

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

Albert Rizaldi, Fabian Immler

March 17, 2025

Abstract

The Vienna Convention on Road Traffic defines the safe distance traffic rules informally. This could make autonomous vehicle liable for safe-distance-related accidents because there is no clear definition of how large a safe distance is. We provide a formally proven prescriptive definition of a safe distance, and checkers which can decide whether an autonomous vehicle is obeying the safe distance rule. Not only does our work apply to the domain of law, but it also serves as a specification for autonomous vehicle manufacturers and for online verification of path planners. This formalization accompanies our paper "A Formally Verified Checker of the Safe Distance Traffic Rules for Autonomous Vehicles". [1]

Contents

1	Safe Distance	2
1.1	Quadratic Equations	2
1.2	Convexity Condition	3
1.3	Movement	4
1.3.1	Continuous Dynamics	4
1.3.2	Hybrid Dynamics	6
1.3.3	Monotonicity of Movement	8
1.3.4	Maximum at Stopping Time	9
1.4	Safe Distance	9
1.4.1	Collision	10
1.4.2	Formalising Safe Distance	11
1.5	Checker Design	13
1.5.1	Prescriptive Checker	15
1.5.2	Approximate Checker	16
1.5.3	Symbolic Checker	18

2	Safe Distance with Reaction Time	18
2.1	Normal Safe Distance	18
2.2	Safe Distance Delta	24
2.3	Checker Design	27
3	Evaluation	29
3.1	Code Generation Setup for Numeric Values	29
3.2	Data Evaluation	31

1 Safe Distance

```

theory Safe-Distance
imports
  HOL-Analysis.Multivariate-Analysis
  HOL-Decision-Procs.Approximation
  Sturm-Sequences.Sturm
begin

```

This theory is about formalising the safe distance rule. The safe distance rule is obtained from Vienna Convention which basically states the following thing.

“The car at all times must maintain a safe distance towards the vehicle in front of it, such that whenever the vehicle in front and the ego vehicle apply maximum deceleration, there will not be a collision.”

To formalise this safe distance rule we have to define first what is a safe distance. To define this safe distance, we have to model the physics of the movement of the vehicle. The following model is sufficient.

$$s = s_0 + v_0 * t + 1 / 2 * a_0 * t^2$$

Assumptions in this model are :

- Both vehicles are assumed to be point mass. The exact location of the ego vehicle is the front-most occupancy of the ego vehicle. Similarly for the other vehicle, its exact location is the rearmost occupancy of the other vehicle.
- Both cars can never drive backward.

```

lemmas [simp del] = div-mult-self1 div-mult-self2 div-mult-self3 div-mult-self4

```

1.1 Quadratic Equations

```

lemma discriminant:  $a * x^2 + b * x + c = (0::real) \implies 0 \leq b^2 - 4 * a * c$ 
  <proof>

```

```

lemma quadratic-eq-factoring:

```

assumes $D : D = b^2 - 4 * a * c$
assumes $nn : 0 \leq D$
assumes $x1 : x_1 = (-b + \text{sqrt } D) / (2 * a)$
assumes $x2 : x_2 = (-b - \text{sqrt } D) / (2 * a)$
assumes $a : a \neq 0$
shows $a * x^2 + b * x + c = a * (x - x_1) * (x - x_2)$
 <proof>

lemma *quadratic-eq-zeroes-iff*:

assumes $D : D = b^2 - 4 * a * c$
assumes $x1 : x_1 = (-b + \text{sqrt } D) / (2 * a)$
assumes $x2 : x_2 = (-b - \text{sqrt } D) / (2 * a)$
assumes $a : a \neq 0$
shows $a * x^2 + b * x + c = 0 \iff (D \geq 0 \wedge (x = x_1 \vee x = x_2))$ (**is** ?z \iff -)
 <proof>

1.2 Convexity Condition

lemma *p-convex*:

fixes $a b c x y z :: \text{real}$
assumes $p\text{-def} : p = (\lambda x. a * x^2 + b * x + c)$
assumes $less : x < y \implies y < z$
and $ge : p x > p y \implies p y \leq p z$
shows $a > 0$
 <proof>

definition *root-in* :: $\text{real} \Rightarrow \text{real} \Rightarrow (\text{real} \Rightarrow \text{real}) \Rightarrow \text{bool}$ **where**

$root\text{-in } m M f = (\exists x \in \{m .. M\}. f x = 0)$

definition *quadroot-in* :: $\text{real} \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow \text{bool}$ **where**

$quadroot\text{-in } m M a b c = root\text{-in } m M (\lambda x. a * x^2 + b * x + c)$

lemma *card-iff-exists*: $0 < \text{card } X \iff \text{finite } X \wedge (\exists x. x \in X)$

<proof>

lemma *quadroot-in-sturm*[code]:

$quadroot\text{-in } m M a b c \iff (a = 0 \wedge b = 0 \wedge c = 0 \wedge m \leq M) \vee$
 $(m \leq M \wedge \text{poly } [:c, b, a] m = 0) \vee$
 $\text{count-roots-between } [:c, b, a] m M > 0$
 <proof>

lemma *check-quadroot-linear*:

fixes $a b c :: \text{real}$
assumes $a = 0$
shows $\neg quadroot\text{-in } m M a b c \iff$
 $((b = 0 \wedge c = 0 \wedge M < m) \vee (b = 0 \wedge c \neq 0) \vee$
 $(b \neq 0 \wedge (\text{let } x = -c / b \text{ in } m > x \vee x > M)))$
 <proof>

lemma *check-quadroot-nonlinear*:
assumes $a \neq 0$
shows *quadroot-in* $m M a b c =$
 $(\text{let } D = b^2 - 4 * a * c \text{ in } D \geq 0 \wedge$
 $((\text{let } x = (-b + \text{sqrt } D) / (2 * a) \text{ in } m \leq x \wedge x \leq M) \vee$
 $(\text{let } x = (-b - \text{sqrt } D) / (2 * a) \text{ in } m \leq x \wedge x \leq M)))$
 $\langle \text{proof} \rangle$

lemma *ncheck-quadroot*:
shows $\neg \text{quadroot-in } m M a b c \longleftrightarrow$
 $(a = 0 \longrightarrow \neg \text{quadroot-in } m M a b c) \wedge$
 $(a = 0 \vee \neg \text{quadroot-in } m M a b c)$
 $\langle \text{proof} \rangle$

1.3 Movement

locale *movement* =
fixes $a v s0 :: \text{real}$
begin

Function to compute the distance using the equation

$$s(t) = s_0 + v_0 * t + 1 / 2 * a_0 * t^2$$

Input parameters:

- s_0 : initial distance
- v_0 : initial velocity (positive means forward direction and the converse is true)
- a : acceleration (positive for increasing and negative for decreasing)
- t : time

For the time $t < 0$, we assume the output of the function is s_0 . Otherwise, the output is calculated according to the equation above.

1.3.1 Continuous Dynamics

definition $p :: \text{real} \Rightarrow \text{real}$ **where**
 $p t = s0 + v * t + 1/2 * a * t^2$

lemma *p-all-zeroes*:
assumes $D: D = v^2 - 2 * a * s0$
shows $p t = 0 \longleftrightarrow ((a \neq 0 \wedge 0 \leq D \wedge ((t = (-v + \text{sqrt } D) / a) \vee t = (-v - \text{sqrt } D) / a)) \vee$
 $(a = 0 \wedge v = 0 \wedge s0 = 0) \vee (a = 0 \wedge v \neq 0 \wedge t = (-s0 / v)))$
 $\langle \text{proof} \rangle$

lemma *p-zero*[simp]: $p\ 0 = s0$
⟨proof⟩

lemma *p-continuous*[continuous-intros]: *continuous-on* $T\ p$
⟨proof⟩

lemma *isCont-p*[continuous-intros]: *isCont* $p\ x$
⟨proof⟩

definition $p' :: \text{real} \Rightarrow \text{real}$ **where**
 $p'\ t = v + a * t$

lemma *p'-zero*: $p'\ 0 = v$
⟨proof⟩

lemma *p-has-vector-derivative*[derivative-intros]: (*p has-vector-derivative* $p'\ t$) (at t within s)
⟨proof⟩

lemma *p-has-real-derivative*[derivative-intros]: (*p has-real-derivative* $p'\ t$) (at t within s)
⟨proof⟩

definition $p'' :: \text{real} \Rightarrow \text{real}$ **where**
 $p''\ t = a$

lemma *p'-has-vector-derivative*[derivative-intros]: (*p' has-vector-derivative* $p''\ t$) (at t within s)
⟨proof⟩

lemma *p'-has-real-derivative*[derivative-intros]: (*p' has-real-derivative* $p''\ t$) (at t within s)
⟨proof⟩

definition *t-stop* :: *real* **where**
 $t\text{-stop} = -v / a$

lemma *p'-stop-zero*: $p'\ t\text{-stop} = (\text{if } a = 0 \text{ then } v \text{ else } 0)$ ⟨proof⟩

lemma *p'-pos-iff*: $p'\ x > 0 \iff (\text{if } a > 0 \text{ then } x > -v / a \text{ else if } a < 0 \text{ then } x < -v / a \text{ else } v > 0)$
⟨proof⟩

lemma *le-t-stop-iff*: $a \neq 0 \implies x \leq t\text{-stop} \iff (\text{if } a < 0 \text{ then } p'\ x \geq 0 \text{ else } p'\ x \leq 0)$
⟨proof⟩

