A Formally Verified Checker of the Safe Distance Traffic Rules for Autonomous Vehicles

Albert Rizaldi*, Fabian Immler*, and Matthias Althoff

Institut für Informatik, Technische Universität München
{rizaldi, immler, althoff}@in.tum.de

Abstract. One barrier in introducing autonomous vehicle technology is the liability issue when these vehicles are involved in an accident. To overcome this, autonomous vehicle manufacturers should ensure that their vehicles always comply with traffic rules. This paper focusses on the safe distance traffic rule from the Vienna Convention on Road Traffic. Ensuring autonomous vehicles to comply with this safe distance rule is problematic because the Vienna Convention does not clearly define how large a safe distance is. We provide a formally proved prescriptive definition of how large this safe distance must be, and correct checkers for the compliance of this traffic rule. The prescriptive definition is obtained by: *1)* identifying all possible relative positions of stopping (braking) distances; *2)* selecting those positions from which a collision freedom can be deduced; and *3)* reformulating these relative positions such that lower bounds of the safe distance can be obtained. These lower bounds are then the prescriptive definition of the safe distance, and we combine them into a checker which we prove to be sound and complete. Not only does our work serve as a specification for autonomous vehicle manufacturers, but it could also be used to determine who is liable in court cases and for online verification of autonomous vehicles' trajectory planner.

1 Introduction

Liability is an important but rarely studied area in autonomous vehicle technology. For example, who should be held liable when a collision involving an autonomous vehicle occurs? In our previous paper [23], we proposed to solve this issue by formalising vehicles' behaviours and traffic rules in higher-order logic (HOL). This formalisation allows us to check formally whether an autonomous vehicle complies with traffic rules. If autonomous vehicles always comply with traffic rules, then they should not be held liable for any accident.

One of the most important traffic rules is to maintain a safe distance between a vehicle and the vehicle in front of it. This notion of safe distance is crucial for traffic simulation, automatic cruise controller (ACC), and safe intersections. Traffic simulation [1] relies on this notion to update the speed and acceleration of each vehicle such that a collision will not occur in the simulation, even when the front vehicle brakes abruptly. ACC [26] and safe intersection systems [16,14] rely on this notion to control the engine and brake module such that a rear-end collision can be avoided.

* Supported by the DFG Graduiertenkolleg 1480 (PUMA)

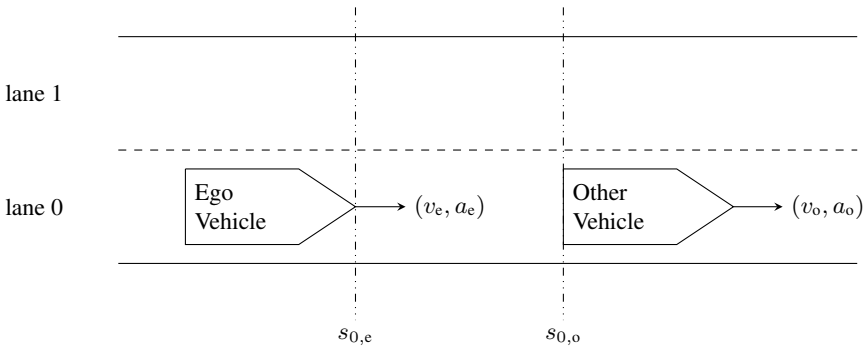

Fig. 1: Scenario for safe distance problem.

The Vienna Convention on Road Traffic defines a 'safe distance' as the distance such that *a collision between vehicles can be avoided if the vehicle in front performs an emergency brake* [24]. Note that this rule states the requirement for safe distance descriptively; there is no prescriptive expression against which a distance can be compared. This makes the process of formally checking the compliance of an autonomous vehicle's behaviour with the safe distance rule problematic.

We follow our previous design decision [23] to use Isabelle/HOL [18] for three reasons. Firstly, it has rich libraries of formalised real analysis which is required to turn the descriptive definition of safe distance into the prescriptive one. Secondly, it allows us to generate code, which we use to evaluate a real data set. Finally, as a theorem prover, Isabelle checks every reasoning step formally and, hence, one only has to trust how we specify the notion of safe distance. Our contributions are as follows:[1]

- We formalise a descriptive notion of safe distance from the Vienna Convention on Road Traffic (Sec. 2).
- We turn this formalised descriptive definition of safe distance into a prescriptive one through logical analysis (Sec. 3).
- We generate executable and formally verified checkers in SML for validating the safe distance rule (Sec. 4).
- We evaluate the US Highway 101 data set from the Next Generation SIMulation (NGSIM) project as benchmark for our checkers (Sec. 5).
- We argue that our prescriptive definition of *safe distance* generalises all definitions of safe distance in the literature (Sec. 6).

We conclude and outline the possible extension of our work in Sec. 7.

2 Formalising Safe Distance from the Vienna Convention

Figure 1 illustrates the scenario for the safe distance problem as defined in the Vienna Convention on Road Traffic. The scenario consists of two vehicles: the *ego* vehicle

[1] Our formalisation is available at `http://home.in.tum.de/~immler/safedistance/index.html`

and the closest vehicle in front of it—which we term *other* vehicle[2]. This scenario is uniquely characterised by six constants: $s_{0,\mathrm{e}}, v_{\mathrm{e}}, a_{\mathrm{e}} \in \mathbb{R}$ from the ego vehicle and $s_{0,\mathrm{o}}, v_{\mathrm{o}}, a_{\mathrm{o}} \in \mathbb{R}$ from the other vehicle. Constants s_0, v, a denote the initial position, initial speed, and maximum deceleration value, respectively, of a vehicle. Note that $s_{0,\mathrm{e}}$ denotes the *frontmost* position of the ego vehicle, while $s_{0,\mathrm{o}}$ denotes the *rearmost* position of the other vehicle. Additionally, we also make the following assumptions:

Assumption 1 *The values of v_{e} and v_{o} are non-negative:* $0 \leq v_{\mathrm{e}} \ \wedge \ 0 \leq v_{\mathrm{o}}$.

Assumption 2 *The values of a_{e} and a_{o} are negative:* $a_{\mathrm{e}} < 0 \ \wedge \ a_{\mathrm{o}} < 0$.

