Towards Runtime Monitoring of Complex System Requirements for Autonomous Driving Functions[*]

Dominik Grundt
dominik.grundt@dlr.de

Anna Köhne
anna.koehne@dlr.de

Ishan Saxena
ishan.saxena@dlr.de

Ralf Stemmer
ralf.stemmer@dlr.de

Bernd Westphal
bernd.westphal@dlr.de

Eike Möhlmann
eike.moehlmann@dlr.de

German Aerospace Center
Institute of Systems Engineering for Future Mobility
Oldenburg, Germany

Autonomous driving functions (ADFs) in public traffic have to comply with complex system requirements that are based on knowledge of experts from different disciplines, e.g., lawyers, safety experts, psychologists. In this paper, we present a research preview regarding the validation of ADFs with respect to such requirements. We investigate the suitability of Traffic Sequence Charts (TSCs) for the formalization of such requirements and present a concept for monitoring system compliance during validation runs. We find TSCs, with their intuitive visual syntax over symbols from the traffic domain, to be a promising choice for the collaborative formalization of such requirements. For an example TSC, we describe the construction of a runtime monitor according to our novel concept that exploits the separation of spatial and temporal aspects in TSCs, and successfully apply the monitor on exemplary runs. The monitor continuously provides verdicts at runtime, which is particularly beneficial in ADF validation, where validation runs are expensive. The next open research questions concern the generalization of our monitor construction, the identification of the limits of TSC monitorability, and the investigation of the monitor's performance in practical applications. Perspectively, TSC runtime monitoring could provide a useful technique in other emerging application areas such as AI training, safeguarding ADFs during operation, and gathering meaningful traffic data in the field.

1 Introduction

The importance of autonomous driving functions (ADFs) for future mobility is increasing rapidly [19, 26]. ADFs have to safely operate in complex and dynamic traffic environments while their internal behavior is often not sufficiently known, e.g., due to the presence of AI-based components. Hence, system validation during their development is necessary for homologation. Here, validation consists of evaluating a representative set of system runs for compliance with system requirements. A promising approach for ADF validation in the automotive domain is scenario-based testing [17, 23, 24]. This method is used for validation with respect to technical system requirements (e.g., limits on maximum acceleration). However, it cannot yet be used for more complex system requirements. In order to act safely in traffic environments, ADFs also need to take into account complex world knowledge about traffic rules and social norms regarding the interaction of traffic participants, resulting in more complex

---

system requirements. Knowledge of several stakeholders, e.g., lawyers, safety experts and psychologists, needs to be integrated into ADF system requirements. In many cases, these system requirements concern the temporal evolution of spatial relations between traffic participants and their environment. In order to objectify requirements derived from such knowledge, formalization is important. Thus, for ADF validation, there is a need for a formalization language that can usefully formalize knowledge from several stakeholders to specify spatio-temporal system requirements and a suitable method that checks the system's behavior against these requirements. Since the validation of ADFs is a time and resource intensive process, runtime monitoring would be particularly beneficial due to continuous calculation of verdicts. Specifically, the continuously calculated verdicts may provide additional insights into cause and effect of violations and can be used as early termination criteria for test runs.

In this paper, we present a research preview for formalizing spatio-temporal system requirements from the automotive domain and a corresponding runtime monitoring concept for the validation of ADFs. To this end, we investigate whether the specification language Traffic Sequence Charts (TSCs) [5, 6] can be used for formalizing such automotive system requirements and how to perform runtime monitoring for this formalism. The paper is organized as follows: In Sec. 2, we discuss related work regarding specification languages and runtime monitoring. In Sec. 3, we investigate the suitability of the TSC language for our purpose. In Sec. 4, we present and discuss our concept for runtime monitoring of TSCs. Finally, in Sec. 5, we conclude and discuss future work.

2 Related Work

In this section, we discuss literature regarding specification languages for traffic scenarios and runtime monitoring. *Specification Languages.* In recent years there has been a trend towards the visual specification of automotive system requirements. Visual specification languages enable a more intuitive understanding and more concise representation of system requirements as compared to textual or symbolic languages. They allow experts from different domains to define, discuss, adapt and further derive system requirements or scenarios, without a deep understanding of symbolic logic. The MLSL formalism [25] enables visual specification of requirements for system controllers of cooperative systems for urban road intersections with a focus on reasoning about authority of movement. The specification language M-SDL [13] focuses on sampling a large number of concrete test cases based on abstract scenarios. While it allows visualization of scenarios, it lacks semantics for its visual elements. The same also applies to the SCENIC framework [14], which can generate large numbers of test scenarios based on scenario descriptions that can also be displayed visually. Here, a scene is generated from code description. In contrast, the TSC language [5, 6] does not have a specific focus on a particular type of traffic scenarios and has been applied to the specification of test scenarios [17, 24] as well as system requirements [4]. While the visual specification of scenarios and system requirements is already used in research and industry, there still is a need for the development of runtime monitoring for visually specified system requirements.