lemma *p'-continuous*[continuous-intros]: *continuous-on* $T\ p'$
⟨proof⟩

lemma *isCont-p'*[*continuous-intros*]: *isCont p' x*
⟨*proof*⟩

definition *p-max* :: *real* **where**
p-max = *p t-stop*

lemmas *p-t-stop* = *p-max-def*[*symmetric*]

lemma *p-max-eq*: *p-max* = *s0 - v² / a / 2*
⟨*proof*⟩

1.3.2 Hybrid Dynamics

definition *s* :: *real* ⇒ *real* **where**
s t = (*if t ≤ 0 then s0*
 else if t ≤ t-stop then p t
 else p-max)

definition *q* :: *real* ⇒ *real* **where**
q t = *s0 + v * t*

definition *q'* :: *real* ⇒ *real* **where**
q' t = *v*

lemma *init-q*: *q 0* = *s0* ⟨*proof*⟩

lemma *q-continuous*[*continuous-intros*]: *continuous-on T q*
⟨*proof*⟩

lemma *isCont-q*[*continuous-intros*]: *isCont q x*
⟨*proof*⟩

lemma *q-has-vector-derivative*[*derivative-intros*]: (*q has-vector-derivative q' t*) (at *t* within *u*)
⟨*proof*⟩

lemma *q-has-real-derivative*[*derivative-intros*]: (*q has-real-derivative q' t*) (at *t* within *u*)
⟨*proof*⟩

lemma *s-cond-def*:
t ≤ 0 ⇒ *s t* = *s0*
0 ≤ t ⇒ *t ≤ t-stop* ⇒ *s t* = *p t*
⟨*proof*⟩

end

locale *braking-movement* = *movement* +

assumes *decel*: $a < 0$
assumes *nonneg-vel*: $v \geq 0$
begin

lemma *t-stop-nonneg*: $0 \leq t\text{-stop}$
 ⟨*proof*⟩

lemma *t-stop-pos*:
assumes $v \neq 0$
shows $0 < t\text{-stop}$
 ⟨*proof*⟩

lemma *t-stop-zero*:
assumes $t\text{-stop} = 0$
shows $v = 0$
 ⟨*proof*⟩

lemma *t-stop-zero-not-moving*: $t\text{-stop} = 0 \implies q\ t = s\ 0$
 ⟨*proof*⟩

abbreviation $s\text{-stop} \equiv s\ t\text{-stop}$

lemma *s-t-stop*: $s\text{-stop} = p\text{-max}$
 ⟨*proof*⟩

lemma *s0-le-s-stop*: $s\ 0 \leq s\text{-stop}$
 ⟨*proof*⟩

lemma *p-mono*: $x \leq y \implies y \leq t\text{-stop} \implies p\ x \leq p\ y$
 ⟨*proof*⟩

lemma *p-antimono*: $x \leq y \implies t\text{-stop} \leq x \implies p\ y \leq p\ x$
 ⟨*proof*⟩

lemma *p-strict-mono*: $x < y \implies y \leq t\text{-stop} \implies p\ x < p\ y$
 ⟨*proof*⟩

lemma *p-strict-antimono*: $x < y \implies t\text{-stop} \leq x \implies p\ y < p\ x$
 ⟨*proof*⟩

lemma *p-max*: $p\ x \leq p\text{-max}$
 ⟨*proof*⟩

lemma *continuous-on-s[continuous-intros]*: *continuous-on* $T\ s$
 ⟨*proof*⟩

lemma *isCont-s[continuous-intros]*: *isCont* $s\ x$
 ⟨*proof*⟩

definition $s' :: \text{real} \Rightarrow \text{real}$ **where**
 $s' t = (\text{if } t \leq t\text{-stop then } p' t \text{ else } 0)$

lemma *s-has-real-derivative*:
assumes $t \geq 0 \ v / a \leq 0 \ a \neq 0$
shows (*s has-real-derivative s' t*) (at t within {0..})
 $\langle \text{proof} \rangle$

lemma *s-has-vector-derivative*[*derivative-intros*]:
assumes $t \geq 0 \ v / a \leq 0 \ a \neq 0$
shows (*s has-vector-derivative s' t*) (at t within {0..})
 $\langle \text{proof} \rangle$

lemma *s-has-field-derivative*[*derivative-intros*]:
assumes $t \geq 0 \ v / a \leq 0 \ a \neq 0$
shows (*s has-field-derivative s' t*) (at t within {0..})
 $\langle \text{proof} \rangle$

lemma *s-has-real-derivative-at*:
assumes $0 < x \ 0 \leq v \ a < 0$
shows (*s has-real-derivative s' x*) (at x)
 $\langle \text{proof} \rangle$

lemma *s-delayed-has-field-derivative*[*derivative-intros*]:
assumes $\delta < t \ 0 \leq v \ a < 0$
shows ($(\lambda x. s (x - \delta))$ has-field-derivative $s' (t - \delta)$) (at t within $\{\delta < ..\}$)
 $\langle \text{proof} \rangle$

lemma *s-delayed-has-vector-derivative*[*derivative-intros*]:
assumes $\delta < t \ 0 \leq v \ a < 0$
shows ($(\lambda x. s (x - \delta))$ has-vector-derivative $s' (t - \delta)$) (at t within $\{\delta < ..\}$)
 $\langle \text{proof} \rangle$

lemma *s'-nonneg*: $0 \leq v \implies a \leq 0 \implies 0 \leq s' x$
 $\langle \text{proof} \rangle$

lemma *s'-pos*: $0 \leq x \implies x < t\text{-stop} \implies 0 \leq v \implies a \leq 0 \implies 0 < s' x$
 $\langle \text{proof} \rangle$

1.3.3 Monotonicity of Movement

lemma *s-mono*:
assumes $t \geq u \ u \geq 0$
shows $s t \geq s u$
 $\langle \text{proof} \rangle$

lemma *s-strict-mono*:
assumes $u < t \ t \leq t\text{-stop} \ u \geq 0$
shows $s u < s t$

<proof>

lemma *s-antimono*:

assumes $x \leq y$

assumes $t\text{-stop} \leq x$

shows $s\ y \leq s\ x$

<proof>

lemma *init-s* : $t \leq 0 \implies s\ t = s\ 0$

<proof>

lemma *q-min*: $0 \leq t \implies s\ 0 \leq q\ t$

<proof>

lemma *q-mono*: $x \leq y \implies q\ x \leq q\ y$

<proof>

1.3.4 Maximum at Stopping Time

lemma *s-max*: $s\ x \leq s\text{-stop}$

<proof>

lemma *s-eq-s-stop*: *NO-MATCH* $t\text{-stop}\ x \implies x \geq t\text{-stop} \implies s\ x = s\text{-stop}$

<proof>

end

1.4 Safe Distance

locale *safe-distance* =

fixes $a_e\ v_e\ s_e :: \text{real}$

fixes $a_o\ v_o\ s_o :: \text{real}$

assumes *nonneg-vel-ego* : $0 \leq v_e$

assumes *nonneg-vel-other* : $0 \leq v_o$

assumes *decelerate-ego* : $a_e < 0$

assumes *decelerate-other* : $a_o < 0$

assumes *in-front* : $s_e < s_o$

begin

lemmas *hyps* =

nonneg-vel-ego

nonneg-vel-other

decelerate-ego

decelerate-other

in-front

sublocale *ego*: *braking-movement* $a_e\ v_e\ s_e$ *<proof>*

sublocale *other*: *braking-movement* $a_o\ v_o\ s_o$ *<proof>*

sublocale *ego-other*: *movement* $a_o - a_e\ v_o - v_e\ s_o - s_e$ *<proof>*

1.4.1 Collision

definition *collision* :: real set \Rightarrow bool **where**

$$\text{collision time-set} \equiv (\exists t \in \text{time-set}. \text{ego.s } t = \text{other.s } t)$$

abbreviation *no-collision* :: real set \Rightarrow bool **where**

$$\text{no-collision time-set} \equiv \neg \text{collision time-set}$$

lemma *no-collision-initially* : no-collision {.. 0}

\langle proof \rangle

lemma *no-collisionI*:

$$(\bigwedge t. t \in S \implies \text{ego.s } t \neq \text{other.s } t) \implies \text{no-collision } S$$

\langle proof \rangle

theorem *cond-1*: ego.s-stop < s_o \implies no-collision {0..}

\langle proof \rangle

lemma *ego-other-strict-ivt*:

assumes ego.s t > other.s t

shows collision {0 ..< t}

\langle proof \rangle

lemma *collision-subset*: collision s \implies s \subseteq t \implies collision t

\langle proof \rangle

lemma *ego-other-ivt*:

assumes ego.s t \geq other.s t

shows collision {0 .. t}

\langle proof \rangle

theorem *cond-2*:

assumes ego.s-stop \geq other.s-stop

shows collision {0 ..}

\langle proof \rangle

abbreviation *D2* :: real **where**

$$D2 \equiv (v_o - v_e)^2 - 2 * (a_o - a_e) * (s_o - s_e)$$

abbreviation *t_D'* :: real **where**

$$t_D' \equiv \text{sqrt } (2 * (\text{ego.s-stop} - \text{other.s-stop}) / a_o)$$

lemma *pos-via-half-dist*:

$$\text{dist } a \ b < b / 2 \implies b > 0 \implies a > 0$$

\langle proof \rangle

lemma *collision-within-p*:

assumes s_o \leq ego.s-stop ego.s-stop < other.s-stop

shows collision {0..} \iff ($\exists t \geq 0. \text{ego.p } t = \text{other.p } t \wedge t < \text{ego.t-stop} \wedge t < \text{other.t-stop}$)