Assumption 3 *The other vehicle is located in front of the ego vehicle:* $s_{0,\mathrm{e}} < s_{0,\mathrm{o}}$.

Continuous dynamics. As specified by the Vienna Convention on Road Traffic, the ego vehicle needs to avoid collision with the other vehicle when both vehicles are braking. To do so, the ego vehicle needs to *predict* its own braking movement and that of the other vehicle over time. We formalise the prediction of this braking movement p with a second-order ordinary differential equation (ODE) [3] $p''(t) = a$ and initial value conditions $p(0) = s_0$ and $p'(0) = v$. The closed-form solution to this ODE is as follows:

$$p(t) \ := \ s_0 + vt + \frac{1}{2}at^2 \ . \tag{1}$$

Hybrid dynamics. Since Eq. (1) is a quadratic equation, it has the shape of a parabola when $a \neq 0$. This implies that a vehicle would move backward after it stops. Hence, Eq. (1) is only valid for the interval $[0, t_{\text{stop}}]$ where t_{stop} is the stopping time. The stopping time t_{stop} is the time when the first derivative of p is zero, that is, $p'(t_{\text{stop}}) = 0$. Substituting t with t_{stop} in the derivative of Eq. (1) results into the following expression for t_{stop}:

$$t_{\text{stop}} \ := \ -\frac{v}{a} \ . \tag{2}$$

Thus, we can extend the movement p of Eq. (1) by introducing discrete jumps (the deceleration makes a jump from $a < 0$ to $a = 0$) into the overall movement s as follows.

$$s(t) \ := \ \begin{cases} s_0 & \text{if } \ t \leq 0 \\ p(t) & \text{if } \ 0 \leq t \leq t_{\text{stop}} \\ p(t_{\text{stop}}) & \text{if } \ t_{\text{stop}} \leq t \end{cases} \tag{3}$$

Two-vehicle scenario. In Fig. 1, we assume that the other vehicle performs an emergency brake with maximum deceleration a_{o}, as specified in the Vienna Convention on Road Traffic. As soon as the other vehicle brakes, the ego vehicle reacts by performing an emergency brake too with maximum deceleration a_{e}. Since an autonomous vehicle can react almost instantly, we assume the reaction time to be zero.

In order to determine whether the distance $s_{0,\mathrm{o}} - s_{0,\mathrm{e}}$ is safe or not, we first use Eq. (3) to predict the movement of the ego vehicle $s_{\mathrm{e}}(t)$ and the other vehicle $s_{\mathrm{o}}(t)$ over

[2] NGSIM has identified the other vehicle for each ego vehicle in the US-101 Highway data set.

[3] We use Lagrange's notation f' and f'' to denote the first and the second derivative of f.

Table 1: Four cases of stopping times and the corresponding equations of $s_{\mathrm{e}}(t) = s_{\mathrm{o}}(t)$.

	$0 \leq t \leq t_{\text{stop,e}}$	$t_{\text{stop,e}} < t$
$0 \leq t \leq t_{\text{stop,o}}$	ⓐ $p_{\mathrm{e}}(t) = p_{\mathrm{o}}(t)$	ⓑ $p_{\mathrm{e}}(t_{\text{stop,e}}) = p_{\mathrm{o}}(t)$
$t_{\text{stop,o}} < t$	ⓒ $p_{\mathrm{e}}(t) = p_{\mathrm{o}}(t_{\text{stop,o}})$	ⓓ $p_{\mathrm{e}}(t_{\text{stop,e}}) = p_{\mathrm{o}}(t_{\text{stop,o}})$

time. Then, a collision will occur if we can find future time t such that $s_{\mathrm{e}}(t) = s_{\mathrm{o}}(t)$. To generalise this predicate, we define *collision* over a set of real numbers $T \subseteq \mathbb{R}$ as follows:

$$\mathit{collision}(T) \;:=\; (\exists t \in T.\; s_{\mathrm{e}}(t) = s_{\mathrm{o}}(t)) \quad . \tag{4}$$

Equations (1) to (3), assumptions 1 to 3, and the definition in (4) above are our formalisation of the safe distance rule from the Vienna Convention on Road Traffic. The remaining results presented in this paper are deduced from there. The deductions are also formally checked by Isabelle theorem prover.

3 Logical Analysis of the Safe Distance Problem

This section analyses the safe distance problem by performing two case distinctions based on stopping times and stopping distances. The first case distinction (Sec. 3.1) is more suitable for *checking* whether there will be a collision or not. The second case distinction (Sec. 3.2) meanwhile is about eliminating the existential quantifier in Eq. (4) and rearranging the resulting formula such that one can obtain lower bounds for the initial distance $s_{0,\mathrm{o}} - s_{0,\mathrm{e}}$ that is still safe. In principle, after resolving the discrete jumps, this quantifier elimination for real arithmetic could be achieved by automatic procedures as implemented in modern computer algebra systems (CASs). There is even a proof-producing procedure implemented in the HOL-Light theorem prover [17]. However, our seven-variable formula appears to be too complex for HOL-Light's quantifier elimination procedure. Therefore, manually finding this lower bound with an interactive theorem prover is necessary. This makes our results more robust against changes in the formalisation and more readable compared to those from CASs'.

3.1 Case Distinction Based on Stopping Times

To check for collisions, we need to find the solution of Eq. (4). However, finding the solution is problematic due to the occurrences of the if-construct in the definition of the overall movement in Eq. (3). Therefore, we perform case distinction based on two stopping times conditions: $0 \leq t \leq t_{\text{stop}}$ and $t_{\text{stop}} < t$ for both vehicles. This produces four cases in total (see Tab. 1). Each case is the equation where $s_{\mathrm{e}}(t) = s_{\mathrm{o}}(t)$ with functions $s_{\mathrm{e}}(t)$ and $s_{\mathrm{o}}(t)$ are substituted as in Eq. (3), depending on which stopping time condition holds. Since each case is a pure quadratic equation, we can use a decision procedure for finding roots of univariate polynomials for each case. A checker based on such a decision procedure is described in Sec. 4.1.

