A Toolbox for Proving and Maintaining Hybrid Specifications

Michael S. Branicky,* Ekaterina Dolginova,** and Nancy Lynch***

Dept. of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139-4307 USA

Abstract. Formal verification in computer science often takes a worst-case view towards performance and uses induction to prove specification invariants. In control theory, robust control takes a worst-case view towards performance; nominal performance proofs often use derivative information to prove invariance of specification sets. In this note, we explore a toolbox for proving (positive) invariance of state-space sets with respect to the actions of dynamical systems. The focus is on dynamical systems given by differential equations, building up to hybrid systems.

1 Introduction

We are interested in the formal verification of safety and performance properties of hybrid systems [1, 8, 17].

In computer science, there is large formal verification literature (e.g., [2, 5, 9, 11, 12, 16, 19]) of discrete-dynamic systems, such as automata. Usually, the proofs that verify safety and performance involve the search for certain formulas which are (proven to be) invariant over the actions of the system. This same style can be translated, almost brute force, into dealing with hybrid systems by including both continuous and discrete system actions [13, 14, 15]. Conceptually, the solution appears clear. However, in chasing these solutions one can often get stuck by thinking of the "traces" or solutions of differential equations (intermixed with discrete steps, of course) as the entities one is to verify.

Herein, we take a step back from this conceptual viewpoint and begin to look for the *mechanics* and *mechanisms* of proof necessary when discrete and continuous actions interact. Specifically, we want to look at tools that enable one to prove the positive invariance of sets with respect to the action of hybrid dynamical systems. This is merely the abstraction of many problems of verification: Is the system performing within safety/performance specifications (specs.)?

* During preparation: Post-doc, Lab. for Information and Decision Systems. Currently: Asst. Prof., Dept. of Electrical Eng. and Applied Physics, Case Western Reserve University, 10900 Euclid Avenue, Cleveland, OH 44106-7221. `branicky@eeap.cwru.edu`
** Undergraduate student, Lab. for Computer Science. `katya@theory.lcs.mit.edu`
*** Prof., Lab. for Computer Science. `lynch@theory.lcs.mit.edu`

Thus, we consider $S \subset \mathbf{R}^n$ (more generally, $S' \subset \mathbf{R}^n \times \mathbf{R}$) to be our (time-varying) specification set for a system Σ. The evolution of Σ is given by a dynamical system $\phi : \mathbf{R}^n \times \mathbf{R} \to 2^{\mathbf{R}^n}$ satisfying the following two properties:

- Initial Condition: $\phi(x, 0) = x$ for all x.
- Transitivity:

$$\phi(x, t_1 + t_2) \in \{x_2 \mid x_1 \in \phi(x, t_1),\ x_2 \in \phi(x_1, t_2)\}$$

The specifications are enforced by requiring the system Σ be such that S is (positively) invariant with respect to the dynamics ϕ, that is

$$\phi(S, t) \subset S, \text{ for all } t \geq 0. \tag{1}$$

The rest of this paper can be seen as making the above notion precise and giving tools for (a) verifying Equation (1), and (b) designing ϕ so that Equation (1) is verified, even in the presence of uncertainty.

When the dynamical system ϕ is a finite automaton, $t \in \mathbf{Z}$, and a spec. is usually given by a state formula whose invariance is verified via an induction proof. This induction is on the discrete actions available to the automata.

We want to develop similar tools in the case the dynamical systems are hybrid (viz., combining automata and differential equations). As a first step, we collect and develop tools here useful in proving invariance when the dynamical systems are given by differential equations. We want the invariance proofs to again be based on induction from local steps: *we want to use derivative information, not solve differential equations.* The reason is that the global information of an ODE solution is generally impossible to find.

The paper is organized as follows. In the next section we review existing theory in our context. Then we develop more directly applicable tools in Section 3. In Sections 4 and 5 we solve a toy example of platoon merge safety. We first formalize the problem with the computer science model of hybrid I/O automata [13] and then prove safety using our system theoretic tools. In this way, we explicitly demonstrate how our tools mesh with both computer science and control theory flavors of reasoning about hybrid systems.

2 The Basics

2.1 On Being Invariant

A formal treatment of positive invariant sets for differential equations appears in Bhatia and Szegö's fine book [4, pp. 306–322]. Their Corollary 3.4.22 (p. 316) follows some preliminaries.