<proof>

lemma *collision-within-eq*:

assumes $s_o \leq \text{ego.s-stop}$ $\text{ego.s-stop} < \text{other.s-stop}$

shows $\text{collision } \{0..\} \longleftrightarrow \text{collision } \{0 .. < \min \text{ego.t-stop } \text{other.t-stop}\}$

<proof>

lemma *collision-excluded*: $(\bigwedge t. t \in T \implies \text{ego.s } t \neq \text{other.s } t) \implies \text{collision } S \longleftrightarrow \text{collision } (S - T)$

<proof>

lemma *collision-within-less*:

assumes $s_o \leq \text{ego.s-stop}$ $\text{ego.s-stop} < \text{other.s-stop}$

shows $\text{collision } \{0..\} \longleftrightarrow \text{collision } \{0 <.. < \min \text{ego.t-stop } \text{other.t-stop}\}$

<proof>

theorem *cond-3*:

assumes $s_o \leq \text{ego.s-stop}$ $\text{ego.s-stop} < \text{other.s-stop}$

shows $\text{collision } \{0..\} \longleftrightarrow (a_o > a_e \wedge v_o < v_e \wedge 0 \leq D2 \wedge \text{sqrt } D2 > v_e - a_e / a_o * v_o)$

<proof>

1.4.2 Formalising Safe Distance

First definition for Safe Distance based on *cond-1*.

definition *absolute-safe-distance* :: *real* **where**

$\text{absolute-safe-distance} = - v_e^2 / (2 * a_e)$

lemma *absolute-safe-distance*:

assumes $s_o - s_e > \text{absolute-safe-distance}$

shows $\text{no-collision } \{0..\}$

<proof>

First Fallback for Safe Distance.

definition *fst-safe-distance* :: *real* **where**

$\text{fst-safe-distance} = v_o^2 / (2 * a_o) - v_e^2 / (2 * a_e)$

definition *distance-leq-d2* :: *real* **where**

$\text{distance-leq-d2} = (a_e + a_o) / (2 * a_o^2) * v_o^2 - v_o * v_e / a_o$

lemma *snd-leq-fst-exp*: $\text{distance-leq-d2} \leq \text{fst-safe-distance}$

<proof>

lemma *sqrt-D2-leq-stop-time-diff*:

assumes $a_e < a_o$

assumes $0 \leq v_e - a_e / a_o * v_o$

assumes $s_o - s_e \geq \text{distance-leq-d2}$

shows $\text{sqrt } D2 \leq v_e - a_e / a_o * v_o$

<proof>

lemma *cond2-imp-pos-vo*:
assumes $s_o \leq \text{ego.s-stop}$ $\text{ego.s-stop} < \text{other.s-stop}$
shows $v_o \neq 0$
<proof>

lemma *cond2-imp-gt-fst-sd*:
assumes $s_o \leq \text{ego.s-stop}$ $\text{ego.s-stop} < \text{other.s-stop}$
shows $\text{fst-safe-distance} < s_o - s_e$
<proof>

Second Fallback for Safe Distance.

definition *snd-safe-distance* :: *real* **where**
 $\text{snd-safe-distance} = (v_o - v_e)^2 / (2 * (a_o - a_e))$

lemma *fst-leq-snd-safe-distance*:
assumes $a_e < a_o$
shows $\text{fst-safe-distance} \leq \text{snd-safe-distance}$
<proof>

lemma *snd-safe-distance-iff-nonneg-D2*:
assumes $a_e < a_o$
shows $s_o - s_e \leq \text{snd-safe-distance} \iff 0 \leq D2$
<proof>

lemma *t-stop-diff-neg-means-leq-D2*:
assumes $s_o \leq \text{ego.s-stop}$ $\text{ego.s-stop} < \text{other.s-stop}$ $a_e < a_o$ $0 \leq D2$
shows $v_e - a_e / a_o * v_o < 0 \iff \text{sqrt } D2 > v_e - a_e / a_o * v_o$
<proof>

theorem *cond-3'*:
assumes $s_o \leq \text{ego.s-stop}$ $\text{ego.s-stop} < \text{other.s-stop}$
shows $\text{collision } \{0..\} \iff (a_o > a_e \wedge v_o < v_e \wedge s_o - s_e \leq \text{snd-safe-distance} \wedge v_e - a_e / a_o * v_o < 0)$
<proof>

definition *d* :: *real* \Rightarrow *real* **where**
 $d \ t = ($
 if $t \leq 0$ *then* $s_o - s_e$
 else if $t \leq \text{ego.t-stop} \wedge t \leq \text{other.t-stop}$ *then* $\text{ego-other.p } t$
 else if $\text{ego.t-stop} \leq t \wedge t \leq \text{other.t-stop}$ *then* $\text{other.p } t - \text{ego.s-stop}$
 else if $\text{other.t-stop} \leq t \wedge t \leq \text{ego.t-stop}$ *then* $\text{other.s-stop} - \text{ego.p } t$
 else $\text{other.s-stop} - \text{ego.s-stop}$
 $)$

lemma *d-diff*: $d \ t = \text{other.s } t - \text{ego.s } t$
<proof>

lemma *collision-d*: $\text{collision } S \iff (\exists t \in S. d \ t = 0)$

<proof>

lemma *collision-restrict*: $\text{collision } \{0..\} \longleftrightarrow \text{collision } \{0..max \text{ ego.t-stop other.t-stop}\}$

<proof>

lemma *collision-union*: $\text{collision } (A \cup B) \longleftrightarrow \text{collision } A \vee \text{collision } B$

<proof>

lemma *symbolic-checker*:

$\text{collision } \{0..\} \longleftrightarrow$
 $(\text{quadroot-in } 0 \text{ (min ego.t-stop other.t-stop) } (1/2 * (a_o - a_e)) (v_o - v_e) (s_o - s_e)) \vee$
 $(\text{quadroot-in ego.t-stop other.t-stop } (1/2 * a_o) v_o (s_o - \text{ego.s-stop})) \vee$
 $(\text{quadroot-in other.t-stop ego.t-stop } (1/2 * a_e) v_e (s_e - \text{other.s-stop}))$
(is - $\longleftrightarrow ?q1 \vee ?q2 \vee ?q3)$
<proof>

end

1.5 Checker Design

definition *rel-dist-to-stop* :: $real \Rightarrow real \Rightarrow real$ **where**

$\text{rel-dist-to-stop } v \ a \equiv - \ v^2 / (2 * a)$

context includes *floatarith-syntax* **begin**

definition *rel-dist-to-stop-expr* :: $nat \Rightarrow nat \Rightarrow \text{floatarith}$ **where**

$\text{rel-dist-to-stop-expr } v \ a = \text{Mult } (\text{Minus } (\text{Power } (\text{Var } v) \ 2)) \ (\text{Inverse } (\text{Mult } (\text{Num } 2) \ (\text{Var } a)))$

definition *rel-dist-to-stop'* :: $nat \Rightarrow \text{float interval option} \Rightarrow \text{float interval option}$
 $\Rightarrow \text{float interval option}$ **where**

$\text{rel-dist-to-stop}' \ p \ v \ a = \text{approx } p \ (\text{rel-dist-to-stop-expr } 0 \ 1) \ [v, a]$

lemma *rel-dist-to-stop'*: $\text{interpret-floatarith } (\text{rel-dist-to-stop-expr } 0 \ 1) \ [v, a] = \text{rel-dist-to-stop } v \ a$

<proof>

definition *first-safe-dist* :: $real \Rightarrow real \Rightarrow real$ **where**

$\text{first-safe-dist } v_e \ a_e \equiv \text{rel-dist-to-stop } v_e \ a_e$

definition *second-safe-dist* :: $real \Rightarrow real \Rightarrow real \Rightarrow real \Rightarrow real$ **where**

$\text{second-safe-dist } v_e \ a_e \ v_o \ a_o \equiv \text{rel-dist-to-stop } v_e \ a_e - \text{rel-dist-to-stop } v_o \ a_o$

definition *second-safe-dist-expr* :: $nat \Rightarrow nat \Rightarrow nat \Rightarrow nat \Rightarrow \text{floatarith}$ **where**

$\text{second-safe-dist-expr } v_e \ a_e \ v_o \ a_o = \text{Add } (\text{rel-dist-to-stop-expr } v_e \ a_e) \ (\text{Minus } (\text{rel-dist-to-stop-expr } v_o \ a_o))$

definition *second-safe-dist'* :: $nat \Rightarrow \text{float interval option} \Rightarrow \text{float interval option}$

