Journal of Internet Services and Applications, 2026, 17:1, doi: 10.5753/jisa.2026.5913

Safe and Protected: Combining Protection Mechanism with Safety Verification In Autonomous Vehicles

José Luis Conradi Hoffmann [**Federal University of Santa Catarina** | ***hoffmann@lisha.ufsc.br***]
Antônio Augusto Fröhlich [**Federal University of Santa Catarina** | ***guto@lisha.ufsc.br***]
Marcus Völp [**University of Luxembourg** | ***marcus.voelp@uni.lu***]
Paolo Milazzo [**University of Pisa** | ***paolo.milazzo@unipi.it***]

Software/Hardware Integration Lab, Institute of Informatics and Statistics, Federal University of Santa Catarina, P.O. Box 476, Campus Universitário, s/n, Trindade, Florianópolis, SC, 88040-900, Brazil.

Received: 14 April 2025 • **Accepted:** 01 October 2025 • **Published:** 21 January 2026

Abstract Protection mechanisms, also known as security mechanisms, in automotive systems are proactive components that continuously monitor vehicle signals to detect early signs of potential faults. For autonomous vehicles, it is essential that safety models, such as Responsibility-Sensitive Safety (RSS), which governs longitudinal and lateral safety, account for these mechanisms to enable timely and effective countermeasures against imminent actuation failures. A typical example is the proactive application of braking to increase longitudinal distance and mitigate the risk of losing braking capability. In this paper, we present a data-centric approach for modeling protection mechanisms using the SmartData framework, which facilitates the automatic derivation of safety properties for real-time formal verification via a Safety Enforcement Unit (SEU). We introduce extensions to RSS proper response strategies, enabling them to anticipate potential actuation constraints by leveraging shared internal states of protection mechanisms and a predictive time-to-trigger metric. We formally demonstrate that our approach preserves compliance with the original RSS safety guarantees by extending its inductive proof structure. Furthermore, we validate the feasibility of our solution through empirical evaluation, showing that the embedded formal verification can automatically extract properties from publish-subscribe message systems and operate at runtime with minimal overhead (less than 1% of platform processing capacity). Finally, we integrate our approach with RSS and a representative protection mechanism within the CARLA simulator to showcase its effectiveness in a realistic autonomous driving environment.

Keywords: Protection Mechanisms, Autonomous Vehicles, Formal Methods.

1 Introduction

Safety assurance and verification processes, such as those incorporated in the ”V” development model exemplified in International Organization for Standardization [2018] 26262, are well-established and indispensable practices for ensuring safety during the design and development of Autonomous Vehicles (AVs) components, as demonstrated by Koopman and Wagner [2016] and Cui *et al*. [2019] in their reviews of the challenges in AV safety. However, as demonstrated by Althoff and Magdici [2016] with the proposal of Responsibility-Sensitive Safety (RSA), and Shalev-Shwartz *et al*. [2017], with the proposal of Responsibility-Sensitive Safety (RSS) the domain of autonomous vehicles, offline testing alone is insufficient to address the inherent dynamism and complexity of safety-critical applications like AVs. These Safety frameworks for AVs have been proposed to facilitate safety verification. These strategies incorporate online verification mechanisms to assess the feasibility of motion planning in accordance with safety parameters. For instance, RSS evaluates safety by ensuring minimum distances relative to surrounding agents to enable corrective actions in response to potentially hazardous decisions made by motion planning algorithms.

Nonetheless, these approaches are predicated on static assumptions regarding vehicle actuation, characterized by fixed minimum and maximum operational limits. They do not account for faults in perception or actuation subsystems, nor do they consider errors from artificial intelligence components involved in object recognition and tracking. Critical perception anomalies—such as the failure to detect or the misclassification of objects—can lead to dangerous scenarios and are typically mitigated through redundancy and cooperative perception, as demonstrate in the works of Huang and Tan [2016] and Lucchetti *et al*. [2023] in the scope of redundancy in AV software stacks, and by Kim *et al*. [2015] and de Lucena and Augusto Fröhlich [2022] in the scope of Vehicle-to-Everything (V2X) applications. However, these countermeasures are not exhaustive in addressing all safety concerns for AVs. Situations may arise in which an AV loses the capacity to brake, accelerate, or steer, due to either the activation of protection mechanisms or component malfunctions.

Failures within actuation systems may breach the predefined safety boundaries, undermining the assumed operational limits and creating a misleading perception of safety, when in fact, the vehicle should transition to a safe operational mode. Consequently, dynamic safety models that adapt to fluctuating operational constraints are essential for advancing toward higher automation levels defined by International Society of Automotive Engineers [2021], specifically Levels

4 and 5. Hoffmann and Fröhlich [2022] introduced an approach that integrates safety model and formal verification with Data-Centric design principles pertaining SmartData, also proposed by Hoffmann and Fröhlich [2025]. SmartData is a data construct with sufficient metadata and clearly defined interfaces that allows for explicitly timing and safety constraints to be defined over the data the system produces and consumes instead of the tasks they run. This method, referred to as SmartData Safety, utilizes data expiration and actuation periodicity to establish timing references. A centralized monitoring entity, designated as the Safety Enforcement Unit (SEU), leverages these timing insights to formulate a set of Signal Temporal Logic (STL) properties that detect non-responsiveness in actuators and sensors. Concurrently, data validity and safety compliance are verified through established safety models.

This paper is an extension of our prior work published in the 13th Latin-American Symposium on Dependable and Secure Computing (LADC) (see Conradi Hoffmann *et al*. [2024b]), where the main contribution was the modeling of novel SmartData constructs for protection mechanism aiming to enhance the adaptability of Safety Models, specifically RSS. Figure 1 presents an overview of the proposed approach. From the protection mechanism's specifications in SmartData, we derive a time-to-trigger metric, which serves to dynamically adjust the RSS parameters in anticipation of potential actuation losses. This allows for RSS to gracefully mitigate the impact of imminent actuation loss, avoiding abrupt interventions that would otherwise degrade the AV performance.

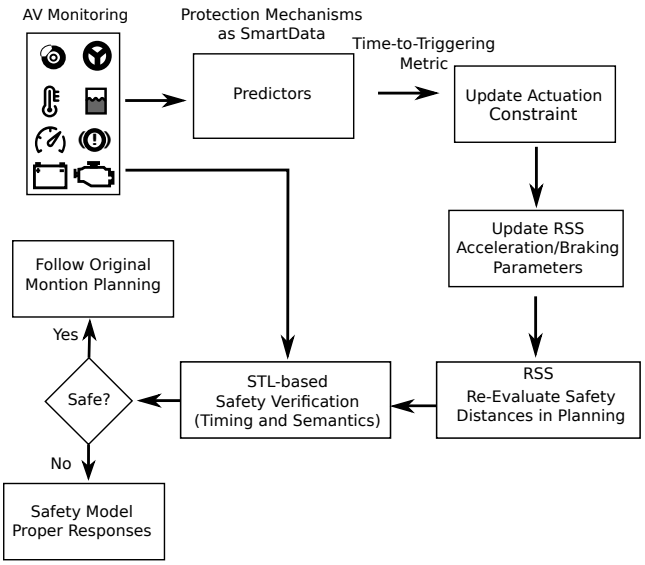

Figure 1. Overview of the proposed solution.

Therefore, the main contribution was to enhance the SmartData Safety design for autonomous vehicles to support protection mechanism, and allow for monitoring and verification through well established STL rules that are automatically derived based on the Data-Centric design of the system through SmartData. The novel actuation constraint-aware RSs was proven to remain consistent with the legal compliance criteria established by RSS.

This paper extension contribution is two-folded: first, we provide a description of the Safety Enforcement Unit implementation and demonstrate how the safety property for RSS is derived at run-time. This implementation description is supported by experimental results of the RSS monitoring and verification at run-time. This first contribution is essential to describe how other can replicate the SEU implementation. Secondly, we illustrate two AV case-studies to corroborate the RSS effectiveness when paired with the proposed constraint-aware adaptability. This contribution is of relevance for practical demonstration of the proposed solution in action and supports the claims for increasing safety of RSS by being aware of actuation constraints.

The remainder of the paper is organized as follows: Section 2 presents the related works. Section 3 presents an overview of the background knowledge that supports this work. Section 4 details the SEU key design choices that allow for low-overhead at online verification. Section 5 presents the modeling of the Protection Mechanism as SmartData and the derivation of the time-to-trigger metric. Section 6 presents the integration of Protection Mechanisms with SEU and the proof that the proposed approach complies with RSS. Section 7 presents two evaluation scenarios using simulators to corroborate the RSS effectiveness when paired with the proposed extensions, specifically exploring overtake maneuvers. Finally, Section 8 presents the final remarks of the paper.

2 Related Works

The RSS framework, introduced by Shalev-Shwartz *et al*. [2017], is a formalized safety concept that interprets the legal principle of Duty of Care through a mathematical lens. This model defines 25 key concepts that outline the responsibilities of drivers to maintain safe distances, covering scenarios from straight roads to complex, unstructured environments, as well as reactions to inappropriate behavior from nearby vehicles. To facilitate its formal verification, the RSS framework relies on specific parameters:

1) ρ, representing the reaction time attributed to the agent.

2) $a_{max,accel}$, $a_{max,brake}$, and $a_{min,brake}$, which denote the upper limits of acceleration and braking, along with the minimum braking capability, applicable to both longitudinal and lateral movements, and differentiated by agent type. We recall to two lemmas derived from Definitions 1 and 6 of the RSS model:

Lemma 1 - Safe longitudinal distance — same direction: Let c_1 be a vehicle which is behind c_2 on the longitudinal axis. Let ρ, $a_{max,brake}$, $a_{max,accel}$, $a_{min,brake}$ be as defined above. Let v_1 , v_2 be the longitudinal velocities of the cars. Then, the minimal safe longitudinal distance between the front-most point of c_1 and the rear-most point of c_2 is:

$$d_{min} = \left[v_1\rho + \frac{a_{max,accel}\rho^2}{2} + \frac{(v_1 + \rho\, a_{max,accel})^2}{2a_{min,brake}} - \frac{v_2^2}{2a_{max,brake}} \right]_+ . \quad (1)$$

where $[x]_+ = max\{0, x\}$.

Lemma 2 - Safe Lateral Distance: Let $a^{lat}_{max,accel}$ and $a^{lat}_{min,brake}$ be road boundaries for lateral acceleration and brake, and assume that a vehicle c_1 is to the left of c_2. Moreover, assume y to be the $\mu - lateralvelocity$ as in Definition 5 of RSS. Define $v_{1,\rho} = v_1 + \rho\, a^{lat}_{max,accel}$ and

$v_{2,\rho} = v_2 - \rho\, a^{lat}_{max,accel}$. The minimal safe lateral distance between the right side of c_1 and the left of c_2 is:

$$d_{min} = y + \left[\frac{v_1 + v_{1,\rho}}{2}\rho + \frac{v_{1,\rho}^2}{2a^{lat}_{min,brake}} - \left(\frac{v_2 + v_{2,\rho}}{2}\rho + \frac{v_{2,\rho}^2}{2a^{lat}_{min,brake}} \right) \right]_+ . \quad (2)$$

Conversely, RSA approaches, as the one proposed by Althoff and Magdici [2016], evaluates safety by examining potential intersections between predicted vehicle trajectories. This method involves forecasting the behavior of surrounding traffic participants to generate occupancy sets, which are then compared against the occupancy set of the ego vehicle's planned trajectory to identify possible collisions. The authors emphasize the inherent complexity of this approach due to the non-linear nature of vehicle dynamics modeling.

Multiple RSA-based approaches have been proposed for online safety verification. Pek *et al*. [2020], for instance, enhance the computational efficiency of predicting future behavior of other traffic participants by filtering out unrealistic or illegal maneuvers from other traffic participants. They integrate this filtered prediction boundary with temporal analysis to ensure the safety of intersection maneuvers. Meanwhile, Gruber and Althoff [2018] expands RSA to handle cases where initial long-term trajectories appear unsafe but may become viable as uncertainty about other vehicles' paths reduces over time. Their strategy involves executing a fail-safe motion plan, initially considering only the beginning of the motion, aiming for the shortest safe fallback trajectory. This trajectory is incrementally refined as computational resources and updated situational data become available. Orzechowski *et al*. [2019] combines RSA with RSS to address the limitations of RSA's worst-case scenario analysis. Specifically, for lane-change scenarios, RSS contributes time gap estimations (derived from safe distance calculations), and their method checks for intersections between the ego vehicle's intended path and the predicted occupancy of adjacent vehicles (using RSA) during the interval $[t_{enter}, t_{gap}]$. Ideally, this time gap t_{gap} would be standardized by legal regulation.

Building on RSS, Sidorenko *et al*. [2022] address a particular longitudinal safety issue where RSS's standard formulations fall short. This occurs when the following vehicle possesses a higher maximum braking capability than the vehicle in front. They introduce a three-step process to derive appropriate safe distance formulas for both cases — when the follower's maximum braking capability is either higher or lower than that of the lead vehicle. Their goal is to achieve reduced safe distances relative to standard RSS by leveraging actual maximum braking capacities rather than the conservative minimum values assumed by the original RSS model.

However, none of these approaches considers the influence of actuator constraints during safety evaluations.

3 Background

This section presents an overview of SmartData, STL, and the Safety Enforcement Unit concept.

3.1 SmartData

SmartData, first proposed by Fröhlich [2018], extends conventional data by incorporating metadata related to timing, location, security, and semantics, while also structuring input, transformation, and output aspects within a data-centric design framework. The key characteristics of SmartData that are pertinent to the methodology proposed in this work include:

1) *Period*: The frequency at which data is collected. This is typically derived from the timing constraints defined by the Critical System's requirements.

2) *Expiry*: This denotes the validity duration of the data, offering a measure of its freshness. Expiry plays a role in both local and global scheduling decisions. Multiple SmartData instances can reference the same physical transducer but possess distinct expiry settings, making this a per-instance property.

3) *Data Semantics*: Data items are annotated with a type descriptor that indicates whether they represent an SI physical quantity or digital data. The Unit provides semantic details such as data size and valid value ranges, which are useful for safety assessments. A data dictionary can further enrich this information by describing different applications for sensors sharing the same unit.