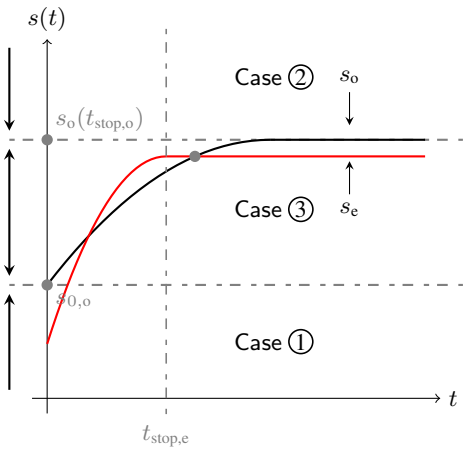

Fig. 2: Three cases obtained from case distinction based on stopping distances.

3.2 Case Distinction Based on Stopping Distances

Figure 2 illustrates the case distinction based on stopping distances. It plots an example of overall movement for the other vehicle $s_o(t)$ and divides this movement into three regions (cases) where a stopping distance of the ego vehicle $s_{stop,e} := s_e(t_{stop,e})$ could be located:

①. $s_{stop,e} < s_{0,o}$;
②. $s_{stop,o} \leq s_{stop,e}$.
③. $s_{0,o} \leq s_{stop,e} < s_{stop,o}$.

These stopping distances can be obtained by substituting t_{stop} in Eq. (2) to s in Eq. (3) for the ego and the other vehicle as follows:

$$s_{stop,e} = s_{0,e} - \frac{v_e^2}{2 \cdot a_e} \quad \text{and} \quad s_{stop,o} = s_{0,o} - \frac{v_o^2}{2 \cdot a_o} \ . \tag{5}$$

For any case in which collision freedom can be deduced, we rearrange the terms and the deduction into the following pattern:

$$s_{0,o} - s_{0,e} > \textit{safe-distance}\,(a_e, v_e, a_o, v_o) \implies \textit{precondition}\,(a_e, v_e, a_o, v_o) \implies \neg\, \textit{collision}\,[0; \infty) \ . \tag{6}$$

This pattern has the interpretation that if the initial distance $s_{0,o} - s_{0,e}$ is bigger than the expression *safe-distance* (a_e, v_e, a_o, v_o) and the *precondition* (a_e, v_e, a_o, v_o) holds, too, then we can guarantee that there will be no collision. We claim that the expression *safe-distance* (a_e, v_e, a_o, v_o) defines the notion of safe distance *prescriptively*; one can easily check whether a collision exists by comparing the initial distance with this expression.

In the rest of this section, we prove three theorems—one for each of these three cases—which determine whether there is a collision or not. As an overview, collision freedom can be deduced in case ① while collision can be deduced in case ②. In case ③, collision depends on further conditions than just the premise $s_{0,\mathrm{o}} \leq s_{\mathrm{stop,e}} < s_{\mathrm{stop,o}}$.

Background formalisation. Consider the quadratic equations of the form $p(x) := ax^2 + bx + c$ with the discriminant $D := b^2 - 4ac$. The analysis of the movement can be carried out with the following well-known mathematical facts about quadratic forms.

– Solution of quadratic equation:

$$D \geq 0 \implies x_{1,2} := \frac{-b \pm \sqrt{D}}{2a} \tag{7}$$

$$a \neq 0 \implies p(x) = 0 \iff (D \geq 0 \wedge (x = x_1 \vee x = x_2)) \tag{8}$$

– Condition for convexity:

$$x < y < z \wedge p(x) > p(y) \leq p(z) \implies a > 0 \tag{9}$$

– Monotonicity:

$$(t \leq u \implies s(t) \leq s(u)) \wedge (t < u \wedge u \leq t_{\mathrm{stop}} \implies s(t) < s(u)) \tag{10}$$

– Maximum at the stopping time:

$$p(t) \leq p(t_{\mathrm{stop}}) \quad \wedge \quad s(t) \leq s(t_{\mathrm{stop}}) \tag{11}$$

Because these are all basic, well-known facts, one can expect that the overhead of using a theorem prover be kept within limits. Indeed, all of the facts in Eq. (7) to (11) can be proved automatically with one of Isabelle's automatic provers: the sum-of-squares methods (ported from Harrison [11]) or rewriting of arithmetic expressions combined with classical reasoning.

Theorems We start with the first theorem for case ① which states that this case implies collision freedom. Intuitively speaking, there will be no collision in this case because the ego vehicle is located so far that it stops before the initial position of the other vehicle.

Theorem 1 (Obvious collision freedom in case ①).

$$s_{\mathrm{stop,e}} < s_{0,\mathrm{o}} \implies \neg\, \mathit{collision}\, [0; \infty) \tag{12}$$

Proof. This is true because $s_{\mathrm{e}}(t) < s_{\mathrm{o}}(t)$ holds for every time $t \geq 0$: $s_{\mathrm{e}}(t) \leq s_{\mathrm{e}}(t_{\mathrm{stop,e}}) = s_{\mathrm{stop,e}} < s_{0,\mathrm{o}} = s_{\mathrm{o}}(0) \leq s_{\mathrm{o}}(t)$ due to transitivity, the assumption, and monotonicity of s in Eq. (10), and the maximum at the stopping time in Eq. (11). □

Since this case implies absence of collision, we can unfold the definition of $s_{\text{stop,e}}$ in Eq. (5) and rearrange Theorem 1 according to the pattern in Eq. (6) into the following safe distance expression:

$$\mathit{safe\text{-}distance}_1 := -\frac{v_{\text{e}}^2}{2 \cdot a_{\text{e}}} \tag{13}$$

For case ②, we first give the following lemma which provides a sufficient condition for a collision in a bounded interval. It follows directly from the continuity of s and an application of the intermediate value theorem for $s_{\text{o}} - s_{\text{e}}$ between 0 and t.

Lemma 1 (Upper bounds on collision time).

$$(s_{\text{e}}(t) \geq s_{\text{o}}(t) \implies \mathit{collision}\,[0;t]) \wedge (s_{\text{e}}(t) > s_{\text{o}}(t) \implies \mathit{collision}\,[0;t))$$

Then, the following theorem states that case ② necessarily implies a collision.