\Rightarrow float interval option \Rightarrow float interval option \Rightarrow float interval option **where**
second-safe-dist' $p\ v_e\ a_e\ v_o\ a_o = \text{approx } p$ (*second-safe-dist-expr* 0 1 2 3) [$v_e, a_e,$
 v_o, a_o]

lemma *second-safe-dist'*:

interpret-floatarith (*second-safe-dist-expr* 0 1 2 3) [v, a, v', a'] = *second-safe-dist*
 $v\ a\ v'\ a'$
 ⟨proof⟩

definition *t-stop* :: real \Rightarrow real \Rightarrow real **where**

t-stop $v\ a \equiv -\ v / a$

definition *t-stop-expr* :: nat \Rightarrow nat \Rightarrow floatarith **where**

t-stop-expr $v\ a = \text{Minus } (\text{Mult } (\text{Var } v) (\text{Inverse } (\text{Var } a)))$

end

definition *s-stop* :: real \Rightarrow real \Rightarrow real \Rightarrow real **where**

s-stop $s\ v\ a \equiv s + \text{rel-dist-to-stop } v\ a$

definition *discriminant* :: real \Rightarrow real \Rightarrow real \Rightarrow real \Rightarrow real \Rightarrow real \Rightarrow real **where**

discriminant $s_e\ v_e\ a_e\ s_o\ v_o\ a_o \equiv (v_o - v_e)^2 - 2 * (a_o - a_e) * (s_o - s_e)$

definition *suff-cond-safe-dist2* :: real \Rightarrow real \Rightarrow real \Rightarrow real \Rightarrow real \Rightarrow real \Rightarrow bool
where

suff-cond-safe-dist2 $s_e\ v_e\ a_e\ s_o\ v_o\ a_o \equiv$
 let $D2 = \text{discriminant } s_e\ v_e\ a_e\ s_o\ v_o\ a_o$
 in $\neg (a_e < a_o \wedge v_o < v_e \wedge 0 \leq D2 \wedge v_e - a_e / a_o * v_o < \text{sqrt } D2$
)

lemma *less-sqrt-iff*: $y \geq 0 \implies x < \text{sqrt } y \iff (x \geq 0 \implies x^2 < y)$

⟨proof⟩

lemma *suff-cond-safe-dist2-code*[code]:

suff-cond-safe-dist2 $s_e\ v_e\ a_e\ s_o\ v_o\ a_o =$
 (let $D2 = \text{discriminant } s_e\ v_e\ a_e\ s_o\ v_o\ a_o$ in
 ($a_e < a_o \implies v_o < v_e \implies 0 \leq D2 \implies (v_e - a_e / a_o * v_o \geq 0 \wedge (v_e - a_e /$
 $a_o * v_o)^2 \geq D2)))$
 ⟨proof⟩

There are two expressions for safe distance. The first safe distance *first-safe-dist* is always valid. Whenever the distance is bigger than *first-safe-dist*, it is guaranteed to be collision free. The second one is *second-safe-dist*. If the sufficient condition *suff-cond-safe-dist2* is satisfied and the distance is bigger than *second-safe-dist*, it is guaranteed to be collision free.

definition *check-precond* :: real \Rightarrow real \Rightarrow real \Rightarrow real \Rightarrow real \Rightarrow real \Rightarrow bool
where

check-precond $s_e\ v_e\ a_e\ s_o\ v_o\ a_o \iff s_o > s_e \wedge 0 \leq v_e \wedge 0 \leq v_o \wedge a_e < 0 \wedge a_o < 0$

lemma *check-precond-safe-distance*:

check-precond $s_e v_e a_e s_o v_o a_o = \text{safe-distance } a_e v_e s_e a_o v_o s_o$
 ⟨proof⟩

1.5.1 Prescriptive Checker

definition *checker* :: *real* \Rightarrow *real* \Rightarrow *real* \Rightarrow *real* \Rightarrow *real* \Rightarrow *real* \Rightarrow *bool* **where**

checker $s_e v_e a_e s_o v_o a_o \equiv$
 let *distance* = $s_o - s_e$;
 precond = *check-precond* $s_e v_e a_e s_o v_o a_o$;
 safe-dist1 = *first-safe-dist* $v_e a_e$;
 safe-dist2 = *second-safe-dist* $v_e a_e v_o a_o$;
 cond2 = *suff-cond-safe-dist2* $s_e v_e a_e s_o v_o a_o$
 in *precond* \wedge (*safe-dist1* $<$ *distance* \vee (*safe-dist2* $<$ *distance* \wedge *distance* \leq
safe-dist1 \wedge *cond2*))

lemma *aux-logic*:

assumes $a \implies b$
assumes $b \implies a \longleftrightarrow c$
shows $a \longleftrightarrow b \wedge c$
 ⟨proof⟩

theorem *soundness-correctness*:

checker $s_e v_e a_e s_o v_o a_o \longleftrightarrow \text{check-precond } s_e v_e a_e s_o v_o a_o \wedge \text{safe-distance.no-collision}$
 $a_e v_e s_e a_o v_o s_o \{0..\}$
 ⟨proof⟩

definition *checker2* :: *real* \Rightarrow *real* \Rightarrow *real* \Rightarrow *real* \Rightarrow *real* \Rightarrow *real* \Rightarrow *bool* **where**

checker2 $s_e v_e a_e s_o v_o a_o \equiv$
 let *distance* = $s_o - s_e$;
 precond = *check-precond* $s_e v_e a_e s_o v_o a_o$;
 safe-dist1 = *first-safe-dist* $v_e a_e$;
 safe-dist2 = *second-safe-dist* $v_e a_e v_o a_o$;
 safe-dist3 = $- \text{rel-dist-to-stop } (v_o - v_e) (a_o - a_e)$
 in
 if \neg *precond* then *False*
 else if *distance* $>$ *safe-dist1* then *True*
 else if $a_o > a_e \wedge v_o < v_e \wedge v_e - a_e / a_o * v_o < 0$ then *distance* $>$ *safe-dist3*
 else *distance* $>$ *safe-dist2*

theorem *checker-eq-checker2*: *checker* $s_e v_e a_e s_o v_o a_o \longleftrightarrow \text{checker2 } s_e v_e a_e s_o$

$v_o a_o$
 ⟨proof⟩

definition *checker3* :: *real* \Rightarrow *real* \Rightarrow *real* \Rightarrow *real* \Rightarrow *real* \Rightarrow *real* \Rightarrow *bool* **where**

checker3 $s_e v_e a_e s_o v_o a_o \equiv$
 let *distance* = $s_o - s_e$;
 precond = *check-precond* $s_e v_e a_e s_o v_o a_o$;
 s-stop-e = $s_e + \text{rel-dist-to-stop } v_e a_e$;

$$\begin{aligned}
& s\text{-stop-}o = s_o + \text{rel-dist-to-stop } v_o \ a_o \\
& \text{in } \text{precond} \wedge (s\text{-stop-}e < s_o \\
& \quad \vee (s_o \leq s\text{-stop-}e \wedge s\text{-stop-}e < s\text{-stop-}o \wedge \\
& \quad \quad (\neg(a_o > a_e \wedge v_o < v_e \wedge v_e - a_e / a_o * v_o < 0 \wedge \text{distance} * (a_o - \\
& a_e) \leq (v_o - v_e)^2 / 2))))
\end{aligned}$$

theorem *checker2-eq-checker3*:

$$\text{checker2 } s_e \ v_e \ a_e \ s_o \ v_o \ a_o \longleftrightarrow \text{checker3 } s_e \ v_e \ a_e \ s_o \ v_o \ a_o$$

<proof>

1.5.2 Approximate Checker

lemma *checker2-def'*: $\text{checker2 } a \ b \ c \ d \ e \ f = ($

$$\begin{aligned}
& \text{let } \text{distance} = d - a; \\
& \text{precond} = \text{check-precond } a \ b \ c \ d \ e \ f; \\
& \text{safe-dist1} = \text{first-safe-dist } b \ c; \\
& \text{safe-dist2} = \text{second-safe-dist } b \ c \ e \ f; \\
& C = c < f \wedge e < b \wedge b * f > c * e; \\
& P1 = (e - b)^2 < 2 * \text{distance} * (f - c); \\
& P2 = -b^2 / c + e^2 / f < 2 * \text{distance} \\
& \text{in } \text{precond} \wedge (\text{safe-dist1} < \text{distance} \vee \\
& \quad \text{safe-dist1} \geq \text{distance} \wedge (C \wedge P1 \vee \neg C \wedge P2)))
\end{aligned}$$

<proof>

lemma *power2-less-sqrt-iff*: $(x::\text{real})^2 < y \longleftrightarrow (y \geq 0 \wedge \text{abs } x < \text{sqrt } y)$

<proof>

schematic-goal *checker-form*: $\text{interpret-form } ?x \ ?y \implies \text{checker } s_e \ v_e \ a_e \ s_o \ v_o \ a_o$