Runtime Monitoring is an established technique for validating system behavior with respect to system requirements. Runtime monitoring methods for temporal logics such as Linear Temporal Logic (LTL) and Signal Temporal Logic (STL) are available in literature. In [2], the authors present such a method for LTL. In the last decade, there has been an increasing interest in runtime monitoring of STL as presented in the works [1, 7, 8, 9, 16]. Additionally, extensions to STL and its runtime monitoring have been developed for spatial as well as spatio-temporal requirements [15, 18, 22]. In [12], timed automata have been used for the verification of real-time specifications based on the logic MITL. In [20], based on programmed scenarios and available temporal properties, monitors are programmed manually and

used to analyze streams generated by the system. Closely related to our concept is the recent work [27], where runtime monitoring for an Adaptive Cruise Control of an autonomous vehicle was presented. Their goal was to monitor robustness of PID controller requirements (invariant over speed and distance) inside the vehicle. A specification was created using STL and the temporal behavior of the controller was monitored. Despite the presence of literature for runtime monitoring of spatio-temporal system requirements, there is a gap between visual specification and existing methods. In this paper, we close this gap by presenting a concept for runtime monitoring of visualized complex spatio-temporal system requirements.

3   TSC for Formalization of Complex System Requirements

As discussed above, there is a need for a specification language that is able to express complex system requirements for autonomous driving functions (ADFs) intuitively, s.t. they can be interpreted and discussed by stakeholders from a wide variety of fields. As is inherent to the traffic domain, system requirements for ADFs (e.g., the execution of a specific maneuver or the compliance with traffic rules) often concern the temporal evolution of spatial relations (like relative positions and distances) between, and the state (like the velocity or activation of a turn signal) of, traffic participants and their environment.

The Traffic Sequence Chart (TSC) language is a formalism that provides visualization of spatio-temporal system requirements with a rigorous formal semantics and tool support [3] for automatic translation to a machine-interpretable format. In this section, we investigate the suitability of the TSC language for the formalization of complex spatio-temporal system requirements for ADF validation and give a simplified overview of syntax and semantics of TSCs. For a detailed account of the TSC language, refer to [5]. Regulations for ADFs are based on expert knowledge from a variety of domains. For discussions between stakeholders, system requirements are often only specified informally in natural language [11]. We now provide an example of such an informal specification. For this example, the system we want to validate is an ADF controlling a car. We formulate a requirement for runs of the system, which gets activated when the car approaches an obstacle in the right lane of a two-lane road with no other traffic:

"The car should change to the left lane, pass by the obstacle and return to the right lane in an orderly fashion within less than 45 seconds, while always keeping a safe distance to the obstacle."

To enable objective verdicts, such informal descriptions need to be formalized. We demonstrate a formalization as a TSC in Fig. 1.
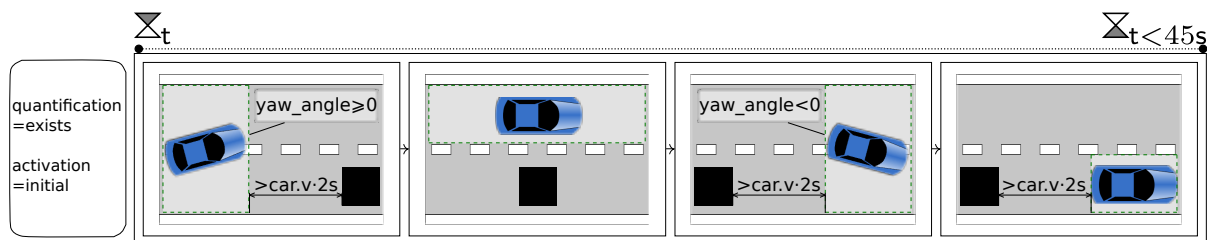

Figure 1: Example TSC of a pass-by maneuver.