We discuss the autonomous system

$$\dot{x} = g(x), \tag{2}$$

where $x \in \mathbf{R}^n$ and with solution denoted by ϕ. Throughout, we assume that for Equation (2), g is continuous on an open set $U \subset \mathbf{R}^n$.

Definition 1. A *curve* γ is a continuous function mapping some interval D_γ into $\mathbf{R}^n$. We always let D_γ represent the domain of the curve γ. If $\mathcal{F}$ is a family of curves, we say $\gamma^* \in \mathcal{F}$ is an *extension* of γ if $D_\gamma \subset D_{\gamma^*}$ and $\gamma = \gamma^*$ on D_γ. We say that γ is *maximal* in $\mathcal{F}$ if the only extension of γ is γ itself; that is maximal refers to the domain and *not* to the function values. For any curve γ, the notation $\gamma(\cdot; t, x)$ means that $t \in D_\gamma$ and $\gamma(t; t, x) = x$.

Definition 2. A set $S \subset U$ is called *positively (negatively) weakly invariant* for Equation (2) if for each $x \in S$ there exists a maximal solution $\phi(\cdot, x) = \phi$ such that $\phi(t, x) \in S$ for all $t \in [0, \sup D_\phi)$ (for all $t \in (\inf D_\phi, 0]$). A set $S \in U$ is called *positively (negatively) invariant* if for each $x \in S$ and each $\phi(\cdot, x)$, $\phi(t, x) \in S$ for all $t \in [0, \sup D_\phi)$ (for all $t \in (\inf D_\phi, 0]$). S is *(weakly) invariant* if it is both positively and negatively (weakly) invariant.

Definition 3. For $S \subset \mathbf{R}^n$, $x \in S$ and $v \in \mathbf{R}^n$, we say v is *subtangential* to S at x if

$$d(S, x + tv)/t \to 0 \text{ as } t \to 0^+.$$

Theorem 4. *The relatively closed set S is positively weakly invariant if and only if $g(x)$ is subtangential to S at x for all $x \in S$.*

Corollary 5. *If each solution $\phi(\cdot, x)$ of Equation (2) is uniquely determined by the initial condition x, then S is positively invariant if and only if $g(y)$ is subtangential to S for all $y \in S$.*

Statements for negative invariance hold replacing g by $-g$ above; invariance when both hold.

2.2 Fences that Hold Solutions

The following is summarized from [10]. Consider the first-order ODE

$$\dot{x} = f(t, x), \tag{3}$$

where $x \in \mathbf{R}$ and with solutions $x = u(t)$. A "fence" is some other function $x = \alpha(t)$ that channels the solutions in the direction of the vector field.

Definition 6. A continuous and continuously differentiable function $\alpha(t)$ is a *lower* (*resp.* upper) *fence* if

$$\dot{\alpha}(t) \leq f(t, \alpha(t)), \qquad [\text{resp. } f(t, \alpha(t)) \leq \dot{\alpha}(t)],$$

A fence is *strong* when the above inequalities are strict. A lower fence is *nonporous* if whenever $\alpha(t_0) \leq u(t_0)$, then $\alpha(t) \leq u(t)$ for all $t > t_0$; reversing the inequalities defines nonporous for upper fences.

Notes: (a) The definition of nonporous in [10] has the second inequality strict. We only require the given, weaker property. (b) Piecewise differentiable fences can be taken care of by checking that the required inequalities hold for both left and right derivatives.

Theorem 7. *A strong fence is nonporous.*

Theorem 8. *If f is Lipschitz with respect to x, then any fence is nonporous.*

3 Moving Forward

3.1 Multi-Dimensional Fences

The theory of fences discussed above is given only for one-dimensional state-spaces, i.e., $x \in \mathbf{R}$ in Equation (3) above. A similar theory exists for a class of ODEs in $\mathbf{R}^n$ known as *monotone systems* [18]. We outline a more general approach, motivated by the well-known concept of a Lyapunov function, below. The approach is distinct from Lyapunov functions in the usual sense, e.g., the function need not be positive definite.

The method starts by identifying with the specification set $S \subset \mathbf{R}^n$ a scalar function $s : \mathbf{R}^n \rightarrow \mathbf{R}$ such that

$$s(x) \text{ is } \begin{cases} > 0, \, x \in \mathrm{int} S \\ = 0, \, x \in S \cap \partial S \\ < 0, \, x \notin S \end{cases} \tag{4}$$

Note that s is a generalized indicator function for the set S, viz. $S = \{x \mid s(x) \geq 0\}$. Thus, we may restrict our attention to showing the invariance of the non-negative reals under the action of our dynamical system.