<proof>

context includes *floatarith-syntax* **begin**

definition *checker'* $p \ s_e \ v_e \ a_e \ s_o \ v_o \ a_o = \text{approx-form } p$

$$\begin{aligned}
& (\text{Conj } (\text{Conj } (\text{Less } (\text{Var } (\text{Suc } (\text{Suc } 0)))) (\text{Var } (\text{Suc } (\text{Suc } (\text{Suc } 0)))))) \\
& \quad (\text{Conj } (\text{LessEqual } (\text{Var } (\text{Suc } (\text{Suc } (\text{Suc } (\text{Suc } (\text{Suc } (\text{Suc } (\text{Suc } 0)))))))))) \\
& (\text{Var } (\text{Suc } (\text{Suc } (\text{Suc } (\text{Suc } 0)))))) \\
& \quad (\text{Conj } (\text{LessEqual } (\text{Var } (\text{Suc } (\text{Suc } (\text{Suc } (\text{Suc } (\text{Suc } (\text{Suc } (\text{Suc } 0)))))))))) \\
& (\text{Var } (\text{Suc } (\text{Suc } (\text{Suc } (\text{Suc } (\text{Suc } (\text{Suc } (\text{Suc } 0)))))))) \\
& \quad (\text{Conj } (\text{Less } (\text{Var } 0) (\text{Var } (\text{Suc } (\text{Suc } (\text{Suc } (\text{Suc } (\text{Suc } (\text{Suc } (\text{Suc } 0)))))))))) \\
& (\text{Less } (\text{Var } (\text{Suc } 0) (\text{Var } (\text{Suc } (\text{Suc } (\text{Suc } (\text{Suc } (\text{Suc } (\text{Suc } (\text{Suc } 0)))))))))) \\
& \quad (\text{Disj } (\text{Less } (\text{Add } (\text{Var } (\text{Suc } (\text{Suc } 0)))) \\
& \quad \quad (\text{Mult } (\text{Minus } (\text{Power } (\text{Var } (\text{Suc } (\text{Suc } (\text{Suc } (\text{Suc } (\text{Suc } 0)))))) \\
& 2)) (\text{Inverse } (\text{Mult } (\text{Var } (\text{Suc } (\text{Suc } (\text{Suc } (\text{Suc } 0)))) (\text{Var } 0)))) \\
& \quad \quad (\text{Var } (\text{Suc } (\text{Suc } (\text{Suc } 0)))))) \\
& \quad (\text{Conj } (\text{LessEqual } (\text{Var } (\text{Suc } (\text{Suc } (\text{Suc } 0)))) \\
& \quad \quad (\text{Add } (\text{Var } (\text{Suc } (\text{Suc } 0)))) \\
& \quad \quad (\text{Mult } (\text{Minus } (\text{Power } (\text{Var } (\text{Suc } (\text{Suc } (\text{Suc } (\text{Suc } (\text{Suc } 0)))))) \\
& 2)) (\text{Inverse } (\text{Mult } (\text{Var } (\text{Suc } (\text{Suc } (\text{Suc } (\text{Suc } 0)))) (\text{Var } 0))))))
\end{aligned}$$

(Conj (Less (Add (Var (Suc (Suc 0)))
 (Mult (Minus (Power (Var (Suc (Suc (Suc (Suc (Suc
 0)))))) 2)) (Inverse (Mult (Var (Suc (Suc (Suc (Suc 0)))) (Var 0))))))
 (Add (Var (Suc (Suc (Suc 0))))
 (Mult (Minus (Power (Var (Suc (Suc (Suc (Suc (Suc (Suc
 0)))))) 2))
 (Inverse (Mult (Var (Suc (Suc (Suc (Suc 0)))) (Var (Suc
 0))))))
 (Disj (LessEqual (Var (Suc 0)) (Var 0))
 (Disj (LessEqual (Var (Suc (Suc (Suc (Suc (Suc 0)))) (Var
 (Suc (Suc (Suc (Suc (Suc (Suc 0)))))))))) (Var
 (Suc (Suc (Suc (Suc (Suc (Suc (Suc (Suc 0))))))))))
 (Disj (LessEqual (Var (Suc (Suc (Suc (Suc (Suc (Suc (Suc
 0))))))))))
 (Add (Var (Suc (Suc (Suc (Suc (Suc 0))))))
 (Minus (Mult (Mult (Var 0) (Inverse (Var (Suc 0))))
 (Var (Suc (Suc (Suc (Suc (Suc (Suc 0))))))))))
 (Less (Mult (Power (Add (Var (Suc (Suc (Suc (Suc (Suc (Suc
 0)))))) (Minus (Var (Suc (Suc (Suc (Suc 0)))))) 2)
 (Inverse (Var (Suc (Suc (Suc (Suc 0))))))
 (Mult (Add (Var (Suc (Suc (Suc 0)))) (Minus (Var (Suc
 (Suc 0)))) (Add (Var (Suc 0)) (Minus (Var 0))))))))))
 ([a_e , a_o , s_e , s_o , Interval' (Float 2 0) (Float 2 0), v_e , v_o , Interval' (Float 0 1)
 (Float 0 1)]) (replicate 8 0)

lemma less-Suc-iff-disj: $i < \text{Suc } x \longleftrightarrow i = x \vee i < x$
 ⟨proof⟩

lemma checker'-soundness-correctness:

assumes $a \in \{\text{real-of-float } al \dots \text{real-of-float } au\}$
assumes $b \in \{\text{real-of-float } bl \dots \text{real-of-float } bu\}$
assumes $c \in \{\text{real-of-float } cl \dots \text{real-of-float } cu\}$
assumes $d \in \{\text{real-of-float } dl \dots \text{real-of-float } du\}$
assumes $e \in \{\text{real-of-float } el \dots \text{real-of-float } eu\}$
assumes $f \in \{\text{real-of-float } fl \dots \text{real-of-float } fu\}$
assumes *chk*: checker' p (Interval' al au) (Interval' bl bu) (Interval' cl cu)
 (Interval' dl du) (Interval' el eu) (Interval' fl fu)
shows checker a b c d e f
 ⟨proof⟩

lemma approximate-soundness-correctness:

assumes $a \in \{\text{real-of-float } al \dots \text{real-of-float } au\}$
assumes $b \in \{\text{real-of-float } bl \dots \text{real-of-float } bu\}$
assumes $c \in \{\text{real-of-float } cl \dots \text{real-of-float } cu\}$
assumes $d \in \{\text{real-of-float } dl \dots \text{real-of-float } du\}$
assumes $e \in \{\text{real-of-float } el \dots \text{real-of-float } eu\}$
assumes $f \in \{\text{real-of-float } fl \dots \text{real-of-float } fu\}$
assumes *chk*: checker' p (Interval' al au) (Interval' bl bu) (Interval' cl cu)
 (Interval' dl du) (Interval' el eu) (Interval' fl fu)
shows checker'-precond: check-precond a b c d e f

and *checker'-no-collision: safe-distance.no-collision c b a f e d {0..}*
 ⟨proof⟩

1.5.3 Symbolic Checker

definition *symbolic-checker* :: *real* ⇒ *real* ⇒ *real* ⇒ *real* ⇒ *real* ⇒ *real* ⇒ *bool*
where

symbolic-checker *s_e* *v_e* *a_e* *s_o* *v_o* *a_o* ≡
 let *e-stop* = - *v_e* / *a_e*;
 o-stop = - *v_o* / *a_o*
 in *check-precond* *s_e* *v_e* *a_e* *s_o* *v_o* *a_o* ∧
 (¬*quadroot-in* 0 (min *e-stop* *o-stop*) (1/2 * (*a_o* - *a_e*)) (*v_o* - *v_e*) (*s_o* - *s_e*) ∧
 ¬*quadroot-in* *e-stop* *o-stop* (1/2 * *a_o*) *v_o* (*s_o* - *movement.p* *a_e* *v_e* *s_e* *e-stop*)
 ∧
 ¬*quadroot-in* *o-stop* *e-stop* (1/2 * *a_e*) *v_e* (*s_e* - *movement.p* *a_o* *v_o* *s_o* *o-stop*))

theorem *symbolic-soundness-correctness*:

symbolic-checker *s_e* *v_e* *a_e* *s_o* *v_o* *a_o* ⇔ *check-precond* *s_e* *v_e* *a_e* *s_o* *v_o* *a_o* ∧
safe-distance.no-collision *a_e* *v_e* *s_e* *a_o* *v_o* *s_o* {0..}
 ⟨proof⟩
end

end

2 Safe Distance with Reaction Time

theory *Safe-Distance-Reaction*

imports

Safe-Distance

begin

2.1 Normal Safe Distance

locale *safe-distance-normal* = *safe-distance* +

fixes *δ* :: *real*

assumes *pos-react*: 0 < *δ*

begin

sublocale *ego2*: *braking-movement* *a_e* *v_e* (*ego.q* *δ*) ⟨proof⟩

lemma *ego2-s-init*: *ego2.s* 0 = *ego.q* *δ* ⟨proof⟩

definition *τ* :: *real* ⇒ *real* **where**

τ *t* = *t* - *δ*

definition *τ'* :: *real* ⇒ *real* **where**

τ' *t* = 1

lemma *τ-continuous*[*continuous-intros*]: *continuous-on* *T* *τ*

$\langle proof \rangle$

lemma *isCont- τ [continuous-intros]*: *isCont* τ x
 $\langle proof \rangle$

lemma *del-has-vector-derivative[derivative-intros]*: (τ *has-vector-derivative* τ' t)
(*at* t *within* u)
 $\langle proof \rangle$

lemma *del-has-real-derivative[derivative-intros]*: (τ *has-real-derivative* τ' t) (*at* t
within u)
 $\langle proof \rangle$

lemma *delay-image*: $\tau \text{ ' } \{\delta..\} = \{0..\}$
 $\langle proof \rangle$

lemma *s-delayed-has-real-derivative[derivative-intros]*:
assumes $\delta \leq t$
shows ((*ego2.s* \circ τ) *has-field-derivative* *ego2.s'* ($t - \delta$) * τ' t) (*at* t *within* $\{\delta..\}$)
 $\langle proof \rangle$

lemma *s-delayed-has-real-derivative' [derivative-intros]*:
assumes $\delta \leq t$
shows ((*ego2.s* \circ τ) *has-field-derivative* (*ego2.s'* \circ τ) t) (*at* t *within* $\{\delta..\}$)
 $\langle proof \rangle$