Dependencies in SmartData, referred to as interest relations, indicate data production dependencies in a manner akin to the publish-subscribe model. These dependencies are specified using the attributes $(unit, dev, period, expiry)$, where dev serves as a disambiguation key from the data dictionary to distinguish between sensors of the same unit type. As described by Fröhlich [2018], additional metadata such as security properties and geographical location can also be attached, but these extensions are beyond the scope of this paper.

In our previous work, Hoffmann *et al*. [2022]; Hoffmann and Fröhlich [2022], we have explored SmartData in the context of Critical Systems design, where SmartData is defined as an abstract set D of datum specifications, denoted by $\Omega_1, \Omega_2, \ldots$, with each data instance expressed as $\omega_i = (t, v)$ combining a timestamp with its value, and $\Omega_i.P$ specifying the sampling period. To express inter-data dependencies within the system, each datum specification Ω_i is associated with a dependency set Ψ_{Ω_i}, which consists of pairs $(\Omega_j, E) \in D \times \mathbb{R}$. Here, $(\Omega_j, E) \in \Psi_{\Omega_i}$ signifies that creating an instance of Ω_i requires an instance of Ω_j produced no more than E time units earlier. Thus, E captures the relative Expiry constraint of Ω_i on Ω_j. Each Ω_i may depend on multiple data sources, each potentially having a unique expiry requirement. Accordingly, we define a System S as the tuple $S = (I, T, O)$, with associated dependencies $\Psi^S = \Psi_{\Omega_1}, \ldots, \Psi_{\Omega_n}$, where $I \subset D$, $T \subset D$, and $O \subset D$, ensuring disjoint sets $I \cap T = I \cap O = T \cap O = \emptyset$. Here, $I \neq \emptyset$ represents inputs from sensor data; T encompasses transformed data within the system; and $O \neq \emptyset$ corresponds to outputs used for actuators.

For clarity in subsequent sections, we adopt the notation ψ_{Ω_i,Ω_j} to refer to the dependency $(\Omega_j, E) \in \Psi_{\Omega_i}$, where $\psi_{\Omega_i,\Omega_j}.E$ designates the expiry component of this dependency pair. Furthermore, the notation $S.T$, $S.O$, and $S.I$ are used to represent the T, O, and I components of the tuple S.

3.2 Signal Temporal Logic

STL, introduced by Maler and Nickovic [2004], is a formalism used to specify properties over dense-time, real-valued signals. STL supports the construction of runtime monitors grounded in the Real-Time temporal logic framework, specifically $MITL_{[a,b]}$. The key STL operators relevant to our discussion include:

$$\begin{aligned}(s,t) \models \varphi_1 U_{[a,b]} \varphi_2 \quad &\leftrightarrow \quad \exists t' \in [t+a, t+b] \mid (s,t') \models \varphi_2 \\ &\quad and\, \forall t'' \in [t,t'], (s,t'') \models \varphi_1 \\ (s,t) \models \Diamond_{[a,b]} \varphi_1 \quad &\leftrightarrow \quad \exists t' \in [t+a, t+b] \mid (s,t') \models \varphi_1\end{aligned}$$

Here, the until operator $U_{[a,b]}$ captures both data dependency and temporal constraints, while the eventually operator $\Diamond_{[a,b]}$ expresses the bounded time interval in which a condition must hold. STL operates over Boolean signals $\mathbb{B}$, requiring input signals to be processed through a domain-specific Boolean filter μ. This filter translates real-valued signals into Boolean values, indicating whether specific system conditions (typically expressed as inequalities) are satisfied.

4 Safety Enforcement Unit

The SEU, introduced by Hoffmann and Fröhlich [2022], works as a centralized monitor with complete awareness of all interest relationships and system data (details on its implementation are provided in Section 4). By integrating SmartData with STL, the SEU enables continuous online formal verification of system behavior. For each SmartData element $\Omega_i \in S.O \cup S.T$, we define a corresponding STL-based verification property:

$$\varphi_{\Omega_i} = \bigwedge_{\psi_{\Omega_i,\Omega_j} \in \Psi_{\Omega_i}} (\mu_{\omega_j} U_{[0,\psi_{\Omega_i,\Omega_j}.E]} \mu(\omega_i)) \tag{3}$$

In this expression, $\mu()$ is the Boolean filter tailored to the SmartData in question. Thus, for every dependency Ω_i has in the system, the SEU checks whether the expiration condition is met—specifically, whether ω_i's generation was appropriately preceded by ω_j within the allowed expiry window E.

A Safety Model can be defined as a collection of rules R that outline the operational safety conditions of the system. Evaluating the safety model involves continuously checking the system's state against each rule $r \in R$. Each rule functions similarly to a Boolean filter $\mu()$, accepting signals as input and outputting a Boolean status. If any rule indicates an unsafe state, the system is expected to initiate the appropriate countermeasures.

To integrate Safety Models into the SEU, we represent each rule $r \in R$ as an Actuator SmartData. Given that the SEU observes all system data, it can dynamically verify every rule in real time. Consequently, the SEU evaluates φ_{Ω_r} using the available (non-expired) input data and the Boolean evaluation of rule r. Should the condition φ_{Ω_r} be violated (i.e., $\mu()$ evaluates to false), the corresponding Actuator SmartData triggers the necessary safety response.

This approach enables us to manage the complexity of the Safety Model outside of STL while still performing comprehensive runtime safety verification within the SEU. The SEU's responsibility is limited to monitoring and triggering appropriate actions based on rule violations; it does not handle the execution of these actions.

The SEU operates continuously, monitoring every SmartData element that consumes data in the system, including safety rules ($r \in R$), transformers ($t \in S.T$), and actuators ($o \in S.O$). Each property is verified according to its designated period, and the corresponding actuation is triggered if a violation occurs. Formally, each SmartData $j \in S.T \cup S.O$ (noting that each rule $r \in R$ is also modeled as an Actuator SmartData) is monitored by the SEU as follows:

$$\varphi_{SEU_{\Omega_j}} = \Diamond[\Omega_j.P, \Omega_j.P]\varphi\Omega_j \tag{4}$$

This ensures that each SmartData element satisfies its safety constraints within every sampling period. Notably, when the SEU detects a safety violation, it may impact multiple actuators, even if they operate at different frequencies. For example, a braking system failure could necessitate responses from both steering and throttle systems. Effective decision-making requires considering the collective outcomes of multiple property evaluations, which is facilitated by the persistent nature of SEU monitoring. Furthermore, the continuous monitoring capability of the SEU enhances system auditability—an important feature for autonomous systems.

4.1 SEU Implementation

The SEU is implemented as a network sniffer within the SmartData Network, enforcing rules for timing verification and safety model validation. The core design of the SEU lies in its integration with SmartData via the Verifiable SmartData Interface and the use of Boolean Filters that implement both timing and safety model rules. The SEU manages trace data using the System State sample concept, controlling the triggering of verification processes and any associated actuation overrides. Figure 2 provides a detailed overview of the SEU's implementation. The SEU background is the SmartData Framework implementation described by Conradi Hoffmann *et al*. [2024c], based on the original SmartData API proposed by Fröhlich [2018]. Following the Data-Centric Design proposed by Hoffmann and Fröhlich [2025], each sensor, data transformation, and actuation is modeled as a SmartData Service that encapsulates the respective low-level interactions with transducers and transforming functions. A SmartData network implementing a publish-subscribe communication protocol that comprises the messages for Interest definition (similar to a subscribe message using SmartData metadata), Response (a sensed or transformed data being published in the network), and Control (a command for proper-responses or actuation override).

The class diagram in Figure 3 details the SEU components depicted in Figure 2. Similar to the SmartData API, the SEU components follow the Concurrent Observer design pattern proposed by Ludwich and Frohlich [2015]. The `SEU` class acts as an `Observer` of the `SmartData Network` through implementations of the SmartData Services and SmartData Network, as described by Conradi Hoffmann *et al*. [2024c]. The `SEU` reacts to *Interest* Messages via the `update()` method while ignoring *Response* Messages. Moreover, it observes

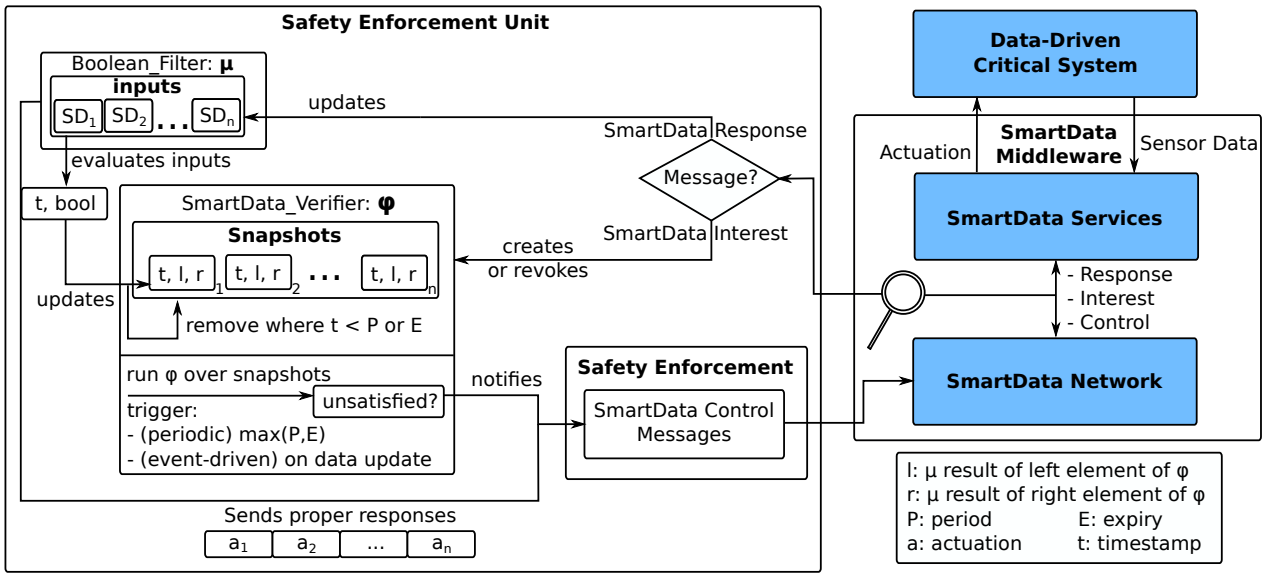

Figure 2. Overview of the components of the Safety Enforcement Unit integrated into the SmartData Framework for online verification and safety enforcement.

each `SmartData Verifier`, an implementation covering the verification of eq. (3).

The implementation considers a slightly adapted version of *Until* operator from STL. We only verify if exists a ω_j that precedes the production of ω_i with no more than E units of time, and that exists a ω_i being produced at least once every E units of time. Algorithm 1 presents the pseudo-algorithm used for each individual interest relationship.

Algorithm 1 Until.evaluate()

Given a trace τ covering at least $\psi_{\Omega_i,\Omega_j}.E$ units of time, returns if the property 3 is satisfied for an individual interest relationship ψ_{Ω_i,Ω_j}.

Require: $\tau^{\psi_{\Omega_i,\Omega_j}.E}$

$left = False$
$right = False$
while $\tau^E \neq \emptyset$ **do**
  $ST = \tau^{\psi_{\Omega_i,\Omega_j}.E}$.remove_head()
  $left = left \vee (\mu(\Omega_j)[ST.ts])$
  $right = right \vee (\mu(\Omega_i)[ST.ts])$
  if $right = True \wedge left = False$ **then**
    $right = False$
return $right = True \wedge left = True$

The SEU itself is responsible for handling the traces and combining the result for the verification of all data dependencies when verifying eq. (4) at every period. These properties check for the timing correctness of data production of all inputs of the system, which in this paper, also includes the internal state of the protection mechanisms PM, modeled inside SmartData for this specific purpose, and using $\mu(pm)$ (eq. (8)) for each specific condition check for the inputs and outputs of the protection mechanism.

The `SEU` comprises a collection of `SmartData_Verifier` and `Boolean Filters`. During `SEU` initialization, system designers provide a list of pseudo-interests, Ψ, for the system, each representing a tuple of two *SmartData::Type*, corresponding to the expected interests, but without timing information. By default, SmartData are assigned a μ `Boolean_Filter` that simply returns true whenever a data arrives, registering this value in the trace. When an *Interest* Message is sent within the `SmartData Network`, the `SEU` creates a `Verifiable_SmartData` instance for both the SmartData and interested SmartData. These `Verifiable_SmartData` are specializations of `remote` SmartData that updates upon receiving a corresponding *Response* Message (i.e., $\Omega_i.Type = (UNIT, dev)$).

A `SmartData_Verifier` is then instantiated using the default μ for the SmartData issuing the Interest Message considering the designated period P. The SmartData of interest is registered within the `SmartData_Verifier` through `register_boolean_filter_interested` with the specified expiry E. If a `SmartData_Verifier` already exists for that interest, only the registration of the new SmartData of interest is done.

Each `Boolean_Filter` acts as an `Observer` of `Verifiable_SmartData`. It holds a list of $\_interests$, represented by `SmartData::Type`, forming its $\_inputs$. Additionally, an attribute $\_sd$ specifies the `SmartData::Type` for filters that do not use $\_inputs$ (i.e., non Safety Models). These attributes are exclusive, where μ uses only $\_sd$, registered via `register_smartdata`. For any `Boolean_Filter` added to the `SEU` through `add_boolean_filter`, a corresponding `SmartData_Verifier` is created with the new `Boolean_Filter` as $interest$ and its $\_period$ attribute as $period$. Whenever a new `Verifiable_SmartData` is created by the `SEU`, the `SEU` also verifies if the *SmartData::Type* matches a *SmartData::Type* in the $\_interests$ list or $\_sd$ of every `Boolean _Filter` in `SEU`'s $\_bfs$ list. If it matches (i.e., `add_input()` is invoked and returned $True$), the *SmartData::Type* is removed from $\_interests$ list of the `Boolean_Filter` and the `Verifiable_ SmartData` is registered in the `SmartData_Verifier` using $\mu_{arrival}$ and the $\_period$ of the `Boolean_Filter` as $expiry$, every safety model and other models integrated to the SmartData is assumed to have $expiry = period$).