Suppose we wish to show that a specification region, given by $s(x) \geq 0$ is invariant over time. Thus, we want to show that $s(t) = s(x(t)) \geq 0$ for all $t > t_0$.

Theorem 9. *Suppose $\dot{s}(t)|_{s=0} \geq 0$. Then $s(t) \geq 0$ is invariant if either*

1. *There exists $\epsilon > 0$ such that $\dot{s}(t) > 0$ for all $s \in (-\epsilon, 0)$,*
2. *$\dot{s}(t)|_{s=0}$ is piecewise differentiable, and there exists $\epsilon > 0$ such that $\dot{s}(t) \geq 0$ for all $s \in (-\epsilon, 0)$,*
3. *$\dot{s}(t)|_{s=0}$ is Lipschitz.*

Proof. For parts (1) and (3), set $f(s,t) = \dot{s}(t)|_s$ and lower fence $\alpha(t) \equiv 0$. Apply Theorems 7 and 8. For part (2), use $\alpha(t) = s(t)$ as a fence for the constant function $f \equiv 0$.

3.2 Towards Robust Verification

We may address robustness issues by enforcing conditions such as

$$\begin{aligned} s(t) &\geq \epsilon_1 > 0 \\ \dot{s}(t)|_{\epsilon_1} &\geq \epsilon_2 > 0 \end{aligned}$$

This provides the basis for "robust verification," in which a nominal system is provably verified by hand, and for which the effects of classes of perturbations (e.g., delay, sensor noise, unmodeled dynamics) can be provably ignored.

Below, we give three variations on a theme for using bounds on invariants in proving the correctness of perturbed systems given proofs for the nominal case and vice versa. Examples will be given in Section 5.2.

Bounding Invariants. Suppose there exists a function $s_L(t) \leq s(t)$, then to prove $s(t) \geq 0$ is invariant, it is sufficient to demonstrate $s_L(t) \geq 0$. Such a tactic is obviously useful in comparing systems; it might also be used if the derivative of $s(t)$ can not be computed but those of lower bound can be.

Bounding the Nomimal Below. Suppose that in our perturbed system, we can only measure $\hat{s}(t)$, but that

$$s(t) \in [\hat{s}(t) - L,\ \hat{s}(t) + H],$$

where $L,\ H \in \mathbf{R}$. Suppose that we have verified a control law $u(s)$ such that $u(0)$ results in $\dot{s}(t) \geq 0$. Then, if we use $u(0)$ whenever $\hat{s}(t) - L \leq 0$, we obtain invariance of $s(t) \geq 0$ via an *implementation relation* (cf. [13]). **Note:** The constant L can be replaced with a function $L(t)$ with the same effect.

Bounding the Perturbed Above. Suppose that in our perturbed system, we can only measure $\hat{s}(t)$, but that

$$\hat{s}(t) \in [s(t) - l,\ s(t) + h],$$

where $l,\ h \in \mathbf{R}$. Suppose that we have a control law that maintains $s(t) \geq l$. Then, we have an existence proof that $\hat{s}$ can be kept above zero. However, such a control law might not be implementable only knowing $\hat{s}$. Again, l and h may be replaced by time-varying functions.

4 Platoon Merge Example–Setup

4.1 Introduction

In [7], a "robust merge platoon maneuver" is described and analyzed. There are four high-level specifications the system should meet:

1. Safety—the platoons are not supposed to collide at a relative speed greater than v_{allow}.
2. The merge should succeed, within a particular amount of time.
3. The merge is optimal, in that there is no other maneuver that could cause the merge to complete faster.
4. Passenger comfort, as measured by bounds on acceleration and jerk, is guaranteed.

Herein, *we only address the issue of safety.* We consider two platoons of vehicles, named 1 and 2, where platoon 1 precedes platoon 2 on a single track. Positions on the track are labeled with nonnegative reals, starting with 0 at a designated beginning point.

We assume the following constants:

– $v_{\text{allow}} \in \mathbf{R}^{\geq 0}$ is the value of the allowable (read also safe or acceptable) collision velocity,
– $a_{\min} \in \mathbf{R}^{\geq 0}$ is the absolute value of the maximum emergency deceleration.

