Multiform Logical Time & Space for Specification of Automated Driving Assistance Systems: Work-in-Progress

Qian Liu, Robert de Simone, Xiaohong Chen, Jing Liu

Multiform Logical Time & Space for Specification of Automated Driving Assistance Systems: Work-in-Progress

Qian Liu*, Robert de Simone†, Xiaohong Chen*, and Jing Liu*
*Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai 200062, China
Email: qianliu@stu.ecnu.edu.cn, xhchen@sei.ecnu.edu.cn, jliu@sei.ecnu.edu.cn
†INRIA Sophia Antipolis Méditerranée, Sophia Antipolis Cedex, France
Email: Robert.de_Simone@inria.fr

***Abstract*—Due to the mobility of autonomous vehicles and changing context through time, the constraints in safe driving rules specification need to be irregularly updated for monitoring the trajectory plan. This is not assumed in the Spatial-Temporal Logic. This paper proposes a novel approach to build the specification of assume-guarantee constraints providing safe driving rules related to time and space, in the context of Automated Driving Assistance Systems (ADAS). The novelty lies in that the specification adopts Multiform Logical Time to express the time constraints and provides spatial events generated by interactions on area trajectory for expressing space constraints. We propose the safe specification patterns at a high-level that provide the required expressiveness for safe driving rules. In these patterns, logical time provides the power of parameterization to express rules, before instantiation in low-level simulation contexts. The specification finally could be used to generate monitors that are executed on lower-level simulation engines with physical and topological features.**

***Index Terms*—Multiform Logical Time, Space, Mobile Cyber-Physical System, Safe Specification Pattern**

I. Introduction

Advanced Driving Assistance System (ADAS) is a typical application of the Mobile Cyber-Physical System (MCPS). ADAS employs various sensors at any time when the car is running to perceive the surrounding environment, collect data, identify, detect, and track static and dynamic objects, thereby monitoring safe driving rules to provide safe driving policies and effectively avoid traffic accidents. Unfortunately, autonomous vehicles have already caused casualties [1]. How to ensure the safety of driving rules for autonomous vehicles with respect to time and space poses grand challenges.

There are already some researches on the specification considering both space and time properties. Most of them are spatial-temporal logics such as Spatio-temporal Logic for closure space (STLCS) [2], Spatio-temporal logical with S4u and Temporal Logical (STL) [3], and Signal Spatio-Temporal Logic(SSTL) [4]. These logics are obtained by extending temporal logic or spatial logic. Moreover, some researchers focus on the safety assurances at lower-level. For example, recently, Responsibility-Sensitive Safety (RSS) [5], a white-box and interpretable mathematical model, is proposed by Mobileye to validate the constraints of safe driving rules at the low-level physical simulator, such as Carla [6], Apollo [7], or Webots/SUMO [8]. However, due to the mobility of autonomous vehicles, where elements (other cars, road pieces, traffic signs) may dynamically enter and exit the scene, the logic and low-level safety assurances cannot deal with dynamically changing constraints in a unified way.

In this paper, we present a safety specification pattern, a high-level abstraction of the safe driving rules, to build specifications for different scenarios. The pattern is instantiated with the trajectory plan, fresh scene as the input which is provided by the low-level physical simulator. Due to the changing scenes, the constraints in specifications are irregularly updated. The constraints in safe driving rules specification relate to time and space. To express the space constraints, we present area trajectory (the area moves through continuous-time), and the spatial events generated by interactions on area trajectories. For time constraints, we adopt multiform logical time as the parameters and extend CCSL (Clock Constraint Specification Language) [9] expressions to express more complex time constraints. Moreover, CCSL relies on the notion of logical clocks [10], which are commonly used to specify both partial orders and causal relationships on events. We thus combine time constraints and space constraints with CCSL.