lemma *s-delayed-has-vector-derivative' [derivative-intros]*:
assumes $\delta \leq t$
shows ((*ego2.s* \circ τ) *has-vector-derivative* (*ego2.s'* \circ τ) t) (*at* t *within* $\{\delta..\}$)
 $\langle proof \rangle$

definition $u :: real \Rightarrow real$ **where**
 u $t =$ ($\text{if } t \leq 0 \text{ then } s_e$
 $\text{else if } t \leq \delta \text{ then } \text{ego.q } t$
 $\text{else } (\text{ego2.s} \circ \tau) t$)

lemma *init-u*: $t \leq 0 \implies u$ $t = s_e$ $\langle proof \rangle$

lemma *u-delta*: u $\delta = \text{ego2.s } 0$
 $\langle proof \rangle$

lemma *q-delta*: $\text{ego.q } \delta = \text{ego2.s } 0$ $\langle proof \rangle$

definition $u' :: real \Rightarrow real$ **where**
 u' $t =$ ($\text{if } t \leq \delta \text{ then } \text{ego.q}' t \text{ else } \text{ego2.s}' (t - \delta)$)

lemma *u'-delta*: u' $\delta = \text{ego2.s}' 0$
 $\langle proof \rangle$

lemma *q'-delta*: $\text{ego}.q' \delta = \text{ego2}.s' 0$ *<proof>*

lemma *u-has-real-derivative*[*derivative-intros*]:
 assumes *nonneg-t*: $t \geq 0$
 shows (*u has-real-derivative u' t*) (*at t within {0..}*)
 <proof>

definition *t-stop* :: *real* **where**
 $t\text{-stop} = \text{ego2}.t\text{-stop} + \delta$

lemma *t-stop-nonneg*: $0 \leq t\text{-stop}$
 <proof>

lemma *t-stop-pos*: $0 < t\text{-stop}$
 <proof>

lemma *t-stop-zero*:
 assumes *t-stop* $\leq x$
 assumes $x \leq \delta$
 shows $v_e = 0$
 <proof>

lemma *u'-stop-zero*: $u' t\text{-stop} = 0$
 <proof>

definition *u-max* :: *real* **where**
 $u\text{-max} = u (\text{ego2}.t\text{-stop} + \delta)$

lemma *u-max-eq*: $u\text{-max} = \text{ego}.q \delta - v_e^2 / a_e / 2$
 <proof>

lemma *u-mono*:
 assumes $x \leq y$ $y \leq t\text{-stop}$
 shows $u x \leq u y$
 <proof>

lemma *u-antimono*: $x \leq y \implies t\text{-stop} \leq x \implies u y \leq u x$
 <proof>

lemma *u-max*: $u x \leq u\text{-max}$
 <proof>

lemma *u-eq-u-stop*: *NO-MATCH* $t\text{-stop} x \implies x \geq t\text{-stop} \implies u x = u\text{-max}$
 <proof>

lemma *at-least-delta*:
 assumes $x \leq \delta$
 assumes $t\text{-stop} \leq x$
 shows $\text{ego}.q x = \text{ego2}.s (x - \delta)$

<proof>

lemma *continuous-on-u*[*continuous-intros*]: *continuous-on T u*
<proof>

lemma *isCont-u*[*continuous-intros*]: *isCont u x*
<proof>

definition *collision-react* :: *real set* \Rightarrow *bool* **where**
collision-react time-set $\equiv (\exists t \in \text{time-set. } u\ t = \text{other.s } t)$

abbreviation *no-collision-react* :: *real set* \Rightarrow *bool* **where**
no-collision-react time-set $\equiv \neg \text{collision-react time-set}$

lemma *no-collision-reactI*:
assumes $\bigwedge t. t \in S \implies u\ t \neq \text{other.s } t$
shows *no-collision-react S*
<proof>

lemma *no-collision-union*:
assumes *no-collision-react S*
assumes *no-collision-react T*
shows *no-collision-react (S \cup T)*
<proof>

lemma *collision-trim-subset*:
assumes *collision-react S*
assumes *no-collision-react T*
assumes $T \subseteq S$
shows *collision-react (S - T)*
<proof>

theorem *cond-1r* : $u\text{-max} < s_o \implies \text{no-collision-react } \{0..\}$
<proof>

definition *safe-distance-1r* :: *real* **where**
safe-distance-1r = $v_e * \delta - v_e^2 / a_e / 2$

lemma *sd-1r-eq*: $(s_o - s_e > \text{safe-distance-1r}) = (u\text{-max} < s_o)$
<proof>

lemma *sd-1r-correct*:
assumes $s_o - s_e > \text{safe-distance-1r}$
shows *no-collision-react {0..}*
<proof>

lemma *u-other-strict-ivt*:
assumes $u > \text{other.s } t$
shows *collision-react {0..<t}*

<proof>

lemma *collision-react-subset*: $\text{collision-react } s \implies s \subseteq t \implies \text{collision-react } t$
<proof>

lemma *u-other-ivt*:
assumes $u \ t \geq \text{other.s } t$
shows $\text{collision-react } \{0 \ .. \ t\}$
<proof>

theorem *cond-2r*:
assumes $u\text{-max} \geq \text{other.s-stop}$
shows $\text{collision-react } \{0 \ .. \}$
<proof>

definition *ego-other2* :: $\text{real} \Rightarrow \text{real}$ **where**
 $\text{ego-other2 } t = \text{other.s } t - u \ t$

lemma *continuous-on-ego-other2*[*continuous-intros*]: $\text{continuous-on } T \ \text{ego-other2}$
<proof>

lemma *isCont-ego-other2*[*continuous-intros*]: $\text{isCont } \text{ego-other2 } x$
<proof>

definition *ego-other2'* :: $\text{real} \Rightarrow \text{real}$ **where**
 $\text{ego-other2}' \ t = \text{other.s}' \ t - u' \ t$

lemma *ego-other2-has-real-derivative*[*derivative-intros*]:
assumes $0 \leq t$
shows (ego-other2 has-real-derivative $\text{ego-other2}' \ t$) (at t within $\{0..\}$)
<proof>

theorem *cond-3r-1*:
assumes $u \ \delta \geq \text{other.s } \delta$
shows $\text{collision-react } \{0 \ .. \ \delta\}$
<proof>

definition *distance0* :: real **where**
 $\text{distance0} = v_e * \delta - v_o * \delta - a_o * \delta^2 / 2$

definition *distance0-2* :: real **where**
 $\text{distance0-2} = v_e * \delta + 1 / 2 * v_o^2 / a_o$

theorem *cond-3r-1'*:
assumes $s_o - s_e \leq \text{distance0}$
assumes $\delta \leq \text{other.t-stop}$
shows $\text{collision-react } \{0 \ .. \ \delta\}$
<proof>

theorem *distance0-2-eq*:
assumes $\delta > \text{other.t-stop}$
shows $(u \delta < \text{other.s } \delta) = (s_o - s_e > \text{distance0-2})$
 $\langle \text{proof} \rangle$

theorem *cond-3r-1'-2*:
assumes $s_o - s_e \leq \text{distance0-2}$
assumes $\delta > \text{other.t-stop}$
shows *collision-react* $\{0 .. \delta\}$
 $\langle \text{proof} \rangle$

definition *safe-distance-3r* :: real **where**
 $\text{safe-distance-3r} = v_e * \delta - v_e^2 / 2 / a_e - v_o * \delta - 1/2 * a_o * \delta^2$

lemma *distance0-at-most-sd3r*:
 $\text{distance0} \leq \text{safe-distance-3r}$
 $\langle \text{proof} \rangle$

definition *safe-distance-4r* :: real **where**
 $\text{safe-distance-4r} = (v_o + a_o * \delta - v_e)^2 / 2 / (a_o - a_e) - v_o * \delta - 1/2 * a_o * \delta^2 + v_e * \delta$

lemma *distance0-at-most-sd4r*:
assumes $a_o > a_e$
shows $\text{distance0} \leq \text{safe-distance-4r}$
 $\langle \text{proof} \rangle$

definition *safe-distance-2r* :: real **where**
 $\text{safe-distance-2r} = v_e * \delta - v_e^2 / 2 / a_e + v_o^2 / 2 / a_o$

lemma *vo-start-geq-ve*:
assumes $\delta \leq \text{other.t-stop}$
assumes $\text{other.s}' \delta \geq v_e$
shows $u \delta < \text{other.s } \delta$
 $\langle \text{proof} \rangle$

theorem *so-star-stop-leq-se-stop*:
assumes $\delta \leq \text{other.t-stop}$
assumes $\text{other.s}' \delta < v_e$
assumes $\neg (a_o > a_e \wedge \text{other.s}' \delta < v_e \wedge v_e - a_e / a_o * \text{other.s}' \delta < 0)$
shows $0 \leq -v_e^2 / a_e / 2 + (v_o + a_o * \delta)^2 / a_o / 2$
 $\langle \text{proof} \rangle$

theorem *distance0-at-most-distance2r*:
assumes $\delta \leq \text{other.t-stop}$
assumes $\text{other.s}' \delta < v_e$
assumes $\neg (a_o > a_e \wedge \text{other.s}' \delta < v_e \wedge v_e - a_e / a_o * \text{other.s}' \delta < 0)$
shows $\text{distance0} \leq \text{safe-distance-2r}$
 $\langle \text{proof} \rangle$

theorem *dist0-sd2r-1*:
assumes $\delta \leq \text{other.t-stop}$
assumes $\neg (a_o > a_e \wedge \text{other.s}' \delta < v_e \wedge v_e - a_e / a_o * \text{other.s}' \delta < 0)$
assumes $s_o - s_e > \text{safe-distance-2r}$
shows $s_o - s_e > \text{distance0}$
<proof>

theorem *sd2r-eq*:
assumes $\delta > \text{other.t-stop}$
shows $(u\text{-max} < \text{other.s } \delta) = (s_o - s_e > \text{safe-distance-2r})$
<proof>

theorem *dist0-sd2r-2*:
assumes $\delta > -v_o / a_o$
assumes $s_o - s_e > \text{safe-distance-2r}$
shows $s_o - s_e > \text{distance0-2}$
<proof>
end

