Open Challenges in the Formal Verification of Autonomous Driving

Paolo Burgio Angelo Ferrando Marco Villani
University of Modena and Reggio Emilia
Department of Physics, Informatics and Mathematics
Modena, Italy
forename.surname@unimore.it

In the realm of autonomous driving, the development and integration of highly complex and heterogeneous systems are standard practice. Modern vehicles are not monolithic systems; instead, they are composed of diverse hardware components, each running its own software systems. An autonomous vehicle comprises numerous independent components, often developed by different and potentially competing companies. This diversity poses significant challenges for the certification process, as it necessitates certifying components that may not disclose their internal behaviour (black-boxes). In this paper, we present a real-world case study of an autonomous driving system, identify key open challenges associated with its development and integration, and explore how formal verification techniques can address these challenges to ensure system reliability and safety.

1 Introduction

The Society of Automotive Engineers (SAE) defines the design goals of autonomous driving across six distinct levels, ranging from Level 0 (L-0) to Level 5 (L-5), as outlined in [27]. These levels represent a spectrum of automation: L-0 denotes no automation, followed by L-1 which includes driver assistance, L-2 for partial automation, L-3 for conditional automation, L-4 for high automation, and culminating in L-5, which signifies full automation. Each level reflects the increasing capability of autonomous systems and their interaction with human drivers. Currently, most commercially available vehicles operate at L-2 automation. This level encompasses features such as adaptive cruise control and lane-keeping assistance, enabling the vehicle to assist the driver while still requiring constant supervision and active engagement. A few manufacturers are exploring L-3 systems, which offer conditional automation under specific circumstances. For example, some modern vehicles equipped with L-3 systems can handle highway driving autonomously, including lane-keeping, speed regulation, and adaptive cruise control, but require the driver to take over when exiting highways or in complex urban environments. Despite such advancements, the industry is still in the early stages of fully implementing higher levels of automation.

According to a recent survey on the subject [17], achieving L-5 autonomy requires the appropriate integration of technologies and efficient communication channels. Realising the full potential of automated driving demands a **reliable**, robust, and widespread mobile network. In this work, we focus on the first item on the list; that is, we are interested in making the components of autonomous driving, as well as their interactions, (more) reliable. To achieve this, we start with a case study of a real-world autonomous driving system and address the issues to enhance its reliability from a formal perspective.

Taking inspiration from [22], we treat the autonomous driving system as a component-based system composed of black-box components that we are not interested in opening (or cannot open). Instead, we focus on how to achieve the formal verification of the resulting heterogeneous system. That is, how the

Matt Luckcuck and Mengwei Xu (Eds.):
Sixth International Workshop on Formal Methods
for Autonomous Systems (FMAS 2024)
EPTCS 411, 2024, pp. 191–200, doi:10.4204/EPTCS.411.13

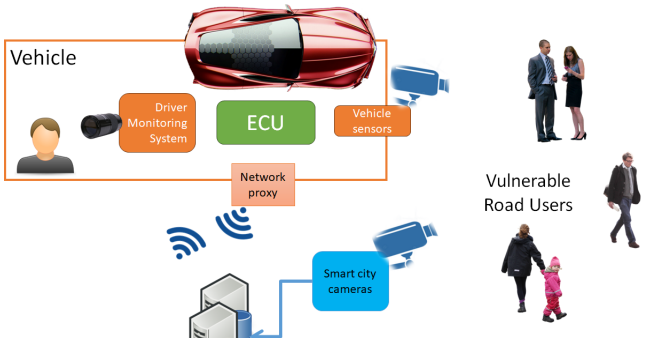

Figure 1: The vehicle architecture and use case

components interact with each other, and how we can verify (and perhaps even enforce) correct behaviour according to well-known standards in autonomous driving.

This paper presents a real-world case study in autonomous driving (Section 2), highlighting key challenges. Section 3 examines how formal methods, particularly formal verification, can address these issues and be integrated into autonomous systems, with Section 4 discussing their limitations. Finally, Section 5 concludes the paper and outlines future directions.

