, 2021). However, finite discrete abstraction approaches are not directly applicable to continuous controllers. Searching for implementation errors in unmanned aerial vehicles attitude controllers based on finite word-length effects (e.g., arithmetic overflows and limit cycles) in the code was presented (Chaves et al., 2018), but functional correctness was not considered. Using BMC, the execution of ACC implementations with a simulation model was verified to not exceed speed limits (Zhang et al., 2013), or to not violate the general safety specification given as a temporal logic specification embedded as a monitor along finite traces (Nenchev, 2023). However, the runtime of these methods is generally exponentially related to the length of the look-ahead horizon for verification. In contrast, by using a pre-computed invariant set to evaluate the safety of the controller, our approach requires only a single step look-ahead in time. Although the computation of the invariant set itself may exhibit exponential complexity inherent to using model checking, this process is conducted separately from the code verification. Furthermore, it can be optimized by reducing complexity at the price of increased conservatism in the safety guarantees.

Instead of providing arguments for absolute correctness, the test coverage of automated driving functions, mostly provided by manual tests and simulation runs in practice, can be extended by searching for specification counterexamples for implementation utilizing reinforcement learning in simulation (Favrin et al., 2020) or by sampling initial conditions from the boundary of a controlled invariant set (Chou et al., 2018). While the latter two approaches are neither sound nor complete, a sound alternative is to (mostly) automatically extract higher-level logic models from code (König et al., 2024), thus enabling exhaustive analysis for identifying potential errors prior to deployment. In a "hybrid" approach for verifying hybrid control systems, a combination of a black-box simulator for trajectories and a white-box transition graph for mode switches was assumed to trade off soundness and scalability (Fan et al., 2017). Although these methods avoid human bias inherent to manual testing and can discover corner cases that may be overlooked otherwise, infinite automated abstraction-based approaches are not guaranteed to scale well for all systems and cannot provide completeness guarantees for large parts of the ODD. Through a compositional approach, a framework for automatic verification of system-level properties that combines software model checking and contract-based analysis was demonstrated in an automotive case study (Cimatti et al., 2023). Upon formalizing functional requirements into contracts with assumptions and assertions expressed in Linear Temporal Logic (LTL), these contracts are assigned to runnable or composite components, and the source code of a module can be verified with respect to the specified contracts. Instead of focusing on individual requirements, CoCoSaFe aims at controller code verification over the maximal safe set of the formal model within the ODD.

3. Preliminaries

In this section, we introduce foundational concepts for CoCoSaFe with a particular focus on automated driving.

Compositional reasoning (Giannakopoulou et al., 2018) is an approach that allows the analysis of complex systems by breaking them down into smaller, more manageable components. In the context of ACC and LKA, compositional reasoning facilitates the verification of individual subsystem controllers (e.g., speed control, lane-keeping) while ensuring that their interactions do not lead to unsafe behaviors, e.g., Cimatti et al. (2023). To decompose the system, suitable AGC have to be defined that describe the assumptions under which a subsystem operates and the guarantees it provides for other subsystems. It is important to note that, rather than employing an AGC to break the system down into (sequential) components, this work utilizes an AGC to decouple the system into subsystems that can function independently or concurrently, similar to Ivanov et al. (2021).

The **operational design domain** of an ADAS defines the specific conditions under which that ADAS is intended to operate (Rajamani, 2011). This may encompass factors such as road types (e.g., highways, urban streets) and traffic scenarios (e.g., presence of other vehicles, pedestrians). Assuming a formally defined state for an automated driving controller, we adopt a notion of its ODD O given as a polyhedral region of the state space where the system can function effectively within their intended operational parameters. However, as the control signals are bounded, the automated driving function can only operate safely within a safe subset S of the ODD O.

To obtain the safe set S, a **control invariant set** (Blanchini, 1999) can be used, which is a subset of the ODD O of a dynamical system where, if the system trajectory starts within this set, it can be controlled to remain within the set for all future time steps. The maximal CIS is the largest such set within O achievable under the given dynamics and control constraints. A valid under-approximation of the CIS guarantees that every state in it can be reached by following a feasible control sequence. Techniques from control theory, such as Lyapunov functions and reachability analysis (Wongpiromsarn et al., 2012), can be employed to compute the CIS. For ACC and LKA, obtaining such sets is crucial for guaranteeing that the vehicle maintains safe distances from other vehicles and stays within the lane.

Bounded model checking is a verification technique used to determine whether a system satisfies certain properties within a finite number of steps (Clarke et al., 2004). This is particularly useful for verifying the correctness of software against safety specifications. For ACC and LKA, by exploring executions of the controller code, BMC can identify potential violations of safety properties for initial states in the safe set, e.g., as used in Zhang et al. (2013) and Nenchev (2023). There are existing software tools, such as UPPAAL (Behrmann et al., 2006) and KeYmaera X (Fulton et al., 2015), that can be used for verifying, runtime monitoring, and synthesizing controllers for CPS. However, none of the currently existing software tools have been used to provide comprehensive functional verification at the code level for automated driving controllers.

4. Code-level verification framework

4.1. Overview of the approach

As outlined in the introduction, CoCoSaFe is structured into the three key stages shown in Fig. 2.

As a first stage, the overall specification and ODD are formalized. This includes defining the automated driving requirements, including the desired behavior and safety constraints, and deriving a suitable model of the environment in which the to-be-verified vehicle controller will operate. In common automated driving system architectures, the driver sets parameters, such as a desired speed and distance to a front object for ACC to manage the longitudinal motion and LKA to manage lateral motion. These systems interact dynamically due to vehicle physics. Their goal is to ensure that acceleration and steering commands keep the vehicle safe for a defined parameter range. Safety requirements follow standards like ISO (Anon, 2018) and UNECE (United Nations Economic Commission for Europe (UNECE), 2023) regulations to ensure target distances, velocities, and vehicle stability are maintained. In addition to considering relevant safety standards, Stage 1 typically involves consulting functional safety experts, and modeling hardware aspects, which might have an impact on safe operation. This is carried out in detail in Section 4.2 for the joint keep distance and keep lane behavior of an automated driving vehicle.

Once the overall analytical specification Σ and ODD O are derived, the automated driving system is decomposed into multiple interacting subsystems. For each subsystem denoted by $(.)$, assumptions are made

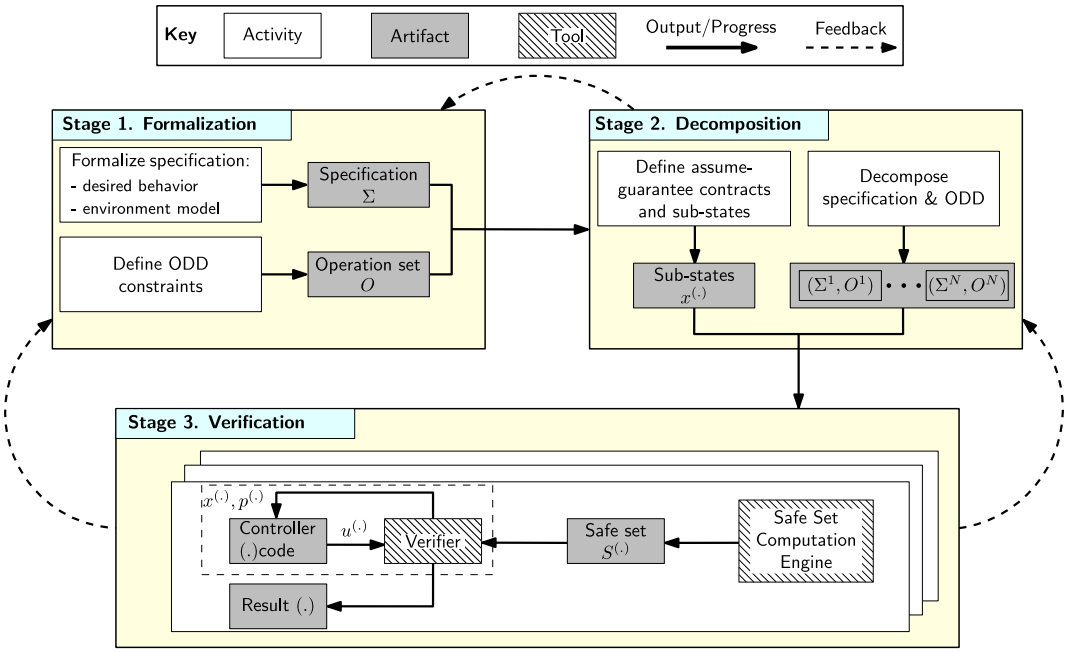

Fig. 2. Detailed overview of the proposed code-level automated driving verification framework. For automated keep distance and lane keeping, for each pair of an operation set and an analytical specification $(O, \Sigma) \in \{(O^{lon}, \Sigma^{lon}), (O^{lat}, \Sigma^{lat})\}$, the safe set $S^{(.)}$ is computed and used to check the safe closed-loop operation of the corresponding controller implementation using a verifier.

about the behavior of the environment or other subsystems (e.g., sensors will provide accurate data). In return, each subsystem guarantees certain behaviors under these assumptions (e.g., the vehicle will safely follow a path, if sensors provide correct data). Using these AGC, decoupled states $x^{(.)}$, analytical specifications $\Sigma^{(.)}$ and ODDs $O^{(.)}$ are obtained for each (sub-)controller of the overall system. This helps in isolating and specifying the behavior of individual controllers in a modular and scalable way. Similarly to Stage 1, this involves considering relevant safety standards as well as consulting functional safety experts, but also requires discussions with algorithm and engineering experts. For the keep distance and keep lane behavior of an automated driving vehicle, in Section 4.3 we utilize a common specification decomposition into a longitudinal and lateral subsystem, where the former is concerned with keeping a suitable distance to a front object, and the later with safely keeping the lane. This allows us to derive two separate analytical specifications for keep distance and keep lane.

The final stage is to verify the automated driving system compositionally, i.e., the correctness of the overall system is established by verifying each subsystem individually. Thus, safe sets $S^{(.)}$ that represent the respective set of all states in which a subsystem can safely operate are obtained. The control signal is provided based on the state $x^{(.)}$ by a software module — the to-be-verified controller, implemented in a general purpose programming language, e.g., C/C++, possibly containing a neural network. All controllers are assumed to be time-invariant and deterministic. For traditional controllers, the Verifier is checking that each controller implementation of a subsystem provides only control signals $u^{(.)}$ at any admissible state $x^{(.)}$, such that its closed-loop operation remains in the safe set $S^{(.)}$ for all possible parameters $p^{(.)}$. For neural-network-based controllers, the verifier consists of three steps, where integration, deployment, and neural network verification are decoupled. If Result (.) is positive for the controllers of all subsystems, this step ensures that the integrated system meets its overall specification and performs reliably and safely. As the joint keep distance and keep lane behavior of an automated driving vehicle is decomposed into two concurrently operating subsystems with respective analytical specifications $\Sigma^{(.)}$ and operation sets $O^{(.)}$, in Section 4.4 we follow the described process for verifying the two subsystems.

Note that during Stages 2 and 3, discrepancies between the overall specification, the ODD, and the contracts may come to light. Consequently, either may require an iterative refinement, denoted by the dashed arrows. For instance, if in Stage 2 the contracts cannot ensure safe operation for the entire overall specification or ODD, the latter might need to be scaled down in Stage 1. If Stage 3 yields an empty safe set for the sub-states and decomposed specifications and ODDs, then it would be necessary to reassess either the contracts from Stage 2 or the entire specification and ODD from Stage 1.