Each `Boolean_Filter` specialization must implement `evaluate()` and `proper_response()`. The

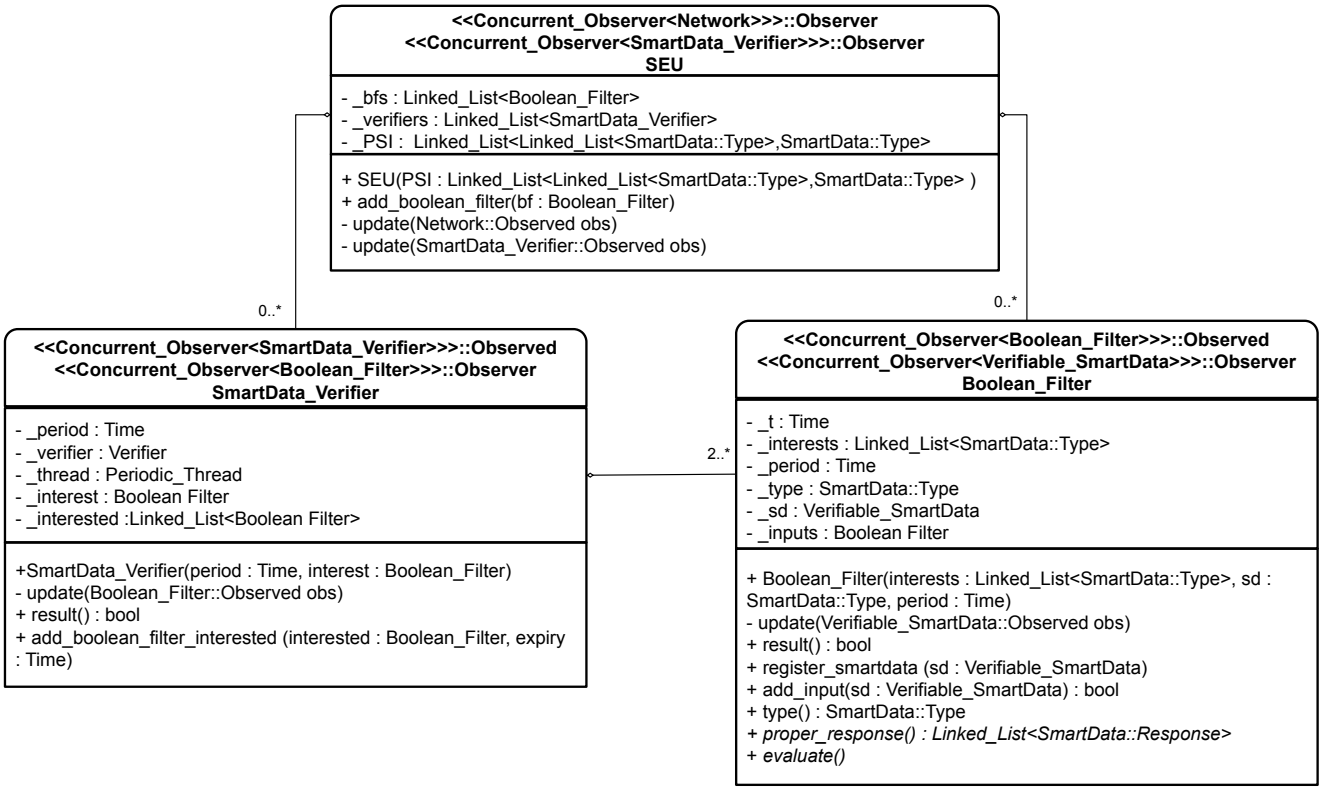

Figure 3. Class diagram of the SEU, Boolean Filter, and SmartData Verifier.

`evaluate()` method performs the μ function itself. The `Verifiable_SmartData update()` triggers a `notify()` that will call the respective `update()` of every `Boolean_Filter` that register to observe it through the method `attach()` (i.e., invoked by `add_input` of the `Boolean_Filter` class). Whenever all inputs for a `Boolean_Filter` are available (updated via `update()`), `evaluate()` is invoked. Completion of `evaluate()` prompts the `Boolean_Filter` to notify any observing `SmartData_Verifier`.

Each safety model's actuation override plan is a list of SmartData *Response* Messages that the `SEU` will issue encapsulated as *Control* Messages to trigger configurations or controls in the `SmartData Services` when the verification conditions dictate.

Each `SmartData_Verifier` includes a `Verifier` implementation, depicted in Figure 4. If the `SmartData_Verifier` has a non-zero $\_period$, it initiates a periodic thread to execute the `evaluate()` method at intervals of $\_period$. If $\_period$ is zero, the verifier operates in an event-driven manner, evaluating on data arrival.

A `Verifier` maintains an array of expiries, sized to the $interests$ list of the $interested$ `Boolean_Filter`. The realization of the property monitors is given by calling `add_input_time` of the `Verifier` implementation. Based on the $expiry$ and to be registered in the $\_expiries$ array at the given position, and the $\_period$, the Verifier will implement verification in accordance with the Algorithm 1.

To enable efficient trace handling, each `Verifier` embeds a list of `Snapshots`, which can be seen as the slice of the system trace that is of interest to the property at hand, represented by the tuple $(t, l[inputs], r)$, where t is the timestamp the SmartData instance was produced, and $l[inputs]$ and r are the result of the Boolean Filter applied to the respective element of the formula, with l being the array of SmartData of interest, and r being the SmartData interested.

Trace handling inside a `Verifier` occurs via `add_sample()` or upon `evaluate()` invocation. During `add_sample()`, the the new snapshot is inserted at the tail of the `Snapshot` list. Since the `update()` of a Boolean Filter will trigger the `update()` of a single element (given by pos or $right$ parameter), the components that were not updated are taken as $False$ (e.g.,$t, [False, \ldots, \mu, \ldots, False], False$ or $t, [False, \ldots, False], \mu$). This operation does not influences the `Verifier` result as we disregard the continuity required by the original Until operator. Finally, the head of the list (i.e., oldest element) is removed if it is too old for all the properties, i.e., $t < ct - max(\_period, max(\_expiries))$, where ct is the current time. This operation is repeated until the list head has $t \geq ct - max(\_period, max(\_expiries))$. The same trace update occurs at the triggering of the periodic evaluation by the periodic thread. This procedure works only as an optimization to improve the performance of algorithm 1, i.e., avoid high-memory consumption and low performance due to never discarding already verified information in the system trace.

4.2 RSS Safety Property

RSS relies on the dynamics of vehicles represented by longitudinal and lateral speeds and location. Thus, with the

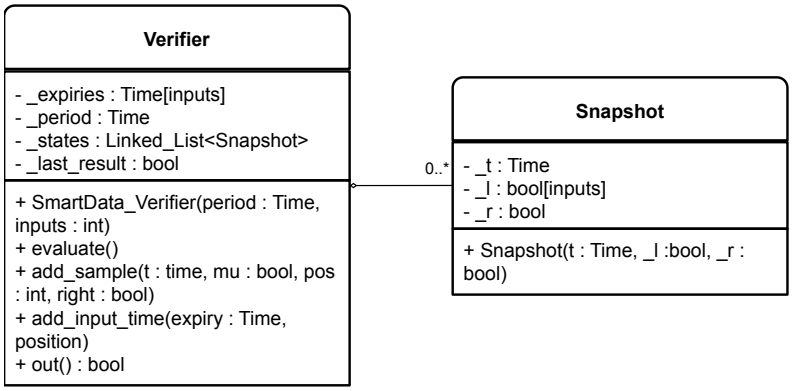

Figure 4. Class diagram of the Verifier implementing algorithm 1.

dynamics for the EGO vehicles[1] and for nearby objects (i.e., other vehicles, pedestrian, and static obstacles), the analysis of longitudinal safety is done by calculating the distance between objects and estimating the minimum safe distance according to the longitudinal speed and road parameters for maximum brake, maximum acceleration, and minimum brake. Whenever the calculated distance is bigger than the minimum distance for all objects, the EGO vehicle is in a safe state. Otherwise, the EGO vehicle must apply the minimum brake. Therefore, to model RSS's rule for Longitudinal Minimum Distance RSS_{lon}, we need to define $Dynamics_{nearby\ objects}$ and $Dynamics_{EGO}$, two SmartData composing the necessary information of dynamics to compute RSS longitudinal safety. We assume a response time $RSS.\rho = 100ms = RSS_{lon}.E = RSS_{lon}.P$, leading to RSS_{lon} being interested in both SmartData above with $P = 100 = E$. The Boolean Filter for RSS_{lon} is defined using $inputs = \{dynamics_{nearby\ objects}, dynamics_{EGO}\}$, the instances of $Dynamics_{nearby\ objects}$ and $Dynamics_{EGO}$, respectively, as:

$$\mu_{RSS_{lon}}(inputs) = \bigwedge_{o \in dynamics_{nearby\ objects}} d(o, dynamics_{EGO}) \geq d_{min}(o, dynamics_{EGO}) \tag{5}$$

where $d(o, dynamics_{EGO})$ is the function that estimates the distance between the EGO and the object, and $d_{min}(o, dynamics_{EGO}$ is given by RSS Lemma 1 through eq. (2).

Finally, the safety property for RSS's Longitudinal Minimum Distance rule RSS_{lon} is automatically derived as:

$$\varphi_{RSS_{lon}} = (\mu_{arrival}(Dynamics_{nearby\ objects}) \; U_{[0,100]} \mu_{RSS_{lon}}(inputs)) \wedge (\mu(Dynamics_{EGO}) U_{[0,100]} \mu_{RSS_{lon}}(inputs)) \tag{6}$$

The property for RSS_{lat} is analogous to the process presented above, using Lemma 2 and eq. (2) instead.

[1] An EGO vehicle is the vehicle of primary interest in the scenario (i.e., the vehicle being verified with RSS).

5 Protection Mechanisms Integration into SmartData Design

The actuation set of an AV encompasses a wide range of functionalities, from elementary tasks such as activating indicator lights to more complex control actions including acceleration, braking, and steering. In this work, we narrow our focus to the subset of actuators directly relevant to the RSS, specifically Steering ($Steering$), Braking ($Brake$), and Throttle ($Throttle$). Collectively, these are denoted as $A = Steering, Brake, Throttle$.

Safety mechanisms—often referred to as protection mechanisms—are designed not solely as reactive interventions but as proactive safeguards. These systems continuously monitor operational signals to detect patterns indicative of potential faults, enabling them to intervene preemptively, prior to fault manifestation. However, such interventions frequently result in the deactivation of vehicle components that are critical for maintaining mobility. Consequently, these protective actions may inadvertently conflict with the assumptions underlying safety models and verification frameworks, such as RSS, which typically rely on static assumptions regarding the availability of actuation capabilities (e.g., defined maximum and minimum thresholds for braking, steering, and throttle control).

Sivakumar and Mohanty [2020] defines a set of concrete examples of protection mechanism for vehicles, such as Brake System Plausibility Device (BSPD), Battery Management System (BMS), and Insulation Monitoring Device (IMD). BSPD is tasked with disabling the vehicle's power system if simultaneous motor power application and aggressive braking are sustained for longer than 0.5 seconds. BMS monitors the thermal and electrical characteristics of battery cells in electric vehicles, including parameters such as cell temperature and voltage. Upon detecting anomalies—such as overvoltage, under-voltage, over-temperature, under-temperature, or communication failures—the BMS responds by isolating the powertrain to prevent hazardous conditions. IMD is another critical protection mechanism, responsible for tracking the insulation resistance between high-voltage components and the low-voltage electrical system; should insulation resistance fall below a predefined threshold, the IMD triggers a full system shutdown. Additional protective functions include the surveillance of charging and discharging circuits,

as well as mechanisms like the brake-over-travel switch.

Beyond hardware-triggered interventions, safety mechanisms can also be informed by automated fault detection algorithms. For instance, Sangha *et al*. [2005] proposes an Artificial Neural Network (ANN) model capable of diagnosing faults related to exhaust gas recirculation and air leakage in the intake manifold pressure system. Likewise, Kong *et al*. [2019] introduces a hybrid-signal-based diagnostic method for detecting overheating within the vehicle's wiring harness. In such scenarios, protective responses might entail moderating the vehicle's acceleration, braking intensity, or steering actions to prevent escalation of the fault and mitigate potential damage to critical components.

5.1 Protection Mechanisms as SmartData

In the context of SmartData system design, protection mechanisms are formalized as Safety Models, denoted by PM, which are instantiated as Actuator SmartData. The role of PM is to monitor the instances ω_i of each relevant data stream involved in its interest relationships $\psi_{PM,\Omega_i} \in \Psi_{PM}$, evaluating specific conditions θ_{Ω_i}, where $\theta_{\Omega_i} : \omega_i \to \mathbb{B}$ represents a predicate function yielding a Boolean outcome. The set of all conditions monitored by PM is denoted as Θ.

When triggered, the protection mechanism selectively disables elements within the actuation set A. This behavior is captured by defining $\Gamma_{PM} = (\gamma_{\alpha_1}, \ldots, \gamma_{\alpha_n})$, where $n = |A|$. Each component γ_{α_i} is a Boolean indicating whether actuation channel α_i is inhibited by PM, for all $i \in [0, \ldots, n]$.

Protection mechanisms can generally be classified into two categories: *i)* mechanisms that react immediately upon detection of a hazardous condition, and *ii)* mechanisms that respond only if the hazardous condition persists for a duration of d time units. Accordingly, a protection mechanism can be defined as a tuple:

$$PM = (\Psi, \Theta, \Gamma, d)$$

where $d \in \mathbb{R}_{\geq 0}$ specifies the required persistence of the monitored condition prior to actuation. When $d = 0$, the mechanism belongs to the first class (immediate action); otherwise, it is categorized as delayed-action.

For immediate-action mechanisms, the activation condition is defined as:

$$\text{If } \left(\bigwedge_{\psi_{PM,\Omega_i} \in \Psi_{PM}} \theta_{\Omega_i} \right) = \text{True, then trigger } PM$$

An illustrative example of this class is the IMD, which can be formalized as:

$$PM_{IMD} = (\Psi, \Theta, \Gamma, 0)$$

where $\Psi = \{\psi_{IMD,IR}\}$, $\Theta = \{\theta_{IR}\}$ with $\theta_{IR} = (ir.v < minimum\_insulation)$, and $\Gamma = (True, True, True)$. Here, IR represents the insulation resistance sensor, and ir refers to its latest sampled value.

For delayed-action mechanisms, activation occurs according to the following logic:

$$\text{If } \left(\bigwedge_{\psi_{PM,\Omega_i} \in \Psi_{PM}} \theta_{\Omega_i} \right) = \text{True,}$$

and persists for d time units, then trigger PM.