The analysis in Appendix I of [7] is done in terms of vehicle velocities. Translating to vehicle positions, their analysis implies that safety is maintained (if $d = 0$) if

$$\Delta x(t) \geq \frac{v_2^2(t) - v_1^2(t) - v_{\text{allow}}^2}{2a_{\min}} \tag{5}$$

where

– x_1, x_2 are the positions of the lead and trailing platoons,
– $\Delta x = x_1 - x_2$, the difference in position of the lead and trailing platoons,
– $v_1 \equiv \dot{x}_1, v_2 \equiv \dot{x}_2$, are the velocities of the lead and trailing platoons.

In the remainder of this section we give a formal model of the platoon safety problem in terms of hybrid I/O automata [13]. In the next section, we prove invariance of Equation 5, in the presence of an abstract controller, using derivative information only.

4.2 Safety

Platoons. We model the system by a hybrid automaton that we call *Platoons*. Each platoon i has a position x_i and a velocity $\dot{x}_i$. The hybrid automaton *Platoons* has the following (non-e [13]) discrete actions:

Input:	Internal:
none	*collide*
Output:	
none	

The variables are:

Input:
none
Output:
$\dot{x}_i \in \mathbf{R}^{\geq 0}$, $i \in \{1, 2\}$, initially arbitrary
$x_i \in \mathbf{R}^{\geq 0}$, $i \in \{1, 2\}$; initially $x_2 = 0$ and x_1 is arbitrary
collided, a Boolean, initially *false*

Internal:
none

Thus, we assume that the velocities are nonnegative—the vehicles will never go backwards. Also, platoon 2 starts at the beginning position on the track.

collide
Precondition:
$x_1 = x_2$
collided = *false*
Effect:
collided = *true*
$\dot{x}_i :=$ arbitrary value, $i \in \{1, 2\}$

We allow for fairly arbitrary behavior when cars collide: the velocities of both vehicles may change arbitrarily.

A trajectory of the hybrid system over the interval I (or *I-trajectory* [13]) w is included among the set of nontrivial trajectories exactly if:

1. *collided* is unchanged in w.
2. $\dot{x}_i$ is an integrable function in w, $i \in \{1, 2\}$.
3. For every $t \in I$, the following are true about $w(t)$:
   (a) $x_2 \leq x_1$.
   (b) The x_i values at t are obtained by integrals, working from $\dot{x}_i$.
4. For every $t \in I$ that is not the right endpoint of I, the following is true about $w(t)$: If $x_1 = x_2$ then $collided = true$.

Thus, we only consider executions in which the platoons do not bypass each other. The *collided* variable just keeps track of the first occurrence of a collision. This will be used in our statement of the correctness property below—we only want to assert what happens the first time a collision occurs.

Safety Condition. We consider the following safety condition on states of *Platoons*:

1. (Safety) If $x_1 = x_2$ and $collided = false$, then $\dot{x}_2 \leq \dot{x}_1 + v_{\text{allow}}$.

Note that this condition is formulated as an invariant assertion.

Let *Safe-Platoons* be the same as *Platoons* except that all the states are restricted to satisfy the safety condition. We will use *Safe-Platoons* as a correctness specification. It says that under all circumstances, the system guarantees that if the platoons ever collide, then the first time they do so, their relative velocity is no more than v_{allow}.

Implementation Structure. For implementations, we consider composed systems consisting of a piece modeling the real world, plus two pieces modeling controllers for the two cars. The real world model is like the *Platoons* automaton, except that the velocities of the two cars are normally controlled by acceleration variables set by two separate controllers. However, once a collision occurs, we uncouple the velocities from the controllers. This corresponds to allowing arbitrary behavior after the first collision.

Define *Controlled-Platoons* to be the same as *Platoons*, except for the following changes. *Controlled-Platoons* includes new input variables:

$\ddot{x}_i \in \mathbf{R}$, $i \in \{1, 2\}$, initially arbitrary

An I-trajectory w is included among the set of nontrivial trajectories of *Controlled-Platoons* exactly if:

1. w is a trajectory of *Platoons*.
2. If $collided = false$ in w then $\dot{x}_i$ is obtained by integration from $\ddot{x}_i$, $i \in \{1, 2\}$.