As in Fig. 1, every TSC consists of a Header and a Chart. In the Header, the start and end of the considered time interval are quantified as the so called *activation mode* and *time quantification*. In our example, the activation mode is *initial* and the time quantification is *exists*, indicating that there must be some initial (starting at activation) time interval in which the car behaves as specified in the Chart. Charts

are constructed from Invariant Nodes, each inscribed with a Spatial View, and may be annotated with an Hourglass Constraint, specifying requirements regarding the duration of the corresponding time interval. In our example, the Chart is a *sequence* of four Invariant Nodes with an Hourglass Constraint of $< 45s$. Within a Spatial View, objects (like cars, obstacles and roads) are represented using object symbols. Constraints on the spatial relations between the represented objects are indicated by the relative positions of their symbols as well as *distance lines* explicitly constraining the allowed distance between objects. Furthermore, objects have attributes (like velocity or yaw angle), which may also appear in constraints.

In the Spatial Views of our example, a car is represented by a blue car symbol and a stationary obstacle by a black square. The first Spatial View is satisfied when the car is at a safe distance of more than $car.v \cdot 2$s (distance covered in 2 seconds) behind the obstacle and is oriented towards the left side of the road, and the second when the car is on the left lane. The third Spatial View is satisfied when the car is at least the specified safe distance ahead of the obstacle and is oriented towards the right side of the road, and the fourth when the car is on the right lane and still at least the specified safe distance ahead of the obstacle. The TSC is satisfied for a given run of a system (starting at the activation of our TSC) if there is an end time t less than 45 seconds, and times $0 = t_0 < t_1 < t_2 < t_3 < t_4 = t$ such that for $i \in \{1,2,3,4\}$, on $[t_{i-1}, t_i)$, the i-th Spatial View is satisfied.

Our example shows that TSCs can, in principle, be used for formalizing complex spatio-temporal system requirements. Additionally, it demonstrates the intuitive visual syntax of TSCs, which satisfies the need for multi-stakeholder collaboration. Besides sequentially connecting Invariant Nodes (or more generally Charts), the formalism provides the syntax and semantics to express *Negation*, *Concurrency* and *Choice* (conceptually logical not, and, or) as well as implications between Charts. In particular, the expression of implications together with a universal quantification of the considered time interval in the Header is useful for the formalization of rules like "whenever a car has not indicated a lane change recently, it must stay on its current lane".

In conclusion, the TSC language is a promising choice for formalizing complex spatio-temporal system requirements in the context of validating ADFs.

4 Runtime Monitoring for the TSC language

In this section, we define our notion of runtime monitoring and based on that describe a novel concept for runtime monitoring of visually specified complex spatio-temporal automotive system requirements, using the TSC language. We define runtime monitoring as the task of checking the satisfaction of a correctness property given a run of a system. In particular, a runtime monitor must decide whether the run satisfies or violates the correctness property as early as possible. At runtime, the monitor iteratively evaluates the current initial segment (prefix) of the run for the following properties: a prefix is called *satisfying* if every continuation satisfies the property, *violating* if no continuation satisfies the property or *inconclusive* otherwise. It returns *violating* or *satisfying* as verdict as soon as either is detected. In TSC runtime monitoring, the correctness property is given by satisfaction of a TSC.
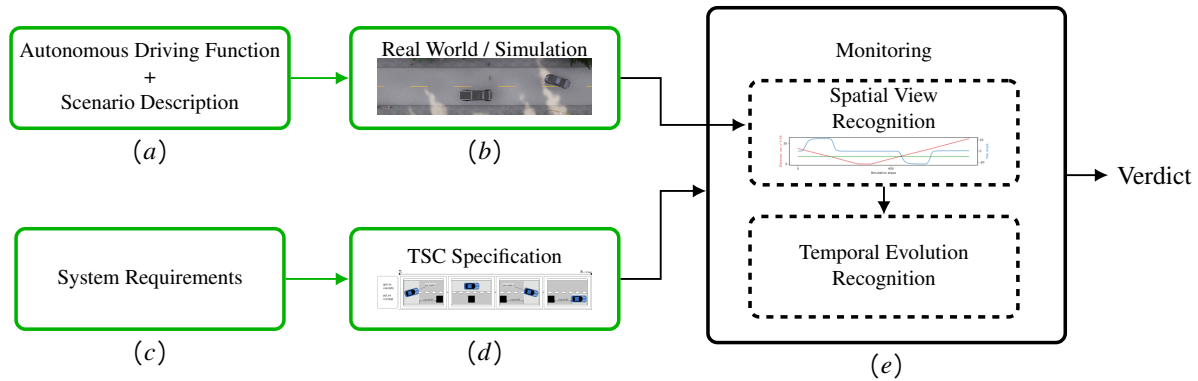
Figure 2: TSC runtime monitoring concept