2 Autonomous Driving Case Study

As a motivational example, we present a case study from the AI4CSM Project[1], funded by the European Commission [33]. This study features a L-3/L-4 autonomous vehicle equipped with advanced sensor fusion capabilities, exemplifying a next-generation automotive platform [29, 35]. The vehicle can interpret both driver status (e.g., drowsiness, distraction) using in-vehicle cameras, and external environmental conditions using on-board sensors and data from city sensors, as illustrated in Figure 1.

In the simplest scenario, the vehicle operates at a low level of automation (*i.e.*, L-2/L-3), with the driver maintaining control. If the vehicle detects a potentially dangerous situation, such as drowsiness or imminent collisions, it triggers a secure takeover strategy, transitioning to L-4 and executing a safety manoeuvre. For our study, we focus on the sensor fusion component, where the perception module running on the on-board Electronic Control Unit (ECU) aggregates information from heterogeneous data streams. Specifically, we implemented a system to monitor the driver using camera-based behavioural analysis, coupled with on-board cameras to inspect the surrounding environment. Additionally, we enhanced the vehicle's perception capabilities by incorporating data from a smart city prototype area, namely the Modena Automotive Smart Area (MASA)[2]. Its structure is shown in Figure 2.

This area includes smart cameras mounted on poles that detect vulnerable road users and analyse or predict their movement trajectories. These data are streamed to the vehicle through the smart city's 4G-5G wireless connectivity. The vehicle's centralised ECU, also known as the Domain Controller, processes this information to determine the most appropriate response, such as emergency braking, complex manoeuvres, or issuing driver warnings. For research purposes, we implemented this use case on a Citroen Mehari. We will now explore the main open challenges that must be addressed to facilitate the industrialisation of these complex systems.

[1] https://ai4csm.eu/
[2] https://www.automotivesmartarea.it/

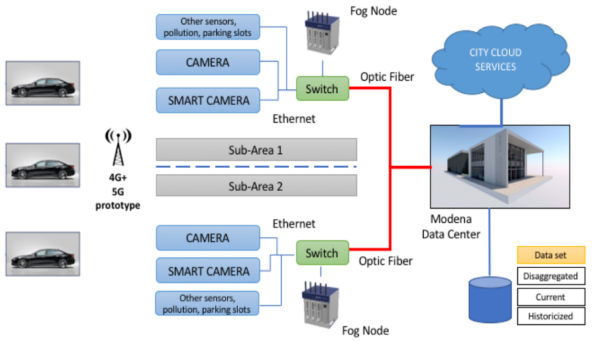

Figure 2: The Modena Automotive Smart Area.

Aggregating probabilistic data sources. One primary open challenge arises from the inherent nature of most software components used for perception. This stage is the first in any autonomous driving stack, where raw sensor data (in our scenario, RGB cameras) are processed to interpret and analyse the driver or the car's surroundings. Numerous algorithms and approaches could be employed, most of which [29, 35, 5, 19, 24] heavily rely on machine learning, deep learning, or, more generally, on heuristics and statistical methodologies to handle the complexity of raw data frames. Additionally, most systems have a hierarchical structure, consisting of sub-components arranged in pipelines. Figure 3 illustrates this decomposition for our camera-based behavioural analysis used for monitoring the driver.

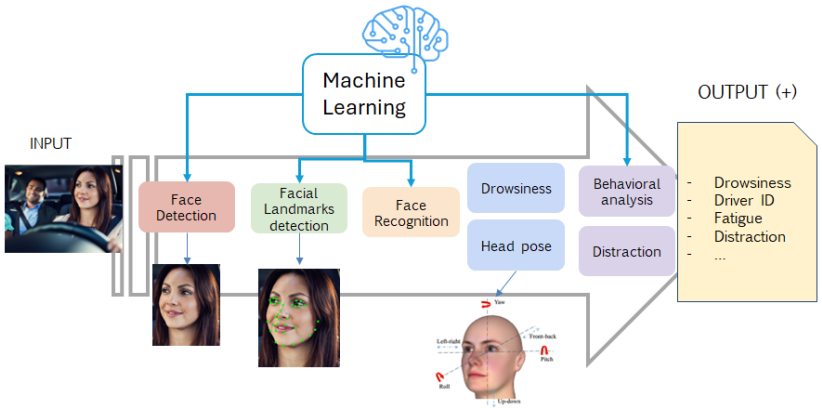

Figure 3: Scheme of the behavioural DMS pipeline.

