Correct-By-Construction Adaptive Cruise Control: Two Approaches

Petter Nilsson[1], Omar Hussien[3], Ayca Balkan[3], Yuxiao Chen[2], Aaron Ames[4]
Jessy Grizzle[1], Necmiye Ozay[1], Huei Peng[2], and Paulo Tabuada[3]

Abstract—**Motivated by the challenge of developing control software provably meeting specifications for real world problems, this paper applies formal methods to adaptive cruise control (ACC). Starting from a Linear Temporal Logic specification for ACC, obtained by interpreting relevant ACC standards, we discuss in this paper two different control software synthesis methods. Each method produces a controller that is correct-by-construction, meaning that trajectories of the closed-loop systems provably meet the specification. Both methods rely on fixed-point computations of certain set-valued mappings. However, one of the methods performs these computations on the continuous state space whereas the other method operates on a finite-state abstraction. While controller synthesis is based on a low-dimensional model, each controller is tested on CarSim, an industry-standard vehicle simulator. Our results demonstrate several advantages over classical control design techniques. First, a formal approach to control design removes potential ambiguity in textual specifications by translating them into precise mathematical requirements. Second, because the resulting closed-loop system is known a priori to satisfy the specification, testing can then focus on the validity of the models used in control design and whether the specification captures the intended requirements. Finally, the set from where the specification (e.g., safety) can be enforced is explicitly computed and thus conditions for passing control to an emergency controller are clearly defined.**

I. Introduction

Adaptive cruise control (ACC) is a driver assistance system that seeks to combine safe following distance with speed regulation. When there is no preceding vehicle in sight, an ACC-equipped vehicle maintains a constant speed set by the driver, just as in a conventional cruise control (CCC) system. When a preceding vehicle is detected, however, an ACC-equipped vehicle changes its control objective to maintaining a safe distance between the two vehicles. For driver comfort, when regulating speed or following distance, the control system is normally limited to using significantly less than the vehicle's maximal deceleration and acceleration capacity. For this reason, ACC is not an active safety system.

ACC has been available on production vehicles since the mid-1990s. All of the ACC vehicles on the market today are of the autonomous type, meaning that the vehicle relies only on information collected on-board. In future (non-autonomous) ACC systems, communication with adjacent vehicles or road-side infrastructure will be used to enhance the responsiveness to lane, speed, or acceleration changes in the preceding vehicle or vehicles. Many of the early ACC systems shut off below a given threshold speed. More recently, full-range (stop-and-go) ACC is available that can bring a vehicle to a full stop and then launch from standstill, and is thus capable of dealing with congested urban traffic. In the last several years, some automakers (e.g., Volvo and Cadillac) have introduced an automated full-braking function to leverage hardware already installed for ACC, tiptoeing the red-line between comfort/convenience systems and active safety systems. This trend is likely to continue as more automated vehicle control functions are introduced on production vehicles.

Even though ACC is a driver convenience feature, manufacturers are likely to treat all software tied to acceleration, deceleration and turning of the vehicle as being related to safety. The increase in safety related functions as manufacturers work toward a fully automated vehicle signifies the need for certification of correctness of the control software that implements such functionality. Current practice for certification is through extensive testing, which constitutes a bottleneck in vehicle design and development cycles. This burden can be partially alleviated by adopting correct-by-construction control software synthesis techniques, where correct by construction means that the control software is guaranteed to meet its formal specifications given a set of assumptions on the physical plant and implementation platform. If the software is known in advance to correctly implement the control specification, then testing can focus on the validity of the assumptions used to model the vehicle and its environment or whether the specification captured the intended requirements.

The main contribution of this paper is to demonstrate how correct-by-construction ACC software can be synthesized using two different techniques. In [1], we synthesized correct-by-construction control software for ACC when the lead vehicle's speed was constant. Here, we extend the task by synthesizing correct-by-construction control software for ACC when the speed of the lead car is variable. This brings in the need to achieve robustness against the unknown acceleration of the preceding vehicle. The formal specifications considered in this paper are given in Linear Temporal Logic, a common specification language for software systems. Two synthesis methods are used, one performing set computations directly on the continuous domain, and a second based on finite-state abstractions. The two resulting correct-by-construction

[1]Dept. of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI. E-mails: {pettni, grizzle, necmiye}@umich.edu. [2]Dept. of Mechanical Engineering, University of Michigan, Ann Arbor, MI. E-mails: {chenyx, hpeng}@umich.edu. [3]Dept. of Electrical Engineering, University of California at Los Angeles, CA. E-mails: {ohussien, abalkan, tabuada}@ucla.edu. [4]Dept. of Electrical Engineering and Dept. of Mechanical Engineering, Georgia Institute of Technology. E-mail: ames@gatech.edu.