Now we describe $Controller_1$. Its input and output variables are:

Input:
$\dot{x}_i \in \mathbf{R}^{\geq 0}$, $i \in \{1, 2\}$
$x_i \in \mathbf{R}^{\geq 0}$, $i \in \{1, 2\}$
collided, a Boolean
Output:
$\ddot{x}_1$

$Controller_1$ has no external actions.

$Controller_1$ is an arbitrary hybrid automaton with the given interface, subject only to the following restrictions:

1. In any trajectory, $\ddot{x}_1$ is a bounded, piecewise continuous (and hence integrable) function.
2. In any state,
   (a) $\ddot{x}_1 \geq -a_{min}$.
   (b) If $\dot{x}_1 = 0$ then $\ddot{x}_1 \geq 0$. (It does not ask that the velocity go negative.)

The interface of $Controller_2$ is analogous. Its input and output variables are:

Input:
$\dot{x}_i \in \mathbf{R}^{\geq 0}$, $i \in \{1, 2\}$
$x_i \in \mathbf{R}^{\geq 0}$, $i \in \{1, 2\}$
collided, a Boolean
Output:
$\ddot{x}_2$

$Controller_2$ has no external actions. $Controller_1$ is an arbitrary hybrid automaton with the given interface, subject only to restrictions on $\ddot{x}_2$ and $\dot{x}_2$ as for $\ddot{x}_1$ and $\dot{x}_1$, resp., in $Controller_1$ above.

The System. Compose *Controlled-Platoons*, $Controller_1$ and $Controller_2$ using hybrid automaton composition.

We are supposed to design an instance of $Controller_2$ so that when it is composed in this way with arbitrary $Controller_1$, the resulting system satisfies the safety condition. We say that it *implements* the *Safe-Platoons* automaton, using a notion of implementation based on preserving hybrid traces. Here, the hybrid trace includes the output variables, which are the positions and velocities of both platoons plus the collided flag. That is enough to ensure that the safety condition of the spec. carries over to the implementation.

A Controller Implementation. We define a specific $Controller_2$, which we call C_2. We describe it very nondeterministically. The interface is already specified. C_2 has no discrete actions, and no internal variables.

In any state of *Platoons*, define

$$\textit{safe-measure} = x_1(t) - x_2(t) - \frac{(\dot{x}_2(t))^2 - (\dot{x}_1(t))^2 - (v_{allow})^2}{2a_{min}}$$

This says that the distance between the two platoons is great enough to allow platoon 2 to slow down sufficiently before hitting platoon 1, even if platoon

1 decelerates at its fastest possible rate.[4] The initial value of $\ddot{x}_2$ is constrained as follows. In the initial state, if *safe-measure* ≤ 0, then $\ddot{x}_2 = -a_{min}$. (Otherwise, $\ddot{x}_2$ is arbitrary.) Therefore, if the position and velocity parameters are on the boundary of a certain "region", then $\ddot{x}_2$ is guaranteed to be the minimum possible—that is, platoon 2 is guaranteed to be decelerating as fast as possible.

C_2 is an arbitrary $Controller_2$, subject only to the following additional restriction on any I-trajectory w:

> If *collided* $=$ *false* in $w(0)$ then for every $t \in I$, the following is true about $w(t)$: If *safe-measure* ≤ 0, then $\ddot{x}_2 = -a_{min}$.

The system we consider, called *Implemented-Platoons* is the composition of *Controlled-Platoons*, an arbitrary $Controller_1$, and C_2.

Correctness. We define a predicate S on states of *Implemented-Platoons*, as follows:

Predicate S: If *collided* $=$ *false* then *safe-measure* ≥ 0.

Note that C_2 is designed to guarantee explicitly that if S is ever violated, or even if it is in danger of being violated (because equality holds), platoon 2 is decelerating as fast as possible. We claim that this strategy is sufficient to guarantee that S is always true:

Lemma 10. *If S is true in the initial state of the system, then S is true in every reachable state.*

Proof. By induction on the number of steps in a hybrid execution. Initially, the claim is true by assumptions. The only (non-e) discrete steps are *collide* and internal steps of $Controller_1$.

1. *collide*
   The effect of the action ensures that *collided* $=$ *true* in the post-state, which makes S true vacuously.
2. Internal step of $Controller_1$
   This does not affect any of the quantities involved in S.