In Fig. 2, we present an overview of our concept for the runtime monitoring for TSCs. The green boxes (*a-d*) and arrows describe artifacts and methods that we consider as given. The contents of (*a*) an autonomous driving function (ADF) and scenario description, (*b*) corresponding runs in a simulation or real world environment, and (*c*) system requirements are already present in the scenario-based testing approach. Based on our discussion in Sec. 3, we propose the use of TSCs for the formalization of system requirements (*d*). The focus of this section is on the dashed black boxes (*e*). The TSC language specifies spatial and temporal aspects explicitly on different levels through Spatial Views and Chart structure respectively. Our concept exploits this property by analyzing them separately through Spatial View Recognition (SVR) and Temporal Evolution Recognition (TER), at runtime. Together, SVR and TER build the runtime monitor. Following, we describe SVR and TER using our example TSC (Fig. 1) and discuss upcoming challenges for the successful realization of our concept.

We define SVR as a method to identify which Spatial Views are satisfied at any specific point of time. SVR is provided, as input, with sensor data based attributes (e.g., a car's yaw angle) extracted from real world/simulation sensors. Additionally, SVR has access to a collection of spatio-temporal system requirements specified using TSCs along with a unique ID for every occurring Spatial View. It serves as a connection between formal specifications and physical/virtual sensor information. The SVR continually produces as output the set of IDs of the Spatial Views that are currently satisfied. Here it should be noted that the SVR output at a given time can include Spatial Views from multiple TSCs or may even be empty, if no Spatial View is satisfied. Additionally, each output is annotated with a timestamp.

If the required attribute values are not provided directly, then the SVR must calculate them internally from the available input data. For our example TSC, the longitudinal distance between car and obstacle as well as the location of the car on the road and with respect to the obstacle will need to be calculated. Spatial Views encode conjunctions of arithmetic constraints on real valued attributes. Once the calculation of attributes is finished, SVR identifies the set of Spatial Views that are currently satisfied and passes this information to TER.

TER iteratively receives the output of SVR as input. Based on this analysis of the spatial aspect of the system requirements, TER analyzes the temporal aspect and decides whether the system run, up to the current time, is *satisfying*, *violating* or *inconclusive* and produces the appropriate output. For our example TSC, we have done this using a timed automaton. Its construction is based on keeping track of which initial segment(s) of the TSC, the system run up to the current time might correspond to in the following sense: A (prefix of a) system run of duration *d fits the TSC up to j* for $j \in \{0,1,2,3,4\}$, iff there exist times $0 = t_0 < t_1 < \ldots < t_{j-1} \le d < t_j$ such that for $i \in \{1,2,\ldots,j\}$ on $[t_{i-1},t_i)$, the *i*-th Spatial

View is satisfied. By definition, a run satisfies the TSC exactly if there is a prefix of this run of duration less than 45, which fits the TSC up to 4. It can be shown that a prefix of a run is: *satisfying*, exactly if the prefix already satisfies the TSC; *inconclusive*, if it is not satisfying, has a duration less than 45 and fits the TSC up to j for some $j < 4$, and *violating* otherwise.

The runtime monitor producing the output *satisfying*, for a specific run, implies that the ADF successfully performs the specified maneuver. In case the output produced is *violating*, this implies that the ADF violates the system requirements. Additionally, the time at which the output *violating* is produced identifies the point of time during the run at which the requirement is first violated. Hence, connecting the runtime monitor to a source of relevant data allows first statements to be made about the validity of the ADF.