This research is supported by the NSF CPS Awards 1239055, 1239037, and 1239085.

controllers are compared by running simulations in Simulink and on a 16 degree of freedom model (with more than 30 continuous states) in CarSim[1]. In subsequent work, we plan to address correct-by-construction control software for lane keeping, as well as correct-by-construction control software when ACC and lane keeping are simultaneously active.

The remainder of the paper is organized as follows. Section II overviews past work on ACC, in terms of controller design and correctness of control software. The simplified vehicle models to be used in the paper are also introduced in this section, which concludes with an interpretation of an ACC standard in terms of a formal specification given in Linear Temporal Logic. Section III develops a correct-by-construction solution by computing polyhedral controlled invariant sets (PCIS), while a solution based on a finite-state abstraction is presented in Section IV. In Section V the two correct-by-construction controllers are compared in Simulink and CarSim simulations. Furthermore, we show how the synthesized software can be used to supervise an existing ACC controller in order to endow it with formal safety guarantees. The controllers have also been implemented on a hardware testbed consisting of radio-controlled (RC) cars; we summarize this implementation in Section VI. Finally, we give concluding remarks in Section VII.

II. Adaptive cruise control

A. Past work

Adaptive cruise control started as an extension of conventional cruise control (CCC), which was how it was described in relevant ISO [2] and SAE [3] standards. In these early design concepts, ACC was a phase or mode of the overall control system, and even "rode-on" existing CCC hardware architecture: in the ACC mode, the speed command to the CCC servo-loop was adjusted to achieve the desired range control objective (e.g., [4]). This nested control architecture resulted in slower response (in comparison to when throttle and brake are controlled directly), which was observed in prototype ACC vehicles in field tests [5]. A comparison between the velocity-command approach and acceleration command approach was analyzed in [6].

The potential of ACC-equipped vehicles for improving traffic flow and safety has been studied extensively since the 1990s [7], [8]. In addition to traffic flow and safety, string stability [9], [10], [11], [12], congestion [13], fuel economy [14], and integration with crash avoidance [15] have also been studied. An extensive survey on ACC designs, including the underlying control concepts, is given in [16]. In recent years, Model Predictive Control (MPC) is widely used in ACC design [17].

Although driver assistance and safety modules such as ACC have been investigated for many years, the emphasis on the correctness of a module's software implementation is much more recent. Safety verification of evasive maneuvers for autonomous vehicles is addressed in [18]. By computing the reachable set of each vehicle under different types of uncertainties and disturbances, safety is ensured whenever the reachable sets for different vehicles are disjoint. The computation of reachability sets for hybrid systems is known to be expensive and in [19] verification is done through counterexample-guided search. Rather than working with a detailed model of the system to be verified, an abstraction is used. Verifying the abstraction leads to counterexamples that may be spurious, i.e., they may not be true behaviors of the real system. However, since reachability analysis on the abstraction is cheaper, a two-step approach is taken: 1) the abstract model is used to obtain counterexamples; 2) the counterexamples are proved or disproved on a detailed model of the system to be verified. The previous approaches to verification fall in the class of model checking; given a specification one checks (by reachability computation or counterexample-guided search) if the specification is met. A different approach, considered in [20], is theorem proving. Here, one writes the assumptions about the system and its environment in a convenient logic and proves a theorem stating that the desired specification follows from the assumptions. A related approach is the use of satisfiability (SAT) and satisfiability modulo theory (SMT) solvers instead of a customized logic as was done in [21].

Unlike all the aforementioned approaches, we do not verify an existing software module. Instead, we synthesize a software module that is guaranteed to satisfy the specification by construction, hence the term correct-by-construction. As we demonstrate, correct-by-construction software can also be used as a supervisor of an existing controller, hence eliminating the need for verification. A similar idea was presented in [22] where a provably correct supervisory intersection collision avoidance controller was constructed by exploiting the order-preserving properties of the dynamics.

A formal synthesis problem consists of a formal specification on a set of variables and a dynamic model which governs the evolution over time of the same set of variables. In the following, we first introduce a dynamic model for a two-car system, then give a formal specification that captures ACC requirements.