Now we consider a trajectory w based on a closed interval $[0, t]$. Since a trajectory cannot change *collided*, and S is vacuously true if *collided* $=$ *true*, we only need to consider the case where *collided* $=$ *false* throughout w. We may

[4] It can be shown [6] that the region

$$safe\text{-}measure = \max\left\{x_1 - x_2 - \frac{(\dot{x}_2)^2 - (\dot{x}_1)^2 - (v_{allow})^2}{2a_{min}}, \dot{x}_1 + v_{allow} - \dot{x}_2\right\}$$

corresponds to the biggest possible safe region. The "relaxation" corresponding to the second argument of the max function says that the relative velocities of the two platoons are already close enough. Invariance has been proven using Theorem 9, part 2.

assume that S is true in $w(0)$. We must show that S is true in $w(t)$. By definition of S, we may assume that *safe-measure* ≥ 0 in state $w(0)$ and must show that this is true in state $w(t)$.

The remaining continuous action arguments are shown in the next section (see Remark 1), using only derivative information.

Lemma 11. *S implies the safety condition.*

Proof. If *collided* = *true*, then S implies the safety condition vacuously. Hence, we must only deal with the case when *collided* = *false*. This involves continuous variables and is shown in the next section (see Remark 2).

5 Platoon Merge Example–Analysis

In this section, we provide a system theoretic proof of the "correctness" of the "abstract controller" for platoon merge proposed in the previous section. Here, *correctness* means that safety is maintained; *abstract controller* means that the control law is specified by a list of (hopefully) minimal constraints. Any actual controller implementation that satisfies these constraints will maintain safety by inclusion. In other words, if we have proved the system correct when it is connected to an *arbitrary* set of behaviors (e.g., which satisfy some constraints or assumptions), then the system will behave correctly when any *particular* behavior within that set is actually connected. Note that this is simply the viewpoint of *worst-case analysis*.

Note: We must only finish the continuous parts missing from Lemmas 10 and 11. From those proofs, we may consider only the case where *collided* = *false* in our continuous analysis below, which yields some simplifications in the resulting $\mathcal{S}$ and s.

5.1 The Nominal Case

Recall that the safety property that we wish to verify is

$$\mathcal{S}: \qquad \text{If } \Delta x(t) = 0, \text{then } v_2(t) - v_1(t) \leq v_{\text{allow}}.$$

This says that if a collision occurs, it must be the case that the relative platoon velocity is less than the allowable collision velocity.

The following restrictions which were made on trajectories of *Implemented-Platoons* above are relevant in this section:

$\mathcal{A}_1$: $\ddot{x}_i \geq -a_{\min}$, $i = 1, 2$.
$\mathcal{A}_2$: $v_i \geq 0$, $i = 1, 2$.

We also use the following "safety invariant," which is a rewriting of Equation (5) above:

$$s(t) \equiv \Delta x(t) + \frac{v_1^2(t) - v_2^2(t) + v_{\text{allow}}^2}{2a_{\min}} \tag{6}$$

Remark 1. $s(t) \geq 0$ is invariant[5] if the following condition is met:

$$\mathcal{C}: \qquad \ddot{x}_2(t) = -a_{\min} \text{ when } s(t) = 0.$$

Proof. Dropping dependence on t, Equation (6) is equivalent to

$$s \equiv x_1 - x_2 + \frac{\dot{x}_1^2 - \dot{x}_2^2 + v_{\text{allow}}^2}{2a_{\min}},$$

so

$$\dot{s} = \dot{x}_1 - \dot{x}_2 + \frac{\dot{x}_1\ddot{x}_1 - \dot{x}_2\ddot{x}_2}{a_{\min}}.$$

But "control law" $\mathcal{C}$ says

$$\begin{aligned} \dot{s}|_{s\leq 0} &= \dot{x}_1 - \dot{x}_2 + \frac{\dot{x}_1\ddot{x}_1 + \dot{x}_2 a_{\min}}{a_{\min}}, \\ &\geq \dot{x}_1 - \dot{x}_2 + \frac{-\dot{x}_1 a_{\min} + \dot{x}_2 a_{\min}}{a_{\min}} \geq 0, \end{aligned}$$

where the last inequality follows from $\mathcal{A}_1$. The conclusion then follows from Theorem 9, Part 2.