To test the feasibility of our concept, we carried out exemplary runs in the autonomous driving simulator CARLA [10]. The runs start with a car approaching an obstacle such that the example requirement as described in Sec. 3 is activated. We continuously computed relevant attribute values from available simulation data and visualized their evolution for one such run in Fig. 3. The red graph represents the distance between car and the stationary obstacle over time, while the blue graph represents variation of the car's yaw angle over time. The green graph describes, at each point of time, the safe distance of the car to the obstacle as defined in the system requirement. Above these graphs, we depict images from the simulation for selected points of time to illustrate the simulation run. The run will be compared against the Spatial Views of the TSC shown in Fig. 1. Their semantics remain the same as described in Sec. 3. The orange lines specify the times during the run at which a particular Spatial View is satisfied. This example shows the possibility to map input data to Spatial Views. Using the output from SVR, TER is able to evaluate the temporal aspect. Hence, SVR and TER together constitute a monitor for complex automotive system requirements specified as TSCs.
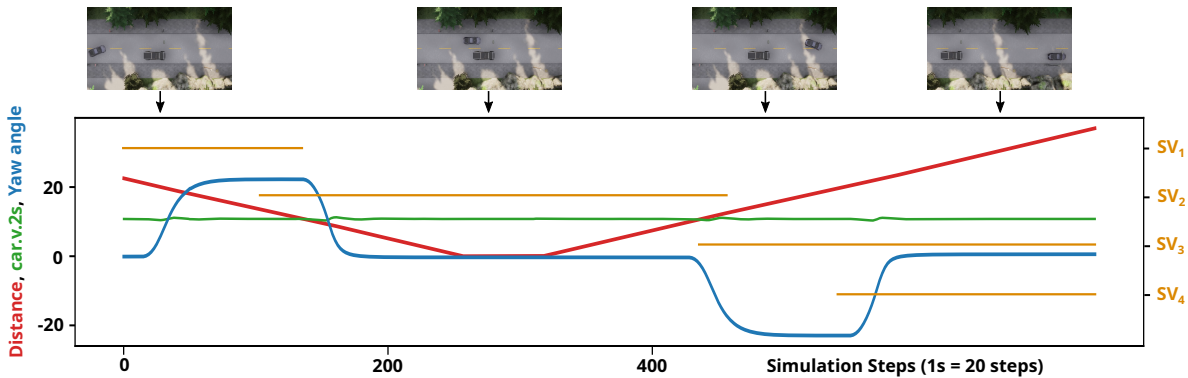

Figure 3: TSC-satisfying simulation run

Discussion. We now discuss current research challenges that have been identified for the successful implementation of our concept. Execution of SVR on resource-constrained hardware at runtime is currently limited by the demand for time and resource efficient calculation of relevant attribute values and performing satisfaction checks for a potentially large set of individual Spatial Views.

Furthermore, to ensure (efficient) monitorability of requirements, there is a need to consider the available sensor data when choosing specific attributes for the specification.

The construction of TERs for general Chart structures and Headers is also a current topic of research. We have constructed TERs for some Charts using timed automata and are currently considering the use of Petri Nets in a more general construction. It needs to be investigated for which subsets of the TSC language these approaches are feasible and further, which subset of the TSC language is monitorable in general. For this purpose, Chart structures using the operations choice, concurrency, negation and implication will need to be considered.

Overall, SVR and TER must be efficiently implemented to be useful on embedded hardware. The efficiency of the monitor can be further improved by additional communication between SVR and TER, e.g., by providing feedback to SVR about which specific Spatial Views are relevant for future verdicts.

5   Conclusion & Future Work

In this paper, we investigated and consequently demonstrated the suitability of the Traffic Sequence Chart (TSC) language for formalizing complex automotive system requirements obtained from multiple stakeholders in the context of autonomous driving function (ADF) validation. We also presented a concept towards runtime monitoring of complex spatio-temporal system requirements, visually specified as TSCs and tested its feasibility. Our novel concept takes advantage of the TSC structure by separating spatial and temporal aspects of complex system requirements to support system validation. For TSC runtime monitoring, we introduce the monitor components, Spatial View Recognition (SVR) and Temporal Evolution Recognition (TER), along with their interfaces. Our concept bridges a gap between visual specifications and runtime monitoring of complex automotive system requirements and combines their advantages. The concept provides a first step towards application of TSC monitoring for validation of ADFs. The availability of runtime monitors for TSCs could significantly increase automation and efficiency of scenario-based testing methods, and thereby support the homologation of ADFs. The concept can also be useful for plausibilization, safeguarding, and training of AI-based driving functions.