A representative example is the BSPD, which can be defined as:

$$PM_{BSPD} = (\Psi, \Theta, \Gamma, 0.5s)$$

where $\Psi = \{\psi_{BSPD,MP}, \psi_{BSPD,B}\}$, $\Theta = \{\theta_{MP}, \theta_B\}$ with $\theta_{MP} = (mp.v > 0)$, $\theta_B = (b.v > hard\_threshold)$, and $\Gamma = (True, True, True)$. In this context, MP refers to the motor powering sensor, B corresponds to the latest braking command issued, and mp and b denote their respective most recent values.

5.2 Estimating Time-to-Triggering

Several protection mechanisms operate by disabling the engine or electrical subsystems once their monitored conditions are violated (e.g., BSPD, IMD, and BMS). To maintain system safety, it is crucial to anticipate and compensate for the potential loss of actuation capability before it becomes critical. Specifically, protection mechanisms belonging to the same category as BSPD provide an explicit time-to-failure or time-to-triggering parameter, denoted as d, which allows for corrective actions to be taken at least d units of time prior to actuator deactivation. However, in scenarios where $d = 0$, no such temporal margin for reaction is available.

Most protection mechanisms are inherently tied to the physical characteristics of the system and thus monitor variables that exhibit temporal evolution—either progressive degradation or escalation—until a safety threshold is exceeded. Examples include increasing temperatures, rising brake pressure, or decreasing tire pressure.

It is fair to assume protection mechanism to share their internal state with safety-related controllers in automotive system, such as Adaptive Cruise Control (ACC) and Advanced Driver Assistance Systems (ADAS). Disabling the execution of complex maneuvers, including overtake, whenever the temperature and voltage of battery system of a BMS, is a suitable and expected behavior for safety controllers. Therefore, having mechanisms to translate protection system state transitions into reaction time (e.g., the time it will take to evolve the current state into one that will disable actuation) is crucial for planning modules to properly regulate certain maneuvers or even promote evasive ones, thus, guaranteeing passengers safety and vehicle integrity. Thus, a lookup table mechanism, for instance, built by system engineers at design time, will provide a dynamic measure of time-to-triggering d expected by our solution.

If such mechanism is not available for a protection mechanism, such as black box system to which only the inputs are known, there is no solution other than employing predictors to promote a view of progressive degradation or escalation until a safety threshold is exceeded. By employing predictive models for these variables, it becomes feasible to estimate a time-to-trigger metric, even for mechanisms that traditionally lack a built-in delay parameter.

Figure 5 illustrates a generalized predictor P and the conceptual framework for deriving a time-to-trigger estimate. Here, h denotes the prediction horizon of P, representing the temporal extent over which the predictor provides reliable forecasts. The variable $t0$ corresponds to the current

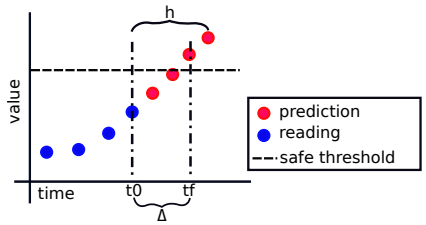

Figure 5. Example of a predictor for protection mechanism and the variables used when calculating constraints. Safe threshold is system engineer defined value for a specific protection mechanism input. *t0* is the current timestamp. *h* is the prediction horizon (e.g., how many time-steps in the future the predictor is able to predict). *tf* is the timestamp of the first prediction that crosses the threshold, and is used alongside *t0* to calculate the remaining time before the predicted safety breach represented as Δ. Note that the choice for red dots for prediction is just a style decision. Furthermore, having predictions after the threshold breach in this illustration is solely to demonstrate that the prediction horizon can be bigger than the remaining time for the safety breach.

time, while tf indicates the future time at which the monitored variable is predicted to cross its safety threshold (as defined by the condition θ_{Ω_i}). The time-to-trigger for the input $\psi_{PM,\Omega_i} \in \Psi_{PM}$ is then given by:

$$\Delta_{\Omega_i} = tf - t0$$

If the predictor does not anticipate any threshold violation within its horizon, we define $tf = \inf$, resulting in $\Delta_{\Omega_i} = \inf$.

5.3 Calculating Compensation of Actuation Constraint

For a protection mechanism PM characterized by $d_{PM} > 0$, the precise assessment of proximity to its potential triggering event is determined by the time at which the condition $\left(\bigwedge_{\psi_{PM,\Omega_i} \in \Psi_{PM}} \theta_{\Omega_i} \right)$ evaluates to true, incremented by the specified verification duration d_{PM}. Let us denote this initial evaluation time as t_{start}. Consequently, the estimated triggering time of the protection mechanism can be expressed as:

$$PM.tf = t_{start} + d_{PM}$$

To prevent system thrashing due to the transient nature of $\left(\bigwedge_{\psi_{PM,\Omega_i} \in \Psi_{PM}} \theta_{\Omega_i} \right)$ potentially reverting to false prior to $PM.tf$, we define a compensation constraint inversely proportional to the remaining time until the expected triggering of PM. This remaining time is given by:

$$PM.ttt = PM.tf - t$$

where t denotes the current evaluation timestamp. Accordingly, the compensation constraint is defined as:

$$PM.C = \frac{d_{PM}}{PM.ttt}$$

For protection mechanisms with an immediate response, i.e., $d_{PM} = 0$, and assuming the availability of a predictor of the form[2]:

$$P_{\Omega_i}\left([\omega_i^{t-k}.v, \ldots, \omega_i^t.v]\right) = \left[\omega_i^{t+1}.v, \ldots, \omega_i^{t+P.h}.v\right]$$

[2]Multi-dimensional predictors can also be employed, in which case the input vector $[\omega_i^{t-k}.v, \ldots, \omega_i^t.v]$ is replaced by a vector of multiple, correlated variables.

where k represents the number of historical samples utilized by P to forecast the subsequent $P.h$ samples of Ω_i, we define Δ_{Ω_i} as the time-to-trigger estimate derived from the prediction. The corresponding compensation constraint for Ω_i is then computed as:

$$\frac{P_{\Omega_i}.h}{\Delta_{\Omega_i}}$$

Since the triggering condition for PM is modeled as:

$$\left(\bigwedge_{\psi \in \Psi_{PM}} \theta_{\Omega_i} \right) = \text{True}$$

the remaining time to triggering is determined as:

$$PM.ttt = \max\left(\Delta_{\Omega_i} \,|\, \forall \psi_{PM,\Omega_i} \in \Psi_{PM}\right)$$

while the global compensation constraint for PM is given by:

$$PM.C = \max\left(\frac{P_{\Omega_i}.h}{\Delta_{\Omega_i}} \,|\, \forall \psi_{PM,\Omega_i} \in \Psi_{PM}\right)$$

Thus, at each decision point, the compensation constraint for PM over the set of actuators A in the AV system can be dynamically updated as follows:

$$\forall i \in [0, \ldots, |A|]:$$
$$PM.C_{\alpha_i} = \begin{cases} PM.C, & \text{if } \gamma_{\alpha_i} \wedge \left(\bigwedge_{\psi \in \Psi_{PM}} \theta_{\psi} \right) \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

6 Constraint-Aware RSS Triggered by SEU

As protection mechanisms are represented as Actuator SmartData, they are subject to verification properties within the SEU, as defined in eq. (4), which periodically evaluates their satisfiability. A system is considered safe with respect to a given protection mechanism if none of its associated conditions $\theta \in \Theta$ hold in the current system state. Thus, the Boolean filter $\mu()$ for a protection mechanism PM can be expressed as:

$$\mu(pm) = \neg \left(\bigwedge_{\psi_{PM,\Omega_i} \in \Psi_{PM}} \theta_{\Omega_i} \right) \quad (8)$$

As outlined in Section 3.2, the primary objective of the SEU is to serve as a runtime verifier of system safety, triggering appropriate responses according to which verification property fails. This design choice intentionally excludes the complexity of embedded safety models from STL formulations. We now formalize the steps to be undertaken whenever the SEU detects a satisfiability violation caused by a protection mechanism PM, specifically when:

$$\bigwedge_{\psi_{PM,\Omega_i} \in \Psi_{PM}} \theta_{\psi_{PM,\Omega_i}} = \text{True}$$

In summary, this process involves updating RSS parameters, re-evaluating the RSS conditions, and executing suitable proper responses. Furthermore, we propose an enhanced set of proper responses that address anticipated brake constraints, enabling compensation for potential brake loss in the near future.

To effectively integrate this framework with RSS, we must dynamically adjust the parameters used in the RSS formulation according to the current status of each protection mechanism PM in the system. To facilitate this, we extend the SmartData system definition to include a set of protection mechanism specifications, denoted by PMS. Whenever an input of any $PM \in S.PMS$ is updated, the system triggers an evaluation of the respective PM, potentially resulting in updated compensation constraints for each actuator relevant to RSS. These constraints are defined as follows:

$$Steering.C = \max\left(PM.C_{Steering} \,|\, \forall PM \in S.PMS\right) \tag{9}$$

$$Throttle.C = \max\left(PM.C_{Throttle} \,|\, \forall PM \in S.PMS\right) \tag{10}$$

$$Brake.ttt = \min\left(PM.ttt \,|\, \forall PM \in S.PMS\right) \tag{11}$$

Rather than constraining the brake actuator via $Brake.C$, we leverage the time-to-trigger metric $Brake.ttt$ to refine the proper response for longitudinal braking. This approach is advantageous as it enables the extension of RSS definitions for proper longitudinal responses to include braking forces exceeding $a^{lon}_{min,brake}$, thereby compensating for the anticipated unavailability of braking (see *Definition 2* and *Definition 3* in the subsequent subsections). Such compensations would not be feasible under the original RSS formulation, which assumes fixed thresholds for safe longitudinal distance. Notably, steering does not require this form of compensation, as RSS assumes instantaneous steering adjustments; thus, it suffices to apply constraints during minimum lateral distance calculations. Although throttle is not directly referenced in RSS proper responses, its constraint must still be updated to ensure motion planning feasibility, as discussed in *Definition 1* of the following subsection.

Following the update of these constraints, the corresponding RSS parameters must also be recalibrated. From this point onward, we denote by $a^{default}$ the static braking/acceleration values assumed in RSS, and by a^{real} the adjusted values that incorporate dynamic constraints. The updates are formalized as:

$$a^{real,lat}_{max,accel} = a^{default,lat}_{max,accel} \times Steering.C \tag{12}$$

$$a^{real,lat}_{min,brake} = a^{default,lat}_{min,brake} \times Steering.C \tag{13}$$

$$a^{real,lon}_{max,accel} = a^{default,lon}_{max,accel} \times Throttle.C \tag{14}$$

It is important to note that these updates apply solely to the ego vehicle's parameters. Constraints applicable to surrounding vehicles are beyond the scope of RSS and, by extension, this paper. This area remains open for future research, particularly in the context of V2X communication.

The following procedure is applied whenever a $PM \in S.PMS$ is updated:

1. Update the constraints and time-to-trigger metrics according to eq. (9), (10), and (11);
2. Recalculate the ego vehicle parameters within RSS using eq. (12), (13), and (14);
3. Re-evaluate the current vehicle state against RSS criteria;
4. If the current state is unsafe but recoverable through RSS proper responses, execute the appropriate response;
5. If there is a potential constraint on braking ($Brake.ttt > 0$), apply the response defined in *Definition 2* if the vehicle is already in an unsafe state, or in *Definition 3* otherwise;
6. Re-execute the motion planning algorithm, incorporating the updated constraints on steering and throttle, and accounting for brake availability only until $Brake.ttt$;
7. Perform RSS verification on the revised motion plan as described in *Definition 1*;
8. If the updated plan still contains unsafe elements or any actuator becomes fully constrained, upon completion of the proper response, re-execute motion planning with the goal of reaching the nearest safe stopping area—provided that actuation capabilities permit. If not, set $Brake.ttt$ to the corresponding time-to-trigger for full constraint and initiate preemptive braking.

6.1 Constraints into Steer and Throttle

In the scenario depicted in Figure 6, we analyze a case in which *RSS* is employed to validate a motion planning output corresponding to an overtaking maneuver. In this situation, considering the full acceleration capabilities of the vehicle ($a^{default}_{max,accel}$, both lateral and longitudinal), the maneuver can be executed without compromising safety. However, if we incorporate knowledge about the vehicle's actual condition—specifically, constraints on its acceleration capability, whether longitudinal or lateral—the maneuver could result in an unsafe state. For example, the vehicle may become unable to return to its original lane in time to avoid a collision, either with oncoming traffic or with the vehicle initially being overtaken, depending on the specific nature of the constraint.

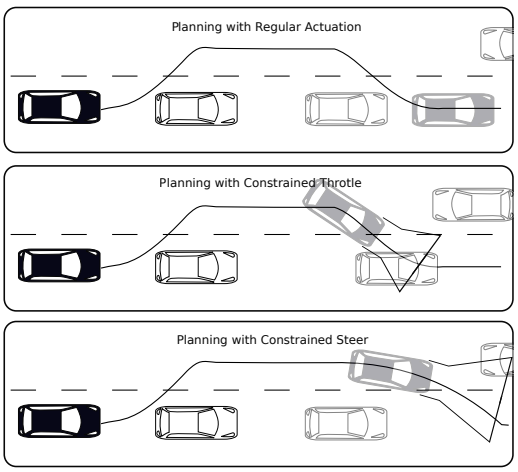

Figure 6. Example of modifying RSS to account for constraints on maximum acceleration.

We assume the maneuver is defined over a time interval $[t0, tf]$, representing the initial and final planned times. Modeling this situation as a safety verification procedure, and following the *RSS* framework along with its naive predic-

tion approach for surrounding vehicle behavior (as defined in *Definition 11* of RSS)[3], we can formally express the safety verification of the motion plan as follows:

Definition 1 (Protection-aware RSS safety verification of Motion Plan): Given a planned vehicle maneuver defined over the interval $[t0, tf]$, and assuming naive prediction models m^{lon} and m^{lat} for the future state of nearby agents set NA (providing discrete-time trajectories of position, velocity, and acceleration), the maneuver is considered safe if:

$$\exists \{a^{lat}\} \leq a^{real,lat}_{max,accel} \wedge \{a^{lon}\} \leq a^{real,lon}_{max,accel} \mid$$
$$\forall c \in NA,\ \forall t \in [t0, tf]:$$
$$dist^{lon}\left(e.m^{lon}(t), c.m^{lon}(t)\right) > dist^{lon}_{min}(e, c) \wedge$$
$$dist^{lat}\left(e.m^{lat}(t), c.m^{lat}(t)\right) > dist^{lat}_{min}(e, c)$$

where $dist_{min}$ denotes the minimum safe longitudinal and lateral distances as per *RSS*. Note that the prediction models m^{lon} and m^{lat} can either follow the naive prediction approach of RSS or more advanced models such as RSA-based predictions, similar to the work of Orzechowski *et al*. [2019] in which RSS and RSA are combined. Violation of this rule would lead to scenarios similar to that illustrated in Figure 6.