Note that the condition $\mathcal{C}$ represents the conditions of the abstract controller mentioned above. Any actual controller (implementation) that satisfies $\mathcal{C}$ will also maintain the invariant $s(t) \geq 0$. It remains to show that the invariant guarantees safety, i.e., it remains to show

Remark 2. $s(t) \geq 0$ implies $\mathcal{S}$.

Proof. First note that due to $\mathcal{A}_2$, $\mathcal{S}$ is automatically satisfied if $v_2 = 0$. Otherwise assume, for contradiction, that S is not met: $\Delta x = 0$ and

$$v_2 > v_{\text{allow}} + v_1.$$

Squaring both sides and again noting $\mathcal{A}_2$ yields the contradiction.

5.2 Perturbed Cases

In this section, we use the tools of Section 3.2 to easily prove safety under various relaxations from the nominal case, including (1) inbound delay, (2) outbound delay, and (3) sensor noise.

Below, we will use approximations of the form

$$\hat{s}(t) \equiv \hat{x}_1(t) - \hat{x}_2(t) + \frac{\hat{v}_1^2(t) - \hat{v}_2^2(t) + v_{\text{allow}}^2}{2a_{\min}}.$$

In this case (and dropping dependence on t)

$$s - \hat{s} \equiv e = (x_1 - \hat{x}_1) - (x_2 - \hat{x}_2) + \frac{(v_1^2 - \hat{v}_1^2) - (v_2^2 - \hat{v}_2^2)}{2a_{\min}}. \qquad (7)$$

Note that if $\overline{e}(t) \geq e(t) \geq \underline{e}(t)$, then $s(t) \geq \hat{s}(t) + \underline{e}(t)$.

[5] Technically, $\{(x_1, x_2, v_1, v_2) \mid s(x_1, x_2, v_1, v_2) \geq 0\}$ is a positive invariant set.

Inbound Delay. This is the lag time in communicating sensor information from the first to second platoons. Hence, if the *inbound delay* is d then

$$\begin{aligned}\hat{x}_1(t) &= x_1(t-d),\ \hat{v}_1(t) = v_1(t-d),\\ \hat{x}_2(t) &= x_2(t), \qquad \hat{v}_2(t) = v_2(t-d),\end{aligned}$$

in Equation 7 above. A simple calculation, taking into account $\mathcal{A}_{1,2}$, shows that $e(t) \geq \underline{e}(t)$ where

$$\underline{e}(t) \equiv v_1(t-d)\tilde{d} - a_{\min}\tilde{d}^2/2 + \frac{[(v_1(t-d) - a_{\min}\tilde{d})^2 - v_1^2(t-d)]}{2a_{\min}} = 0,$$

$\tilde{d} = \min\{d, v_1(t-d)/a_{\min}\}$, and the last equality follows after some algebra. So, by the arguments in Section 3.2, safety is maintained as long as full-braking is invoked for platoon 2 whenever $\hat{s}(t) \leq 0$.

Outbound Delay. In this case, the acceleration command for platoon 2 at time t, namely $\ddot{x}_2(t)$, is based on sensed readings at time $t-d$ if the *outbound delay* is d. Hence, we have Equation 7 *but with all variables at time* $t+d$. The situation is similar to inbound delay above in that one may compute bounds on $\hat{s}(t+d)$ versus $s(t+d)$ as a function of positions and velocities measured up to time t. For x_1, $\dot{x}_1$ this case is the same as above, and we can simply use

$$\hat{x}_1(t+d) = x_1(t), \qquad \hat{v}_1(t+d) = v_1(t).$$

This case is different than the previous one, though, in the sense that second platoon's controller may take into account the command it has sent over the past d time units in estimating its *own* position and velocity d units hence (assuming no collisions in between, of course). Summarizing, it uses

$$\begin{aligned}\hat{x}_2(t+d) &= x_2(t) + v_2(t)d + \int_0^d (d-\sigma)\ddot{x}_2(t-d+\sigma)\,d\sigma,\\ \hat{v}_2(t+d) &= v_2(t) + \int_0^d \ddot{x}_2(t-d+\sigma)\,d\sigma.\end{aligned}$$

Invariance is maintained if we command full-braking at all times t (i.e., set $\ddot{x}_2(t) = -a_{\min}$ for all times t) such that $\hat{s}(t+d) \leq 0$.