Remark 1. The framework proposed in this paper focuses on high-level controllers, such as ACC and LKA. Controller stability with respect to a formal model is implicitly guaranteed, if the controller keeps the system within the corresponding invariant set. In addition, it is assumed that the employed low-level controllers are stable (or their stability has been proven or verified) and their safety-relevant dynamical behavior is included in the formal model. Further, perfect perception is assumed, implying that the controllers have access to complete and accurate information about their environment at all times. Several aspects of imperfect sensing and other dynamical effects can be integrated into a formal model, e.g., as shown in Nenchev (2025).

In the following sections, we describe in detail the key stages of CoCoSaFe for verifying high-level controllers.

4.2. Stage 1: Formalization

Deriving an analytical specification and ODD suitable for formal verification is a challenging task for the complete chain of effects from the sensors to the actuators of an automated keep distance and lane keeping vehicle. We start by identifying the primary use cases for the system and analyze environmental factors like other traffic participants and road representations that affect system performance. In addition, operational constraints related to vehicle dynamics and sensor capabilities are derived. This includes adhering to existing industry standards and engaging with stakeholders, including engineers and safety experts, to validate the derived specification and ODD, and iteratively refine it based on testing and real-world data.

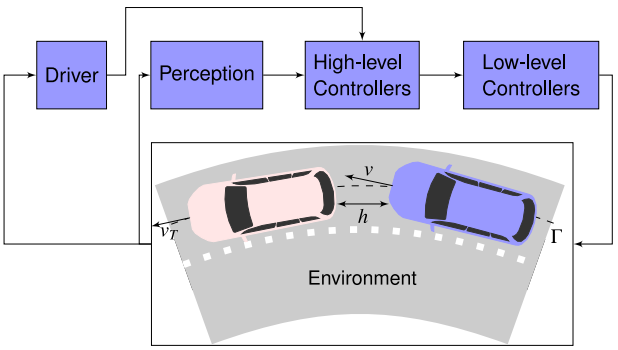

Fig. 3. A common ADAS architecture with typical high-level controllers ACC and LKA. ACC provides the acceleration a and LKA the curvature κ to the low-level controllers of the ego vehicle (blue).

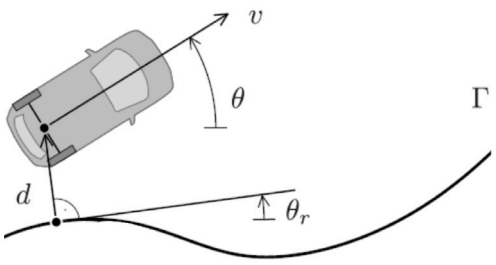

Fig. 4. Lateral vehicle kinematics (Wu et al., 2024).

Consider a common ADAS architecture shown in Fig. 3, as, for example, used in Widmann et al. (2000). The driver activates or deactivates the high-level controllers ACC and LKA, and provides a desired velocity v_d and a desired time headway t_{h_d} for the former. The desired velocity is the target velocity for the automated driving vehicle (also referred to as the ego vehicle). The distance between the relevant physical limits of the front object (e.g. the rear bumper of a vehicle) and the front bumper of the ego vehicle is the headway h. The time headway t_h is the amount of time after which the target object and the ego vehicle will collide, given the current headway h, when the target object suddenly stops and the ego vehicle maintains its current ego velocity v, i.e., $t_h = h/v$. The desired time headway t_{h_d} to the target object corresponds to the relative distance that eventually needs to be maintained. The goal of lane-keeping is to control the vehicle to follow the center line of the current lane by providing a suitable curvature κ. The information about the target object and the road is measured by sensors such as radars, cameras, or a lidar. This information is utilized in the ACC to produce an acceleration and in the LKA to obtain a curvature for reaching desired states for the ego vehicle. The acceleration commands from the ACC are used by the low-level controllers such as the engine control unit and/or power train, the transmission controller and the brake controller, the curvature signal from LKA is used by the low-level controller to produce a steering angle. In most driving scenarios, the longitudinal motion is controlled by the throttle and brake inputs, while the lateral motion is determined by the steering input. Thus, a common assumption for comfort driving functions is to decouple their effects: ACC controls the longitudinal subsystem and LKA deals with the lateral subsystem.

To obtain an analytical longitudinal specification, the vehicle's relevant dynamics (Rajamani, 2011) are obtained by non-linear force-balance equations, combined with exact feedback linearization to compensate non-linearities for the low-level effect chain and the variable vehicle mass (Ioannou and Chien, 1993). Inspired by Nilsson et al. (2016), where correct-by-construction ACCs were synthesized for formal models, the state of the longitudinal dynamics has to contain v, v_T, and h. As the slip angle can be assumed to be zero in the rear axle center when driving slowly and is generally neglected when driving fast, the direct dependence of the longitudinal dynamics on the lateral dynamics is neglected.

Consider an automated driving vehicle that has to keep its lane (Fig. 4). Instead of considering a dynamical single-track model for lateral motion, we opt for linearized relative kinematics to a reference curve Γ (denoted by the dashed curve in Fig. 3 and the solid curve in Fig. 4, respectively), which denotes the center of the lane, as it captures relevant safety aspects. The vehicle's rear axle center is used as a reference point, and the curvature as the model's control input. For that, d represents the normal signed distance between the reference curve Γ and the center of the rear axis of the vehicle (also called the cross-track error), and θ is the vehicle orientation. The orientation θ_r is the normal distance between the reference curve Γ and the vehicle's rear axle center position, serving as a disturbance input to the model. The state has to contain θ and d for the lateral dynamics. Thus, the vehicle movement is described by the dynamical model:

$$\Sigma : \dot{v} = a, \dot{v_T} = a_T, \dot{h} = v_T - v, \dot{d} = v(\theta - \theta_r), \dot{\theta} = v\kappa. \tag{1}$$

According to the relevant ISO standard (Anon, 2018), the ACC computes a, so that the ego vehicle velocity v reaches the desired driver velocity v_d, or so that the headway h to the target object driving with velocity v_T stays above a specified minimal value h_{min} and the current time headway stays above a specified minimal time headway $t_{h_{min}}$. In set speed mode, the target object is irrelevant, and the only safety requirement is the physically limited control. Since control limitations can be guaranteed by a simple limiter, in this work, we focus on the so-called time gap or keep distance operation of the ACC. Further, without loss of generality, the ego acceleration a, the target object acceleration a_T are limited by $a \in [a_{min}, a_{max}]$ and $a_T \in [a_{T,min}, a_{T,max}]$, respectively.

For lane keeping, the UNECE Regulation 79 (United Nations Economic Commission for Europe (UNECE), 2023) includes requirements for Automated Commanded Steering Functions (ACSF). While the regulation deals with all functions related to automated steering, the nominal operation of LKA we consider is closest to the category B2 system. Category B2 pertains specifically to LKA that provides continuous support to the driver by keeping the vehicle within a lane without requiring input from the driver for steering for extended periods of time. As mandated by typical road and car widths, the specification is that the car should remain within d_{max} meters of the center of the lane, i.e. $|d| \leq d_{max}$. We also require that θ remains in its corresponding admissible bound, as well as the curvature κ, the curvature of the road θ_r, i.e., $\kappa \in [\kappa_{min}, \kappa_{max}], \theta_r \in [\theta_{min}, \theta_{max}]$.

Collecting all requirements, the overall ODD for joint keep distance and keep lane operation is denoted by

$$\begin{aligned} O = \{ & h/v \geq t_{h_{min}} \wedge h_{min} \leq h \leq h_{max} \wedge \\ & v_{T,min} \leq v_T \leq v_{T,max} \wedge v_{min} \leq h \leq v_{max} \wedge \\ & |d| \leq d_{max} \wedge \theta_{min} \leq \theta \leq \theta_{max} \wedge \\ & a_{min} \leq a \leq a_{max} \wedge a_{T,min} \leq a_T \leq a_{T,max} \wedge \\ & \kappa_{min} \leq \kappa \leq \kappa_{max}, \theta_{min} \leq \theta_r \leq \theta_{max} \}. \end{aligned} \tag{2}$$

Note that the longitudinal and lateral motion interact dynamically in the derived analytical specification (1). The overall system is composed by an ACC, an LKA, and a formal model (1), such that the composition should guarantee for the operation to remain in (2).

4.3. Stage 2: Decomposition

System decoupling is often achieved through modular design, where each subsystem encapsulates specific functionality, and by defining interfaces that capture responsibilities between modules. To determine suitable AGC, subsystem interactions are analyzed and the assumptions and guarantees for each module are articulated. System safety

can then be ensured through compositional reasoning, which involves demonstrating that the safety of each individual subsystem guarantees the overall safety of the system. This involves breaking down the overall system into subsystems, verifying each of them individually, and then ensuring that their integration preserves the overall system safety. CoCoSaFe uses the common decomposition of automated driving, e.g., as assumed in Rajamani (2011) and most relevant safety standards, given by a longitudinal and a lateral vehicle guidance subsystem.[2] We first derive an AGC, which allows a decomposition of the overall analytical specification and ODD derived in Section 4.2.

4.3.1. Assume-guarantee contracts and substates

An AGC serves as a means to formally specify and verify the interactions between different subsystems. By defining assumptions about the environment in which a subsystem operates and guarantees about its behavior, these contracts facilitate modular verification. This modularity is crucial for managing complexity in large software systems, enabling scalable and more efficient verification processes. In the domain of automated driving, a common approach is to split the system into a longitudinal and a lateral subsystem.

Ensuring a proper speed while navigating a curve is crucial for maintaining vehicle stability and safety. The longitudinal velocity of the vehicle must be such that the required centripetal force does not exceed the available friction force. Assuming a coefficient of friction between the tires and the road μ and with g denoting the gravitation coefficient, the maximum safe speed in a curve can be guaranteed by

$$v \leq \sqrt{\frac{g\mu}{\kappa}}. \tag{3}$$

Note that in the absence of a preview for the curvature of the road, this inequality will generally impose a conservative restriction on the ODD (2). A potential solution is discussed in Remark 2. As (3) is an additional system requirement, we augment the previously derived ODD (2) by (3). Going back to Fig. 2, this corresponds to following the dashed arrow from Stage 2 back to Stage 1. Since this requirement augmentation does not require further refinement of the overall formal ODD or specification, we proceed with defining the AGCs. To that end, we assume that a non-empty maximal CIS exists for (1) within (2) and (3).

Considering the model (1), the subsystem states can be defined in a straight forward manner by $x^{lon} = [v, v_T, h]^T$ for the longitudinal and $x^{lat} = [\theta, d]^T$ for the lateral subsystem, with respective dynamics.

Now we can derive separate analytical specifications, which capture the desired behavior of their respective controllers ACC and LKA, operating side-by-side.