For a future application of our concept to scenario-based testing of ADFs, the challenges discussed in Sec. 4 need to be addressed. Also, integrated monitoring of TSCs that characterize proper driving or critical traffic scenarios could be used to auto-trigger switching of automated driving levels, provide warning to the driver or to the driving function developers. Additionally, our concept can also help to improve the efficiency of AI driving function training, in particular for Active Learning and Reinforcement Learning [21], e.g., by supplying an additional criterion for early termination of training runs. Last but not least, the above application cases are not limited to only the automotive domain. TSCs may also be used for the formalization of requirements from other traffic domains such as maritime, railway and aerospace. While some of these may benefit from a domain-specific extension of the Spatial View formalism, our monitoring approach is, in principle, independent of the domain.

References

[1] Ezio Bartocci, Jyotirmoy V. Deshmukh, Alexandre Donzé, Georgios Fainekos, Oded Maler, D. Ničković & Sriram Sankaranarayanan (2018): *Specification-Based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications*. In: Lectures on Runtime Verification, DOI: `10.1007/978-3-319-75632-5_5`.

[2] Andreas Bauer, Martin Leucker & Christian Schallhart (2011): *Runtime Verification for LTL and TLTL*. ACM Trans. Softw. Eng. Methodol. 20, DOI: `10.1145/2000799.2000800`.

[3] Jan Becker, Tjark Koopmann, Birte Neurohr, Christian Neurohr, Lukas Westhofen, Boris Wirtz, Eckard Böde & Werner Damm (2022): *Simulation of Abstract Scenarios: Towards Automated Tooling in Criticality Analysis*, pp. 42–51. DOI: `10.5281/zenodo.5907154`.

[4] Jan Steffen Becker (2020): *Partial Consistency for Requirement Engineering with Traffic Sequence Charts*. In: *Combined Proceedings of the Workshops at Software Engineering 2020 Co-located with the German Software Engineering Conference 2020 (SE 2020), Innsbruck, Österreich, March 05, 2020*, CEUR Workshop Proceedings 2581. Available at `http://ceur-ws.org/Vol-2581/ase2020paper1.pdf`.

[5] Werner Damm, Stephanie Kemper, Eike Möhlmann, Thomas Peikenkamp & Astrid Rakow (2017): *Traffic Sequence Charts - From Visualization to Semantics*. AVACS Technical Report (117). Available at `http://www.avacs.org/fileadmin/Publikationen/Open/avacs_technical_report_117.pdf`.

[6] Werner Damm, Eike Möhlmann, Thomas Peikenkamp & Astrid Rakow (2018): *A Formal Semantics for Traffic Sequence Charts*, pp. 182–205. Springer International Publishing, Cham, DOI: `10.1007/978-3-319-95246-8_11`.

[7] Jyotirmoy V. Deshmukh, Alexandre Donzé, Shromona Ghosh, Xiaoqing Jin, Garvit Juniwal & Sanjit A. Seshia (2017): *Robust Online Monitoring of Signal Temporal Logic*. Form. Methods Syst. Des., p. 5–30, DOI: `10.1007/s10703-017-0286-7`.

[8] Alexandre Donzé, Thomas Ferrère & Oded Maler (2013): *Efficient Robust Monitoring for STL*. In: *Computer Aided Verification*, Springer International Publishing, pp. 264–279, DOI: `10.1007/978-3-642-39799-8_19`.

[9] Alexandre Donzé & Oded Maler (2010): *Robust Satisfaction of Temporal Logic over Real-Valued Signals*. In: *International Conference on Formal Modeling and Analysis of Timed Systems*, Springer, pp. 92–106, DOI: `10.1007/978-3-642-15297-9_9`.

[10] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez & Vladlen Koltun (2017): *CARLA: An Open Urban Driving Simulator*. In: *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, DOI: `10.48550/ARXIV.1711.03938`.

[11] United Nations Economic Commission for Europe (2021): *UN Regulation No 157 – Uniform provisions concerning the approval of vehicles with regards to Automated Lane Keeping Systems [2021/389]*. Available at `http://data.europa.eu/eli/reg/2021/389/oj`.

[12] Thomas Ferrère, Oded Maler, Dejan Ničković & Amir Pnueli (2019): *From Real-Time Logic to Timed Automata*. J. ACM 66, DOI: `10.1145/3286976`.

[13] foretellix (2019): *Open M-SDL*. `https://www.foretellix.com/open-language/`. Accessed: 2022-04-22.