Sensor Noise. Here, we measure $\hat{x}_i$ and $\hat{v}_i$. We assume (the same) uniform noise bound on the measurements of all variables, e.g., $|x_1(t) - \hat{x}_1(t)| \leq \delta$ and similarly for the other three measurements. Unequal bounds follow easily. Then, we have $e(t) \geq \underline{e}$ where

$$\underline{e} \equiv -\delta - \delta + \frac{v_1^2 - (v_1+\delta)^2 - v_2^2 + (v_2 - m)^2}{2a_{\min}} = -2\delta - \frac{2v_1\delta + 2v_2 m + \delta^2 - m^2}{2a_{\min}},$$

where $m = \min(v_2, \delta)$.

Again, safety is maintained as long as full-braking is invoked for platoon 2 whenever $\hat{s}(t) + \underline{e} \leq 0$. If this is too conservative, a real application could get better estimates by considering intervals of measurements and the dynamic relationship of measured variables.

References

1. Panos Anstaklis *et al.*, editors. *Hybrid Systems II.* vol. 999, *Lecture Notes in Computer Science.* Springer, New York, 1995.
2. A. Benveniste and G. Berry, guest editors. *Proc. of the IEEE*, 79(9), 1991. Special Issue on The Synchronous Approach to Reactive and Real-Time Systems.
3. A. Benveniste and P. Le Guernic. Hybrid dynamical systems theory and the signal language. *IEEE Transactions on Automatic Control*, 35(5), 1990.
4. N. P. Bhatia and G. P. Szegö. *Dynamical Systems: Stability Theory and Applications*, vol. 35 of *Lecture Notes in Mathematics.* Springer, Berlin, 1967.
5. J. W. de Bakker *et al.*, editors. *Real-Time: Theory in Practice*, vol. 600 of *Lecture Notes in Computer Science.* Springer, New York, 1991.
6. E. Dolginova and N. Lynch. "Safety Verification for Automated Platoon Manuevers: A Case Study." In *Proc. of the International Workshop on Hybrid and Real-Time Systems (HART '97)*, Grenoble, France, March 1997. To appear.
7. J. Frankel *et al.* "Robust Platoon Maneuvers for AVHS," California PATH report, UCB, 1995. Preprint.
8. R. L. Grossman *et al.*, editors. *Hybrid Systems*, vol. 736 of *Lecture Notes in Computer Science.* Springer, New York, 1993. .
9. N. Halbwachs. *Synchronous Programming of Reactive Systems.* Kluwer Academic, Boston 1993.
10. J.H. Hubbard and B.H. West. *Differential Equations: A Dynamical Systems Approach.* Springer, New York, 1990.
11. M. Joseph, editor. *Formal Techniques in Real-Time and Fault-Tolerant Systems*, vol. 331 of *Lecture Notes in Computer Science.* Springer, New York, 1988.
12. N. Lynch. Distributed Algorithms. Morgan Kaufmann, San Francisco, 1996.
13. N. Lynch *et al.* Hybrid I/O Automata. In R. Alur, T. Henzinger, and E. Sontag, editors, *Hybrid Systems III*, pp. 496–510, Lecture Notes in Computer Science, vol. 1066, Springer, 1996.
14. O. Maler, Z. Manna, and A. Pnueli. From timed to hybrid systems. In de Bakker et al. [5], pp. 447–484.
15. Z. Manna and A. Pnueli. Verifying hybrid systems. In Grossman et al. [8], pp. 4–35.
16. Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems.* Springer, New York, 1991.
17. A. Pnueli and J. Sifakis, guest editors. *Theoretical Computer Science*, 138(1), 1995. Special Issue on Hybrid Systems.
18. H.L. Smith. *Monotone Dynamical Systems: An Introduction to the Theory of Competitive and Cooperative Systems.* Mathematical Surveys and Monographs, Vol. 41. American Mathematical Society, Providence, RI, 1995.
19. Y.-J. Wei and P.E. Caines. Hierarchical COCOLOG for finite machines. In G. Cohen and J-P. Quadrat, editors, *Proc. 11th INRIA International Conference on the Analysis and Optimization of Systems*, vol. 199 of *Lecture Notes in Control and Information Sciences*, pp. 29–38, New York, 1994. Springer.
20. J. Lygeros, D. Godbole, S. Sastry. A verified hybrid control design for automated vehicles. Preprint.
21. A. Puri and P. Varaiya. Verification of hybrid systems using abstractions. Preprint.

This article was processed using the LaTeX macro package with LLNCS style