4.3.2. Decoupled analytical specifications

With the formal specification and ODD, and the AGC, we obtain decoupled analytical specifications for longitudinal and lateral automated vehicle motion as follows. The continuous differential Eqs. (1) are transformed into discrete time difference equations by exact discretization with an equidistant sampling period t_s. A zero-order hold is used at a time instant t for the duration of t_s for a and a_T, which are denoted by a_t and $a_{T,t}$ in the discrete time domain. Exact discretization ensures that safety proofs on the discrete time system hold for the corresponding trajectories of the original continuous time system. The continuous state variable $x^{(\cdot)}$ is replaced by the corresponding discrete time version $x_t^{(\cdot)}$ at a discrete time instant t. Analogously, the same applies for κ and θ_r with κ_t and $\theta_{r,t}$ in the discrete time domain for the lateral model. Thus, the assumed longitudinal model is

$$\Sigma^{lon}: x_{t+1}^{lon} = A^{lon}x_t^{lon} + B^{lon}a_t + E^{lon}a_{T,t}, \text{ with}$$
$$A^{lon} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -t_s & t_s & 1 \end{bmatrix}, B^{lon} = \begin{bmatrix} t_s \\ 0 \\ -\frac{1}{2}t_s^2 \end{bmatrix}, E^{lon} = \begin{bmatrix} 0 \\ t_s \\ \frac{1}{2}t_s^2 \end{bmatrix}. \tag{4}$$

An assumption for ACC is that the current curvature will remain within feasible limits, guaranteed by LKA. In addition, the inequality (3) ensures the maintenance of appropriate speed for the current curvature. Extracting the remaining longitudinally relevant part of the overall ODD, the longitudinal ODD is

$$\begin{aligned} O^{lon} = \{ & h/v \geq t_{h_{min}} \wedge h_{min} \leq h \leq h_{max} \wedge \\ & v_{T,min} \leq v_T \leq v_{T,max} \wedge v_{min} \leq v \leq v_{max} \wedge \\ & a_{min} \leq a_t \leq a_{max} \wedge a_{T,min} \leq a_{T,t} \leq a_{T,max}\} \wedge \\ & v \leq \sqrt{g\mu/\kappa} \wedge \kappa_{min} \leq \kappa \leq \kappa_{max}, \end{aligned} \tag{5}$$

where the inequality $v \leq \sqrt{g\mu/\kappa}$ comes from (3).

Similarly, the lateral model is

$$\Sigma^{lat}: x_{t+1}^{lat} = A^{lat}x_t^{lat} + B^{lat}\kappa_t + E^{lat}\theta_{r,t}, \text{ with}$$
$$A^{lat} = \begin{bmatrix} 1 & 0 \\ v_t t_s & 1 \end{bmatrix}, B^{lat} = \begin{bmatrix} v_t t_s \\ \frac{1}{2}v_t t_s^2 \end{bmatrix}, E^{lat} = \begin{bmatrix} 0 \\ -v_t t_s \end{bmatrix}. \tag{6}$$

An assumption for LKA is that the velocity controlled by ACC will be within admissible bounds. Analogously to the longitudinal model, by extracting the remaining laterally relevant part of the overall ODD, the lateral ODD is

$$\begin{aligned} O^{lat} = \{ & |d| \leq d_{max} \wedge \theta_{min} \leq \theta \leq \theta_{max} \wedge \\ & \kappa_{min} \leq \kappa \leq \kappa_{max}, \theta_{min} \leq \theta_r \leq \theta_{max}\} \wedge \\ & v_{min} \leq v \leq v_{max}. \end{aligned} \tag{7}$$

4.3.3. Correctness of compositional reasoning

While the longitudinal model (4) is independent of the lateral one, the longitudinal ODD O^{lon} (5) depends on the lateral variable κ through the inequality (3). The following proposition provides an under-approximation for the longitudinal CIS.

Proposition 1. *Assume that for all admissible curvature values $\kappa \in [\kappa_{min}, \kappa_{max}]$ there exists a non-empty under-approximation of the maximal CIS of the longitudinal dynamics (4) within (5), and consider the partitioning of the curvature range into N^{lon} equally sized segments $[\kappa_n, \kappa_{n+1})$, where $\kappa_n = \kappa_{min} + n(\kappa_{max} - \kappa_{min})/N^{lon}$ for $n \in \{1, \dots, N^{lon} - 1\}$. Let $\hat{O}_n^{lon}$ denote the version of (5) obtained for $\kappa_{min} = \kappa_{max} = \kappa_n$. Then, the family of pairs $(\Sigma^{lon}, \hat{O}_n^{lon})$, $n \in \{1, \dots, N^{lon} - 1\}$, constitutes a valid under-approximation for the maximal longitudinal CIS.*

Proof. As $\kappa_{n+1} > \kappa_n$, we have $\sqrt{g\mu/\kappa_{n+1}} < \sqrt{g\mu/\kappa_n}$. Thus, the maximal CIS of (4) within $\hat{O}_{n+1}^{lon}$ is a subset of the maximal CIS of (4) within $\hat{O}_n^{lon}$. Thus, with the segmentation by tangent planes $\hat{O}_n^{lon}$, an under-approximation of the maximal CIS for every admissible curvature $\kappa \in [\kappa_n, \kappa_{n+1})$ is given by the maximal CIS of (4) within $\hat{O}_{n+1}^{lon}$. Then, the conjunction of all corresponding maximal CIS of (4) within the tangent planes $\hat{O}_n^{lon}$ is an under-approximation for the maximal longitudinal CIS. □

In the lateral analytical specification, the state v_t of the longitudinal model appears in the system matrix (6). The next proposition provides an under-approximation for the lateral CIS.

Proposition 2. *Assume that for all admissible velocity values $v \in [v_{min}, v_{max}]$, $v_{min} > 0$ there exists a non-empty under-approximation of the maximal CIS of the lateral dynamics (6) within (7), and consider the partitioning of the velocity range into N^{lat} equally sized segments $[v_n, v_{n+1})$,*

[2] The approach can easily be used for control systems with additional subsystems, as discussed later in the paper.

where $v_n = v_{min} + n(v_{max} - v_{min})/N^{lat}$ *for* $n \in \{1, \dots, N^{lat} - 1\}$. *Let* $\hat{O}_n^{lat}$ *denote the version of* (7) *obtained for* $v_{min} = v_{max} = v_n$. *Then, the family of pairs* $(\Sigma^{lat}, \hat{O}_n^{lat})$, $n \in \{1, \dots, N^{lon} - 1\}$, *constitutes a valid under-approximation for the maximal lateral CIS.*

Proof. By applying a state transformation

$$T(v_t) = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{v_t t_s} \end{bmatrix}$$

to the state of (6), the influence of v_t can be isolated only to the input matrices in a linear multiplicative manner, and the state transition matrix A^{lat} becomes constant. Thus, for any two velocities $v_n < v_{n+1}$, the backward reachable set of the dynamics with v_{n+1} is contained within the corresponding set for v_n. Therefore, the maximal CIS for v_{max} is an under-approximation of every maximal CIS of a model (6) for a velocity $v_t \in [v_{min}, v_{max}]$. Following a similar argumentation, the maximal CIS for v_{n+1} is an under-approximation of any maximal CIS of (6) for a velocity $v_t \in [v_n, v_{n+1})$. Then, the conjunction of all corresponding maximal CIS of (6) within the tangent planes $\hat{O}_n^{lat}$ is an under-approximation for the maximal lateral CIS. □

Note that while the maximal CIS for v_{max} could be used for the whole velocity range, it would be overly conservative. Proposition 2 provides a fine-grained piecewise under-approximation obtained by computing the CISs of N^{lat} models. This leads to the following theorem, which ensures that the decoupled verification of the controllers guarantees the overall system properties.

Theorem 1. *Assume that there exists a non-empty under-approximation of the CIS of the exactly discretized:*

- *longitudinal model* (4) *within the ODD* (5) *for all admissible curvature values* $\kappa \in [\kappa_{min}, \kappa_{max}]$;
- *lateral model* (6) *within the ODD* (7) *for all admissible velocity values* $v \in [v_{min}, v_{max}]$.

Then, the decoupled verification of the longitudinal and lateral controllers – each with its respective under-approximated CIS – ensures that the overall continuous model (1) *exactly discretized with sampling period* t_s *satisfies* (2) *and* (3).

Proof. The inequalities (2) and (3) represent the conjunction of (5) and (7). The conjunction of the dynamical models (4) and (6) represents the exactly discretized version of the continuous model (1) with sampling period t_s. Propositions 1 and 2 establish sound under-approximations of the CIS for the longitudinal dynamics and the CIS for the lateral dynamics, respectively. Consequently, the conjunction of these two invariant sets forms a subset of the maximal CIS of (1) within (2) and (3). Since both verification processes are based on independent conditions, and given that (5) and (7) only partially overlap in terms of the admissible sets for κ and v, there is no feedback loop that could lead to circular reasoning. Therefore, the decoupled verification of the longitudinal and lateral controllers, each with their respective CIS, ensures the overall system properties described by (1)–(3). □

In the following section, we compute the proposed separate safe sets for compositional verification.

4.4. Stage 3: Verification

For verifying the safe operation of each subsystem (.), where $(.) \in \{lon, lat\}$, with respect to its analytical specification and ODD, CoCoSaFe uses set invariance. For control systems, this means finding a control signal that is able to render a set invariant, i.e., a controlled invariant set. If all control signals produced by the controller yield following states within the safe set $S^{(.)}$, the controller can be certified as safe with respect to the analytical specification and the operation set. Thus, once such a safe set is obtained, checking safety over (in)finite simulation traces over time is effectively transformed into a set containment problem (denoted by the dashed area of Stage 3 in Fig. 2). Practically, set containment at code level is accomplished by a state-of-the-art BMC for traditional controllers, or a three-step procedure employing a BMC and a DNN checker for neural-network-based controllers. We first turn to obtaining the safe sets.

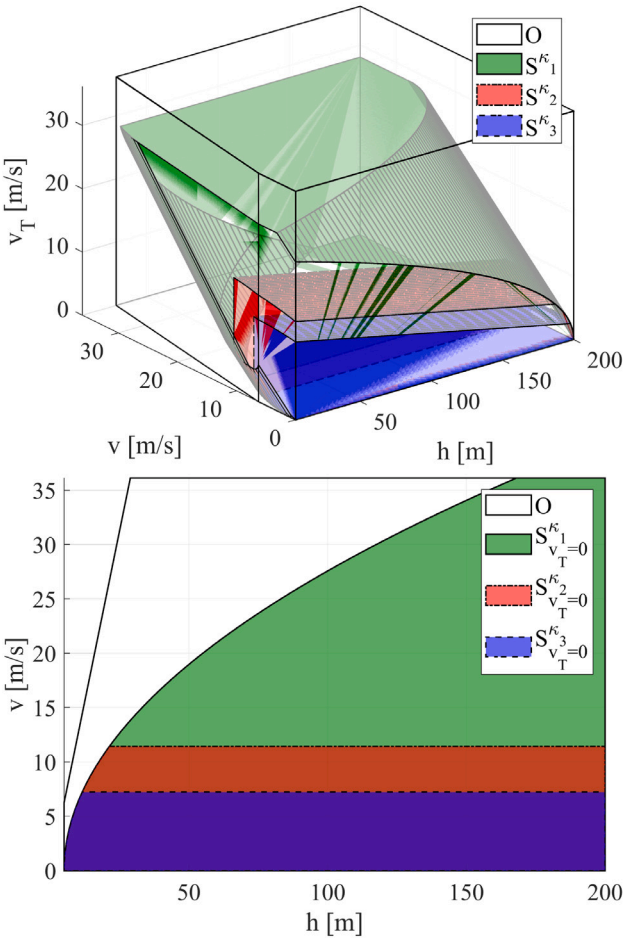

Fig. 5. Maximal longitudinal safe sets for $\kappa_1 = 0.006, \kappa_2 = 0.06$, and $\kappa_3 = 0.15$ in 3D (a) and their 2D slices for $v_T = 0$.