Theorem 2 (Obvious collision in case ②).

$$s_{\text{stop,e}} \geq s_{\text{stop,o}} \implies \mathit{collision}\,[0;\infty)$$

Proof. Since by definition $s_{\text{stop,e}} = s_{\text{e}}(t_{\text{stop,e}})$ and $s_{\text{stop,o}} = s_{\text{o}}(t_{\text{stop,o}})$, setting $t := \max\{t_{\text{stop,e}}, t_{\text{stop,o}}\}$ in Lemma 1 above proves that this case implies a collision. □

Since case ② implies collision, no safe distance expression is produced from the logical analysis of case ②.

We now consider case ③, where the ego vehicle stops behind the other vehicle. There can still be a collision, i.e. the movement of the ego vehicle can intersect the movement of the other vehicle and still stop behind the other vehicle (see Fig. 2). The following lemma states that a collision (if any) in case ③ must occur while *both* cars are still moving. This lemma therefore allows us to reduce the reasoning to the continuous part p of the movement s.

Lemma 2 (Collision within stopping times in case ③).

$$s_{0,\text{o}} \leq s_{\text{stop,e}} < s_{\text{stop,o}} \implies$$
$$\mathit{collision}\,[0;\infty) \iff \mathit{collision}\ (0; \min\{t_{\text{stop,e}},\ t_{\text{stop,o}}\}) \tag{14}$$

Proof. The "$\Longleftarrow$"-part is obvious and we only prove the "$\Longrightarrow$"-part. If a collision happens at time t while one of the vehicles has already stopped, then it must be the ego vehicle which has stopped ($t_{\text{stop,e}} < t$). Also, we have $s_{\text{e}}(t_{\text{stop,e}}) > s_{\text{o}}(t_{\text{stop,e}})$ because according to Eq. (10), s_{o} is strictly increasing in $[t_{\text{stop,e}}; t]$ (see Fig. 2). Then, Lemma 1 yields a suitable witness for an earlier collision $t' < t_{\text{stop,e}}$. The whole proof takes just about 80 lines in the formalisation. □

Then, the following theorem characterises the conditions for ensuring a collision in case ③.

Theorem 3 (Conditional collision in case ③).

$$s_{0,\text{o}} \leq s_{\text{stop,e}} < s_{\text{stop,o}} \implies \mathit{collision}\,[0;\infty) \iff$$
$$a_{\text{o}} > a_{\text{e}} \wedge v_{\text{o}} < v_{\text{e}} \wedge s_{0,\text{o}} - s_{0,\text{e}} \leq \frac{(v_{\text{o}} - v_{\text{e}})^2}{2 \cdot (a_{\text{o}} - a_{\text{e}})} \wedge t_{\text{stop,e}} < t_{\text{stop,o}} \tag{15}$$

Proof. ("$\Longleftarrow$".) Case ③ and Eqs. (7), (8), and (15) yield a root of $p_\mathrm{o} - p_\mathrm{e}$, which is contained in the interval $(0; \min\{t_\text{stop,e}, t_\text{stop,o}\})$. The root is therefore also a root of $s_\mathrm{o} - s_\mathrm{e}$ and therefore witnesses *collision* $[0; \infty)$.

Only if ("$\Longrightarrow$")-part of the conclusion. From *collision* $[0; \infty)$, we obtain a root t with $s_\mathrm{o}(t) - s_\mathrm{e}(t) = 0$. Then, Lemma 2 allows us to deduce $p_\mathrm{o}(t) - p_\mathrm{e}(t) = 0$ and condition (9) for convexity (for $p_\mathrm{o} - p_\mathrm{e}$ at times $0 < t < \min\{t_\text{stop,e}, t_\text{stop,o}\}$) yields $a_\mathrm{o} > a_\mathrm{e}$. This gives, together with the fact that the discriminant of $p_\mathrm{o} - p_\mathrm{e}$ is nonnegative according to Eq. (8), the remaining conjuncts of Eq. (15) after some arithmetic manipulations and reasoning. The whole proof takes about 130 lines in the formalisation. □

In order to unify this theorem with the pattern in (6), we negate the logical equivalence in (15) and rearrange the theorem as follows.

$$s_{0,\mathrm{o}} - s_{0,\mathrm{e}} > \frac{v_\mathrm{o}^2}{2 \cdot a_\mathrm{o}} - \frac{v_\mathrm{e}^2}{2 \cdot a_\mathrm{e}} \implies s_{0,\mathrm{o}} - s_{0,\mathrm{e}} > \frac{(v_\mathrm{o} - v_\mathrm{e})^2}{2 \cdot (a_\mathrm{o} - a_\mathrm{e})} \implies s_{0,\mathrm{o}} \leq s_\text{stop,e} \implies$$
$$(a_\mathrm{o} > a_\mathrm{e} \wedge v_\mathrm{o} < v_\mathrm{e} \wedge t_\text{stop,e} < t_\text{stop,o}) \implies \neg\, \textit{collision}\, [0; \infty) \quad (16)$$

This reformulation fits the pattern in (6) and now we have two possible safe distance expressions and one for precondition:

$$\textit{safe-distance}_2 := \frac{v_\mathrm{o}^2}{2 \cdot a_\mathrm{o}} - \frac{v_\mathrm{e}^2}{2 \cdot a_\mathrm{e}} \qquad \textit{safe-distance}_3 := \frac{(v_\mathrm{o} - v_\mathrm{e})^2}{2 \cdot (a_\mathrm{o} - a_\mathrm{e})} \ ,$$
$$\textit{precondition} := s_{0,\mathrm{o}} \leq s_\text{stop,e} \wedge (a_\mathrm{o} > a_\mathrm{e} \wedge v_\mathrm{o} < v_\mathrm{e} \wedge t_\text{stop,e} < t_\text{stop,o}) \quad (17)$$

To choose between these two expressions, we use the following lemma which determines their relative position.

Lemma 3 (Relative position of safe distance expressions).

$$a_\mathrm{o} > a_\mathrm{e} \implies \textit{safe-distance}_2 \leq \textit{safe-distance}_3$$

Proof. We prove this lemma by multiplying both sides with the multiplier $2 \cdot (a_0 - a_\mathrm{e})$ which is positive. Then, we reason backwards by performing arithmetical reasoning which eventually leads to $0 \leq (t_\text{stop,e} - t_\text{stop,o})^2$ which is always true. □

With this lemma, we choose *safe-distance*$_3$ when the *precondition* holds. Otherwise, it must be the case that $\neg\, (a_\mathrm{o} > a_\mathrm{e} \wedge v_\mathrm{o} < v_\mathrm{e} \wedge t_\text{stop,e} < t_\text{stop,o})$—since we assume case ③. Then, Theorem 3 ensures that *safe-distance*$_2$ is indeed a prescriptive definition of safe distance.