Every block of this system has a specific performance metric, typically expressed in Frames-Per-Second (FPS), and a nominal accuracy, indicating the reliability of the information produced by the (sub)component. Our Driver Monitoring System (DMS) is a white-box component, developed in-house, which allows us complete access to its internals for tuning and modifications. However, in realistic industrial scenarios, most software modules will be developed by different companies and often implemented as "black-box" ECUs. Therefore, we identify the first open challenge.

Open Challenge 1: the need to compose a hierarchy of probabilistic software modules to formally measure and derive the overall system's resulting accuracy.

Deploy on embedded systems. Deploying intelligence in automation use cases requires two key components: powerful computational hardware and numerous sensor modules to accurately interpret

the surrounding and in-cabin environment in a timely manner. This presents a significant challenge for automotive engineers, who must integrate TOPS-greedy[3] software components onto power-efficient boards, ideally featuring many-core data processors such as those from NVIDIA Orin [25] or AMD XILINX [7, 1]. Figure 4 illustrates the target architecture of next-generation ECUs.

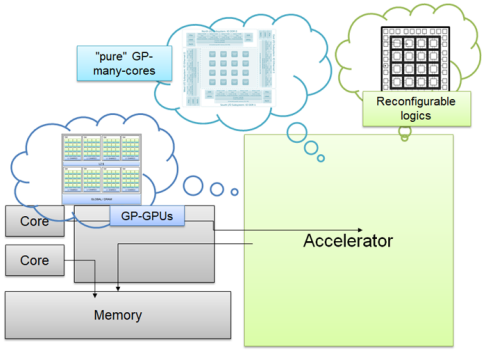

Figure 4: Generic architecture next-generation ECUs.

These systems employ multi-core host platforms, which include both Real-Time and non-Real-Time core ISAs (Instruction Set Architectures), which define the set of instructions a processor can execute. These are coupled with data-crunching architectures such as GPGPUs [25], reconfigurable arrays [1], or application-specific circuitry to implement processing algorithms directly in hardware. Such a complex architecture presents two main challenges.

Open Challenge 2: to devise efficient strategies for mapping software components onto the available computing cores, exploiting redundancy and voting schemes to enhance overall system reliability.

Intuitively, most of the algorithms we employ can potentially run on various cores, and finding the optimal mapping must be handled in the most efficient manner.

3 Formal Methods to the Rescue

In the previous section, we discussed a real-world case study in autonomous driving, highlighting challenges in integrating autonomous systems into road infrastructure and progressing towards L-5 capabilities. Here, we explore how formal methods, particularly formal verification, address these challenges. For a comprehensive overview of formal verification techniques in autonomous systems, see [21].

3.1 Open Challenge 1: Heterogeneous Composition of Untrustworthy Components

The first challenge involves managing components with varying levels of reliability. Some components are open-source (white-box), allowing full access and modifications, while others are closed-source (black-box), restricting access to their internal workings. To address this challenge, we propose exploiting formal verification techniques. Specifically, as outlined in [20], we employ formal verification methods focusing on three key areas (called also recipes): verification of decision-making components, AI-based components, and the enforcement of safety claims.

[3]They require a high number of Tera Operations Per Second (TOPS) to process complex algorithms, such as those used in artificial intelligence and sensor fusion.

To address this challenge, we propose the use of heterogeneous verification techniques [22, 4, 6, 26]. These techniques are based on the Assume-Guarantee principle, where each component of the heterogeneous system is defined in terms of its *assumptions* (what the component expects from the system to function correctly) and its *guarantees* (what the component provides to the system upon correct execution). This methodology allows us to abstract away the implementation details of the various system components, enabling the system designer to focus on their integration. As long as the assumptions and guarantees of a component are documented and made available, it can be implemented as either a white-box or black-box component. By employing these verification techniques, it is possible to formally verify the proper integration of multiple components, potentially developed by different parties [22, 4, 6, 26].