4.4.1. Computing safe sets

To obtain the safe set, we employ invariance properties, where the goal is to avoid unsafe states in $O^{(.)}$ at all times. For that, we utilize the maximal robust CIS, i.e., the set of all states inside of $O^{(.)}$ for which there exists a controller that can guarantee that the future trajectory remains safe under admissible worst-case values of a_T. We compute the CIS (or an under-approximation thereof) for a subsystem as a polytope $S^{(.)}$ represented by a finite number of inequalities N_S with corresponding matrices A_x^c and B_x^c, i.e.,

$$S^{(.)} = \{x | A_x^c x \leq B_x^c\}, A_x^c \in \mathbb{R}^{N_S \times 3}, B_x^c \in \mathbb{R}^{N_S}. \tag{8}$$

For any state $x_t^{(.)} \in S^{(.)}$ at time t, there exists at least one admissible control action with $u_t^{(.)}$, such that the following state $x_{t+1}^{(.)}$ according to the analytical specification remains within $S^{(.)}$. CoCoSaFe is compatible with any off-the-shelf CIS computation method which provides an under-approximation of the actual robust CIS of the analytical specifications.

Longitudinally, (4) is a discrete-time linear system with input vector a_t and disturbance $a_{T,t}$, and the maximal robust CIS is computed

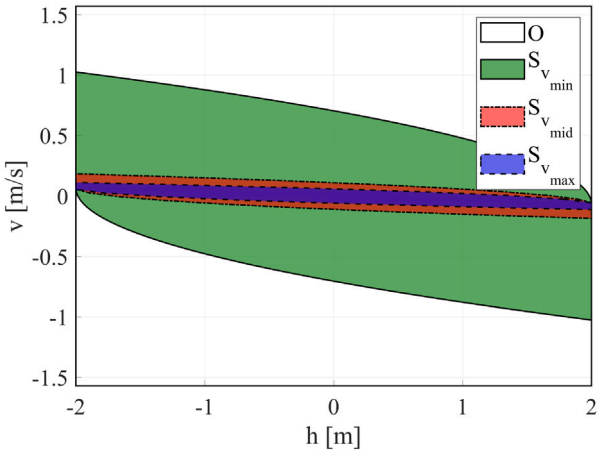

Fig. 6. Lateral CIS' for $v_{min} + v_s, v_{mid} = 0.5(v_{min} + v_{max})$, and v_{max}.

for each of the N^{lon} ODDs using the fixed-point algorithm proposed in Herceg et al. (2013). Fig. 5 illustrates the maximal longitudinal safe sets for different values of κ and its 2-dimensional slices for $v_T = 0$ using the parameters of the case study (Section 5.2.3). Note that the volume of the sets decreases significantly with a growing curvature. This approximation of the maximal robust CIS with the analytical specification (4) for the ODD (5) is used for verification.

Analogously, the robust maximal CIS for the lateral discrete-time linear system (6) with disturbance $\theta_{r,t}$ is computed using the methods from Herceg et al. (2013). Fig. 6 shows approximate lateral sets for three admissible longitudinal velocity values and parameters as introduced in Section 5.2.3. Note that the volume of the sets decreases significantly with increasing velocity. With the safe sets now computed, the next step is to verify the controller implementations for each subsystem.

Remark 2. For simplicity, we assumed only a single step "preview" for the target vehicle acceleration and the lane orientation in this paper. However, most state-of-the-art automated driving solutions employ previews for these signals over multiple steps ahead in time obtained by sensors to decrease conservatism in computing the following control action and prevent excessively cautious behavior. For example, the longitudinal control signal a_t can be picked depending on $a_{T,t}, a_{T,t+1}, \dots, a_{T,t+k}$. Note that a system with such a multi-step preview can be converted to a standard linear system by augmenting the state space with additional states corresponding to the preview of the external signal $a_{T,t}, a_{T,t+1}, \dots, a_{T,t+k^{lon}}$. Similarly, an extended system can also be obtained for the lateral subsystem with a preview of the external signal $\theta_{r,t}, \theta_{r,t+1}, \dots, \theta_{r,t+k^{lat}}$.

4.4.2. Verifying a controller implementation

The decoupled analytical specifications for ACC and LKA do not interfere with each other except for the effect of the longitudinal velocity on the lateral behavior and the curvature on the longitudinal behavior. These interactions were considered when computing the safe sets, which ensures that subsystems' interactions do not violate their respective assumptions and guarantees. Consequently, all admissible values of the lateral variable κ_t are considered while verifying the longitudinal controller. Note that a variable κ_t implies a variable corresponding safe set for verification (Fig. 5). Analogously, all admissible values of the longitudinal velocity v_t are considered during lateral controller verification (implying a variable corresponding lateral safe set). As shown in Section 4.3.3, if both the longitudinal and the lateral controllers pass verification with respect to the corresponding safe sets individually, their compositional behavior is also safe.

```
#include <assert.h>
#define n m p_min p_max ...
double Control(double state[n], double p[m]){...}

double x[n]; double p[m];

bool IsInSafeSet(double x[n], double p[m]){...}

bool IsInNextSafeSet(double x[n], double u,
                     double p[m]){...}

void Generate(double x[n], double p[m]){
    const double s = __VERIFIER_assign_random();
    __VERIFIER_assume(!isNaN(s));
    x[0]=s;
    ... //same for all x and p
    __VERIFIER_assume(IsInSafeSet(x,p));
}

void harness() {
    Generate(x,p);
    double u = Control(x,p);
    assert(IsInNextSafeSet(x,u,p));
}
```

Fig. 7. Verification harness for the function Control using a BMC.

An individual controller implementation can be deemed safe when it operates only within the bounded domain of the CIS for the analytical specification. For any possible state $x^{(\cdot)}$ in the safe set $S^{(\cdot)}$, the control output $u^{(\cdot)}$ of the to-be-verified controller is checked if it produces a following state, which is also inside of $S^{(\cdot)}$. As (4) and (6) are linear dynamical systems and $S^{(\cdot)}$ is a polytope, the following state is within the CIS if

$$A_x^c \begin{bmatrix} A^{(\cdot)} & B^{(\cdot)} \end{bmatrix} \begin{bmatrix} x^{(\cdot)} \\ u^{(\cdot)} \end{bmatrix} \leq B_x^c \tag{9}$$

holds. A bounded model checker is utilized for this. By using a BMC the code is also implicitly checked for common programming and security errors like integer overflows, out of bound array access, illegal pointer de-references, occurrence and treatment of exceptions, the presence of undefined behavior etc., in addition to checking safety. By using an invariant set to evaluate the safety of the controller, our approach requires only a single step look-ahead in time. Thus, the bounded roll-out by the BMC is primarily related to loops in the code. Thus, the BMC bound needs to be sufficiently large for the specific implementation to ensure that all safety-relevant behaviors are captured. As we use an underestimation of the actual CIS, the proposed verification procedure is sound, as discussed in Section 4.3.2. If the computed CIS is exact, the procedure is also complete.

An exemplary verification harness for verifying the function `Control` with respect to a safe set with state dimension `n` and a parameter vector of size `m` using a BMC is shown in Fig. 7. A state `x` and parameters `p` are checked if they are contained in the safe set according to (8) in function `IsInSafeSet`, or in the following state given a control input `u` according to (9) in the function `IsInNextSafeSet`, respectively. Note that the dependence of the check functions on parameters `p` results from the employed approximations of the maximal CIS depending on κ_t and v_t for the longitudinal and the lateral subsystem, respectively, as described in Section 4.4.1. The state variables `x` and parameters `p` are chosen randomly by the BMC in the function `Generate` and constrained to their feasible bounds `p_min` `p_max` and the safe set using `assume` statements and `IsInSafeSet`. Based on these variables, the controller produces an output `u`. Then, the safety

properties for the following state are checked by `IsInNextSafeSet` using an `assert` statement.

Remark 3. If the to-be-verified controller contains internal states, these have to be made accessible for modification by the BMC within the admissible bounds. A possible way to achieve this is to augment the state of the analytical specification and to consider them already in the safe set computation. This comes at the price of an increased CIS computation complexity, but is beneficial for verification with the BMC. The approach is showcased in Section 5, where the proportional integral controller has an internal state for error integration.

4.4.3. Verification decomposition for large DNN controllers

As BMC enumerates possible branches during program state-space exploration, it is not guaranteed that verifying a neural network will terminate in a reasonable amount of time. A verification timeout is likely even for moderately sized neural networks, as shown in the case study. To mitigate this limitation, we propose a three-step decomposition approach. First, the DNN integration is checked for a representative set of states within the safe set. Second, the original DNN is replaced by a simpler DNN with the same inputs and outputs, and BMC is used to check that all code required for operational deployment (DNN model loading, DNN inference, etc.) works as expected. Finally, the safety of the actual DNN is checked using an off-the-shelf DNN verifier. The overall process is complete with respect to the maximal safe set of the formal model and ODD under the assumption that the BMC can discover all deployment code-related flaws for the chosen states (Step 1 & 2), and that the safety check of the DNN by the DNN verifier itself is complete (Step 3).

In the first step, a finite set of individual states $x_t^{(\cdot)}$ in $S^{(\cdot)}$ is chosen to evaluate the DNN controller, focusing specifically on its deployment code. Heuristic criteria for achieving complete test coverage are particularly effective for selecting these states, as they aim to execute a significant portion of the code at least once, including error handling paths and exceptional cases. Then, the controller's output is checked to yield a following state that remains in $S^{(\cdot)}$ using a BMC for each of the selected states. This assures that the input–output behavior of the overall controller, including the supporting code, performs as expected before performing the DNN verification.

In the second step, we take inspiration from Elboher et al. (2023) that proposes to verify a large DNN by initially reducing it to a simpler, smaller DNN for verification (abstraction), and iteratively making it more complex as needed (refinement). Therefore, we replace the original DNN with a simpler DNN with the same inputs and outputs, and use BMC to check that all code required for operational deployment (DNN model loading, DNN inference, etc.) works as expected. For our purposes, we assume that an abstraction approach was used to obtain the smaller DNN, and that the supporting code needed for the original DNN is being fully executed for the deployment of the smaller DNN. This ensures that the supporting code is checked for implementation flaws. In this step, the controller operation is not required to satisfy (9), since the smaller DNN model might not fulfill the safety specification.

As a final third step, we use an off-the-shelf DNN verifier to check the safety of the actual DNN. In addition to a neural network, a verification query to the verifier includes the property to be checked — for any point in $S^{(\cdot)}$, the DNN has to produce a controller output that satisfies (9). The property has to be given in the form of linear or nonlinear constraints on the network's inputs and outputs, which is readily the case for (8) and (9). The verification problem thus reduces to finding an assignment of DNN input values that satisfies all the constraints simultaneously, or confirming that no such assignment exists. Only if the verification is successful in all three steps, the safety specification is considered as satisfied.

Table 1
Overview of controllers for empirical evaluation.

Controller type	Abbr.	Func.	Code lines	Parameters
Switching Proportional	SPC	ACC	10	$k_p = 3$
Non-Linear	NLC	ACC	15	$a_{com} = 1.5$ m/s^2
Model-predictive	MPC	ACC	5521	$N = 5$
Neural-network	NNC	ACC	1672	151 parameters
Proportional-integral	LPC	LKA	8	poles 0.3, 0.4, 0.5
Non-linear Geometric	LGC	LKA	12	$l = 2.8$ m, $k = 2$
Model-predictive	LMC	LKA	1521	$N = 3$, $w_r = 0.1$

5. Empirical evaluation

5.1. Evaluation methodology

We carried out extensive experiments to evaluate the safety and correctness of ADAS controller implementations with respect to formal specifications within common real-world ODDs. Following Fig. 2, the formalization step of the overall specification and ODD for automated driving in a lane with front targets was carried out in Section 4.2. Then, the detailed decomposition process into separate specifications and ODDs for the keep distance and keep lane behaviors was described in Section 4.3. Finally, the verification process for the two subsystems operating concurrently was detailed in Section 4.4, where we described how the compositional approach ensures the integrated system's safety based on separate safe sets and model checking.