2.2 Safe Distance Delta

locale *safe-distance-no-collision-delta* = *safe-distance-normal* +
assumes *no-collision-delta*: $u \delta < \text{other.s } \delta$
begin

sublocale *delayed-safe-distance*: *safe-distance* $a_e v_e \text{ ego.q } \delta a_o \text{ other.s}' \delta \text{ other.s } \delta$
<proof>

lemma *no-collision-react-initially-strict*:
assumes $s_o \leq u\text{-max}$
assumes $u\text{-max} < \text{other.s-stop}$
shows *no-collision-react* $\{0 <..< \delta\}$
<proof>

lemma *no-collision-react-initially*:
assumes $s_o \leq u\text{-max}$
assumes $u\text{-max} < \text{other.s-stop}$
shows *no-collision-react* $\{0 .. \delta\}$
<proof>

lemma *collision-after-delta*:
assumes $s_o \leq u\text{-max}$
assumes $u\text{-max} < \text{other.s-stop}$
shows *collision-react* $\{0 ..\} \longleftrightarrow \text{collision-react } \{\delta..\}$
<proof>

lemma *collision-react-strict*:
assumes $s_o \leq u\text{-max}$

assumes $u\text{-max} < \text{other.s-stop}$
shows $\text{collision-react } \{\delta \dots\} \longleftrightarrow \text{collision-react } \{\delta < \dots\}$
 $\langle \text{proof} \rangle$

lemma *delayed-other-s-stop-eq*: $\text{delayed-safe-distance.other.s-stop} = \text{other.s-stop}$
 $\langle \text{proof} \rangle$

lemma *delayed-cond3'*:
assumes $\text{other.s } \delta \leq u\text{-max}$
assumes $u\text{-max} < \text{other.s-stop}$
shows $\text{delayed-safe-distance.collision } \{0 \dots\} \longleftrightarrow$
 $(a_o > a_e \wedge \text{other.s}' \delta < v_e \wedge \text{other.s } \delta - \text{ego.q } \delta \leq \text{delayed-safe-distance.snd-safe-distance}$
 $\wedge v_e - a_e / a_o * \text{other.s}' \delta < 0)$
 $\langle \text{proof} \rangle$

lemma *delayed-other-t-stop-eq*:
assumes $\delta \leq \text{other.t-stop}$
shows $\text{delayed-safe-distance.other.t-stop} + \delta = \text{other.t-stop}$
 $\langle \text{proof} \rangle$

lemma *delayed-other-s-eq*:
assumes $0 \leq t$
shows $\text{delayed-safe-distance.other.s } t = \text{other.s } (t + \delta)$
 $\langle \text{proof} \rangle$

lemma *translate-collision-range*:
assumes $s_o \leq u\text{-max}$
assumes $u\text{-max} < \text{other.s-stop}$
shows $\text{delayed-safe-distance.collision } \{0 \dots\} \longleftrightarrow \text{collision-react } \{\delta \dots\}$
 $\langle \text{proof} \rangle$

theorem *cond-3r-2*:
assumes $s_o \leq u\text{-max}$
assumes $u\text{-max} < \text{other.s-stop}$
assumes $\text{other.s } \delta \leq u\text{-max}$
shows $\text{collision-react } \{0 \dots\} \longleftrightarrow$
 $(a_o > a_e \wedge \text{other.s}' \delta < v_e \wedge \text{other.s } \delta - \text{ego.q } \delta \leq \text{delayed-safe-distance.snd-safe-distance}$
 $\wedge v_e - a_e / a_o * \text{other.s}' \delta < 0)$
 $\langle \text{proof} \rangle$

lemma *sd-2r-correct-for-3r-2*:
assumes $s_o - s_e > \text{safe-distance-2r}$
assumes $\text{other.s } \delta \leq u\text{-max}$
assumes $\neg (a_o > a_e \wedge \text{other.s}' \delta < v_e \wedge v_e - a_e / a_o * \text{other.s}' \delta < 0)$
shows $\text{no-collision-react } \{0 \dots\}$
 $\langle \text{proof} \rangle$

lemma *sd2-at-most-sd4*:
assumes $a_o > a_e$

shows $\text{safe-distance-}2r \leq \text{safe-distance-}4r$
(proof)

lemma $\text{sd-}4r\text{-correct}$:
assumes $s_o - s_e > \text{safe-distance-}4r$
assumes $\text{other.s } \delta \leq u\text{-max}$
assumes $\delta \leq \text{other.t-stop}$
assumes $a_o > a_e$
shows $\text{no-collision-react } \{0..\}$
(proof)

Irrelevant, this Safe Distance is unreachable in the checker.

definition $\text{safe-distance-}5r :: \text{real where}$
 $\text{safe-distance-}5r = v_e^2 / 2 / (a_o - a_e) + v_o^2 / 2 / a_o + v_e * \delta$

lemma $\text{sd-}5r\text{-correct}$:
assumes $s_o - s_e > \text{safe-distance-}5r$
assumes $u\text{-max} < \text{other.s-stop}$
assumes $\text{other.s } \delta \leq u\text{-max}$
assumes $\delta > \text{other.t-stop}$
shows $\text{no-collision-react } \{0..\}$
(proof)

lemma $\text{translate-no-collision-range}$:
 $\text{delayed-safe-distance.no-collision } \{0 ..\} \longleftrightarrow \text{no-collision-react } \{\delta ..\}$
(proof)

lemma delayed-cond1 :
assumes $\text{other.s } \delta > u\text{-max}$
shows $\text{delayed-safe-distance.no-collision } \{0 ..\}$
(proof)

theorem $\text{cond-}3r\text{-}3$:
assumes $s_o \leq u\text{-max}$
assumes $u\text{-max} < \text{other.s-stop}$
assumes $\text{other.s } \delta > u\text{-max}$
shows $\text{no-collision-react } \{0 ..\}$
(proof)

lemma $\text{sd-}2r\text{-correct-for-}3r\text{-}3$:
assumes $s_o - s_e > \text{safe-distance-}2r$
assumes $\text{other.s } \delta > u\text{-max}$
shows $\text{no-collision-react } \{0..\}$
(proof)

lemma $\text{sd-}3r\text{-correct}$:
assumes $s_o - s_e > \text{safe-distance-}3r$
assumes $\delta \leq \text{other.t-stop}$
shows $\text{no-collision-react } \{0 ..\}$

<proof>

lemma *sd-2-at-least-sd-3*:

assumes $\delta \leq \text{other.t-stop}$

shows *safe-distance-3r* \geq *safe-distance-2r*

<proof>

end

2.3 Checker Design

We define two checkers for different cases:

- one checker for the case that $\delta \leq \text{other.t-stop}$ ($\text{other.t-stop} = -v_o / a_o$)
- a second checker for the case that $\delta > \text{other.t-stop}$

definition *check-precond-r1* :: $\text{real} \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow \text{bool}$ **where**

check-precond-r1 $s_e v_e a_e s_o v_o a_o \delta \longleftrightarrow s_o > s_e \wedge 0 \leq v_e \wedge 0 \leq v_o \wedge a_e < 0 \wedge a_o < 0 \wedge 0 < \delta \wedge \delta \leq -v_o / a_o$

definition *safe-distance0* :: $\text{real} \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow \text{real}$ **where**

safe-distance0 $v_e a_o v_o \delta = v_e * \delta - v_o * \delta - a_o * \delta^2 / 2$

definition *safe-distance-1r* :: $\text{real} \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow \text{real}$ **where**

safe-distance-1r $a_e v_e \delta = v_e * \delta - v_e^2 / a_e / 2$

definition *safe-distance-2r* :: $\text{real} \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow \text{real}$ **where**

safe-distance-2r $a_e v_e a_o v_o \delta = v_e * \delta - v_e^2 / 2 / a_e + v_o^2 / 2 / a_o$

definition *safe-distance-4r* :: $\text{real} \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow \text{real}$ **where**

safe-distance-4r $a_e v_e a_o v_o \delta = (v_o + a_o * \delta - v_e)^2 / 2 / (a_o - a_e) - v_o * \delta - 1 / 2 * a_o * \delta^2 + v_e * \delta$

definition *safe-distance-3r* :: $\text{real} \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow \text{real}$ **where**

safe-distance-3r $a_e v_e a_o v_o \delta = v_e * \delta - v_e^2 / 2 / a_e - v_o * \delta - 1 / 2 * a_o * \delta^2$

definition *checker-r1* :: $\text{real} \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow \text{bool}$ **where**

checker-r1 $s_e v_e a_e s_o v_o a_o \delta \equiv$

let distance = $s_o - s_e$;

precond = *check-precond-r1* $s_e v_e a_e s_o v_o a_o \delta$;

vo-star = $v_o + a_o * \delta$;

t-stop-o-star = $-vo-star / a_o$;

t-stop-e = $-v_e / a_e$;

safe-dist0 = *safe-distance-1r* $a_e v_e \delta$;

safe-dist1 = *safe-distance-2r* $a_e v_e a_o v_o \delta$;