B. Design model

The ACC vehicle is modeled as a (lumped) point mass m moving along a straight line at velocity v. The net action of braking and engine torque applied to the wheels is lumped as a net force F_w acting on the mass of the vehicle, while the combined aerodynamic and rolling resistance is gathered into a net force F_r,

$$m\dot{v} = F_w - F_r. \tag{1}$$

In the above equation, F_w is viewed as the control input and is assumed to be bounded by

$$-0.3mg \le F_w \le 0.2mg, \tag{2}$$

where g is the gravitational constant. Such a bound is consistent with non-emergency braking and acceleration, and thus with the "driver comfort" notion of ACC. The term F_r is represented by $F_r = f_0 + f_1 v + f_2 v^2$ for constants f_0, f_1 and f_2 that can be empirically measured [7]. We limit the

[1]CarSim is a vehicle simulation package that is widely used in industry. It is a registered trademark of Mechanical Simulation Corporation, Ann Arbor, MI.

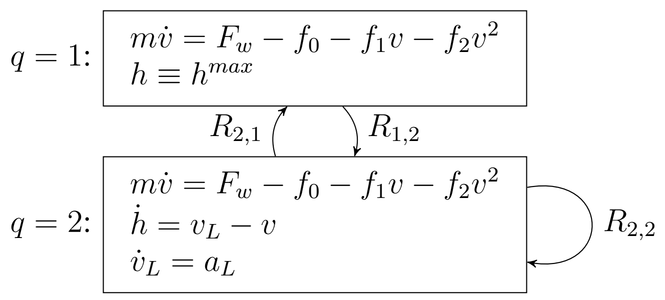


Fig. 1. Hybrid system model for ACC.

admissible velocities to a bounded set $\mathcal{V} = [v^{min}, v^{max}]$ with $v^{min} \geq 0$.

To include a lead vehicle in the system description, we assume that the ACC vehicle is equipped with a radar that measures headway h up to a maximal value h^{max}. When no lead car is present, the radar shows its maximal value h^{max}. This leads to a hybrid system model with a discrete state $q \in \{1, 2\}$. We call the state $q = 1$ the `no lead car` state, and call $q = 2$ the `lead car` state. This latter discrete state is used to model the situation when a lead car is present within the radar range, and therefore contains an additional continuous state v_L, corresponding to lead car velocity. The continuous dynamics in the state $q = 1$ are those of equation (1) while the continuous dynamics in state $q = 2$ contain two additional differential equations

$$\begin{aligned} \dot{h} &= v_L - v, \\ \dot{v}_L &= a_L. \end{aligned} \tag{3}$$

Here v_L is the velocity of the lead car and a_L is its acceleration.

The two discrete states have different continuous state spaces. The state $q = 1$ has state space $\mathcal{V} \times \{h^{max}\}$ while state $q = 2$ has state space $\mathcal{V} \times \mathcal{H} \times \mathcal{V}_L$, where $\mathcal{H} = [0, h^{max}]$, and the lead car velocity range $\mathcal{V}_L$ is given below in the form of an assumption.

In practice, $q = 2$ when there is a car within the radar range, and $q = 1$ otherwise. Switching between the two modes is thus governed by lane changes undertaken by other cars, which are modeled using reset maps $R_{1,2} : \mathcal{V} \times \{h^{max}\} \to 2^{\mathcal{V} \times \mathcal{H} \times \mathcal{V}_L}$, $R_{2,1} : \mathcal{V} \times \mathcal{H} \times \mathcal{V}_L \to 2^{\mathcal{V}} \times \{h^{max}\}$ and $R_{2,2} : \mathcal{V} \times \mathcal{H} \times \mathcal{V}_L \to 2^{\mathcal{V} \times \mathcal{H} \times \mathcal{V}_L}$:

$$\begin{aligned} &R_{1,2}(v, h^{max}) = \{(v, \bar{h}, \bar{v}_L) : (\bar{h}, \bar{v}_L) \in \mathcal{H} \times \mathcal{V}_L\}, \\ &R_{2,1}(v, h, v_L) = \{(v, h^{max})\}, \\ &R_{2,2}(v, h, v_L) = \{(v, \bar{h}, \bar{v}_L) : (\bar{h}, \bar{v}_L) \in \mathcal{H} \times \mathcal{V}_L\}. \end{aligned} \tag{4}$$

Here $R_{1,2}$ models a transition from the `no lead car` state $q = 1$ to the `lead car` state $q = 2$, where the headway is initialized to some $\bar{h} \in \mathcal{H}$ and the lead car speed v_L to some $\bar{v}_L \in \mathcal{V}_L$. Similarly, $R_{2,2}$ models situations where the radar reading suddenly changes as a result of lane changes undertaken by cars in front.

The switching is assumed to be non-deterministic, except for the case when h reaches h^{max} in state $q = 2$, in which case a forced transition to $q = 1$ occurs. The complete hybrid model is shown schematically in Fig. 1.