To this end, and to evaluate the generalizability of our approach, we performed case studies involving the application of CoCoSaFe to two qualitatively different classes of ADAS controllers (adaptive cruise and lane keeping), and for each class we considered controllers that employ significantly different control techniques which are widely used in the automotive domain. The key features of the controllers examined in the two categories are summarized in Table 1. This includes four different adaptive cruise, one of which was neural-network-based, and three lane-keeping assist controllers, one of which had an internal state. The details of these controllers, all of which are taken from literature, are presented in Section 5.2.3. Each of these controllers was implemented in C++. All verification experiments used:

- a sampling time of $t_s = 0.2\,s$ for the analytical specifications;
- CBMC 5.95.1 (Clarke et al., 2004) as a bounded model checker with a timeout of 1 h and a loop unwinding limit of 1000. Accordingly, `__VERIFIER_assume` is replaced by `__CPROVER_assume` and `__VERIFIER_assign_random` by `nondet_double`, respectively, in the verification harness (Fig. 7);
- a standard workstation with an Intel Core i7-11850H CPU with 64 GB DDR4 RAM.

Regarding the decomposition approach for neural network controllers: we first check the actual DNN for two individual states, the supporting code is checked using the BMC with a simplified neural network with the same input and output interface, but only 3 neurons with random parameters in the hidden layer. Furthermore, Marabou (Katz et al., 2019) was used as a DNN verifier to check the actual DNN implementing the neural-network-based controller.

5.2. Experimental results

In this section, CoCoSaFe as presented in Section 4 is evaluated in case studies with different automated driving controllers for adaptive cruise control and lane keeping.

Table 2
Parameters used in the case study.

Parameter	Value	Description
a_{max}	2 m/s^2	Maximum ego acceleration
a_{min}	−4 m/s^2	Minimum ego acceleration
$a_{T,max}$	1 m/s^2	Maximum target acceleration
$a_{T,min}$	−2 m/s^2	Minimum target acceleration
$v_t, v_{T,t}$	[1, 130] km/h	Range of velocities
h_{max}	200 m	Maximum headway
h_{min}	5 m	Minimum headway
$t_{h_{min}}$	0.8 s	Minimum time headway
d_{min}	−2 m	Minimum lateral distance
d_{max}	2 m	Maximum lateral distance
θ_{max}	$\pi/2$ rad	Maximum angle
θ_{min}	$-\pi/2$ rad	Minimum angle
κ_{min}	−0.15 m^{-1}	Minimum curvature
κ_{max}	0.15 m^{-1}	Maximum curvature
$\theta_{r,max}$	0.1 rad	Maximum road angle
$\theta_{r,min}$	−0.1 rad	Minimum road angle

5.2.1. Stage 1: Formalization

The first stage of CoCoSaFe requires defining the parameters necessary for the overall formal specification, shown in Table 2. Since driver parameters v_d and t_{h_d} can take integer values in their respective ranges, the longitudinal controller verification can be performed individually for each of the possible combinations in parallel. For simplicity, we consider a fixed pair of a desired ego velocity $v_d = 130$ km/h and a desired time headway $t_{h_d} = 1.8$ s in the following.

5.2.2. Stage 2: Decomposition

Using the AGC corresponding to (3) with the coefficient of friction $\mu = 0.8$ and the gravitational constant $g = 9.81$ m/s^2, we obtained separate analytical specifications for the longitudinal and the lateral subsystem. The analytical specification (4) and (5) of ACC can be easily encoded by means of the velocity of the ego vehicle v, the velocity of the target object v_T and the headway h, all contained in the longitudinal state x^{lon}. Similarly, the analytical specification (6) and (7) can be readily encoded by means of the ego vehicle lateral deviation d and the steering angle θ, both contained in the lateral state x^{lat}.

5.2.3. Stage 3: Verification

Finally, in the third stage of CoCoSaFe, we compute the safe sets to perform verification. As described in Section 4.4.1, we computed $N^{lon} = 10$ longitudinal safe sets S_n^{lon}, where $n \in \{1, \dots, N^{lon}\}$, for the longitudinal analytical specification over the relevant range of the curvature $\kappa \in [\kappa_{min}, \kappa_{max}]$. The safe sets are captured by an overall number of inequalities ranging from 79 for κ_{min} to 546 for κ_{max}. Computing a safe set took between 0.2 min for κ_{min} and 6 min for κ_{max}. This procedure is required to be executed only once and the resulting safe sets are reused for evaluating the safety of all ACC controllers. The target acceleration $a_{T,t}$ and the curvature κ_t are chosen freely within their respective bounds for the longitudinal verification by the BMC using `assume` statements. The individual states to check the integration of the DNN are $x_t \in \{[0, 0, 0], [15, 5, 5]\}$.

Similarly, we computed $N^{lat} = 13$ lateral safe sets S_n^{lat}, where $n \in \{1, \dots, N^{lat}\}$, for the analytical lateral specification with longitudinal velocities $v_t \in [v_{min}, v_{max}]$, as described in Section 4.4.1. The safe sets are captured by an overall number of inequalities ranging from 8 for v_{max} to 74 for v_{min}. The calculation of a single safe set for the analytical specification using the nominal state x^{lat} required roughly 30 s. For the augmented state $\tilde{x}^{lat}$ with an integrator, the computation of a single safe set took approximately 1 min. This procedure is required to be executed only once and the resulting safe sets are reused for evaluating the safety of all lateral controllers. Both the longitudinal velocity v_t and the orientation of the road $\theta_{r,t}$ are freely chosen within their limits for lateral verification by the BMC using `assume` statements.

Adaptive cruise control results. The following four common classes of adaptive cruise controllers, which are widely used for automated driving, are considered for verification:

1. A Switching Proportional Controller (SPC) with the gain $k_P = 3$ and

   $$a_t = k_P(v_t - \min(v_d, h_t/t_{h_d})),$$

   where the min-expression takes care of the time gap and adapt speed modes.
2. A NonLinear Controller (NLC) known as the Intelligent Driver Model (Treiber et al., 2000):

   $$a_t = a_{max}\left(1 - \left(\frac{v_t}{v_d}\right)^{\delta} - \left(\frac{d(x_t^{lon})}{d_{T,t}}\right)^2\right),$$
   $$d(x_t^{lon}) = h_t + v_t t_{h_d} + \frac{v_t(v_t - v_{T,t})}{2\sqrt{a_{max}a_{com}}},$$

   where $d_{T,t} = 1.8v_t$ is the desired distance between the two vehicles, which is around half of the current ego vehicle's velocity v_t in km/h (the recommended minimum distance according to German traffic rules) and $a_{com} = 1.5$ m/s^2 is the absolute value of the comfortable acceleration.
3. A Model Predictive Controller (MPC) (Naus et al., 2010) using the model (4) with an optimization horizon of $N = 5$ steps. Let t denote the current time step and $\tilde{t}$ the optimization horizon time. The acceleration of the target object is initially $a_{T,\tilde{t}=0} = a_{T,t}$ and is assumed to be $a_{T,\tilde{t}} = 0$ for the remaining optimization horizon with $\tilde{t} > 0$. Note that a preview of the acceleration of the target object over the whole planning horizon can be considered using the approach outlined in Remark 2. With the initial state x_t^{lon}, the following quadratic program is solved at each state in a receding horizon manner:

   $$\min_{a_{\tilde{t}}} \sum_{\tilde{t}=0}^{N} (\|v_{\tilde{t}} - \min(v_d, h_{\tilde{t}}/t_{h_d})\| + \|a_{\tilde{t}}\|),$$
   $$\text{s.t. } \forall \tilde{t} \in [0, N], (4), x_{\tilde{t}} \in O_x^{lon}; a_{\tilde{t}} \in O_u^{lon};$$
   $$\forall \tilde{t} \in [1, N], a_{T,\tilde{t}} = 0;$$
   $$x_{\tilde{t}}^{lon} = x_t^{lon}, a_{T,0} = a_{T,t}.$$

   The controller is implemented using the Multi-Parametric Toolbox (Herceg et al., 2013). An explicit solution of the optimal control problem comprising 348 state feedback controllers over the relevant ODD is exported to C.
4. A Neural Network Controller (NNC) (Zhu et al., 2020), which combines imitation learning of recorded demonstrations and optimizing a reward function incorporating safety, efficiency, and comfort metrics to maximize cumulative rewards through simulation trials. Deep Deterministic Policy Gradient (DDPG) is utilized to learn an actor network together with a critic network. We focus on verifying the actor with an input x_t and an output a_t. The actor has one hidden layer with 30 neurons. For all layers, the Rectified Linear Unit (ReLU) activation function was used.

Table 3 shows results from the development-time verification of the longitudinal controllers. The MPC was the only one with a successfully verified safety, while the SPC and the NLC were falsified. This is largely due to the fact that the SPC only considers the current headway h_t and the velocity of the ego vehicle v_t, the NLC additionally the velocity of the target vehicle $v_{T,t}$, and only the MPC the full state information of the model including the acceleration of the target $a_{T,t}$ as well as the analytical specification and the ODD explicitly.

Simply using BMC on the NNC timed out for all safe sets. A similar result was reported in Scheibler et al. (2015) for small DNNs controlling a cart pole system, presumably caused by the nonlinearity

Table 3
Longitudinal controller code lines and falsification/verification times.

Controller type	Code lines	Falsification time [min]	Verification time [min]
SPC	10	3.0	–
NLC	15	5.1	–
MPC	5521	–	6.1
NNC	1672	timeout	timeout
NNC*	1672	21.4	–

NNC: times using BMC only for verification
NNC*: cumulative time for decomposition-based verification.

and noninvertability of the DNN. However, using the decomposition proposed in Section 4.4.3, we falsified the NNC by a counterexample. The bounded model checking of the supporting code with the auxiliary neural network was carried out in 10.4 min, and 5 min for the two selected individual states for the actual DNN. The verification of the actual DNN using Marabou took only 1 min.

All falsifying counterexamples denote an insufficient ego vehicle deceleration, while the target object decelerates with a_{min} and the time headway is not large enough. While the PC decelerates slightly in this case, the NC and the NNC decelerate with nearly a_{min}.