Overall definition. To sum up our logical analysis, *safe-distance*$_1$ always holds as a prescriptive definition of the safe distance. Expression *safe-distance*$_3$ holds when it is case ③ and *precondition* holds, while *safe-distance*$_2$ is valid when it is still case ③ but *precondition* does not hold.

4 Designing Sound Checkers for the Safe Distance Rule

We use the analyses from Sec. 3 to guide the design of sound and complete abstract checkers in Sec. 4.1; these checkers are defined in terms of real numbers and other non-executable constructs. We then show how to turn them into executable checkers by using exact rational arithmetic, symbolic decision procedures, or interval arithmetic in Sec. 4.2.

4.1 Abstract Checkers

We design two checkers here: a descriptive and a prescriptive version. Both checkers are derived from the case distinction in Sec. 3.1 and Sec. 3.2, respectively.

Descriptive Checker From the case distinction based on stopping times in Sec. 3.1, we conclude that the problem of detecting collision is reduced into the problem of finding solutions for each entry in Tab. 1 in the corresponding time interval. This is formalised with the predicate *has-root-in*, defined as $f(t)$ *has-root-in* $T \iff \exists t \in T.\ f(t) = 0$. A checker based on this approach can then be defined as follows.

$$\mathit{checker}_{\mathrm{d}} := \neg \begin{pmatrix} p_{\mathrm{e}}(t) - p_{\mathrm{o}}(t)\ \textit{has-root-in}\ [0; \min\{t_{\mathrm{stop,e}}, t_{\mathrm{stop,o}}\}] & \vee \\ p_{\mathrm{e}}(t_{\mathrm{stop,e}}) - p_{\mathrm{o}}(t)\ \textit{has-root-in}\ [t_{\mathrm{stop,e}}; t_{\mathrm{stop,o}}] & \vee \\ p_{\mathrm{o}}(t_{\mathrm{stop,o}}) - p_{\mathrm{e}}(t)\ \textit{has-root-in}\ [t_{\mathrm{stop,o}}; t_{\mathrm{stop,e}}] & \vee \\ p_{\mathrm{e}}(t_{\mathrm{stop,e}}) - p_{\mathrm{o}}(t_{\mathrm{stop,o}})\ \textit{has-root-in}\ [\max\{t_{\mathrm{stop,e}}, t_{\mathrm{stop,e}}\}; \infty) & \end{pmatrix}$$

The following theorem ensures that the checker is both sound and complete. It follows immediately from the definitions of braking movement p, stopping time t_{stop}, predicate *has-root-in*, and predicate *collision*:

Theorem 4 (Correctness of abstract descriptive checker).

$$\mathit{checker}_{\mathrm{d}} \iff \neg \mathit{collision}\,[0; \infty)$$

Prescriptive Checker From the case distinction based on stopping distances in Sec. 3.2, we have defined three expressions of safe distances. Each expression has associated preconditions for which the expression is valid. We can design the prescriptive checker from these expressions as follows:

$$\begin{aligned} \mathit{checker}_{\mathrm{p}} :=\ & \textbf{let}\ \mathit{dist} = s_{0,\mathrm{o}} - s_{0,\mathrm{e}}\ \textbf{in} \\ & \textbf{if}\ \underline{\mathit{dist} > \mathit{safe\text{-}distance}_1}\ \textbf{then}\ \mathit{True} \\ & \textbf{else if}\ a_0 > a_{\mathrm{e}} \wedge v_{\mathrm{o}} < v_{\mathrm{e}} \wedge t_{\mathrm{stop,e}} < t_{\mathrm{stop,o}}\ \textbf{then}\ \underline{\mathit{dist} > \mathit{safe\text{-}distance}_3} \\ & \textbf{else}\ \underline{\mathit{dist} > \mathit{safe\text{-}distance}_2} \end{aligned}$$

The following theorem states that the prescriptive checker is also sound and complete.

Theorem 5 (Correctness of abstract prescriptive checker).

$$\mathit{checker}_{\mathrm{p}} \iff \neg \mathit{collision}\,[0; \infty)$$

Proof. The soundness follows from the Theorem 1, 2, and 3 in Sec. 3 while the completeness comes from the fact that case ①, ②, and ③ cover all possible cases. □

4.2 Executable Checkers

A fragment of HOL can be seen as a functional programming language. When we talk about *executable* specifications, we talk about specifications within that fragment. In principle, such specifications could be evaluated inside Isabelle's kernel. For a more efficient evaluation, Isabelle/HOL comes with a code generator [10], which translates executable specifications to code for (functional) programming languages like SML, OCaml, Scala, or Haskell. We will generated the code for SML to evaluate the US-101 Highway data set in Sec. 5.

The aforementioned checkers $\mathit{checker}_{\mathrm{d}}$ and $\mathit{checker}_{\mathrm{p}}$ are formally proved correct, but are not executable, because they involve e.g., real numbers or quantifiers over real numbers (via *has-root-in*). We therefore refine them towards executable formulations. To this end, Isabelle provides a variety of techniques, and we explore the use of the following:

1. *Exact arithmetic on rational numbers.*
   Exact arithmetic on rational numbers can be directly used for $\mathit{checker}_{\mathrm{p}}$ if all parameters are rational numbers. It requires, however, the manual work of formalising the analysis presented in Section 3.2.
2. *Decision procedure for finding roots of univariate polynomials.*
   By contrast, using a decision procedure based on Sturm sequences for *has-root-in* in $\mathit{checker}_{\mathrm{d}}$ requires almost no manual reasoning. However, it has to be used as a black-box method and might not be easy to extend, if it is required.
3. *Interval arithmetic.*
   With interval arithmetic, one can include uncertainties into parameters of the model and could even address non-polynomial problems. Numerical uncertainties can, however, cause the checkers to be incomplete.