In addition to Assume-Guarantee reasoning, model checking and formal methods can play a crucial role in verifying component integration and ensuring system reliability [8, 3, 10, 16]. Component-Based Software Engineering (CBSE) methodologies also provide a framework for assembling reliable systems from diverse components [30]. Moreover, adhering to safety and certification standards, such as ISO 26262, is essential for validating the safety of automotive software systems [14].

To complete the verification process, we envision the use of Runtime Verification (RV) [2], a technique for monitoring and analysing the execution of a system at runtime to ensure it adheres to specified properties. RV can check and enforce adherence to all assumptions and guarantees of the components. As highlighted in [22], the Assume-Guarantee verification methodology focuses on verifying the resulting distributed system and the integration of its components. However, it relies on the assumption that the assumptions and guarantees of each component (which may be black-boxes) are satisfied. To bridge this gap and provide a robust verification technique suitable for the heterogeneous nature of the autonomous driving domain, we need to employ additional verification methods to ensure the proper behaviour of individual components. By confirming that each component behaves correctly, we can validate the entire system's integration and maintain its formal assurances.

It is important to note that the use of RV in this context is not entirely straightforward. The components of an autonomous driving system may exhibit a certain level of uncertainty, meaning that the information they provide may not always be precise. For example, as illustrated in Figure 3, the steps that process the camera input to determine the driver's level of drowsiness, distraction, fatigue, etc., are inherently uncertain. These steps rely on Machine Learning models, which offer results with varying levels of confidence. Additionally, this uncertainty is not limited to the current information provided but may also encompass temporal aspects. For instance, a component might predict that the driver will fall asleep in five minutes, with a given level of confidence. Due to these factors, it is unrealistic to rely solely on standard RV approaches for verifying component conformance. Instead, techniques that incorporate RV with uncertainty must be considered, such as those discussed in [34, 32, 11]. For a comprehensive survey on this topic, the reader may refer to [31]. Additionally, the complexity of verifying machine learning and AI components within autonomous systems presents unique challenges. Ensuring the reliability of non-deterministic algorithms requires specialised verification techniques [23]. Addressing these challenges will enable the development of robust, reliable, and safe autonomous driving systems.

3.2 Open Challenge 2: Efficient Strategies for Mapping the Distributed Computation

Addressing this challenge involves the use of formal verification techniques to ensure that the mapping strategies are both efficient and reliable (since we are in a real-time system). Formal verification can be employed to systematically verify that the software components are optimally distributed across the computing cores, and that redundancy and voting mechanisms are correctly implemented to enhance fault tolerance and system robustness. By formally verifying these strategies, we can guarantee the sys-

tem meets its performance and reliability requirements, even in the presence of component failures or uncertainties. Indeed, fault-tolerant designs, which incorporate redundancy and voting schemes, play a crucial role in mitigating the impact of component failures. The study presented in [9] provides valuable insights into the application of formal verification techniques to validate the correctness of these designs. By systematically verifying that fault-tolerant hardware meets specified reliability requirements, the authors demonstrate the effectiveness of formal methods in identifying design errors that traditional testing might overlook. Although [9] does not originate from the domain of autonomous driving, it provides a valuable foundation for addressing the open challenge discussed here. The paper presents methodologies for formal verification of fault-tolerant hardware designs, which are crucial for ensuring the reliability and robustness of systems with heterogeneous components. By adapting these verification techniques, we can systematically validate the correctness and reliability of the complex, integrated systems used in autonomous vehicles.

Formal verification can be used to prove that the redundancy and voting mechanisms are correctly implemented and that they effectively enhance system reliability; for example, in [28], the authors discuss fault-tolerance techniques that include redundancy and efficient scheduling policies. Formal verification ensures that these techniques are correctly applied, thereby enhancing the reliability of the system.