According to *Definition 8* of RSS (Lateral Proper Response), in a dangerous lateral scenario, the vehicle on the left must apply a deceleration of at most $-a^{default,lat}_{min,brake}$, while the vehicle on the right must apply at least $a^{default,lat}_{min,brake}$ until their relative lateral velocity (μ-*lateral-velocity*) reaches zero—meaning no further lateral convergence.

Assuming the ego vehicle anticipates constraints in its steering actuator, we first consider the μ-*lateral-velocity* definition from RSS, which incorporates a distance tolerance to avoid oscillations in lateral positioning. If vehicles are not closing the lateral gap (i.e., μ-*lateral-velocity* $= 0$), imposing a constraint on $a^{real,lat}_{min,brake}$ will not affect the safe distance maintenance. If the situation is already classified as dangerous, both vehicles are expected to be applying the appropriate proper responses, ensuring convergence toward μ-*lateral-velocity* $= 0$, as prescribed by *Definition 8* of RSS.

However, if μ-*lateral-velocity* > 0 and the situation was previously safe, constraining $a^{real,lat}_{min,brake}$ may compromise the ability to maintain a safe lateral distance. In such cases, the revised minimum distance could precipitate a dangerous situation, prompting the ego vehicle to engage its proper response protocol as defined by RSS, thus aiming to reduce the lateral velocity to zero. Once μ-*lateral-velocity* $= 0$ is achieved, the continuation of any hazardous condition (μ-*lateral-velocity* > 0) would depend solely on the actions of the other vehicle, given that the ego vehicle will have fulfilled its required response obligations (i.e., lateral velocity has been neutralized).

[3] *Definition 11* in RSS specifies: "The longitudinal or lateral state of a road agent is defined by its position, velocity, and acceleration, denoted by p_0, v_0, a_0. The future state, assuming a naive prediction, is as follows. Let $\tau = -v_0/a_0$ if v_0 and a_0 have opposite signs, or $\tau = \inf$ otherwise. The acceleration at time t is a_0 for $t \in [0, \tau]$ and zero thereafter. The velocity at time t is v_0 plus the integral of acceleration, and the position is p_0 plus the integral of velocity."

6.2 Constraints into Brake

The *RSS* framework assumes that the minimum safe distance is defined such that the rear vehicle can brake at $a^{default,lon}_{min,brake}$ and still avoid collision, even if the front vehicle decelerates at $a^{default,lon}_{max,brake}$. According to *Definition 4* of *RSS* (Longitudinal Proper Response), in a dangerous longitudinal scenario where both vehicles are traveling in the same direction, the rear vehicle is required to apply at least $-a^{default,lon}_{min,brake}$ until the situation is resolved. In the case of vehicles moving in opposite directions, the vehicle traveling in the correct direction must decelerate by at most $-a^{default,lon}_{min,brake,correct}$, while the one traveling in the wrong direction should accelerate at least $a^{default,lon}_{min,brake}$ (note: acceleration is considered negative and braking positive since the vehicle is traveling in the incorrect direction). Therefore, only braking constraints influence the proper response in dangerous longitudinal situations.

The approach proposed herein is built for adaptability, consistently assuming a worst-case scenario: the vehicle is expected to lose its braking capability after $Brake.ttt$ units of time. Consequently, whenever $Brake.ttt < tfs$, where $tfs = v/a^{default,lon}_{min,brake}$ represents the time required to reach a complete stop, actuation becomes infeasible under the assumption that the protection mechanism will trigger after $Brake.ttt$ time units. Therefore, upon detecting the potential activation of a protection mechanism, the vehicle must adjust its current acceleration to align with the available braking window, $Brake.ttt$.

It is important to note that simply applying $a^{default,lon}_{max,brake}$ constitutes a feasible and legally acceptable proper response under RSS, assuming all vehicles comply with RSS principles. This follows from the RSS assumption that any vehicle behind the ego vehicle maintains a safe distance and is capable of braking, even if the ego vehicle applies $a^{default,lon}_{max,brake}$. However, as this situation is predictive in nature, and the actual triggering of the protection mechanism may not occur exactly after $Brake.ttt$ time units, persistently applying $a^{default,lon}_{max,brake}$ could be overly conservative and detrimental to system performance. As a solution, we propose two strategies: one for scenarios where the ego vehicle is already executing a proper response (*Definition 2*), and another for situations where the vehicle is presently in a safe state (*Definition 3*).

Definition 2 (Protection-aware Proper Response for Longitudinal Braking): Let $c1$ represent the ego vehicle, following another vehicle $c2$. Let d_{min} denote the minimum safe distance as per *Definition 1*. Assuming the current distance between $c1$ and $c2$ is $d \geq d_{min}$, consider the worst case where $c2$ brakes at $-a^{default,lon}_{max,brake}$ until a full stop, and $d = d_{min}$. The standard RSS proper response remains feasible if:

$$tfs = \frac{v1}{a^{default,lon}_{min,brake}} \leq Brake.ttt$$

Otherwise, the braking must be intensified to at least:

$$a^{correct,lon}_{brake} = -\frac{v1}{Brake.ttt}$$

If $a^{correct,lon}_{brake} > a^{default,lon}_{max,brake}$, the correction is infeasible, necessitating an evasive legal maneuver as per RSS *Definition 12*. This reasoning applies equally to vehicles traveling in

opposite directions, adjusting braking signals according to vehicle orientation.

Definition 12 of RSS outlines two conditions for a maneuver to be deemed legal:

- *At the initiation of the maneuver, the ego vehicle's longitudinal and lateral accelerations must comply with the basic proper response constraints (see Definition 10) relative to all other road users.*
- *Throughout the maneuver, the lateral acceleration (according to m^{lat}) must not exceed $a_{max,accel}^{default,lat}$, and the longitudinal acceleration (according to m^{lon}) must remain within $[-a_{max,brake}^{default,lon}, a_{max,accel}^{default,lon}]$.*

Definition 10 of RSS specifies the *Basic Proper Response to Dangerous Situations*, which mandates the execution of either a lateral or longitudinal proper response whenever an unsafe situation is detected (e.g., $d < d_{min}$, with d_{min} as defined by *Lemmas* 1 and 2 in Section 2).

Remark 1. *Under normal circumstances, vehicle $c1$ would apply $-a_{min,brake}^{default,lon}$ until a complete stop is achieved. The time required for this maneuver is $tfs = v1/a_{min,brake}^{default,lon}$. Since we are already at the decision-making stage, we disregard ρ (the system response time). Therefore, if $Brake.ttt > 0$ and $Brake.ttt \geq tfs$, the standard proper response remains feasible. Conversely, if $Brake.ttt < tfs$, braking becomes infeasible beyond $Brake.ttt$. To address this, the ego vehicle must intensify its braking to achieve $tfs \leq Brake.ttt$. This adjusted braking force is calculated as $a_{brake}^{correct,lon} = v1/Brake.ttt$. If this exceeds $a_{max,brake}^{default,lon}$, the correction becomes infeasible, necessitating an evasive maneuver in line with RSS Definition 12.*

Definition 3 (Protection-aware Correction for Future Proper Response Longitudinal Braking): Assume the same initial conditions as in *Definition 2*, but vehicle $c2$ is not braking at the present time, and both vehicles are traveling in the same direction. In the worst-case scenario, at time t, $d = d_{min}$. Now, suppose that at $t + \rho$, $c2$ initiates braking at $-a_{max,brake}^{default,lon}$, and during the interval $[t, t+\rho]$, $c1$ accelerates at $a_{max,accel}^{default,lon}$. At $t + \rho$, the velocity of $c1$ becomes:

$$v1' = v1 + a_{max,accel}^{default,lon} \cdot \rho$$

According to *Definition 2*, the proper response at $t + \rho$ is feasible if:

$$Brake.ttt \geq tfs' = \frac{v1'}{a_{min,brake}^{default,lon}}$$

The acceleration correction required at time t is then:

$$a = v1/(Brake.ttt - \rho) \tag{15}$$

$$a_{brake}^{correct,lon} = \begin{cases} a_1 \leq a_{max,accel}^{default,lon}, \; if Brake.ttt - \rho \\ \geq tfs'_{max} \\ 0 \; , \; if \; a = a_{max,accel}^{default,lon} \\ \frac{(Brake.ttt - p) * a_{max,brake}^{default,lon} - v1}{p} \; , \\ Otherwise \end{cases} \tag{16}$$

Remark 2. *RSS Definition 4 assumes the ego vehicle ($c1$) can accelerate by $a_{max,accel}^{default,lon}$ over $[t, t + \rho]$ and still remain safe, perceiving the need to act only after ρ time units. Thus, at $t + \rho$, $v1'$ is given by $v1' = v1 + a_{max,accel}^{default,lon} \cdot \rho$. Should $c2$ brake with $-a_{max,brake}^{default,lon}$ from $t + \rho$ and the predicted triggering condition hold, feasibility requires $Brake.ttt \geq tfs'$. The actuation at $t + \rho$ remains valid if $a_{brake}^{correct,lon} \leq a_{max,brake}^{default,lon}$. To anticipate this at time t, we deduct ρ from $Brake.ttt$, leading to $Brake.ttt - \rho \geq tfs'$.*

To avoid unnecessary braking at time t, if $Brake.ttt - \rho \geq tfs'_{max}$ (where $tfs'_{max} = v1'/a_{max,brake}^{default,lon}$), the ego vehicle may accelerate freely up to $a_{max,accel}^{default,lon}$. If $a = v1/(Brake.ttt - \rho) = a_{max,accel}^{default,lon}$, no positive acceleration is feasible, but braking is not yet required. Otherwise, if $a = v1/(Brake.ttt - \rho) > a_{max,accel}^{default,lon}$, preemptive braking is necessary to ensure feasibility of the future proper response.

Considering braking over p time units, with initial velocity $v1$, the velocity at $t + p$ becomes:

$$vf = p \cdot a_1 + v1 \tag{17}$$

The braking must satisfy:

$$\exists a_{min,brake}^{default,lon} \leq a_2 \leq a_{max,brake}^{default,lon} \mid a_2 = \frac{vf}{Brake.ttt - p} \tag{18}$$

Substituting eq. (17) into eq. (18) and assuming minimization of braking (i.e., $a_2 = a_{max,brake}^{default,lon}$), we derive:

$$Brake.ttt - p = \frac{p \cdot a_1 + v1}{a_{max,brake}^{default,lon}} \tag{19}$$

Rearranging:

$$a_1 = \frac{(Brake.ttt - p) \cdot a_{max,brake}^{default,lon} - v1}{p} \tag{20}$$

Thus, whenever $a \leq a_{max,brake}^{default,lon}$, the future proper response at $t + p$ remains feasible.[4] *Otherwise, an evasive maneuver is required, per RSS Definition 12.*

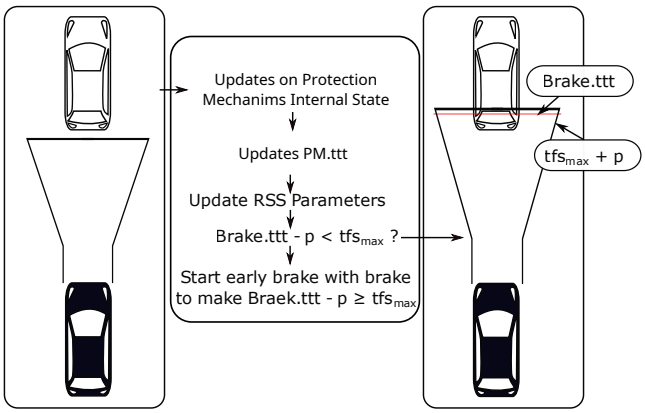

Figure 7. Example of modifying RSS to include constraints on minimum braking capability.

Figure 7 illustrates an application of *Definition 3*. In scenarios where the brake constraint triggers such that $Brake.ttt -$

[4] This formulation assumes $a_{max,brake}^{default,lon}$ is achievable unless constrained. If so, this value should be updated accordingly.

$\rho < tfs_{max}$, the system may face a future unsafe state if early braking is not executed. In these cases, the proper response prescribed by *Definition 3* must be applied.

For non-dangerous scenarios involving $c1$ and $c2$ traveling in opposite directions, evaluation of the future need for correction should consider constraints imposed on steering or throttle actuation, as covered by *Definition 1*.

6.3 Handling Overtake Maneuvers

To avoid performance trashing during complex maneuvers, such as overtaking, due to early braking, we base on our previous work (Conradi Hoffmann *et al*. [2024a]) to set proper responses using both braking and steering to attempt recovering from such situations without impairing performance. Note that even without these maneuver specific proper responses, if all vehicles follow RSS, by *Definition 1*, the verified motion plan alongside *Definition 2* and *Definition 3* already guarantee braking without crashing whenever feasible. Thus, this is just an attempt to avoid impairing the performance even further.

The execution of an overtake maneuver is affected by steering and acceleration. Nevertheless, the proper response may require braking if steering or acceleration are constrained. This paper addresses steering and braking constraints. No optimization is done using positive longitudinal acceleration, and it is left for future work. The possible conditions are split into scenarios with and without opposite traffic concerning the safety of each action. Then, overtake is analyzed based on its current stage, namely, leaving/outside the original lane (lateral velocity $Vl_{ego} \geq 0$) and returning to the original lane ($Vl_{ego} < 0$). Furthermore, critical steering constraints and critical braking constraints are also considered. In terms of overtake safety, we define the safe distance to return to the lane (SDRL) as:

$$SDRL = L_{ego} + Dmin_{ego,over} \tag{21}$$

where L_{ego} is the length of the ego vehicle, and $Dmin_{ego,over}$ is the longitudinal safe distance between ego and the overtaken vehicle, given by RSS. The notion of safe distance from RSS (given by $Dmin$) assumes the front-most point of the vehicle behind and the back-most point of the vehicle in front. Therefore, the length of the vehicle must be considered as if the vehicle was already in the original lane.