Exact rational arithmetic. All the operations occurring in $\mathit{checker}_{\mathrm{p}}$ could be executed on rational numbers. Under the assumption that all the parameters are rational numbers, $\mathit{checker}_{\mathrm{p}}$ can be executed using the standard approach of *data-refinement* [9] for real numbers in Isabelle/HOL. That is, the code generator is instructed to represent real numbers as a data type with a constructor $\mathit{Ratreal} : \mathbb{Q} \to \mathbb{R}$. Then, operations on the rational subset of the real numbers are defined by pattern matching on the constructor, and performing the corresponding operation on rational numbers. For example, addition $+_{\mathbb{R}}$ on *Ratreal*-constructed real numbers can be implemented with addition $+_{\mathbb{Q}}$ on rational numbers: $\mathit{Ratreal}(p) +_{\mathbb{R}} \mathit{Ratreal}(q) = \mathit{Ratreal}(p +_{\mathbb{Q}} q)$. Therefore, as long as the input is given as rational numbers, code generation for $\mathit{checker}_{\mathrm{p}}$ works without further manual setup. Correctness follows from Theorem 5.

Sturm sequences. A different approach can be followed by looking at the prescriptive formulation $\mathit{checker}_{\mathrm{d}}$. To evaluate *has-root-in*, we can resort to a decision procedure based on *Sturm sequences* which been formalised in Isabelle [5]. The interface to this decision procedure is an executable function $\mathit{count\text{-}roots}(p, I)$, which returns the number of roots of a given univariate polynomial p in a given interval I. It satisfies the proposition p *has-root-in* $T \longleftrightarrow (\mathit{count\text{-}roots}(p, I) > 0)$ and can therefore be used as an executable specification for the occurrences of *has-root-in* in $\mathit{checker}_{\mathrm{d}}$. Correctness follows from Theorem 4.

Interval arithmetic. The previous two approaches both assume that the parameters are given as exact rational numbers. One could argue that this is an unrealistic assumption, because real-world data cannot be measured exactly. For this checker, we therefore allow intervals of parameters. Isabelle's *approximation* [12] method allows us to interpret $\mathit{checker}_{\mathrm{p}}$ (a formula with inequalities over real numbers) as an expression in interval arithmetic. The resulting checker $\mathit{checker}_{\mathrm{i}}$ takes a Cartesian product of intervals as enclosure for the parameters as input.

Theorem 6 (Correctness of Checker).
If $(s_{\mathrm{e}}, v_{\mathrm{e}}, a_{\mathrm{e}}, s_{\mathrm{o}}, v_{\mathrm{o}}, a_{\mathrm{o}}) \in S_{\mathrm{e}} \times V_{\mathrm{e}} \times A_{\mathrm{e}} \times S_{\mathrm{o}} \times V_{\mathrm{o}} \times A_{\mathrm{o}}$*, then*

$$\mathit{checker}_{\mathrm{i}}\,(S_{\mathrm{e}}, V_{\mathrm{e}}, A_{\mathrm{e}}, S_{\mathrm{o}}, V_{\mathrm{o}}, A_{\mathrm{o}}) \implies \neg \mathit{collision}[0; \infty)$$

Proof. The theorem follows directly from the correctness of *approximation*. □

Note that we lose completeness in this approach; the checker could fail to prove collision-freedom because of imprecision in the approximate calculations. Such imprecision occurs because of, e.g., finite precision calculations or case distinctions that cannot be resolved. It might be that in a case distinction $a < b \vee a \geq b$, none of the two disjuncts can be proved (consider e.g. $a \in [0; 1], b \in [0; 1]$) with just interval arithmetic. Tracking dependencies between input variables or interval constraint propagation approaches could alleviate this problem.

5 Data Analysis of the Safe Distance Problem

The traffic data used in this evaluation are obtained from the Next Generation SIMulation (NGSIM) project of the U.S. Department of Transportation Federal Highway Administration (FHWA). We specifically focus on the data set for the US Highway 101 (US-101). The length of the study area is about 640 meters with five lanes in total and the data was collected for 45 minutes. For every identified car, the data set provides information such as the position, speed, acceleration, length of the vehicle, and distance to the other vehicle with a time resolution of $0.1\,\mathrm{s}$.

The US-101 data set does not provide any information about the maximum deceleration of the vehicles. The maximum deceleration value can be obtain from the values of tyre friction on dry condition. We take these values from the domain of traffic collision reconstruction [6] which has been used by lawyers in court [3]. The tyre friction values for automobile and motorcycle are $\mu_{\text{motor}} = 0.75$ and $\mu_{\text{auto}} = 0.8$, respectively. As for truck and bus, we take the value from [19], i.e., $\mu_{\text{truck}} = 0.7$. By assuming $g = 9.8\,\mathrm{m\,s^{-2}}$, these tyre friction values correspond to maximum deceleration values of $a_{\text{motor}} = -7.35\,\mathrm{m\,s^{-2}}$, $a_{\text{auto}} = -7.84\,\mathrm{m\,s^{-2}}$, and $a_{\text{truck}} = -6.86\,\mathrm{m\,s^{-2}}$.

We evaluate three executable checkers: (1) the exact rational arithmetic-based prescriptive checker, (2) the Sturm sequences-based descriptive checker, and (3) the interval arithmetic-based prescriptive checker. Interval arithmetic-based checker is parameterised with uncertainty u which represents the measurement error in the data. This parameter, however, does not represent the error due to floating-point computation which is handled internally by the *approximation* decision procedure in this interval

Table 2: Number of detected safe distance situations and time performance of each checker (for $N = 3{,}915{,}006$ data points).

Checker	u	Safe dist. (%)	Time
$\text{Descriptive}_{\text{Sturm}}$	-	$99.74\,\%$	$1068.32\,\text{s}$
$\text{Prescriptive}_{\text{exact}}$	-	$99.74\,\%$	$168.93\,\text{s}$
$\text{Prescriptive}_{\text{interval}}$	7	$99.05\,\%$	$352.73\,\text{s}$
$\text{Prescriptive}_{\text{interval}}$	5	$97.48\,\%$	$323.56\,\text{s}$
$\text{Prescriptive}_{\text{interval}}$	3	$90.92\,\%$	$324.23\,\text{s}$

arithmetic-based checker. Each time this checker evaluates an arithmetic expression, the interval of each evaluated subexpression is enlarged accordingly so as to include the error due to fixed precision of floating-point numbers.