The work in [15] provides a comprehensive framework for the formal verification of distributed Resource Management (RM) schemes in many-core systems using probabilistic model checking. This research is particularly relevant to our work in the context of autonomous driving systems, which also require efficient resource allocation across multiple computing cores. The authors demonstrate the use of the PRISM model checker [18] to analyse and compare the performance and reliability of different RM schemes. They emphasise the limitations of traditional simulation methods, which are inherently exhaustive, and advocate for formal verification to ensure completeness and accuracy. In our study, similar formal verification techniques can be applied to optimise the mapping of software components onto the available computing cores in autonomous vehicles. By leveraging the probabilistic analysis methods described in [18], we could systematically evaluate the robustness and performance efficiency of our proposed resource management strategies in autonomous driving systems.

4 Limitations of Applying Formal Methods in Autonomous Systems

While Formal Methods provide a promising approach for integrating heterogeneous components and efficient mapping in autonomous systems, there are significant practical limitations to their use. The main challenges include the need for specialised knowledge, scalability issues, interpretability of results, and difficulties in handling uncertain environments, as well as cost-benefit trade-offs in development.

One key limitation is the high barrier to entry, as FM often requires deep expertise in formal logic and verification techniques. This specialised skill set is not commonly available within standard engineering teams, and the process of specifying and verifying systems can be time-consuming. Despite the development of new tools aimed at making FM more accessible, their capabilities are still evolving and often require substantial refinement to meet industry needs.

Scalability and complexity also present major obstacles. Autonomous systems comprise numerous interacting components, leading to a state space that grows exponentially, making exhaustive verification computationally expensive or infeasible. Techniques like compositional reasoning and modular verification attempt to manage this, but they require careful abstraction, which may oversimplify or overlook critical behaviours. Moreover, model checking can be computationally intensive, particularly when applied to real-time or resource-constrained systems, and Assume-Guarantee reasoning hinges on accurate

assumptions that are challenging to guarantee in practice.

Another practical challenge is the interpretability of verification results. Outputs from FM tools, such as model checkers or runtime verifiers, may highlight specification violations without providing clear solutions, requiring domain expertise to resolve. When AI and machine learning components are involved, this issue is further complicated by their probabilistic and non-deterministic nature, making the analysis of verification results particularly difficult.

The dynamic and uncertain environment in which autonomous systems operate adds further complexity. Perception modules relying on sensor fusion and AI-based decision-making are context-dependent and produce inherently uncertain outputs. Traditional FM approaches struggle to define precise specifications in such cases. Techniques that incorporate probabilistic reasoning or uncertainty-aware models are being explored [34, 32, 11], but their practical applicability to real-world systems remains under active research and development.

Integrating FM into existing development workflows also poses a significant challenge, as it requires a careful cost-benefit analysis. The process of formally specifying, modeling, and verifying components demands significant time and resources, potentially extending the development lifecycle. While FM provides strong safety and reliability assurances – critical for autonomous vehicles – these benefits must be weighed against their scalability and adaptability to system updates and changes.

Lastly, verifying AI and ML components remains an open challenge, as their non-deterministic behaviour does not fit within traditional FM frameworks. Specialised techniques are needed to manage varying levels of confidence and probabilistic outputs in AI models [23]. Thus, hybrid approaches combining formal verification with testing and validation are necessary for comprehensive system assurance.

In summary, FM offers a rigorous foundation for ensuring system reliability and safety in autonomous vehicles, but practical limitations such as scalability, required expertise, result interpretation, and integration into dynamic environments must be addressed to enable their effective use in real-world applications.

5 Conclusions and Future Work

In this short paper, we present a real-world case study in the autonomous driving domain, identify key open challenges, and discuss how formal verification techniques can address these issues.

We focus on two primary challenges hindering the achievement of L-5 automation in autonomous vehicles. The first is the heterogeneous composition of black-box components, which affects system reliability. The second challenge involves the proper mapping of computations within a highly dynamic, real-time distributed system. We propose how existing formal verification techniques can be leveraged to address these issues and explore their implications, along with potential adaptations for integration into autonomous driving systems.

Our aim is to highlight these open challenges in the autonomous driving domain and demonstrate how formal verification techniques can theoretically enhance system reliability. For future work, we intend to build upon the insights reported here by applying some of the discussed techniques and methodologies to our case study, leveraging tools like those proposed in [12, 13] to further explore their practical applicability and impact on system robustness.