[14] Daniel J. Fremont, Tommaso Dreossi, Shromona Ghosh, Xiangyu Yue, Alberto L. Sangiovanni-Vincentelli & Sanjit A. Seshia (2019): *Scenic: a language for scenario specification and scene generation*. In: *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, ACM, DOI: `10.1145/3314221.3314633`.

[15] Iman Haghighi, Austin Jones, Zhaodan Kong, Ezio Bartocci, Radu Gros & Calin Belta (2015): *SpaTeL: A Novel Spatial-Temporal Logic and Its Applications to Networked Systems*. In: *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, HSCC '15, Association for Computing Machinery, p. 189–198, DOI: `10.1145/2728606.2728633`.

[16] Stefan Jakšić, Ezio Bartocci, Radu Grosu, Reinhard Kloibhofer, Thang Nguyen & Dejan Ničković (2015): *From Signal Temporal Logic to FPGA Monitors*. In: *2015 ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, IEEE, pp. 218–227, DOI: `10.1109/MEMCOD.2015.7340489`.

[17] Andrea Leitner (2020): *ENABLE-S3: Project Introduction*, pp. 13–23. Springer International Publishing, Cham, DOI: `10.1007/978-3-030-14628-3_4`.

[18] Tengfei Li, Jing Liu, JieXiang Kang, Haiying Sun, Wei Yin, Xiaohong Chen & Hui Wang (2020): *STSL: A Novel Spatio-Temporal Specification Language for Cyber-Physical Systems*. In: *2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS)*, pp. 309–319, DOI: `10.1109/QRS51102.2020.00048`.

[19] Yifang Ma, Zhenyu Wang, Hong Yang & Lin Yang (2020): *Artificial Intelligence Applications in the Development of Autonomous Vehicles: A Survey*. IEEE/CAA Journal of Automatica Sinica 7, pp. 315–329, DOI: `10.1109/JAS.2020.1003021`.

[20] Rupak Majumdar, Aman Mathur, Marcus Pirron, Laura Stegner & Damien Zufferey (2021): *Paracosm: A Test Framework for Autonomous Driving Simulations*. In: *Fundamental Approaches to Software Engineering*, Springer International Publishing, Cham, pp. 172–195, DOI: `10.1007/978-3-030-71500-7_9`.

[21] Piergiuseppe Mallozzi, Ezequiel Castellano, Patrizio Pelliccione, Gerardo Schneider & Kenji Tei (2019): *A Runtime Monitoring Framework to Enforce Invariants on Reinforcement Learning Agents Exploring Complex Environments*. In: *2019 IEEE/ACM 2nd International Workshop on Robotics Software Engineering (RoSE)*, pp. 5–12, DOI: `10.1109/RoSE.2019.00011`.

[22] Laura Nenzi, Luca Bortolussi, Vincenzo Ciancia, Michele Loreti & M. Massink (2017): *Qualitative and Quantitative Monitoring of Spatio-Temporal Properties with SSTL*. Logical Methods in Computer Science 14, DOI: `10.23638/LMCS-14(4:2)2018`.

[23] Christian Neurohr, Lukas Westhofen, Tabea Henning, Thies de Graaff, Eike Möhlmann & Eckard Böde (2020): *Fundamental Considerations around Scenario-Based Testing for Automated Driving*. In: *2020 IEEE Intelligent Vehicles Symposium (IV)*, pp. 121–127, DOI: `10.1109/IV47402.2020.9304823`.

[24] PEGASUS: *PEGASUS Project*. `https://www.pegasusprojekt.de/de/`. Accessed: 2022-04-25.

[25] Maike Schwammberger (2021): *Distributed Controllers for Provably Safe, Live and Fair Autonomous Car Manoeuvres in Urban Traffic*. Ph.D. thesis, Oldenburg University, Germany. Available at `https://oops.uni-oldenburg.de/4961/`.

[26] Ekim Yurtsever, Jacob Lambert, Alexander Carballo & Kazuya Takeda (2020): *A Survey of Autonomous Driving: Common Practices and Emerging Technologies*. IEEE Access 8, DOI: `10.1109/ACCESS.2020.2983149`.

[27] Eleni Zapridou, Ezio Bartocci & Panagiotis Katsaros (2020): *Runtime Verification of Autonomous Driving Systems in CARLA*. In Jyotirmoy Deshmukh & Dejan Ničković, editors: *Runtime Verification*, Springer International Publishing, Cham, pp. 172–183, DOI: `10.1007/978-3-030-60508-7_9`.