Definition 4 (Proper responses without opposite traffic):

1. If $Steering.C = 1$, the only possible outcome is to apply $a^{real,lon}_{min,brake}$ until reaching a full stop.
2. If $Steering.C < 1$ and $Brake.ttt > tfs$ proceed as follows:
   2.a) If $Vl_{ego} \geq 0$, create enough space to return to the original lane until $D(ego, over) \geq SDRL$, which can be achieved by applying $a^{real,lon}_{min,brake}$ to $a^{real,lon}_{max,brake}$ based on $Vl_{ego}/Brake.ttt$. Moreover, simultaneously apply $a^{real,lat}_{min,brake}$ until $Vl_{ego} = 0$. Then, go to step $4.2.c$.
   2.b) If $Vl_{ego} < 0$, go to step $4.2.c$.
   2.c) Apply $a^{real,lat}_{min,brake}$ until the vehicle returns to the center of the original lane. Finish the maneuver by applying $a^{real,lat}_{min,accel}$ until $S = 0$.
3. After executing the proper response, re-run the motion planning module, setting the new goal to the closest stopping area if possible. Otherwise, apply $a^{real,lon}_{min,brake}$ until a full stop is achieved.

Suppose another vehicle is in the original lane behind the vehicle being overtaken during the execution of proper response $4.2.a$, this vehicle is expected to follow the principle of common sense rule 5 envisioned by RSS (*"If you can avoid an accident without causing another one, you must do it."*) and comply with the action by creating enough space for the ego vehicle to get back to the original lane, applying $a^{default,lon}_{min,brake}$. The evaluation of feasibility of $4.2.a$ considers the time necessary to complete it versus the time the vehicle in the opposite direction would crash into the ego vehicle. These times are calculated as follows:

First, estimate the time needed to create the necessary distance from the overtaken vehicle using $ego_a = a^{default,lon}_{min,brake}$ if the overtaken vehicle applies positive acceleration, or $ego_a = 0$ otherwise. Assuming the vehicle being overtaken, at the worst case, would be braking with $a^{default,lon}_{min,brake}$, t_d, the time to create distance $\geq SDRL$ is given by solving the following equation:

$$\frac{(-a^{default,lon}_{min,brake})}{2} * t^2 + (V_{ego} - V_{over}) * t - SDRL = 0 \tag{22}$$

Where $V_{ego}\ and\ V_{over}$ are the speed of the ego vehicle and the vehicle on the right, respectively. On the other hand, if, in the worst case, the vehicle being overtaken is applying $a_{overtaken} = 0$, the time to create distance $\geq SDRL$ (t_d) assuming the ego vehicle is braking using $a^{real,lon}_{min,brake}$ is given by solving the following equation:

$$\frac{(-a^{real,lon}_{min,brake})}{2} * t^2 + (V_{ego} - V_{over}) * t - SDRL = 0 \tag{23}$$

Finally, the time needed to leave the current lane and return to the original lane (t_{ld}) is a function of the current lateral velocity Vl_{ego} and the maximum steering angle possible to be obtained with $a^{real,lat}_{min,brake}$. As disclosed by RSS, lateral acceleration is assumed to change instantaneously with a turn in the steering wheel. Therefore, in the proper response proposed here, the constraint into the steering wheel will result in a constant acceleration of $a^{real,lat}_{min,brake}$ in lateral velocity during its execution. $(D(ego, lc))$ is the length of the line traced from the center of the original lane, the current position of the vehicle, using the fixed steering angle. The time required to travel this distance is given by solving the following equation:

$$\frac{(a^{real,lat}_{min,brake})}{2} * t^2 + (V_{ego}) * t - (D(ego, lc)) = 0 \tag{24}$$

Thus, given the sum of t_d and d_{ld} as $t_{4.2.a}$, the operation is feasible only when the vehicle in the opposite direction can brake sufficiently to avoid crashing into the ego vehicle while they are still in the same lane. The time t_{bc} needed for the vehicle in the opposite direction to brake assuming $a^{default,lon}_{min,brake}$ is given by solving:

$$\frac{(-a^{default,lon}_{min,brake}) + a_{ego,lon} * t^2}{2} + (V_{OP} + V_{ego}) * t - (D(ego, op)) = 0 \tag{25}$$

During the execution of proper response $4.2.b$, the distance between the ego and the vehicle being overtaken is expected to be at least $SDRL$. Otherwise, ego should not be allowed to return to the original lane (e.g., already be heading to original lane[5]). Therefore, the time required to complete $4.2.b$ is $t_{4.2.b} = t_{ld}$. The analysis of feasibility of the proper responses given in Definition 4 is defined as:

Definition 5 (Feasibility of proper responses in the presence of Opposite Traffic):

1. If $Brake.ttt \geq tfs$, proper response 4.1 is safe.
2. If $Brake.ttt < tfs$, proper response 4.1 is safe whenever $D(ego, op) >= Dmin(ego, op)$ considering the updated parameter (e.g., brake with maximum brake until $Brake.ttt$ and slow down by simply not accelerating is sufficient to full stop before crashing).
3. Let $t_{4.2.a}$ be the time required to complete step $4.2.a$ given by equations (22 or 23) and equation (24). Let t_{bc} be the minimum time remaining before crashing into another vehicle in the opposite direction, assuming that it applies a proper response according to RSS (e.g., $a_{min,brake}^{default,lon}$), given by equation (25). Proper response $4.2.a$ is safe whenever $t_{4.2.a} < t_{bc} < Brake.ttt$.
4. Let $t_{4.2.b}$ be the time required to complete step $4.2.b$ given by equation (24). Let t_{bc} be as defined above. Proper response $4.2.b$ is safe whenever $t_{4.2.b} < t_{bc} < Brake.ttt$.
5. Whenever $4.2.a$ or $4.2.b$ are unsafe, abort the proper responses and follow the original RSS proper response for unsafe longitudinal distance with opposite traffic, applying brake according to $Brake.ttt$. This is safe whenever $D(ego, op) >= Dmin(ego, op)$ considering the updated parameter.

The aforementioned procedure can be applied to get vehicles back into a safe state during an overtake with and without opposite traffic, with partial and critical steering constraints and partial braking constraints. Moreover, critical brake constraint is supported in $4.2.b$ whenever $t_{4.2.b} < t_{bc}$, and in $4.1.a$, whenever $t_{4.2.a} < t_{bc}$. Other proper responses may be proposed if there is a valid road to the left of the ego vehicle during an overtake. This condition is out of the scope of this paper and is left for future work.

6.4 Proving Compliance with RSS

We will now prove that the actions proposed in *Definition 1*, *Definition 2*, and *Definition 3* are compliant with *Definition 12* of RSS. Note that in *Definition 1*, we are only assuming the possibility of constrained actuation on $a_{max,brake}^{real,lat}$, $a_{max,accel}^{real,lat}$, and $a_{max,accel}^{real,lon}$ to the point they will only decrease up to 0 (e.g., the actuation is fully constrained). On the other hand, for *Definitions 2* and *3*, we never assume accelerations outside the range $[-a_{max,brake}^{default,lon}, a_{max,accel}^{default,lon}]$. Thus, the second condition of *Definition 12* of RSS will never be broken.

Definition 10 of RSS points to the proper responses for longitudinal and lateral movement, regulated by *Definitions 4 and 8* of RSS, respectively. *Definition 1* in itself, aims at verifying *Definition 10* of RSS throughout discrete steps into a motion plan. Moreover, in Section 6.1, we demonstrate that the notion of constrained $a_{max,brake}^{real,lat}$ and $a_{max,accel}^{real,lat}$ will promote an early actuation based on *Definitions 8* of RSS to enforce safety previous to the loss of steering movement. In this way, maneuvers that would be impacted by a potential triggering of a protection mechanism during its execution (e.g., changing lanes or performing an overtake, similar to the examples presented in Figure 6), will be rejected during planning phase based on the formulation proposed in *Definition 1* of this paper. Thus, the motion planning algorithm should create new planning considering the constraints.

Nevertheless, the other vehicles may put themselves in the opposite direction of the road, and vehicles in front may start braking at $a_{max,brake}^{default,lon}$, as they are unaware of the constraints of the ego vehicle. In both scenarios, this is the exact point where our approach affects RSS, guaranteeing the safety condition assumed by *Definitions 4* of RSS (i.e., *"until the situation is non-dangerous again"*) even when considering a worst-case scenario of a full stop while considering a possible fault in the near future. *Definitions 3*, focuses on guaranteeing safety by anticipating a braking procedure in the presence of a potential triggering of a protection mechanism. *Definitions 2* focuses on adapting the braking up to $a_{max,brake}^{default,lon}$ based on the distance in time to the possible fault.

In this way, we not only comply with the first and second conditions of *Definition 12* of RSS but make the first one robust to the triggering of protection mechanisms. We may now write: "The longitudinal and lateral accelerations of the ego vehicle at the initial time of the maneuver satisfy the proper response constraints applied to him, considering all agents and the protection mechanisms that could be triggered during the maneuver execution."

We will now prove this conclusion using an inductive proof, making a simple addition to cover our adjustments in the inductive proof presented in *Lemma 5* of RSS:

RSS's Lemma 5 *Consider a multi-lane road where all lanes share the same geometry. Suppose that at any time $t > 0$, whenever a prediction of the triggering of a protection mechanism results in a time-triggering metric > 0, the ego vehicle will update RSS parameters accordingly. Moreover, suppose that at all times, all cars on the road comply with the basic proper response as given in Definition 10 of RSS. Then, there will be no collisions, even in the presence of a protection mechanism triggering.*

Proof. We will prove that for any pair of cars, c_1, c_2, there is a sequence of increasing times, $0 = t0 \leq t1 < t2 < t3 <$ ..., such that for every time t_i, $i \geq 1$, there is no collision between c_1 and c_2 in the time interval $[t_{i-1}, t_i]$, and at time t_i the situation between c_1 and c_2 is not dangerous. The basis of the induction is the earliest time, t_1, in which one of the cars is starting to drive. By the definition of proper response, and considering that possible triggers for protection mechanisms have been predicted and RSS parameters have been adjusted, t_1 is not a dangerous situation, and it is clear that there cannot be an accident from t_0 to t_1. Suppose the inductive assumption holds for $t1 < ... < t_i$. So, at time t_i, the situation is non-dangerous. Let t_b be the earliest time after t_i for which the situation becomes dangerous. If no such t_b exists, then there will be no accidents in the time interval $[ti, \infty)$ (because,

[5]This assumption holds as overtakes are expected to follow traffic rules, and therefore, perform overtake while in curves or areas with no visibility.

before an accident occurs, the situation must be dangerous), hence we are done. Otherwise, suppose that t_b exists due to one of the cars being affected by steering constraints. Thus, t_b is perceived only by this car. In this scenario, the car under constraints initiates a proper response according to *Definition 10* of RSS. Then there is a time $t_c \geq t_b$ for which either the situation becomes non-dangerous, and we are done. Suppose the situation is also considered dangerous for both cars since the constrained car already started its proper response and compensated for its steer constraint before triggering the protection mechanisms. In that case, the situation falls under the regular RSS proper response. There is a time $t_d \geq t_c$ for which the situation becomes non-dangerous, or the relative longitudinal velocity of the two cars is non-positive, or the relative lateral velocity of the two cars is non-positive. In the former case, we set $t_{i+1} = t_d$. Note that, for the first case, if the situation becomes non-dangerous at t_c, we have $t_d = t_c$. In the latter cases, we set t_{i+1} to be the earliest time after t_d in which the cars are again not in a dangerous situation, and if no such time exists, it must mean that the cars would never collide in the time interval $[t_d, \infty)$. In all cases, by the definitions of proper response and safe distance, and *Definitions 2* and *3* proposed here, there will be no accident in the time interval $[t_b, t_d]$. Moreover, there can be no accidents in the time intervals $[t_i, t_b]$ and $[t_d, t_{i+1}]$. Hence, our inductive argument is concluded. Finally, it is crucial to note that the definitions of proper response imply no contradictions between the proper response of c_1 relative to c_i and relative to c_j, for any other two cars c_i, c_j.

Therefore, assuming all cars in the scenario are RSS-compliant, reacting to faults, whether predicted or pointed by Protection Mechanisms would not generate unsafe situations as all vehicles respect safe distances according to road parameters for acceleration and braking.

We note that having big Δ_{Ω_i} will imply small braking compensations as the early deceleration is modeled to be the minimum sufficient to allow full stop before reaching $Brake.ttt$ of 0. Thus, if at the next prediction there is no more signs of triggering the protection mechanism in the near future, the false positive impact is minimum to the system performance. Nonetheless, if such operation is done, for instance, during an overtake with traffic in the opposite direction and the completion of the maneuver is unfeasible (via Definition 1), the proposed overtake maneuver will be engaged to either return to the original lane or engaged a full stop before crashing in the current lane, the latter being feasible whenever all vehicles follows RSS constraints as assumed in *Lemma 5*[6].

Thus, even if a predictor presents false positives, early deceleration might impair traffic performance but will not reduce safety.

6.5 Overtake Proper Responses Compliance with Legal Evasive Maneuvers

The following paragraphs demonstrate compliance of the proposed overtake-specific proper responses with legal evasive maneuvers defined by RSS.

First, constraining an actuation to its limit will reduce the original minimum and maximum to 0. Furthermore, the scenarios presented here are always handled considering the constrained actuation, therefore, $\forall a_{lat} \in proper\_responses a_{lat} < a_{max,accel}^{default,lat}$ and $\forall a_{lon} \in proper\_responses - a_{max,brake}^{default,lon} < a_{lon} < a_{max,accel}^{default,lon}$.

The second requirement for an evasive maneuver to be considered legal by RSS is that the longitudinal and lateral accelerations of the ego vehicle should satisfy the baseline proper response constraints applied to him with respect to all other agents. Since this paper considers the possibility of actuation to be impaired by faults, some constraints may naturally disrespect RSS proper responses, for instance, braking with at least $a_{min,brake}^{default,lon}$ whenever facing a dangerous longitudinal situation will never be achieved if $a_{min,brake}^{real,lon} < a_{min,brake}^{default,lon}$. To this end, the initial steps described in the start of Section 6 consist of updating RSS parameters and re-evaluating the situation to overcome this issue.