Lane keeping assist results. We investigated three classes of controllers for meeting the lane keeping requirements:

(1) A Lateral Proportional-integral Controller (LPC), given as

$$\kappa_t = K^T \tilde{x}_t^{lat}$$

where $\tilde{x}^{lat}$ denotes the state extended by the accumulated distance error e, i.e., $\tilde{x}^{lat} = [x^T, e]^T$ and assuming that the dynamics (6) augmented by an integrator for the lateral deviation, i.e., $e_{t+1} = e_t + d_t$. The controller is designed by placing the poles of the closed-loop system in 0.3, 0.4, and 0.5 using the method from J. Kautsky and Dooren (1985). The control signal is saturated additionally to account for the actuation limits (7);

(2) A Lateral Geometric Controller (LGC), also known as Stanley controller (Hoffmann et al., 2007), which focuses on minimizing the lateral distance and the heading errors between the vehicle and a desired path. An adapted version of the controller in terms of the vehicle's curvature is given as:

$$\kappa_t = -\frac{1}{l}\tan\left[\theta_t - \theta_{r,t} + \tan^{-1}\left(\frac{kd_t}{v_t}\right)\right],$$

where $l = 2.8\,m$ is the wheelbase of the vehicle and $k = 2$ is a tuning parameter. Similarly, the control signal is saturated to account for the actuation limits (7);

(3) A Lateral Model-predictive Controller (LMC) for the model (6) and the optimization horizon with $N = 3$ samples. Let t denote the current time step and $\tilde{t}$ the optimization horizon time. The orientation of the road and the longitudinal velocity are assumed to be given as preview vectors over the optimization horizon, i.e., $\theta_{r,\tilde{t}}|_{[t,t+N-1]}$ and $v_{\tilde{t}}|_{[t,t+N-1]}$, respectively. To decrease the complexity of controller computation, we assume that both preview vectors are constant over the optimization horizon, i.e., $\forall\tilde{t} \in [0, N-1], \theta_{r,\tilde{t}} = \theta_{r,t}, v_{\tilde{t}} = v_t$. Note that a preview of both vectors over the whole planning horizon can be considered using the approach outlined in Remark 2. With the initial state x_t^{lat} and the weight $w_r = 0.1$, the following quadratic program is solved at each state in a receding horizon manner:

$$\begin{aligned}
&\min_{\kappa_{\tilde{t}}} \sum_{\tilde{t}=0}^{N-1} ((x_{\tilde{t}}^{lat})^T x_{\tilde{t}}^{lat} + w_r \kappa_{\tilde{t}}^2),\\
&\text{s.t. } \forall\tilde{t} \in [0, N], v_{\tilde{t}} = v_t, (6), x_{\tilde{t}}^{lat} \in O_x^{lat}, \kappa_{\tilde{t}} \in O_u^{lat};\\
&\quad \forall\tilde{t} \in [0, N-1], \theta_{r,\tilde{t}} = \theta_{r,t}, \theta_{r,t} \in O_u^{lat};\\
&\quad x_{\tilde{t}}^{lat} = x_t^{lat}.
\end{aligned}$$

Table 4
Lateral controller code lines and falsification/verification times.

Controller type	Code lines	Falsification time [min]	Verification time [min]
LPC	8	2.5	–
LGC	12	4.9	–
LMC	1521	–	7.8

The controller for the Linear Parameter Varying (LPV) system (6) is implemented using the method proposed in Besselmann and Löfberg (2012). The explicit solution of the optimal control problem comprises 144 state feedback controllers over the relevant ODD and is exported to C.

Table 4 summarizes the verification/falsification results for the above three lateral controllers. Similar to the longitudinal case, the model-predictive LMC is the only one to pass the verification, while the LPC and the LGC were falsified. This is largely due to the fact that the LPC and the LGC only consider the current state x_t^{lat}, and only the LMC additionally the orientation of the road $\theta_{r,t}$ as well as the analytical specification and the ODD explicitly. The LGC with an increased tuning parameter $k = 4$ also passes the verification. Note that the LPC was falsified using safe sets with the augmented state $\tilde{x}_t^{lat}$, as outlined in Remark 3.

All falsifying counterexamples denote an insufficient ego vehicle curvature provided by the controller, while driving at relatively high speed and with a relatively high initial orientation difference to the reference orientation.

5.3. Threats to validity

The following potential validity threats highlight areas where the CoCoSaFe study's results might be vulnerable to misinterpretation or might not fully capture the intended safety guarantees across diverse driving scenarios and controller types.

Internal validity threats may be due to the conditional nature of the evidence obtained from CoCoSaFe, which only demonstrates that safety is maintained within the invariant set of the analytical specification. This raises concerns about the validity of the conclusions regarding the controller's behavior in the field. To mitigate this threat, the analytical specifications were taken from literature (Rajamani, 2011) and they were validated against real-world data in Hoffmann et al. (2007), Menzel et al. (2018) to ensure that they capture relevant dynamics for ACC and LKA.

The specific implementation of the controllers also play a crucial role in influencing verification. Features such as templates, loops, and recursive functions can introduce additional challenges that may compromise the termination of the verification process in reasonable time. To alleviate this, the MISRA (Motor Industry Software Reliability Association) coding standard (Anon, 2023) was used to enhance code safety and quality.

Further, BMCs analyze the software's behavior only up to a predefined bound. Errors that occur beyond this loop unrolling limit might be missed. To address this, only finite loops were used in the code and the bound for the BMC was set to the maximum size of the for-loops present in the controllers.

Construct validity threats may occur in defining the ODD and parameter set for model checking. A trade-off between precision and complexity may lead to an inadequate representation of the system being verified. To reduce these threats, the ODD was chosen to encompass the maximal view range of common ADAS sensors and maximal velocity, acceleration, orientation and curvature operation ranges for ACC and LKA.

Tool limitations present another challenge, as the assumptions necessary for verification may not be feasible to incorporate into the

deployed code. To mitigate this issue, CoCoSaFe operates at the input–output boundaries of the controllers being verified, ensuring that these are accessible for specifying assumptions and assertions. Additionally, many verifiers are not robust against all possible flaws. For instance, floating-point errors have shown to be particularly difficult when it comes to ensuring the overall safety of DNNs (Jia and Rinard, 2021). To tackle this challenge, we combined tools with a proven track record in verifying code level including floating-point errors, such as CBMC (Clarke et al., 2004), as well as a dedicated neural network verifier, like Marabou (Katz et al., 2019).

Furthermore, depending on an analytical specification restricts the generalization of the verification outcomes. We emphasize that CoCoSaFe, like most formal verification methods, should be supplemented with data-driven testing to improve its applicability to complex behaviors that arise solely in real-world scenarios.

External validity threats may arise if our approach is not applicable to other ADAS controllers. To mitigate this threat, we evaluated the approach for two qualitatively different classes of ADAS controllers (cruise control and lane keeping), and for each class we considered controllers that use several distinct methods commonly employed in the automotive domain. Nevertheless, further case studies, for instance the application to other controller types or varied operational settings (such as urban driving or adverse weather conditions), are required to ensure that these threats are fully addressed. The particular assumptions and models applied might not be transferable to other areas, which could limit the framework's broader usability. Nevertheless, since similar subsystem decompositions used in this study are prevalent in fields such as aerospace (McRuer et al., 2014), we anticipate that CoCoSaFe can also be applied in that context.

6. Discussion

Using the CoCoSaFe framework presented in Section 4, we could achieve more realistic and, most importantly, a safe closed-loop behavior for many initially falsified controllers. The considered automated driving controllers were used to control traffic agents in an in-house simulation environment for automated driving development. The experimental results presented in the previous sections suggest that our approach offers several benefits beyond this application.

Independent Subsystem Analysis. Our compositional verification approach allows the analysis of each subsystem independently, which improves the scalability and feasibility of verification for large systems with multiple concurrently operating subsystems. The longitudinal and lateral subsystems can be verified in parallel, which reduces the overall verification time. Assuming that the analytical specification is valid, verified subsystems can be reused in different products without the need to re-verify them. This also enables incremental verification, where changes or updates to one subsystem do not necessitate re-verifying the entire system. This is particularly relevant for over-the-air updates in large and diverse vehicle fleets — if only the ACC is updated, only its verification and interaction with LKA needs re-evaluation, not the entire system. Verifying subsystems individually also supports isolating and debugging safety violations and might reduce maintenance efforts by focusing on specific subsystems.

Identifying Code-Level Safety Flaws. We were able to find safety flaws at code level for all controllers except for the model-predictive-based ones (Tables 3 and 4). These safety deficits may have remained uncovered with testing alone.

Explainable Falsifying Samples. The acquired falsifying samples correspond to driving scenarios that provide feedback to improve the controllers considered, either by deriving additional automated tests or by revisiting algorithmic solutions.

Enhancing System-Level Analysis. Applying formal verification even only to some software modules greatly supports the overall system-level analysis and design, e.g., by providing hints where a dedicated supervisor component might be required as a safety assurance measure. Note that in practice, the computed safe sets can be used as a supervisor (runtime monitor) to ensure the safe operation of a controller.

Identifying Gaps in Training Data. For neural network controllers, our approach allows falsifying examples to be used to indicate possible gaps in the collected training data or undesirable biases in the current training stage.

Feasible Pipeline integration. Our framework can be integrated to support verification within the ADAS development process. As the only manual modeling step is related to deriving the analytical specification and verifying controller implementations are possible in an automated manner within minutes, our solution can be included in Continuous Integration (CI) pipelines, where it is connected to consecutive versions of the to-be-deployed controller software.

Broad Applicability for ADAS. As falsification or verification was possible for all seven considered controller classes, the proposed framework is expected to be suitable for a broad range of controllers. The method is not limited to the analytical specifications (4) and (6). Although this paper uses a linear system for which computing a CIS is known to have polynomial complexity (Raković et al., 2007), the verification approach presented can be easily used for analytical specifications and operation sets, which allow computing a CIS. Obtaining the CIS is possible for many nonlinear systems, e.g., (Fiacchini et al., 2010). Even though our work focuses on safe keep distance and lane keeping of automated driving vehicles, its key ideas can be transferred to other CPS.

While the presented scheme has great potential for automated safety verification of many safety-critical controllers, several aspects of CoCoSaFe can be extended in future work.

Extension to Additional ADAS elements. While our current focus has been on the verification of controller code, by integrating other ADAS components we can evaluate how the controllers interact for various ADAS functionalities. This holistic approach not only strengthens the verification process but also addresses potential edge cases and interactions that may arise in real-world scenarios. Recent empirical studies have indicated that an ensemble failure predictor outperforms individual simulators in forecasting the failures of a digital twin Biagiola et al. (2024). Thus, a possible way to enhance generalization could be using an alternative, potentially non-analytical specification. Another alternative is to extend the approach to consider uncertainties in the system model, e.g., by using probabilistic (Calinescu et al., 2018; Gerasimou et al., 2021) and/or statistical model checking.

Specification complexity. Achieving a suitable analytical specification for controller verification requires balancing precision and complexity, with hybrid system formulations being a promising option (Wongpiromsarn et al., 2012). If the resulting safe set is highly complex, characterized by many defining inequalities, the verification process may take considerable time. One potential solution is to under-approximate this complex invariant set using simpler invariants, as suggested in Anevlavis et al. (2023), or by employing a lower-complexity parametric shape.

Using more complex dynamical system models as analytical specifications can render the computation of the CIS infeasible or make bounded model checking intractable. Even with linear models, the complexity of the analytical specification impacts computational effort and the size of the operational domain for verification. For nonlinear systems, some studies use convex approximations to reduce complexity (Fiacchini et al., 2010), while others leverage structural properties of polynomial systems to compute exact sets (Ben Sassi and Girard, 2012). However, since the maximal controlled invariant set of a nonlinear system is often non-convex, the resulting CIS can be conservative. Future work could explore whether other desired properties of closed-loop controller operation can be captured by invariants and verified using our approach, such as minimizing unnecessary fluctuations in control signals in camera-based systems when images are stable.

7. Conclusion

We introduced CoCoSaFe, a Compositional Code-level formal Safety verification Framework for automated driving controller implementations. CoCoSaFe relies on computing controlled invariant sets for analytical specifications of the closed-loop operation within the operational design domain. This enables a modular and parallelizable verification of concurrently operating subsystem controller implementations by utilizing a bounded model checker. Furthermore, by proposing a three-stage verification decomposition, we verify neural network-based controllers, for which off-the-shelf bounded model checkers have timed out.

We demonstrated the effectiveness of our method by an extensive case study with various types of traditional and neural-network-based controllers. The experimental results confirm that adaptive cruise and lane keeping controllers can be verified offline within a time frame of minutes on a regular computer, thus emphasizing the low computational overhead of the framework for cyber–physical systems.