II. Framework

Figure 1 presents the framework of our approach. We provide a safe specification pattern to build safe driving rules specification by instantiation with the fresh scene provided by a low-level physical simulator. The observer could monitor the trajectory plan with the specification, and then output whether it is safe or not. The safe driving rules specification consists of time constraints and space constraints, which are all expressed with CCSL specifications. In the following, we first present how to express the time constraints and space constraints in the specification, and the safe specification pattern expression. Then a simple case is used to show how to instantiate the specification pattern with a scene. Finally we present how to dynamically update the constraints on our monitor while receiving a fresh scene.

TABLE I
SYNTAX AND SEMANTICS OF SPATIAL RELATIONS AND EVENTS

Type	Syntax		Semantics (Area A and B move through time, $\epsilon \in \mathbb{R}^+$ is small enough)
Spatial Relations	$A \triangleq B$	$\Longleftrightarrow$	$\forall p \in \mathbb{U}, p \in A \Longleftrightarrow p \in B$
	$A \sqsubseteq B$	$\Longleftrightarrow$	$\forall p \in \mathbb{U}, p \in B \Longrightarrow p \in A$
	$A \ominus B$	$\Longleftrightarrow$	$\forall p \in \mathbb{U}, \neg(p \in B \wedge p \in A)$
	$A \sqcap B$	$\Longleftrightarrow$	$\exists p \in \mathbb{U}, p \in A \wedge p \in B$
Spatial Events	$join(A, B)$	$\Longleftrightarrow$	$\{t \in \mathbb{R}^+ \mid A(t) \ominus B(t) \wedge A(t+\epsilon) \sqcap B(t+\epsilon)\}$
	$disjoin(A, B)$	$\Longleftrightarrow$	$\{t \in \mathbb{R}^+ \mid A(t) \sqcap B(t) \wedge A(t+\epsilon) \ominus B(t+\epsilon)\}$
	$include(A, B)$	$\Longleftrightarrow$	$\{t \in \mathbb{R}^+ \mid (A(t) \sqcap B(t) \vee A(t) \ominus B(t)) \wedge A(t+\epsilon) \sqsubseteq B(t+\epsilon)\}$
	$exclude(A, B)$	$\Longleftrightarrow$	$\{t \in \mathbb{R}^+ \mid A(t) \sqsubseteq B(t) \wedge (A(t+\epsilon) \sqcap B(t+\epsilon) \vee A(t+\epsilon) \ominus B(t+\epsilon))\}$

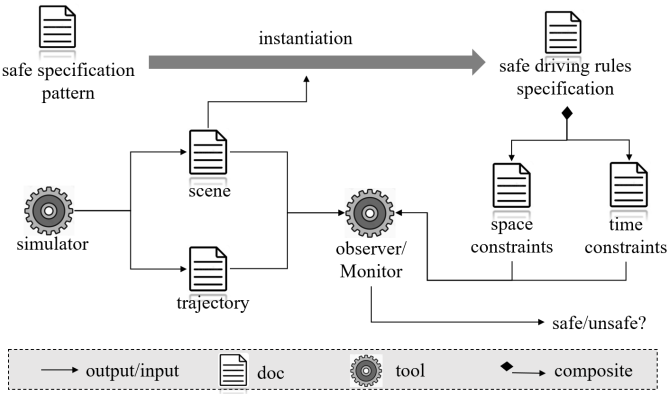

Fig. 1. Overview of our framework

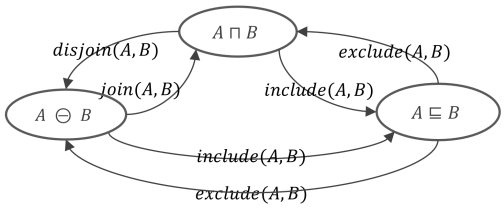

Fig. 2. Transitions between spatial relations

A. Safe Specification Pattern

Space and Time in Specification: The safe driving rules specification involves the elements in the scene, which need to be considered when constructing the specifications. In our setting, a *scene* consists of a certain number of (active elements) "players" such as cars, pedestrians, bikers, (passive elements) "objects", for instance, road lanes and crossings, traffic signs, and a specific "ego car".