Our proper responses still assume respecting other nearby vehicles considering the feasibility analysis of the proper responses through the time boundaries established by *Definition 5*, and therefore, covering vehicles in the opposite direction. Vehicles behind the ego are expected to cope with RSS proper responses and respond to its braking by keeping a safe longitudinal distance. This assumption does not disrupt RSS safety analysis since braking is bounded within the default minimum and maximum. Furthermore, safe distances to vehicles on the left of the ego vehicles will never be broken as long as the ego is not applying positive lateral acceleration. An exception to this scenario is a possible steering constraint when leaving the original lane, where the vehicle applies a positive lateral acceleration. Nevertheless, the proposed proper responses act on the limit of available steering and braking (if not critical) to avoid crashes whenever feasible. This scenario is covered in *Definition 4.2.a*, where the $Vl_{ego} \geq 0$, and the first action is to brake longitudinally until $D(ego, over) \geq SDRL$ and $Vl_{ego} = 0$. Note that if steering constraint is critical, the first condition *4.1* will rely on braking. Moreover, the crash is unavoidable if the brake is critical to the point where it cannot stop the vehicle before crashing into another vehicle or the road boundaries. Thus, the proper responses are characterized as legal maneuvers and cope with the original RSS.

7 Constraint-Aware RSS in CARLA

To demonstrate the new proper responses in action, we integrate the Safety Enforcement Unit, and RSS, with CARLA. Car Learning to Act (CARLA), proposed by Dosovitskiy *et al.* [2017], is a simulator that has grown in popularity over the last few years. CARLA promotes detailed scenarios for simulating AVs. Moreover, in recent versions, CARLA supports integrating with other simulation tools, including SUMO. CARLA provides tools for converting maps, trajectories, and vehicle blueprints to integrate with SUMO. Finally, the simulation synchronization is done periodically via a Client Application Programming Interface (API) of CARLA, enabling reproducibility of results.

In our implementation, the RSS module is integrated with

[6] Safe once Definition 1 handles unsafe situations that would be foresight before engaging a motion plan, and, if the motion plan is allowed to take place, the distances before loosing an actuator are always safe if all cars react according to RSS constraints.

Figure 8. Glance at CARLA simulated scenario.

CARLA at the control interface level, specifically before the planning module transmits the future waypoints to the PID controller. At this stage, the RSS module overrides the next-step target speed and steering angle according to the computed proper responses for detected unsafe conditions—that is, whenever the actual distance falls below the minimum safe distances calculated by RSS. Consequently, this integration requires, at a minimum, the real-time dynamics of the ego vehicle (position and velocity) as well as those of surrounding objects.

The decision-making module is designed under the assumption that protection mechanisms continuously share their internal states and that predictive models are available to RSS. With this information, the Constraint-Aware RSS is capable of dynamically updating both actuation constraints and time-to-trigger metrics at runtime. Based on these updates, the module selects the appropriate response strategy: it applies either *Definition 2* or *Definition 3* whenever *Brake.ttt* is greater than zero, or otherwise relies on *Definition 1*. When anticipating full constraint of an actuation channel, it adjusts braking behavior accordingly with respect to *Brake.ttt*.

During simulation, we use an electric vehicle as the EGO, and induce a battery cell over-voltage fault to evaluate the system's performance. Assuming that the BMS shares its internal state with the RSS module inside the SEU, and that either a look-up-table or a predictor P is used for actively monitoring system health, the fault is anticipated to cross the safety threshold within a 1-second horizon. At this point in the simulation, the ego vehicle is positioned immediately prior to initiating an overtaking maneuver, replicating the scenario illustrated in Figure 6, and corresponding to the exact frame shown in Figure 8.

7.1 Safe Distance Keeping Experimental Results

Figure 9 illustrates the evolution of the inter-vehicle distance during simulation, alongside the minimum safe distance as defined by both the original RSS and the proposed Constraint-Aware RSS. Correspondingly, Figure 10 depicts the vehicle speeds for the same scenario, comparing the behaviors under both frameworks. In this scenario, the standard RSS fails to respond promptly and continues along its initial trajectory, crossing into adjacent lanes before applying emergency braking, which is ultimately triggered by a system shutdown.

In contrast, the Constraint-Aware RSS updates its behavior in accordance with eq. (5.3), initiating preemptive braking

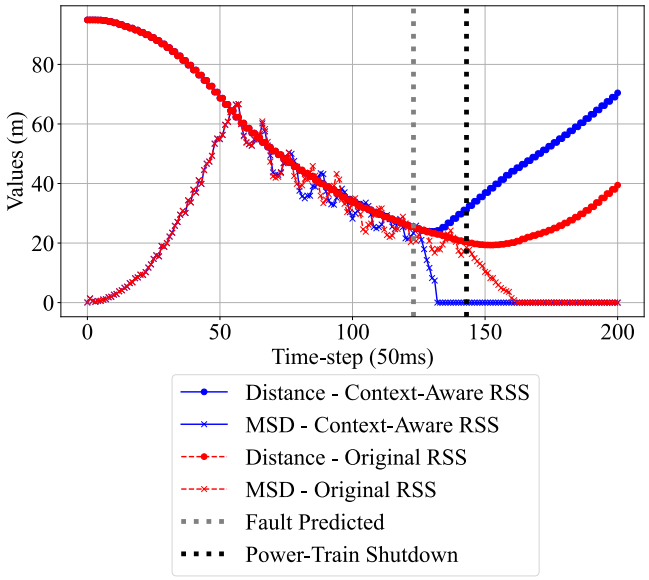

Figure 9. EGO vehicle's distance to vehicle in front and the Minimum Safe Distance (MSD) according to RSS.

as soon as predictive conditions indicate imminent actuation loss—specifically, when steering, throttle, and braking capabilities are forecasted to be fully constrained before the maneuver can be completed. Leveraging the *Brake.ttt* metric, the system anticipates the loss of actuation and proactively avoids executing the overtake maneuver, opting instead for early braking.

At simulation time step 57, the leading vehicle begins to decelerate, prompting the ego vehicle to also reduce its speed. As the relative speed decreases and closing distance narrows, the vehicle governed by the original RSS attempts an overtaking maneuver. However, it loses the capability to complete this action at time step 143 due to a full system shutdown, which had been predicted by the fault predictor P one second earlier (at step 123). It is important to note that, in the simulation, the system shutdown results in a wheel lock of the electric vehicle model, bringing the vehicle to a complete stop within a few steps (by time step 164).

The speed profiles further illustrate the advantage of the Constraint-Aware RSS: the ego vehicle equipped with this system begins preemptive braking precisely at time step 123, coinciding with the fault prediction, and successfully comes to a halt before the shutdown occurs. Moreover, it maintains controlled steering to remain within its lane. In contrast, the ego vehicle operating under the original RSS only reaches a full stop later, due to the mechanical wheel lock induced by the shutdown, by which point it has already veered out of its lane.

7.2 Overtake Proper Response Results

The previous experiment tackled a view of the system considering a fault being detect in the early moments of an overtake maneuver. Now, we will demonstrate an experiment in which the fault is detect after the vehicle left the original lane and is facing traffic in the opposite direction. First of all, we must take into consideration that, if the maneuver is engaged, by *Definition 1*, the plan is safe and the distance in all steps is safe in regards to RSS minimum distance. Therefore, at any moment that an unsafe situation is found, the vehicles are

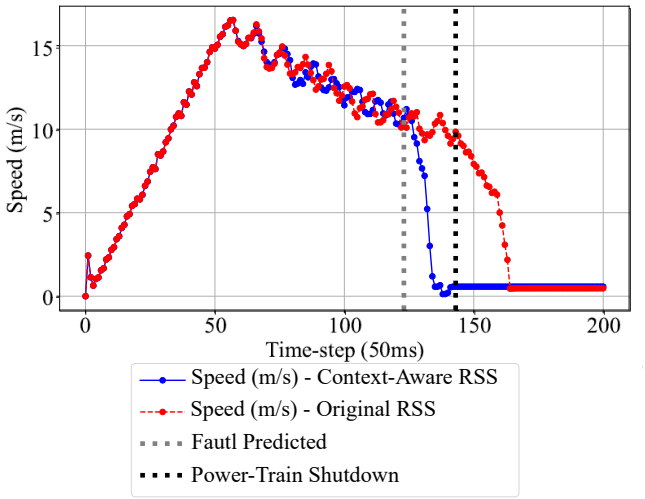

Figure 10. EGO vehicle's speed considering original RSS and Constraint-Aware RSS.

able to return to a safe state (e.g., full stop) by applying the respective braking proper response in accordance to *Definition 2* and *3*.

The scenario follows a constant turn-rate and acceleration (CTRA) model with time-indexed fault injections around lane-change, specifically mid maneuver, when the ego vehicle is on the opposite lane attempting to gain sufficient distance to return to the original lane and complete the overtake. Figure 11 presents the regular scenario, in which the vehicle does not faces any fault during the overtake. The initial configuration is: 1) opposite vehicle driving at $17m/s$; 2) vehicle to be overtaken (red) driving at $10m/s$; 3) ego vehicle (blue) initially driving at $10m/s$, accelerating at $3.5m/s^2$ ($a^{real,lon}_{max,accel}$), with a turn rate of $\pm 0.25\ rad/s$, applied during lane changes, respectively. The simulation step is 100 milliseconds. The figure illustrates time via a gradient of the dot colors, white being the first time-step, and the respective color being the final time-step (e.g., when ego vehicle fully stops). After finishing the overtake, the ego vehicle brakes with $a^{real,lon}_{max,brake}$ until a full stop is achieved.

In this experiment, we simulate a fault at 2 seconds of simulation, when the ego vehicle arrives at the center of the opposite lane. The BMS protection mechanism updates its internal state, and either via a look-up-table search or a predictor, a 1 second time-to-trigger is identified. Figure 12 presents the first scenario, where the same proper response used in the previous example (preemptively braking) is used. By doing so, this allows for both ego vehicle and the vehicle in the opposite direction to achieve full stop without crashing.

When considering the overtake-specific proper response, specifically the one described by *Definition 4* and *Definition 5*, the proper response will attempt returning to the original lane by using steering. Figure 13 presents the scenario where the proper response 4.2.a is attempted. Note that the ego vehicle is not able to create enough space (SDLR) and return to the original lane in time (e.g., $t_{4.2.a} < Brake.ttt$). Thus, the proper response should be braking according to $Brake.ttt$ instead, according to *Definition 5.5*. Nevertheless, it was able to fully stop the vehicle without crashes. Since safe distances to all nearby vehicles is taken into consideration, the vehicle will either attempt this maneuver or fully stop in the lane, which is feasible due to *Definition 1*, *2,* and *3*.

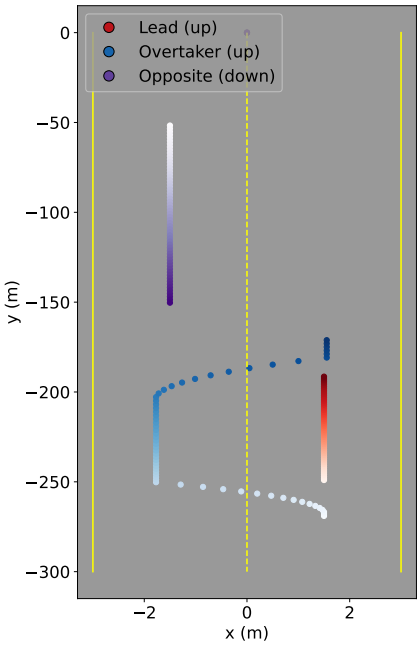

Figure 11. Complete overtake maneuver in a three vehicle scenario with one vehicle in the opposite direction going down (purple), the vehicle that will be overtaken (red), and the ego vehicle performing the overtake (blue). A gradient from white to the aforementioned color is used to represent time. The simulation step is 100 milliseconds.

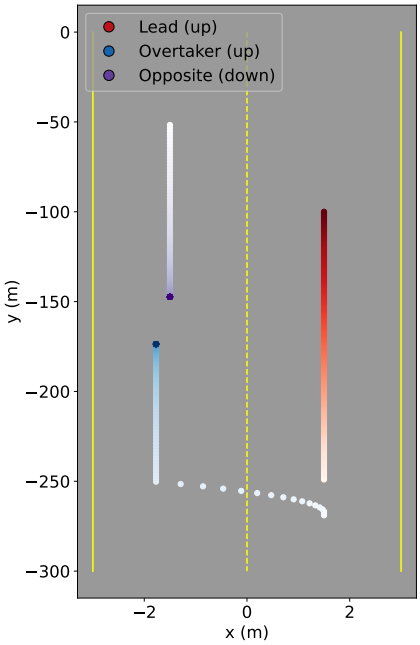

Figure 12. Overtake maneuver where a fault is perceived at 2 seconds of simulation. RSS proper response and preemptively braking promote a safe full stop for both ego vehicle (blue) and the vehicle in the opposite direction (purple). The same color semantics from previous figure is used in this figure to represent time.

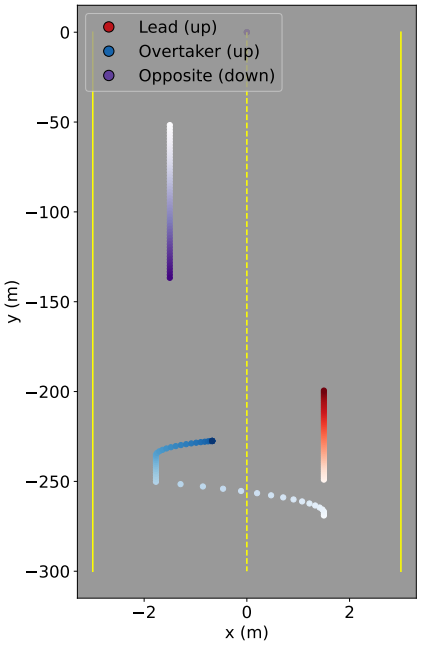

Figure 13. Overtake maneuver where a fault is perceived at 2 seconds of simulation. By applying *Definition 4.2.a* proper response, the vehicle attempts to return to the original lane but fails since the fault takes place before the maneuver is completed. Nonetheless, the vehicle is fully stopped without crashing and freeing more space for other vehicles in the opposite direction to attempt evasive maneuvers.