Two aspects are measured for each checker: the number of detected safe distance situations and the CPU time for checking the whole data set. The measurement is performed with an Intel i5-4330M $2.80\,\text{GHz}$ processor and $12\,\text{GB}$ of RAM. We draw four conclusions from the results in Tab. 2:

1. Both $\text{prescriptive}_{\text{exact}}$ and $\text{descriptive}_{\text{Sturm}}$ checkers detect the same number of safe distance situations. This is not surprising since we have formally proved the correctness of both checkers and they use exact arithmetic.
2. Interval arithmetic-based checkers detect fewer safe distance situations than the other checkers do. This shows that this checker is more conservative and incomplete.
3. The number of safe distance scenario detected decreases as the uncertainty parameter u decreases. This is because the uncertainty parameter u corresponds to the uncertainty value of 2^{-u} and, hence, a decrease of uncertainty parameter u is equivalent to an increase of the uncertainty value.
4. The prescriptive checker has a better time performance than the descriptive checker. This is understandable because the descriptive checker is based on a more general decision procedure (Sturm sequences), and the prescriptive checker is heavily tuned for this safe distance problem.
5. The $\text{prescriptive}_{\text{exact}}$ checker detects safe distance situations approximately *two* times faster than those $\text{prescriptive}_{\text{interval}}$ checkers. This is because the critical factor in the time performance of these two types of checkers lies in the computation of *dist* in $\mathit{checker}_{\text{p}}$. Subtracting an interval by another interval essentially consists of *two* exact arithmetic subtractions — one each for the lower bound and upper bound.

Two caveats regarding the results from Tab. 2 are worth mentioning here. First, when the $\text{prescriptive}_{\text{exact}}$ and $\text{descriptive}_{\text{Sturm}}$ checker return *False*, they do not detect a collision but a guaranteed-to-happen collision if the ego and the other vehicles brake with full deceleration. Second, when the $\text{prescriptive}_{\text{interval}}$ checkers return *False*, no conclusion can be drawn concerning the potential collision due to violating safe distance rule (see Thm. 6). This inconclusive answer is because either the uncertainty u for the data is too large or the precision for the floating-point approximation is too limited.

6 Related Work

In this section, we compare our formalisation with results from the domain of transportation engineering and formal verification. One notable difference between our work with the others is that we ignore the reaction time for the ego vehicle. Hence, when comparing our work with others, those parameters are set to zero. In general, all related works discussed here except the work by Goodloe et al. [8] are incomplete, and those in the domain of transportation engineering (discussed here) are not formally proved.

In the domain of traffic engineering, there are two areas which are related to our work: traffic simulation and collision warning. Mazda and PATHS algorithms [1]—for collision warning—and Gipps's model [7]—for traffic simulation—formulate the notion of safe distance which exactly match our second definition of safe distance in Eq. (17). Qu et al. [22] analyse the safe distance problem by applying a technique from molecular dynamics. Unlike the case distinction in our work, they have three cases which depend on the relationship between v_e and v_o. Their notion of safe distance for case $v_\mathrm{e} > v_\mathrm{o}$ and $v_\mathrm{e} = v_\mathrm{o}$ matches exactly with our second definition of safe distance in Eq. (17). However, their notion of safe distance when $v_\mathrm{e} < v_\mathrm{o}$ does not match with any of our definitions of safe distance due to different assumptions. A more detailed analysis for the safe distance problem is given by Chen et al. [4]. If we consider their single lane scenario only, their definitions of safe distance for stationary and decelerating case exactly match our first and second definition of safe distance in Eq. (13) and Eq. (17), respectively.

The related work described up until now always assume that the maximum deceleration for all vehicles is the same. Therefore, none of the works described previously matches our third definition of safe distance. Wilson [25] performs case distinction based on the stopping times and graphically identifies the region called "envelope of opportunity" for each case. This envelope of opportunity divides the plot between the reaction time and the deceleration of the ego vehicle into safe and unsafe region. The envelope of opportunity for $t_{\text{stop,e}} > t_{\text{stop,o}}$ and $t_{\text{stop,e}} < t_{\text{stop,o}}$ match our second and third definition of safe distance in Eq. (17), respectively.

Loos et al. [15] verify ACC formally in KeYmaera where, in their model of ACC, they axiomatise that a safe distance is formalised as the second safe distance definition in Eq. (17). This safe distance definition is then modified to take into account all possible impacts of control decisions for the future of reaction time, and then setting it as an invariant for the controller. They then use the proof calculus for the quantified differential dynamic logic (Qd$\mathcal{L}$) [21] to prove that the controller maintains this invariant, which in turn implies the axiomatised safe distance in Eq. (17) by transitivity. Our work completes theirs by proving that this axiomatised safe distance is indeed safe. However, their controller is safe on the assumption that all vehicles have the same braking performance.

Although Goodloe et al. [8] formally verify programs for aerospace applications, namely airborne conflict detection and resolution (CD&R), their approach is in general very similar to ours. Their objective is to verify whether a checker correctly determines that two aircraft maintain a minimum separation distance. Similar to our work, they also define an abstract checker, prove its soundness and completeness in PVS theorem prover, derive a concrete checker in C, and prove that the refinement from abstract to

concrete checker is correct in Frama-C. Our work differs in the step to convert from abstract to concrete checker. Thanks to the code generation facility in Isabelle, we can generate the concrete checker *automatically* in SML.

7 Conclusion and Future Work

We have formalised descriptive and prescriptive versions of the safe distance traffic rule from the Vienna Convention on Road Traffic. For each version, we have also derived two corresponding abstract checkers, which operate on real numbers, and proved their soundness and completeness. The prescriptive checker is refined further into a concrete checker in SML which operates on rational numbers. Interval arithmetic is used here to ensure that it preserves the soundness property despite the error due to the limited precision of floating-point numbers. We then use these two checkers together with the Sturm sequences-based checker to evaluate the US-101 Highway data set from NGSIM.