Each of these generic element types define a number of area zones of relevance for safety, such as ego car includes: 1) front_collision_area; 2) front_danger_area; 3) front_warning_area; and 4) front_safe_area (and similarly for right, left, and back areas), denoted as $type_{direction}(ego)$. Then we present area trajectories in Definition 1 as these areas move through continuous-time in simulation.

Definition 1: Given an area A, the area trajectory of A is a total function $A : \mathbb{R}^+ \rightarrow 2^{\mathbb{U}}$, where $\mathbb{U}$ is the universe space, $\mathbb{R}^+$ represents the continuous (physical) time. $A(t)$ is the area A at time t.

To express the relations between these areas, we present spatial relations, including equal ($\triangleq$), belongs to ($\sqsubseteq$), disjoint ($\ominus$), overlap ($\sqcap$). This allows us to define spatial events (*join*, *disjoin*, *include*, *exclude*), generated by interactions between area trajectories, to express transition between the relations, as shown in Figure 2. The syntax and semantics of spatial relations and events are as shown in Table I.

The specification also needs CCSL expressions to express the more complex time-related requirements, such as "within some seconds", "after some seconds", "after some meters", and so on. We thus extend two CCSL expressions $withIn$ and $after$.

- $c_1 \triangleq c_2\ withIn\ d$, where d is multiform logical time. It means if c_2 ticks, then the next tick of c_1 should occur within the time d.
- $c_1 \triangleq c_2\ after\ d$, where d is multiform logical time. It means if c_2 ticks, then the next tick of c_1 should occur after the time d.

Expression of Pattern: Based on the spatial events, we can use logical time events in simple safe specification patterns to express time and space constraints. A simple pattern uses Esterel as the formalism expression, as shown in Table II. In the pattern, there are three template events (logical clock), $StartCheck$, $Recover$, and $Failure$, which are instantiated with the scene as the input and CCSL expressions in the low-level physical simulation context. It states that once a $StartCheck$ temporal condition is detected, initializing the realization that there might be a latent problem to be monitored, then one awaits for any of two subsequent events: a positively resolving $Recover$ event, stating that things went back to normal, or a catastrophic $Failure$ event, notifying a potential collision of a timed-out delay for remaining too long in the problematic model. In CCSL constraints, the pattern states that *Failure* never occur, either *StartCheck* or *Recover* ticks and *StartCheck* ticks before *Recover*.

B. Instantiation

In the ADAS context, we need to consider various scenarios and elements, so we build predefined safe driving rules specification for each typical scenario or each type of player/object

TABLE II
SAFE SPECIFICATION PATTERN AND FORMALISM EXPRESSION

Safety Specification Pattern
Once *StartCheck* temporal condition is detected, a monitor is initialized to detect possible latent problems. The monitor stops either with a positive issue, when receiving the *Recover* event, or a negative one, when receiving a *Failure* event (collision or time-out, for instance).
Esterel formalism expression
await $StartCheck$ ***then (await*** $Recover$***) abort*** $Failure$

with the pattern by using CCSL specification to instance the three template events. Thus, once receiving a fresh scene, ADAS recognizes the type and areas of objects/players in the scenario precisely, if the specification of this scene is in predefined specifications, then it generates the space and time constraints related to the objects/players and scenario from the predefined specification. When there is no predefined specification for the scene, ADAS need the machine learning to help it build specification for the current scene and store into the knowledge repository, and then generating the constraints.