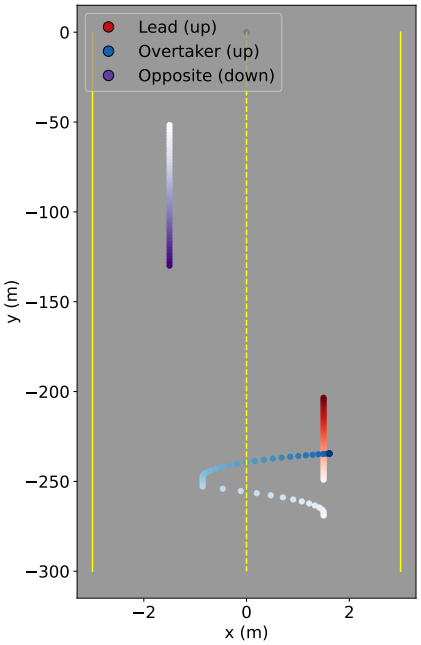

Figure 14. Overtake maneuver where a fault is perceived at 1.5 seconds of simulation. By applying *Definition 4.2.a* proper response, the vehicle is able to return to the original lane before the 1 second to return to the original lane but fails since the fault takes place before the maneuver is completed. Nonetheless, the vehicle is fully stopped without crashing and freeing more space for other vehicles in the opposite direction to attempt evasive maneuvers.

Finally, we also demonstrate the scenario where the vehicle is not yet in the middle of the opposite lane, and applies the proper response given by *Definition 4.2.a* to safely return to the original lane and avoid further performance losses. Figure 14 presents the scenario where the fault is perceived at 1.5 seconds instead of 2, and the vehicle is able to return to the original lane safely within 1 second (e.g., $t_{4.2.a} \geq Brake.ttt$).

7.3 SEU Verification Latency

This evaluation examines the SEU's performance in providing low-overhead, real-time monitoring and timing verification of data, based on the RSS safety principles discussed earlier. Table 1 offers a detailed breakdown of the latency involved in both updating traces and verifying data within the SEU. Whenever new data is detected on the network, the SEU immediately triggers the `update()` function of the corresponding Verifiable SmartData. This operation transfers the message contents into an internal buffer, which is then processed by the Boolean Filters μ assigned to monitor this specific SmartData. In this evaluation scenario, all SmartData instances are being monitored, with a special focus on the filters $\mu_{RSS_{lon}}$ and $\mu_{RSS_{lat}}$, which assess longitudinal and lateral safety conditions for both the EGO Motion Vector and the Objects Fuser. These checks follow the RSS response time of $100ms$ established before.

The $Dynamics_{EGO}$ module estimates the vehicle's motion using IMU and GNSS data over time. Its output is critical for the Camera, LiDAR, and Object Fuser modules, as it supports the generation of the object list used for RSS verification. Consequently, there are four monitoring requests directed at $Dynamics_{EGO}$. When a Boolean Filter's `update()` function is called, it runs the μ function and notifies any safety properties that depend on that filter, thereby logging a new state sample in the property's trace through `STL_Verifier.update()`.

In Table 1, the first column categorizes the type of data, while the second column reports the average latency for executing the `update()` function for each data type. For multi-dimensional data like 3D positions, accelerations, and angular velocities, the reported values represent the combined averages of their individual SmartData components. The third column summarizes the average latency of the `evaluate()` function, aggregated across all properties that utilize the respective data. These averages are based on measurements collected over 2000 execution cycles. The final row of the table provides the cumulative sum of these average latencies. All evaluations were conducted on the NVIDIA Jetson AGX Orin 64 ECU, which serves as the computing platform for vision-based tasks (such as object detection and tracking using LiDAR point clouds and camera imagery) in a real-world autonomous vehicle prototype, developed as part of the SmartData on Wheels project[7]. Considering a response time of RSS of $100ms$, the verification procedure is able to cope with the periodicity consuming less than 1% of the platform computational power.

8 Final Remarks

This paper presented an approach to improve RSS to be robust to faults by leveraging the knowledge behind vehicular protection mechanisms. Vehicular Protection mechanisms

[7] `https://lisha.ufsc.br/SDAV+Overview`

Table 1. Latency of trace update and safety verification in NVIDIA Jetson Orin ECU.

Data (Ω_j)	Avg `update()`	Avg `evaluate()`
GNSS Location (X,Y,Z)	$29.796\mu s$	$15.075\mu s$
IMU Accelerations x3	$34.118\mu s$	$15,144\mu s$
IMU Heading	$11.287\mu s$	$4.192\mu s$
IMU Angular Velocities x3	$42.765\mu s$	$16.003\mu s$
$Dynamics_{EGO}$	$35.02\mu s$	$19.47\mu s$
$Dynamics_{nearby\ objects}$ CAM	$13.2\mu s$	$3.244\mu s$
$Dynamics_{nearby\ objects}$ LiDAR	$16.53\mu s$	$4.235\mu s$
$Dynamics_{nearby\ objects}$ Fuser	$24.018\mu s$	$3.43\mu s$
Total Latency	$206.734\mu s$	$80.793\mu s$

are designed to be proactive, constantly monitoring the signals to identify tendencies generated prior to a fault in order to actuate by either turning off a component or limiting its actuation range.

Our approach is based on predicting when a protection mechanism will be triggered. We model a time-to-trigger metric that allows for an early reaction to the possibility of losing an actuation in the near future, ensuring the system is always prepared for potential faults. The baseline approach relies on either having a direct time-to-trigger metric whenever intrinsic of the protection mechanism, or, when this is not available a design time look-up-table for protection mechanisms internal states, and finally, in the worst case, relying on predictors. However, predictors have intrinsic prediction errors that may lead to a false positive of a future triggering of a protection mechanism. Therefore, two new proper responses are proposed to compensate for a possible loss of the braking system. They focus on braking with the minimum braking required to achieve a complete stop before losing the ability to brake. Thus, by avoiding hard braking, we minimize the thrashing of the AV performance due to false positives. Nevertheless, having predictors with minimal errors is still crucial to avoid trashing the AV performance. To reduce the impact of performances losses on more complex maneuvers, such as overtake, we have combined the proposed approach with overtake-specific proper responses that also consider using steering to recover from actuator loss.

Finally, by modeling the protection mechanisms as SmartData, we can extract formal verification rules automatically, without need for expert knowledge in formal methods from system programmers. The monitoring of the state of protection mechanisms and triggering of the proper responses can be done at run time automatically by the SEU with low overhead (< 1% of platform processing power).

For future works, we envision evasive maneuvers to supplement decision-making when a proper response is deemed unfeasible and novel approaches to improve system performance in the face of false positives, such as using side lanes. Furthermore, expanding the proposed solutions to scenarios with non-autonomous traffic, such as pedestrian and regular drivers, is an open challenge to be addressed in the next steps of the research.

Acknowledgements

This paper is an extension of the conference paper ”José Luis Conradi Hoffmann, Antônio Augusto Fröhlich, Marcus Völp, and Paolo Milazzo. 2024. Using Vehicular Protection Mechanisms to Enable Fault-Aware Safety Verification of Autonomous Vehicles. In Proceedings of the 13th Latin-American Symposium on Dependable and Secure Computing (LADC ’24). Association for Computing Machinery, New York, NY, USA, 55–64. https://doi.org/10.1145/3697090.3697101”. We would like to acknowledge Murillo Guindani, from the Software/Hardware Integration Lab for supporting the simulation configuration in CARLA.

Funding

This research was partially funded by FUNDEP Rota 2030 project AutoDL (29271.03.01/2023.04-00).

Authors' Contributions

JLCH, AAF, MV, and PM contributed to the conception of this study. JLCH performed the experiments. AAF, MV, and PM contributed with the revision of the manuscript. JLCH is the main contributor and writer of this manuscript. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Availability of data and materials

The source code for the SEU implementation is available at LISHA's GitLab.

References

Althoff, M. and Magdici, S. (2016). Set-based prediction of traffic participants on arbitrary road networks. *IEEE Transactions on Intelligent Vehicles*, 1(2):187–202. DOI: 10.1109/TIV.2016.2622920.

Conradi Hoffmann, J. L., Augusto Fröhlich, A., and Völp, M. (2024a). Enhancing rss to be fault tolerant during overtaking maneuvers. In *IECON 2024 - 50th Annual Conference of the IEEE Industrial Electronics Society*, pages 1–6. DOI: 10.1109/IECON55916.2024.10905937.

Conradi Hoffmann, J. L., Fröhlich, A. A., Völp, M., and Milazzo, P. (2024b). Using vehicular protection mechanisms to enable fault-aware safety verification of autonomous

vehicles. In *Proceedings of the 13th Latin-American Symposium on Dependable and Secure Computing*, LADC '24, page 55–64, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3697090.3697101.

Conradi Hoffmann, J. L., Passig Horstmann, L., and Fröhlich, A. A. (2024c). Transparent integration of autonomous vehicles simulation tools with a data-centric middleware. *Design Automation for Embedded Systems*, 28(1):45–66. DOI: 10.1007/s10617-023-09280-w.

Cui, J., Sabaliauskaite, G., Liew, L. S., Zhou, F., and Zhang, B. (2019). Collaborative analysis framework of safety and security for autonomous vehicles. *IEEE Access*, 7:148672–148683. DOI: 10.1109/access.2019.2946632.

de Lucena, M. M. and Augusto Fröhlich, A. (2022). Modeling misbehavior detection timeliness in vanets. In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, Stuttgart, Germany. IEEE. DOI: 10.1109/ETFA52439.2022.9921605.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16. DOI: 10.48550/arXiv.1711.03938.

Fröhlich, A. A. (2018). SmartData: an IoT-ready API for sensor networks. *International Journal of Sensor Networks*, 28(3):202. DOI: 10.1504/ijsnet.2018.096264.

Gruber, F. and Althoff, M. (2018). Anytime safety verification of autonomous vehicles. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 1708–1714, Maui, HI, USA. IEEE. DOI: 10.1109/ITSC.2018.8569950.

Hoffmann, J. L. C. and Fröhlich, A. A. (2022). Smartdata safety: Online safety models for data-driven cyber-physical systems. In *48th Annual Conference of the IEEE Industrial Electronics Society*, pages 1–6, Brussels, Belgium. IEEE. DOI: 10.1109/IECON49645.2022.9969074.

Hoffmann, J. L. C. and Fröhlich, A. A. (2025). Smartdata: Toward the data-driven design of critical systems. *IEEE Access*, 13:41865–41886. DOI: 10.1109/ACCESS.2025.3548542.

Hoffmann, J. L. C., Horstmann, L. P., Wagner, M., Vieira, F., de Lucena, M. M., and Fröhlich, A. A. (2022). Using formal methods to specify data-driven cyber-physical systems. In *2022 IEEE 31st International Symposium on Industrial Electronics (ISIE)*, pages 643–648, Anchorage, AK, USA. IEEE. DOI: 10.1109/ISIE51582.2022.9831686.

Huang, J. and Tan, H.-S. (2016). Control system design of an automated bus in revenue service. *IEEE Transactions on Intelligent Transportation Systems*, 17(10):2868–2878. DOI: 10.1109/tits.2016.2530760.

International Organization for Standardization (2018). ISO 2626: Road vehicles – functional safety. Available at: `https://www.iso.org/obp/ui/#iso:std:iso:26262:-1:ed-2:v1:en`.

Kim, S.-W., Qin, B., Chong, Z. J., Shen, X., Liu, W., Ang, M. H., Frazzoli, E., and Rus, D. (2015). Multivehicle cooperative driving using cooperative perception: Design and experimental validation. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):663–680. DOI: 10.1109/TITS.2014.2337316.

Kong, W., Luo, Y., Qin, Z., Qi, Y., and Lian, X. (2019). Comprehensive fault diagnosis and fault-tolerant protection of in-vehicle intelligent electric power supply network. *IEEE Transactions on Vehicular Technology*, 68(11):10453–10464. DOI: 10.1109/TVT.2019.2921784.

Koopman, P. and Wagner, M. (2016). Challenges in autonomous vehicle testing and validation. *SAE International Journal of Transportation Safety*, 4(1):15–24. DOI: 10.4271/2016-01-0128.

Lucchetti, F., Graczyk, R., and Völp, M. (2023). Toward resilient autonomous driving—an experience report on integrating resilience mechanisms into the apollo autonomous driving software stack. *Frontiers in Computer Science*, 5:1–11. DOI: 10.3389/fcomp.2023.1125055.

Ludwich, M. K. and Frohlich, A. A. (2015). Proper handling of interrupts in cyber-physical systems. In *2015 International Symposium on Rapid System Prototyping (RSP)*, pages 83–89, Piscataway, New Jersey, USA. IEEE. DOI: 10.1109/RSP.2015.7416551.

Maler, O. and Nickovic, D. (2004). Monitoring temporal properties of continuous signals. In Lakhnech, Y. and Yovine, S., editors, *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 152–166, Berlin, Heidelberg. Springer Berlin Heidelberg. DOI: 10.1007/978-3-540-30206-3_12.

of Automotive Engineers, I. S. (2021). Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. DOI: 10.4271/j3016_202104.

Orzechowski, P. F., Li, K., and Lauer, M. (2019). Towards responsibility-sensitive safety of automated vehicles with reachable set analysis. In *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*, pages 1–6, Graz, Austria. IEEE. DOI: 10.1109/ICCVE45908.2019.8965069.

Pek, C., Manzinger, S., Koschi, M., and Althoff, M. (2020). Using online verification to prevent autonomous vehicles from causing accidents. *Nature Machine Intelligence*, 2(9):518–528. DOI: 10.1038/s42256-020-0225-y.

Sangha, M., Gomm, J., Yu, D., and Page, G. (2005). Fault detection and identification of automotive engines using neural networks. *IFAC Proceedings Volumes*, 38(1):272–277. 16th IFAC World Congress. DOI: 10.3182/20050703-6-CZ-1902.01933.

Shalev-Shwartz, S., Shammah, S., and Shashua, A. (2017). On a formal model of safe and scalable self-driving cars. *CoRR*, abs/1708.06374:1–37. DOI: http://arxiv.org/abs/1708.06374.

Sidorenko, G., Fedorov, A., Thunberg, J., and Vinel, A. (2022). Towards a complete safety framework for longitudinal driving. *IEEE Transactions on Intelligent Vehicles*, 7(4):809–814. DOI: 10.1109/TIV.2022.3209910.

Sivakumar, A. and Mohanty, P. (2020). Electronic system design of a formula student electric car. In *2020 IEEE International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)*, pages 115–120, Udupi, India. IEEE. DOI: 10.1109/DISCOVER50404.2020.9278091.