$safe-dist2 = safe-distance-4r a_e v_e a_o v_o \delta;$
 $safe-dist3 = safe-distance-3r a_e v_e a_o v_o \delta$
in
if $\neg precondition$ then *False*
else if $distance > safe-dist0 \vee distance > safe-dist3$ then *True*
else if $(a_o > a_e \wedge vo-star < v_e \wedge t-stop-e < t-stop-o-star)$ then $distance >$
 $safe-dist2$
else $distance > safe-dist1$

theorem *checker-r1-correctness:*

$(checker-r1 s_e v_e a_e s_o v_o a_o \delta \longleftrightarrow check-precond-r1 s_e v_e a_e s_o v_o a_o \delta \wedge$
 $safe-distance-normal.no-collision-react a_e v_e s_e a_o v_o s_o \delta \{0..\})$
<proof>

definition *check-precond-r2* :: $real \Rightarrow real \Rightarrow real \Rightarrow real \Rightarrow real \Rightarrow real \Rightarrow real$
 $\Rightarrow bool$ **where**

$check-precond-r2 s_e v_e a_e s_o v_o a_o \delta \longleftrightarrow s_o > s_e \wedge 0 \leq v_e \wedge 0 \leq v_o \wedge a_e < 0$
 $\wedge a_o < 0 \wedge 0 < \delta \wedge \delta > -v_o / a_o$

definition *safe-distance0-2* :: $real \Rightarrow real \Rightarrow real \Rightarrow real \Rightarrow real$ **where**

$safe-distance0-2 v_e a_o v_o \delta = v_e * \delta + 1 / 2 * v_o^2 / a_o$

definition *checker-r2* :: $real \Rightarrow real \Rightarrow real \Rightarrow real \Rightarrow real \Rightarrow real \Rightarrow real \Rightarrow bool$
where

$checker-r2 s_e v_e a_e s_o v_o a_o \delta \equiv$
let $distance = s_o - s_e;$
 $precond = check-precond-r2 s_e v_e a_e s_o v_o a_o \delta;$
 $safe-dist0 = safe-distance-1r a_e v_e \delta;$
 $safe-dist1 = safe-distance-2r a_e v_e a_o v_o \delta$

in
if $\neg precondition$ then *False*
else if $distance > safe-dist0$ then *True*
else $distance > safe-dist1$

theorem *checker-r2-correctness:*

$(checker-r2 s_e v_e a_e s_o v_o a_o \delta \longleftrightarrow check-precond-r2 s_e v_e a_e s_o v_o a_o \delta \wedge$
 $safe-distance-normal.no-collision-react a_e v_e s_e a_o v_o s_o \delta \{0..\})$
<proof>

Combine the two checkers into one.

definition *check-precond-r* :: $real \Rightarrow real \Rightarrow real \Rightarrow real \Rightarrow real \Rightarrow real \Rightarrow real \Rightarrow$
 $bool$ **where**

$check-precond-r s_e v_e a_e s_o v_o a_o \delta \longleftrightarrow s_o > s_e \wedge 0 \leq v_e \wedge 0 \leq v_o \wedge a_e < 0 \wedge$
 $a_o < 0 \wedge 0 < \delta$

definition *checker-r* :: $real \Rightarrow real \Rightarrow real \Rightarrow real \Rightarrow real \Rightarrow real \Rightarrow real \Rightarrow bool$
where

$checker-r s_e v_e a_e s_o v_o a_o \delta \equiv$
let $distance = s_o - s_e;$

```

precond      = check-precond-r se ve ae so vo ao δ;
vo-star     = vo + ao * δ;
t-stop-o-star = -vo-star / ao;
t-stop-e    = -ve / ae;
t-stop-o    = -vo / ao;
safe-dist0  = safe-distance-1r ae ve δ;
safe-dist1  = safe-distance-2r ae ve ao vo δ;
safe-dist2  = safe-distance-4r ae ve ao vo δ;
safe-dist3  = safe-distance-3r ae ve ao vo δ
in
  if ¬ precond then False
  else if distance > safe-dist0 then True
  else if δ ≤ t-stop-o ∧ distance > safe-dist3 then True
  else if δ ≤ t-stop-o ∧ (ao > ae ∧ vo-star < ve ∧ t-stop-e < t-stop-o-star)
then distance > safe-dist2
  else distance > safe-dist1

```

theorem checker-eq-1:

```

checker-r se ve ae so vo ao δ ∧ δ ≤ -vo / ao ↔ checker-r1 se ve ae so vo ao
δ
⟨proof⟩

```

theorem checker-eq-2:

```

checker-r se ve ae so vo ao δ ∧ δ > -vo / ao ↔ checker-r2 se ve ae so vo ao δ
⟨proof⟩

```

theorem checker-r-correctness:

```

(checker-r se ve ae so vo ao δ ↔ check-precond-r se ve ae so vo ao δ ∧
safe-distance-normal.no-collision-react ae ve se ao vo so δ {0..})
⟨proof⟩

```

end

3 Evaluation

theory Evaluation

imports

Safe-Distance

HOL-Library.Float

begin

3.1 Code Generation Setup for Numeric Values

definition real-div-down :: nat ⇒ int ⇒ int ⇒ real **where**

real-div-down p i j = truncate-down (Suc p) (i / j)

definition real-div-up :: nat ⇒ int ⇒ int ⇒ real **where**

real-div-up p i j = truncate-up (Suc p) (i / j)

context includes *float.lifting* **begin**

lift-definition *float-div-down* :: *nat* \Rightarrow *int* \Rightarrow *int* \Rightarrow *float* **is** *real-div-down*
<proof>

lift-definition *float-div-up* :: *nat* \Rightarrow *int* \Rightarrow *int* \Rightarrow *float* **is** *real-div-up*
<proof>
end

lemma *compute-float-div-up*[*code*]: *float-div-up* *p i j* = - *float-div-down* *p (-i) j*
including *float.lifting*
<proof>

lemma *compute-float-div-down*[*code*]:
float-div-down *prec m1 m2* = *lapprox-rat* (*Suc prec*) *m1 m2*
including *float.lifting* *<proof>*

definition *real2-of-real* :: *nat* \Rightarrow *real* \Rightarrow (*real* * *real*) **where**
real2-of-real *p x* = (*truncate-down* (*Suc p*) *x*, *truncate-up* (*Suc p*) *x*)

context includes *float.lifting* **begin**

lift-definition *float2-of-real* :: *nat* \Rightarrow *real* \Rightarrow *float* \times *float* **is** *real2-of-real*
<proof>
end

definition *float2-opt-of-real* :: *nat* \Rightarrow *real* \Rightarrow *float interval option* **where**
float2-opt-of-real *prec x* = *Interval'* (*fst* (*float2-of-real* *prec x*)) (*snd* (*float2-of-real* *prec x*))

hide-const (**open**) *Fraction-Field.Fract*

lemma *real-of-rat-Fract*[*simp*]: *real-of-rat* (*Fract a b*) = *a / b*
<proof>

lemma [*code*]: *float2-of-real* *p* (*Ratreal r*) =
(*let* (*a*, *b*) = *quotient-of* *r* *in*
(*float-div-down* *p a b*, *float-div-up* *p a b*))
including *float.lifting*
<proof>

fun *real-of-dec* :: *integer* \times *integer* \Rightarrow *real* **where**

real-of-dec (*m*, *e*) =
real-of-int (*int-of-integer* *m*) *
(*if* *e* \geq 0 *then* 10^{\wedge} (*nat-of-integer* *e*) *else* *inverse* (10^{\wedge} (*nat* ($-$ (*int-of-integer* *e*))))))

lemma *real-of-dec* (*m*, *e*) = *int-of-integer* *m* * 10^{powr} (*int-of-integer* *e*)
<proof>

3.2 Data Evaluation

definition *trans6* **where**

$$\text{trans6 } c \text{ chk } \text{ se } \text{ ve } \text{ ae } \text{ so } \text{ vo } \text{ ao} = \\ \text{chk } (c \text{ se}) (c \text{ ve}) (c \text{ ae}) (c \text{ so}) (c \text{ vo}) (c \text{ ao})$$

definition *checker-dec* **where**

$$\text{checker-dec } \text{chk } p \text{ u} = \\ \text{trans6 } (\text{float2-opt-of-real } (\text{nat-of-integer } u) \text{ o real-of-dec}) (\text{chk } (\text{nat-of-integer } p))$$

definition *checker-interval* = *checker-dec checker'*

definition *checker-symbolic* = *trans6 real-of-dec symbolic-checker*

definition *checker-rational* = *trans6 real-of-dec checker*

lemmas[*code*] = *movement.p-def*

<ML>

The precision of the input data is roughly 12 and yields similar performance as Sturm

<ML>

Number of data points:

- data01: 1121215
- data02: 1341135
- data03: 1452656

<ML>

Precision: 12, Uncertainty: 7 digits

<ML>

end

References

- [1] A. Rizaldi, F. Immler, and M. Althoff. A formally verified checker of the safe distance traffic rules for autonomous vehicles. In S. Rayadurgam and O. Tkachuk, editors, *NASA Formal Methods*, pages 175–190, Cham, 2016. Springer International Publishing.