Acknowledgements

The authors have received funding from ECSEL JU project AI4CSM (GA N.101007326) and the Chips JU project ShapeFuture (GA N.101139996).

References

[1] AMD Xilinx (2022): *Zynq UltraScale+ MPSoC ZCU102 Evaluation Kit*. Available at `https://www.xilinx.com/products/boards-and-kits/ek-u1-zcu102-g.html`. Accessed on October 2024.

[2] Ezio Bartocci, Yliès Falcone, Adrian Francalanza & Giles Reger (2018): *Introduction to Runtime Verification*. In Ezio Bartocci & Yliès Falcone, editors: *Lectures on Runtime Verification - Introductory and Advanced Topics*, *Lecture Notes in Computer Science* 10457, Springer, pp. 1–33, doi:10.1007/978-3-319-75632-5_1.

[3] Saddek Bensalem, Marius Bozga, Thanh-Hung Nguyen & Joseph Sifakis (2010): *Compositional verification for component-based systems and application*. *IET Softw.* 4(3), pp. 181–193, doi:10.1049/IET-SEN.2009.0011.

[4] Albert Benveniste, Benoît Caillaud, Dejan Nickovic, Roberto Passerone, Jean-Baptiste Raclet, Philipp Reinkemeier, Alberto L. Sangiovanni-Vincentelli, Werner Damm, Thomas A. Henzinger & Kim G. Larsen (2018): *Contracts for System Design*. *Found. Trends Electron. Des. Autom.* 12(2-3), pp. 124–400, doi:10.1561/1000000053.

[5] Roberto Cavicchioli, Riccardo Martoglia & Micaela Verucchi (2022): *A Novel Real-Time Edge-Cloud Big Data Management and Analytics Framework for Smart Cities*. *Journal of Universal Computer Science* 28(1), p. 3 – 26, doi:10.3897/jucs.71645.

[6] Adrien Champion, Arie Gurfinkel, Temesghen Kahsai & Cesare Tinelli (2016): *CoCoSpec: A Mode-Aware Contract Language for Reactive Systems*. In Rocco De Nicola & Eva Kühn, editors: *Software Engineering and Formal Methods - 14th International Conference, SEFM 2016, Held as Part of STAF 2016, Vienna, Austria, July 4-8, 2016, Proceedings*, *Lecture Notes in Computer Science* 9763, Springer, pp. 347–366, doi:10.1007/978-3-319-41591-8_24.

[7] Kwon Neung Cho, Jeongeun Kim, Do Young Choi, Young Hyun Yoon, Jung Hwan Oh & Seung Eun Lee (2021): *An FPGA-Based ECU for Remote Reconfiguration in Automotive Systems*. *Micromachines* 12, doi:10.3390/mi12111309. Available at `https://www.mdpi.com/2072-666X/12/11/1309`.

[8] E. M. Clarke, O. Grumberg & D. A. Peled (1999): *Model Checking*. MIT Press.

[9] Luis Entrena, Antonio J. Sanchez-Clemente, Luis Ángel García-Astudillo, Marta Portela-García, Mario García-Valderas, Almudena Lindoso & Roberto Sarmiento (2023): *Formal Verification of Fault-Tolerant Hardware Designs*. *IEEE Access* 11, pp. 116127–116140, doi:10.1109/ACCESS.2023.3325616.

[10] Yliès Falcone, Mohamad Jaber, Thanh-Hung Nguyen, Marius Bozga & Saddek Bensalem (2015): *Runtime verification of component-based systems in the BIP framework with formally-proved sound and complete instrumentation*. *Softw. Syst. Model.* 14(1), pp. 173–199, doi:10.1007/S10270-013-0323-Y.

[11] Angelo Ferrando & Vadim Malvone (2022): *Runtime Verification with Imperfect Information Through Indistinguishability Relations*. In Bernd-Holger Schlingloff & Ming Chai, editors: *Software Engineering and Formal Methods - 20th International Conference, SEFM 2022, Berlin, Germany, September 26-30, 2022, Proceedings*, *Lecture Notes in Computer Science* 13550, Springer, pp. 335–351, doi:10.1007/978-3-031-17108-6_21.