In future work, we plan to apply CoCoSaFe to additional types of automotive controllers, e.g. those responsible for lane changing or interactive urban driving, as well as other application domains, such as mobile robotic platforms and manipulators. Furthermore, specification complexity and additional ADAS elements will be considered in our future work, as outlined in Section 6.

CRediT authorship contribution statement

Vladislav Nenchev: Writing – review & editing, Writing – original draft, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Conceptualization. **Calum Imrie:** Writing – review & editing, Writing – original draft, Visualization, Conceptualization. **Simos Gerasimou:** Writing – review & editing, Writing – original draft, Visualization, Methodology, Conceptualization. **Radu Calinescu:** Writing – review & editing, Writing – original draft, Visualization, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Vladislav Nenchev reports travel was provided by Lloyd's Register Foundation. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was partly supported by the Assuring Autonomy International Programme, a partnership between Lloyd's Register Foundation and the University of York, UK. Radu Calinescu's work was also funded by the EPSRC project EP/V026747/1 'UKRI Trustworthy Autonomous Systems Node in Resilience', and by the QUALIFICA project of the Institute for Software Engineering and Software Technology 'Jose María Troya Linero' at the University of Málaga (QUAL21-010 UMA). Vladislav Nenchev's work was also supported by the University of the Bundeswehr Munich, Germany. We thank the anonymous reviewers for their valuable feedback, which has contributed to the improvement of the paper.

Data availability

The authors do not have permission to share data.

References

Adelt, J., Gebker, J., Herber, P., 2024. Reusable formal models for concurrency and communication in custom real-time operating systems. Int. J. Softw. Tools Technol. Transf. 26, 229–245. http://dx.doi.org/10.1007/s10009-024-00743-4.

Alam, A., Gattami, A., Johansson, K.H., Tomlin, C.J., 2014. Guaranteeing safety for heavy duty vehicle platooning: Safe set computations and experimental evaluations. Control Eng. Pract. 24, 33–41.

Anevlavis, T., Liu, Z., Ozay, N., Tabuada, P., 2023. Controlled invariant sets: Implicit closed-form representations and applications. IEEE Trans. Autom. Control 1–16.

Anon, 2018. Intelligent transport systems - adaptive cruise control systems - performance requirements and test procedures. ISO Standard 15622:2018.

Anon, 2023. MISRA C++:2023 Guidelines for the Use of C++ in Critical Systems. Motor Industry Software Reliability Association (MISRA), Warwickshire, UK.

Behere, S., Törngren, M., 2016. A functional reference architecture for autonomous driving. Inf. Softw. Technol. 73, 136–150. http://dx.doi.org/10.1016/j.infsof.2015.12.008.

Behrmann, G., David, A., Larsen, K.G., Hakansson, J., Petterson, P., Yi, W., Hendriks, M., 2006. UPPAAL 4.0. In: Proceedings of the 3rd International Conference on the Quantitative Evaluation of Systems. QEST '06, IEEE Computer Society, USA, pp. 125–126. http://dx.doi.org/10.1109/QEST.2006.59.

Ben Sassi, M.A., Girard, A., 2012. Computation of polytopic invariants for polynomial dynamical systems using linear programming. Automatica 48 (12), 3114–3121. http://dx.doi.org/10.1016/j.automatica.2012.08.014.

Benvenuti, L., Ferrari, A., Mazzi, E., Vincentelli, A.L., 2008. Contract-based design for computation and verification of a closed-loop hybrid system. In: Proc, of 11th Int. Workshop on Hybrid Systems: Computation and Control. HSCC, pp. 58–71.

Besselmann, T., Löfberg, J., 2012. Explicit MPC for LPV systems: stability and optimality. IEEE Trans. Autom. Control.

Biagiola, M., Stocco, A., Riccio, V., Tonella, P., 2024. Two is better than one: digital siblings to improve autonomous driving testing. Empir. Softw. Eng. 29 (4), 72.

Blanchini, F., 1999. Set invariance in control. Automatica 35 (11), 1747–1767.

Calinescu, R., Češka, M., Gerasimou, S., Kwiatkowska, M., Paoletti, N., 2018. Efficient synthesis of robust models for stochastic systems. J. Syst. Softw. 143, 140–158.

Calinescu, R., Imrie, C., Mangal, R., Rodrigues, G.N., Pasareanu, C.S., Santana, M.A., Vázquez, G., 2024. Controller synthesis for autonomous systems with deep-learning perception components. IEEE Trans. Softw. Eng. 50 (6), 1374–1395. http://dx.doi.org/10.1109/TSE.2024.3385378.

Camilli, M., Mirandola, R., Scandurra, P., 2023. Enforcing resilience in cyber-physical systems via equilibrium verification at runtime. ACM Trans. Auton. Adapt. Syst. 18 (3).

Chaves, L., Bessa, I.V., Ismail, H., dos Santos Frutuoso, A.B., Cordeiro, L., de Lima Filho, E.B., 2018. Dsverifier-aided verification applied to attitude control software in unmanned aerial vehicles. IEEE Trans. Reliab. 67 (4), 1420–1441.

Chou, G., Sahin, Y.E., Yang, L., Rutledge, K.J., Nilsson, P., Ozay, N., 2018. Using control synthesis to generate corner cases: A case study on autonomous driving. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 37 (11), 2906–2917. http://dx.doi.org/10.1109/TCAD.2018.2858464.

Chrszon, P., Maurer, P., Saleip, G., Müller, S., Fischer, P.M., Gerndt, A., Felderer, M., 2023. Applicability of model checking for verifying spacecraft operational designs. In: ACM/IEEE 26th Int. Conf. on Model Driven Engineering Languages and Systems. MODELS, pp. 206–216.

Cimatti, A., Cristoforetti, L., Griggio, A., Tonetta, S., Corfini, S., Di Natale, M., Barrau, F., 2023. EVA: a tool for the compositional verification of AUTOSAR models. In: Sankaranarayanan, S., Sharygina, N. (Eds.), Tools and Algorithms for the Construction and Analysis of Systems. Springer Nature Switzerland, Cham, pp. 3–10.

Clarke, E., Kroening, D., Lerda, F., 2004. A tool for checking ANSI-C programs. In: Jensen, K., Podelski, A. (Eds.), Tools and Algorithms for the Construction and Analysis of Systems. TACAS 2004, In: Lecture Notes in Computer Science, vol. 2988, Springer, pp. 168–176.

Dawson, C., Gao, S., Fan, C., 2023. Safe control with learned certificates: A survey of neural Lyapunov, barrier, and contraction methods for robotics and control. IEEE Trans. Robot. 39 (3), 1749–1767. http://dx.doi.org/10.1109/TRO.2022.3232542.

DeCastro, J., Liebenwein, L., Vasile, C.-I., Tedrake, R., Karaman, S., Rus, D., 2020. Counterexample-guided safety contracts for autonomous driving. In: Algorithmic Foundations of Robotics XIII: Proceedings of the 13th Workshop on the Algorithmic Foundations of Robotics 13. Springer, pp. 939–955.

Durand, T., Fazekas, K., Weissenbacher, G., Zwirchmayr, J., 2021. Model checking AUTOSAR components with CBMC. In: 2021 Formal Methods in Computer Aided Design. FMCAD, IEEE, pp. 96–101.

Elboher, Y.Y., Cohen, E., Katz, G., 2023. On applying residual reasoning within neural network verification. Softw. Syst. Model. 1–16. http://dx.doi.org/10.1007/s10270-023-01138-w.

Fan, C., Qi, B., Mitra, S., Viswanathan, M., 2017. DryVR: Data-driven verification and compositional reasoning for automotive systems. In: Majumdar, R., Kunˇ cak, V. (Eds.), Computer Aided Verification. pp. 441–461.

Favrin, A., Nenchev, V., Cenedese, A., 2020. Learning to falsify automated driving vehicles with prior knowledge. IFAC-PapersOnLine http://dx.doi.org/10.1016/j.ifacol.2020.12.664, IFAC World Congress 2020 (IFAC'2020), Berlin.

Fiacchini, M., Alamo, T., Camacho, E., 2010. On the computation of convex robust control invariant sets for nonlinear systems. Automatica 46 (8), 1334–1338. http://dx.doi.org/10.1016/j.automatica.2010.05.007.

Fulton, N., Mitsch, S., Quesel, J.-D., Völp, M., Platzer, A., 2015. KeYmaera X: An axiomatic tactical theorem prover for hybrid systems. In: Automated Deduction-CADE-25: 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings 25. Springer, pp. 527–538.

Garrido, F., Resende, P., 2022. Review of decision-making and planning approaches in automated driving. IEEE Access 10, 100348–100366. http://dx.doi.org/10.1109/ACCESS.2022.3207759.

Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., Vechev, M., 2018. AI2: Safety and robustness certification of neural networks with abstract interpretation. In: 2018 IEEE Symposium on Security and Privacy. SP, pp. 3–18. http://dx.doi.org/10.1109/SP.2018.00058.

Gerasimou, S., Cámara, J., Calinescu, R., Alasmari, N., Alhwikem, F., Fang, X., 2021. Evolutionary-guided synthesis of verified pareto-optimal MDP policies. In: 36th IEEE/ACM International Conference on Automated Software Engineering. ASE, IEEE, pp. 842–853.

Giannakopoulou, D., Namjoshi, K.S., Păsăreanu, C.S., 2018. Compositional reasoning. In: Handbook of Model Checking. Springer International Publishing, Cham, pp. 345–383.

Grundt, D., Jurj, S.L., Hagemann, W., Kröger, P., Fränzle, M., 2022. Verification of sigmoidal artificial neural networks using iSAT. In: International Workshop on Symbolic-Numeric Methods for Reasoning About CPS and IoT. pp. 45–60. http://dx.doi.org/10.4204/EPTCS.361.6.

Herceg, M., Kvasnica, M., Jones, C.N., Morari, M., 2013. Multi-parametric toolbox 3.0. In: European Control Conference. ECC, pp. 502–510. http://dx.doi.org/10.23919/ECC.2013.6669862.

Hoffmann, G.M., Tomlin, C.J., Montemerlo, M., Thrun, S., 2007. Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing. In: Proc. of American Control Conference. ACC, pp. 2296–2301. http://dx.doi.org/10.1109/ACC.2007.4282788.

Huang, X., Kwiatkowska, M., Wang, S., Wu, M., 2017. Safety verification of deep neural networks. In: Majumdar, R., Kunčak, V. (Eds.), Computer Aided Verification. Springer Int. Publishing, Cham, pp. 3–29.

Ioannou, P., Chien, C., 1993. Autonomous intelligent cruise control. IEEE Trans. Veh. Technol. 42 (4), 657–672. http://dx.doi.org/10.1109/25.260745.

Ivanov, R., Jothimurugan, K., Hsu, S., Vaidya, S., Alur, R., Bastani, O., 2021. Compositional learning and verification of neural network controllers. ACM Trans. Embed. Comput. Syst. 20 (5s), http://dx.doi.org/10.1145/3477023.

J. Kautsky, N.K.N., Dooren, P.V., 1985. Robust pole assignment in linear state feedback. Internat. J. Control 41 (5), 1129–1155.

Jacumet, R., Rathgeber, C., Nenchev, V., 2023. Analytical safety bounds for trajectory following controllers in autonomous vehicles. In: Proc. of Int. Conf. on Control, Decision and Information Technologies. CoDIT, pp. 730–735.