Figure 3 presents a simple scenario. It shows that ego car following the car $front\_car$. In this case, we consider three states (safe, danger, collision) of ego car, and thus use two type areas (danger and collision) to build the safe driving rules specification for the *front* direction, which is expressed as:

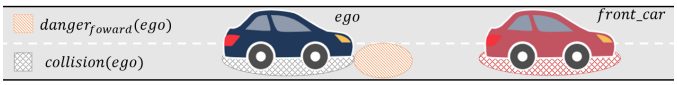

Fig. 3. Example of safe distance

$$1.StartCheck = join(danger_{front}(ego), front\_car)$$
$$2.Recover = disjoin(danger_{front}(ego), front\_car) \triangleq join(danger_{front}(ego), front\_car)\ withIn\ max\_resp\_time,$$
$$3.Failure = join(collison(ego), front\_car) \triangleq join(danger_{front}(ego), front\_car)\ after\ max\_resp\_time,$$

which states that once the front danger area of ego car overlaps with $front\_car$, then the front danger area should disjoint within max response time, otherwise, a collision would happen. In this specification, "the danger area of ego car overlaps with another car" is the distance between the ego car and the front cars is less than the safe distance. So the "recovery" is the distance between the ego car and the front car greater than the safe distance within a max response time, otherwise, the ego car will collide with the front car. When instantiation for this scene, ADAS needs to provide the instance of the front car and ego car and compute the max response time.

C. Monitor

The specification could be used to generate monitors that are executed on lower-level simulation engines with physical and topological features. With the areas and area trajectories of the elements in the scene of ego car, we could generate the spatial events. Therefore, we can use CCSL to combine space constraints and time constraints and monitor whether the trajectory plan is safe or not. Different from traditional monitor generation, the constraints in the specification irregularly updated requires automated monitor generation and update. Traditionally, once receiving fresh scene information, ADAS rebuilds everything to generate the constraints, but it takes too many resources and is inefficient. Using the updating constraints, our monitor only concerns about the newly appeared elements instead of everything. Therefore, it can be expected that the performance of our monitor would be better.

III. CONCLUSION

In this paper, we present a safe specification pattern in Esterel formalism expression to building specification of safe driving rules in the context of automated driving. The pattern is instantiated by CCSL specifications. The space constraints in the specification are expressed with the primitive events/clocks that are generated by encounters of well-defined area trajectories. We thus combine space and time constraints with CCSL, thereby providing a unified approach to deal with time and space constraints. The specification finally could be used to generate monitors that are executed on lower-level simulation engines with physical and topological features.

ACKNOWLEDGMENT

This work is partially supported by funding under National Key Research and Development Project 2017YFB1001800, NSFC 61972150.

REFERENCES

[1] T. Economist, "Why uber's self-driving car killed a pedestrian," 2017.

[2] V. Ciancia, G. Grilletti, D. Latella, M. Loreti, and M. Massink, "An experimental spatio-temporal model checker," in *Software Engineering and Formal Methods - SEFM 2015 Collocated Workshops: ATSE, HOFM, MoKMaSD, and VERY*SCART, York, UK, September 7-8, 2015, Revised Selected Papers*, ser. Lecture Notes in Computer Science, D. Bianculli, R. Calinescu, and B. Rumpe, Eds., vol. 9509. Springer, 2015, pp. 297–311.

[3] D. Gabelaia, R. Kontchakov, A. Kurucz, F. Wolter, and M. Zakharyaschev, "Combining spatial and temporal logics: expressiveness vs. complexity," *Journal of Artificial Intelligence Research*, vol. 23, pp. 167–243, 2005.

[4] L. Nenzi, L. Bortolussi, V. Ciancia, M. Loreti, and M. Massink, "Qualitative and quantitative monitoring of spatio-temporal properties," in *Runtime Verification*. Springer, 2015, pp. 21–37.

[5] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *arXiv preprint arXiv:1708.06374*, 2017.

[6] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[7] B. Ltd., "Apollo," https://apollo.auto/.

[8] C. Ltd., "Webots for automobiles," https://cyberbotics.com/doc/automobile/sumo-interface.

[9] C. André and F. Mallet, "Specification and verification of time requirements with ccsl and esterel," in *Proceedings of the 2009 ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems*, 2009, pp. 167–176.

[10] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Communications of the ACM*, vol. 21, no. 7, pp. 558–565, 1978.