[12] Angelo Ferrando & Vadim Malvone (2024): *Hands-on VITAMIN: A Compositional Tool for Model Checking of Multi-Agent Systems*. In Marco Alderighi, Matteo Baldoni, Cristina Baroglio, Roberto Micalizio & Stefano Tedeschi, editors: *Proceedings of the 25th Workshop "From Objects to Agents", Bard (Aosta), Italy, July 8-10, 2024*, *CEUR Workshop Proceedings* 3735, CEUR-WS.org, pp. 148–160. Available at `https://ceur-ws.org/Vol-3735/paper_12.pdf`.

[13] Angelo Ferrando & Vadim Malvone (2024): *VITAMIN: A Compositional Framework for Model Checking of Multi-Agent Systems*. *CoRR* abs/2403.02170, doi:10.48550/ARXIV.2403.02170. arXiv:2403.02170.

[14] International Organization for Standardization (2018): *ISO 26262-1:2018. Road vehicles – Functional safety*.

[15] Shafaq Iqtedar, Osman Hasan, Muhammad Shafique & Jörg Henkel (2016): *Formal probabilistic analysis of distributed resource management schemes in on-chip systems*. In Luca Fanucci & Jürgen Teich, editors: *2016*

Design, Automation & Test in Europe Conference & Exhibition, DATE 2016, Dresden, Germany, March 14-18, 2016, IEEE, pp. 930–935. Available at https://ieeexplore.ieee.org/document/7459441/.

[16] Daniel Karlsson, Petru Eles & Zebo Peng (2007): *Formal verification of component-based designs*. *Des. Autom. Embed. Syst.* 11(1), pp. 49–90, doi:10.1007/S10617-006-9723-3.

[17] Manzoor Ahmed Khan, Hesham El-Sayed, Sumbal Malik, Muhammad Talha Zia, Muhammad Jalal Khan, Najla Alkaabi & Henry Alexander Ignatious (2023): *Level-5 Autonomous Driving - Are We There Yet? A Review of Research Literature*. *ACM Comput. Surv.* 55(2), pp. 27:1–27:38, doi:10.1145/3485767.

[18] Marta Z. Kwiatkowska, Gethin Norman & David Parker (2002): *PRISM: Probabilistic Symbolic Model Checker*. In Tony Field, Peter G. Harrison, Jeremy T. Bradley & Uli Harder, editors: *Computer Performance Evaluation, Modelling Techniques and Tools 12th International Conference, TOOLS 2002, London, UK, April 14-17, 2002, Proceedings*, *Lecture Notes in Computer Science* 2324, Springer, pp. 200–204, doi:10.1007/3-540-46029-2_13.

[19] Songtao Liu, Di Huang & andYunhong Wang (2018): *Receptive Field Block Net for Accurate and Fast Object Detection*. In: *The European Conference on Computer Vision (ECCV)*, doi:10.1007/978-3-030-01252-6_24.

[20] Matt Luckcuck (2023): *Using formal methods for autonomous systems: Five recipes for formal verification*. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability* 237(2), pp. 278–292, doi:10.1177/1748006X211034970.

[21] Matt Luckcuck, Marie Farrell, Louise A. Dennis, Clare Dixon & Michael Fisher (2019): *Formal Specification and Verification of Autonomous Robotic Systems: A Survey*. *ACM Comput. Surv.* 52(5), pp. 100:1–100:41, doi:10.1145/3342355.

[22] Matt Luckcuck, Marie Farrell, Angelo Ferrando, Rafael C. Cardoso, Louise A. Dennis & Michael Fisher (2022): *A Compositional Approach to Verifying Modular Robotic Systems*. *CoRR* abs/2208.05507, doi:10.48550/ARXIV.2208.05507. arXiv:2208.05507.

[23] G. Marcus & E. Davis (2019): *Rebooting AI: Building Artificial Intelligence We Can Trust*. Pantheon Books.