Jia, K., Rinard, M., 2021. Exploiting verified neural networks via floating point numerical error. In: Drăgoi, C., Mukherjee, S., Namjoshi, K. (Eds.), Int. Static Analysis Symposium. Springer International Publishing, Cham, pp. 191–205.

Katz, G., Huang, D.A., Ibeling, D., Julian, K., Lazarus, C., Lim, R., Shah, P., Thakoor, S., Wu, H., Zeljić, A., Dill, D.L., Kochenderfer, M.J., Barrett, C., 2019. The marabou framework for verification and analysis of deep neural networks. In: Dillig, I., Tasiran, S. (Eds.), Computer Aided Verification. Springer International Publishing, Cham, pp. 443–452.

Kianfar, R., Falcone, P., Fredriksson, J., 2013. Safety verification of automated driving systems. IEEE Intell. Transp. Syst. Mag. 5 (4), 73–86.

Kojchev, S., Klintberg, E., Fredriksson, J., 2020. A safety monitoring concept for fully automated driving. In: 2020 IEEE 23rd International Conference on Intelligent Transportation Systems. ITSC, pp. 1–7. http://dx.doi.org/10.1109/ITSC45102.2020.9294307.

König, L., Heinzemann, C., Griggio, A., Klauck, M., Cimatti, A., Henze, F., Tonetta, S., Küperkoch, S., Fassbender, D., Hanselmann, M., 2024. Towards safe autonomous driving: Model checking a behavior planner during development. In: Finkbeiner, B., Kovács, L. (Eds.), Tools and Algorithms for the Construction and Analysis of Systems. Springer Nature Switzerland, Cham, pp. 44–65.

Lopez, D.M., Choi, S.W., Tran, H.-D., Johnson, T.T., 2023. NNV 2.0: The neural network verification tool. In: Enea, C., Lal, A. (Eds.), Computer Aided Verification. Springer Nature Switzerland, Cham, pp. 397–412.

Lygeros, J., Godbole, D.N., Sastry, S., 1996. A verified hybrid controller for automated vehicles. In: Proceedings of 35th IEEE Conference on Decision and Control, vol. 2, pp. 2289–2294.

Ma, Y., Sun, C., Chen, J., Cao, D., Xiong, L., 2022. Verification and validation methods for decision-making and planning of automated vehicles: A review. IEEE Trans. Intell. Veh. 7 (3), 480–498.

Marchetto, A., Scanniello, G., 2024. A rapid review on software vulnerabilities and embedded, cyber-physical, and IoT systems. In: Kadgien, R., Jedlitschka, A., Janes, A., Lenarduzzi, V., Li, X. (Eds.), Product-Focused Software Process Improvement. Springer Nature Switzerland, Cham, pp. 468–477.

McRuer, D.T., Graham, D., Ashkenas, I., 2014. Aircraft Dynamics and Automatic Control. Princeton University Press, Princeton.

Mehdipour, N., Althoff, M., Tebbens, R.D., Belta, C., 2023. Formal methods to comply with rules of the road in autonomous driving: State of the art and grand challenges. Automatica 152, 110692. http://dx.doi.org/10.1016/j.automatica.2022.110692.

Menzel, T., Bagschik, G., Maurer, M., 2018. Scenarios for development, test and validation of automated vehicles. In: 2018 IEEE Intelligent Vehicles Symposium. IV, pp. 1821–1827.

Naus, G., Ploeg, J., Van de Molengraft, M., Heemels, W., Steinbuch, M., 2010. Design and implementation of parameterized adaptive cruise control: An explicit model predictive control approach. Control Eng. Pract. 18 (8), 882–892. http://dx.doi.org/10.1016/j.conengprac.2010.03.012.

Nenchev, V., 2021. Automated behavior modeling for verifying safety-relevant modules. In: Proc. of IEEE Int. Conf. on Robotic Computing. IRC, pp. 92–95.

Nenchev, V., 2023. Model checking embedded adaptive cruise controllers. Robot. Auton. Syst. 167, 104488.

Nenchev, V., 2025. One stack, diverse vehicles: Checking safe portability of automated driving software. In: 2025 IEEE/SICE International Symposium on System Integration. SII, pp. 764–769. http://dx.doi.org/10.1109/SII59315.2025.10870905.

Nenchev, V., Imrie, C., Gerasimou, S., Calinescu, R., 2024. Code-level safety verification for automated driving: A case study. In: Formal Methods. FM'24, In: Lecture Notes in Computer Science (LNCS), vol. 14934, Springer, pp. 356–372.

Nilsson, P., Hussien, O., Balkan, A., Chen, Y., Ames, A.D., Grizzle, J.W., Ozay, N., Peng, H., Tabuada, P., 2016. Correct-by-construction adaptive cruise control: Two approaches. IEEE Trans. Control Syst. Technol. 24 (4), 1294–1307.

Păsăreanu, C.S., Gopinath, D., Yu, H., 2019. Compositional verification for autonomous systems with deep learning components. In: Safe, Autonomous and Intelligent Vehicles. Springer, pp. 187–197.

Paterson, C., Wu, H., Grese, J., Calinescu, R., Pasareanu, C.S., Barrett, C.W., 2021. DeepCert: Verification of Contextually Relevant Robustness for Neural Network Image Classifiers. In: Habli, I., Sujan, M., Bitsch, F. (Eds.), Computer Safety, Reliability, and Security - 40th International Conference, SAFECOMP 2021, York, UK, September 8-10, 2021, Proceedings. In: Lecture Notes in Computer Science, vol. 12852, Springer, pp. 3–17. http://dx.doi.org/10.1007/978-3-030-83903-1_5.

Pek, C., Manzinger, S., Koschi, M., Althoff, M., 2020. Using online verification to prevent autonomous vehicles from causing accidents. Nat. Mach. Intell. 2 (9), http://dx.doi.org/10.1038/s42256-020-0225-y.

Rajamani, R., 2011. Vehicle dynamics and control. In: Mechanical Engineering Series, Springer US.

Raković, S., Kerrigan, E., Mayne, D., Kouramas, K., 2007. Optimized robust control invariance for linear discrete-time systems: Theoretical foundations. Automatica 43 (5), 831–841. http://dx.doi.org/10.1016/j.automatica.2006.11.006.

Ruoss, A., Baader, M., Balunović, M., Vechev, M., 2021. Efficient certification of spatial robustness. In: Thirty-Fifth AAAI Conference on Artificial Intelligence. pp. 2504–2513.

Santa Cruz, U., Shoukry, Y., 2022. NNLander-VeriF: A neural network formal verification framework for vision-based autonomous aircraft landing. In: Deshmukh, J.V., Havelund, K., Perez, I. (Eds.), NASA Formal Methods. Springer International Publishing, Cham, pp. 213–230.

Scheibler, K., Winterer, L., Wimmer, R., Becker, B., 2015. Towards verification of artificial neural networks. In: Heinkel, U., Rößler, M., Kriesten, D. (Eds.), Proceedings of the 18th Workshop "Methoden Und Beschreibungssprachen Zur Modellierung Und Verifikation Von Schaltungen Und Systemen". MBMV, Technische Universität Chemnitz, Germany, Chemnitz, Germany, pp. 30–40.

Scholtes, M., Westhofen, L., Turner, L.R., Lotto, K., Schuldes, M., Weber, H., Wagener, N., Neurohr, C., Bollmann, M.H., Körtke, F., Hiller, J., Hoss, M., Bock, J., Eckstein, L., 2021. 6-layer model for a structured description and categorization of urban traffic and environment. IEEE Access 9, 59131–59147.

Staron, M., 2021. Automotive Software Architectures: An Introduction. Springer Cham.

Stursberg, O., Fehnker, A., Han, Z., Krogh, B.H., 2004. Verification of a cruise control system using counterexample-guided search. Control Eng. Pract. 12, 1269–1278.

Sun, X., Khedr, H., Shoukry, Y., 2019. Formal verification of neural network controlled autonomous systems. In: Proc. 22nd ACM Int. Conf. on Hybrid Systems: Computation and Control. HSCC '19, ACM, New York, NY, USA, pp. 147–156.

Tian, Y., Pei, K., Jana, S., Ray, B., 2018. DeepTest: Automated testing of deep-neural-network-driven autonomous cars. In: Proceedings of the 40th International Conference on Software Engineering. ICSE '18, Association for Computing Machinery, New York, NY, USA, pp. 303–314. http://dx.doi.org/10.1145/3180155.3180220.

Treiber, M., Hennecke, A., Helbing, D., 2000. Congested traffic states in empirical observations and microscopic simulations. Phys. Rev. E 62, 1805–1824.

United Nations Economic Commission for Europe (UNECE), 2023. UN Regulation No. 79: Uniform provisions concerning the approval of vehicles with regard to steering equipment.

Widmann, G.R., Daniels, M.K., Hamilton, L., Humm, L., Riley, B., Schiffmann, J.K., Schnelker, D.E., Wishon, W.H., 2000. Comparison of lidar-based and radar-based adaptive cruise control systems. SAE Trans. 109, 126–139.

Wongpiromsarn, T., Mitra, S., Lamperski, A., Murray, R.M., 2012. Verification of periodically controlled hybrid systems: Application to an autonomous vehicle. ACM Trans. Embed. Comput. Syst. 11 (S2), http://dx.doi.org/10.1145/2331147.2331163.

Wu, H.-J., Nenchev, V., Rathgeber, C., 2024. Automatic parameter tuning of self-driving vehicles. In: 2024 IEEE Conference on Control Technology and Applications. CCTA, pp. 555–560. http://dx.doi.org/10.1109/CCTA60707.2024.10666632.

Zhang, Y., Xie, F., Dong, Y., Zhou, X., Ma, C., 2013. Cyber/physical co-verification for developing reliable cyber-physical systems. In: 2013 IEEE 37th Annual Computer Software and Applications Conference. pp. 539–548.

Zhu, M., Wang, Y., Pu, Z., Hu, J., Wang, X., Ke, R., 2020. Safe, efficient, and comfortable velocity control based on reinforcement learning for autonomous driving. Transp. Res. Part C: Emerg. Technol. 117, 102662. http://dx.doi.org/10.1016/j.trc.2020.102662.

Vladislav Nenchev is Professor of Embedded Systems at the University of the Bundeswehr Munich, Germany. Prior to this, he was with the Automated and Autonomous Driving Division of BMW Group in Munich, Germany. His research interests include motion planning, optimal control and formal verification for automated driving and robotics. He develops both classical and AI-based methods to enhance the safety and reliability of cyber-physical systems.

Calum Imrie is a Researcher with the Centre for Assuring Autonomy, the University of York, UK. He investigates robotics and autonomous systems with a particular focus on the safety and assurance of deploying these systems. This includes both the learning phases, such as reinforcement learning, and at runtime particularly for AI components, and self-adaptive mechanisms. He has a special interest in robotics and autonomous systems being utilized for managing and protecting the environment.

Simos Gerasimou is Associate Professor of Computer Science at the University of York, UK. His research focuses on developing rigorous tool-supported approaches using model-based analysis, simulation and formal verification to support the engineering of trustworthy software for autonomous systems. His areas of expertise include safe AI, software engineering for AI and AI for software engineering, model-driven adaptive and autonomous systems, and assurances for autonomous systems.

Radu Calinescu is Professor of Computer Science at the University of York, UK. His research interests include formal methods for self-adaptive, autonomous, secure and dependable software, cyber–physical and AI systems, and in performance and reliability software engineering. He is an active promoter of formal methods at runtime as a way to improve the integrity and predictability of self-adaptive, autonomous and AI systems and processes.