Our work serves as an example of how one can use theorem provers, especially Isabelle, to turn a vague requirement from a legal text into a more precise and concrete specification. Isabelle, as a framework, also provides us with a unified platform to prove theorems, to design a checker, to prove the soundness of the checker, and to generate the (functional) code automatically. From the evaluation of the data set, we found that at least 90% of the time, each traffic participant—if we assume them to be autonomous vehicles—obeys the safe distance rule. *Our work advances the state-of-the-art by providing a unique combination of formally proved and complete safe distance definitions which generalise all definitions in the literature, formally proved checkers without strict assumptions on braking performance, and real data evaluation.*

We wish to extend this work by considering the reaction time of the ego vehicle. It might also be interesting to see how our third definition of safe distance can be incorporated into the controller in [15] when considering vehicles with different braking performance. To make the reasoning easier, we would like to have more automation for real arithmetic in Isabelle/HOL. We assume that the verification could be more organised by following a dedicated calculus for hybrid systems [20], which could be embedded in Isabelle/HOL. Our checker could also be extended with reachability analysis [2,13] in order to verify a continuous trace. Lastly, aligned with our previous work in formalisation of traffic rules [23], we wish to increase the number of formalised traffic rules such that the liability issue can be deduced automatically with our checkers.

References

1. Aghabayk, K., Sarvi, M., Young, W.: A state-of-the-art review of car-following models with particular considerations of heavy vehicles. Transport Reviews 35(1), 82–105 (2015)
2. Althoff, M., Dolan, J.: Online verification of automated road vehicles using reachability analysis. IEEE Transactions on Robotics 30(4), 903–918 (Aug 2014)
3. American Prosecutors Research Institute: Crash Reconstruction Basics for Prosecutors: Targeting Hardcore Impaired Drivers. Author (2003)
4. Chen, C., Liu, L., Du, X., Pei, Q., Zhao, X.: Improving driving safety based on safe distance design in vehicular sensor networks. International Journal of Distributed Sensor Networks 2012(Article ID 469067) (2012)

5. Eberl, M.: A decision procedure for univariate real polynomials in Isabelle/HOL. In: Proceedings of the 2015 Conference on Certified Programs and Proofs, CPP 2015, Mumbai, India, January 15-17, 2015. LNCS, vol. 9035, pp. 75–83. Springer (2015)
6. Fricke, L.: Traffic Accident Reconstruction. The Traffic Accident Investigation Manual, Vol. 2, Northwestern University Center for Public Safety (1990)
7. Gipps, P.: A behavioural car-following model for computer simulation. Transportation Research Part B: Methodological 15(2), 105 – 111 (1981)
8. Goodloe, A., Muñoz, C.A., Kirchner, F., Correnson, L.: Verification of numerical programs: From real numbers to floating point numbers. In: NASA Formal Methods. LNCS, vol. 7871, pp. 441–446. Springer (2013)
9. Haftmann, F., Krauss, A., Kunčar, O., Nipkow, T.: Data refinement in Isabelle/HOL. In: Interactive Theorem Proving. LNCS, vol. 7998, pp. 100–115. Springer (2013)
10. Haftmann, F., Nipkow, T.: Code generation via higher-order rewrite systems. In: Functional and Logic Programming. LNCS, vol. 6009, pp. 103–117. Springer (2010)
11. Harrison, J.: Verifying nonlinear real formulas via sums of squares. In: Theorem Proving in Higher Order Logics. LNCS, vol. 4732, pp. 102–118. Springer (2007)
12. Hölzl, J.: Proving inequalities over reals with computation in Isabelle/HOL. In: Proc. of the ACM SIGSAM International Workshop on Programming Languages for Mechanized Mathematics Systems. pp. 38–45 (2009)
13. Immler, F.: Verified reachability analysis of continuous systems. In: Tools and Algorithms for the Construction and Analysis of Systems. LNCS, vol. 9035, pp. 37–51. Springer (2015)
14. Kowshik, H., Caveney, D., Kumar, P.: Provable systemwide safety in intelligent intersections. IEEE Transactions on Vehicular Technology 60(3), 804–818 (March 2011)
15. Loos, S.M., Platzer, A., Nistor, L.: Adaptive cruise control: Hybrid, distributed, and now formally verified. In: Formal Methods. LNCS, vol. 6664, pp. 42–56. Springer (2011)
16. Loos, S., Platzer, A.: Safe intersections: At the crossing of hybrid systems and verification. In: IEEE Conference on Intelligent Transportations Systems. pp. 1181–1186 (Oct 2011)
17. McLaughlin, S., Harrison, J.: A proof-producing decision procedure for real arithmetic. In: Automated Deduction. LNCS, vol. 3632, pp. 295–314. Springer (2005)
18. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL - A Proof Assistant for Higher-Order Logic, LNCS, vol. 2283. Springer (2002)
19. Olsen, R.A.: Pedestrian injury issues in litigation. In: Karwowski, W., Noy, Y.I. (eds.) Handbook of Human Factors in Litigation, pp. 15–1–15–23. CRC Press (Dec 2004)
20. Platzer, A.: Logical Analysis of Hybrid Systems. Springer Berlin Heidelberg (2010)
21. Platzer, A.: A complete axiomatization of quantified differential dynamic logic for distributed hybrid systems. Logical Methods in Computer Science 8(4), 1–44 (2012)
22. Qu, D., Chen, X., Yang, W., Bian, X.: Modeling of car-following required safe distance based on molecular dynamics. Mathematical Problems in Engineering 2014(Article ID 604023) (2014)
23. Rizaldi, A., Althoff, M.: Formalising traffic rules for accountability of autonomous vehicles. In: IEEE Conference on Intelligent Transportation Systems. pp. 1658–1665 (2015)
24. Vanholme, B., Gruyer, D., Lusetti, B., Glaser, S., Mammar, S.: Highly automated driving on highways based on legal safety. IEEE Transactions on Intelligent Transportation Systems 14(1), 333–347 (2013)
25. Wilson, B.H.: How soon to brake and how hard to brake: Unified analysis of the envelope of opportunity for rear-end collision warnings. In: Enhanced Safety of Vehicles. vol. 47 (2001)
26. Xiao, L., Gao, F.: A comprehensive review of the development of adaptive cruise control systems. Vehicle System Dynamics 48(10), 1167–1192 (2010)