[24] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong & Jayant Kalagnanam (2023): *A Time Series is Worth 64 Words: Long-term Forecasting with Transformers*. In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, OpenReview.net. Available at https://openreview.net/forum?id=Jbdc0vTOcol.

[25] NVIDIA (2022): *NVIDIA Jetson AGX Orin Series*. Available at https://www.nvidia.com/content/dam/en-zz/Solutions/gtcf21/jetson-orin/nvidia-jetson-agx-orin-technical-brief.pdf. Accessed: October 2024.

[26] Ivan Ruchkin, Joshua Sunshine, Grant Iraci, Bradley R. Schmerl & David Garlan (2018): *IPL: An Integration Property Language for Multi-model Cyber-physical Systems*. In Klaus Havelund, Jan Peleska, Bill Roscoe & Erik P. de Vink, editors: *Formal Methods - 22nd International Symposium, FM 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 15-17, 2018, Proceedings*, *Lecture Notes in Computer Science* 10951, Springer, pp. 165–184, doi:10.1007/978-3-319-95582-7_10.

[27] I SAE (2021): *Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles j3016 202104*. *Society of Automotive Engineers* 41.

[28] Sepideh Safari, Mohsen Ansari, Heba Khdr, Pourya Gohari-Nazari, Sina Yari-Karin, Amir Yeganeh-Khaksar, Shaahin Hessabi, Alireza Ejlali & Jörg Henkel (2022): *A Survey of Fault-Tolerance Techniques for Embedded Systems From the Perspective of Power, Energy, and Thermal Issues*. *IEEE Access* 10, pp. 12229–12251, doi:10.1109/ACCESS.2022.3144217.

[29] Society of Automotive Engineers (SAE) (2024): *Sensor fusion expanding in step with advancing vehicle sophistication*. Available at https://www.sae.org/news/2024/02/sensor-fusion-trends. Accessed on October 2024.

[30] Clemens A. Szyperski, Dominik Gruntz & Stephan Murer (2002): *Component software - beyond object-oriented programming, 2nd Edition*. Addison-Wesley component software series, Addison-Wesley. Available at https://www.worldcat.org/oclc/248041840.

[31] Rania Taleb, Sylvain Hallé & Raphaël Khoury (2023): *Uncertainty in runtime verification: A survey*. *Comput. Sci. Rev.* 50, p. 100594, doi:10.1016/J.COSREV.2023.100594.

[32] Rania Taleb, Raphaël Khoury & Sylvain Hallé (2021): *Runtime Verification Under Access Restrictions*. In Simon Bliudze, Stefania Gnesi, Nico Plat & Laura Semini, editors: *9th IEEE/ACM International Conference on Formal Methods in Software Engineering, FormaliSE@ICSE 2021, Madrid, Spain, May 17-21, 2021*, IEEE, pp. 31–41, doi:10.1109/FORMALISE52586.2021.00010.

[33] Ovidiu Vermesan, Reiner John, Patrick Pype, Gerardo Daalderop, Kai Kriegel, Gerhard Mitic, Vincent Lorentz, Roy Bahr, Hans Erik Sand, Steffen Bockrath & Stefan Waldhör (2021): *Automotive Intelligence Embedded in Electric Connected Autonomous and Shared Vehicles Technology for Sustainable Green Mobility*. *Frontiers in Future Transportation* 2, doi:10.3389/ffutr.2021.688482.

[34] Shaohui Wang, Anaheed Ayoub, Oleg Sokolsky & Insup Lee (2011): *Runtime Verification of Traces under Recording Uncertainty*. In Sarfraz Khurshid & Koushik Sen, editors: *Runtime Verification - Second International Conference, RV 2011, San Francisco, CA, USA, September 27-30, 2011, Revised Selected Papers*, *Lecture Notes in Computer Science* 7186, Springer, pp. 442–456, doi:10.1007/978-3-642-29860-8_35.

[35] Zhangjing Wang, Yu Wu & Qingqing Niu (2020): *Multi-Sensor Fusion in Automated Driving: A Survey*. *IEEE Access* 8, pp. 2847–2868, doi:10.1109/ACCESS.2019.2962554.