We make the following assumptions on lead car behavior:

(A.1) The velocity v_L of the lead car is always in the interval $\mathcal{V}_L = [v_L^{min}, v_L^{max}]$, where $v_L^{min} \geq v^{min}$. Furthermore, its acceleration is bounded by $a_L \in [a_L^{min}, a_L^{max}]$.

(A.2) There is at most one lead car within the radar range at all times, i.e., $R_{2,2}(v, h, v_L) = (v, h, v_L)$ for all $(v, h, v_L) \in \mathcal{V} \times \mathcal{H} \times \mathcal{V}_L$.

In Section VII we comment on the reasonableness of these assumptions and how they can be relaxed.

C. Formal ACC specification

In this section we formalize the adaptive cruise control requirements using linear temporal logic (LTL). Introducing the *time headway* ω, defined as $\omega = h/v$, we summarize the requirements defined by the International Organization of Standardization, see [2, Chapter 6], as follows:

1) ACC operates in two modes: the `set speed` mode and the `time gap` mode;
2) in `set speed` mode, a preset desired speed v^{des} eventually needs to be maintained;
3) in `time gap` mode, a desired time headway ω^{des} to the lead vehicle eventually needs to be maintained; and the time headway needs to satisfy $\omega \geq \omega^{min}$ at all times,
4) The system is in `set speed` mode if $v^{des} \leq h/\omega^{des}$, otherwise it is in `time gap` mode;
5) independently of the mode, the input constraint (2) needs to be satisfied at all times.

We proceed with the translation of requirements 1)–5) into LTL. The basic building blocks of an LTL specification are the so-called *atomic propositions*. The set of atomic propositions represents the quantities necessary to express a desired behavior. These atomic propositions are identified with subsets of the state, input and/or output spaces of the hybrid system in the sense that an atomic proposition is satisfied whenever the state, input and/or output of the hybrid system belongs to the set corresponding to that proposition. For the ACC system, the desired behavior is given in terms quantities related to velocity v and headway h, and in terms of the input F_w. Thus, we will identify atomic propositions with subsets of $v - h - F_w$ space, that is, subsets of $(\mathcal{V} \times \mathcal{H}) \times \mathbb{R}$.

First, to be able to refer to the two modes of ACC in the specification, we introduce an atomic proposition M_1 that is satisfied when the system is in `set speed` mode and an atomic proposition M_2 that is satisfied when the system is in `time gap` mode. We formally define specification modes M_1 and M_2 as follows:

$$\begin{aligned} M_1 &= \left\{(v, h, F_w) : v^{des} \leq h/\omega^{des}\right\}, \\ M_2 &= \left\{(v, h, F_w) : v^{des} > h/\omega^{des}\right\}. \end{aligned}$$

We also introduce the atomic propositions G_1, G_2, S_1, S_2 and S_U. The safe sets S_1 and S_2 are used to express constraints that need be satisfied at all times in mode M_1 and M_2, respectively. The set S_U defines the input constraint which must be satisfied at all times. These three sets are given by

$$\begin{aligned} S_1 &= \mathcal{V} \times \{h^{max}\} \times \mathbb{R}, \\ S_2 &= \left\{(v, h, F_w) : v \leq h/\omega^{min}\right\}, \\ S_U &= \{(v, h, F_w) : -0.3mg \leq F_w \leq 0.2mg\}. \end{aligned}$$

Next, we define target sets G_1 and G_2 to express requirements 2) and 3). Here, multiple interpretations are possible, depending on whether one interprets "maintain" as *not exceeding* or as *stay in the vicinity of*. Below, both these interpretations are detailed, and we later show how they impact performance by synthesizing controllers for each.

a) Target sets with upper bounds: If we interpret *maintain a bound* as satisfying an upper bound only, then in mode M_1, a desired upper bound on velocity v^{des} should eventually be achieved which is expressed by $\widetilde{G}_1 = \{(v,h,F_w) : v \le v^{des}\}$. Correspondingly, $\widetilde{G}_2 = \{(v,h,F_w) : v \le h/\omega^{des}\}$ collects all states with an upper bound on velocity that implies time headway above[2] ω^{des}. This set should be reached and kept invariant if the ACC is in mode M_2 indefinitely. The target sets $\widetilde{G}_1$ and $\widetilde{G}_2$, together with mode sets and the unsafe set, are depicted in Fig. 2a.

b) Target sets with upper and lower bounds: Instead of just respecting upper bounds on velocity, the target sets can be constrained to a vicinity of a given speed. Introducing additional lower bounds on v bounds time headway from above[2] in `time gap` mode, which may be desirable in order to deter other cars from cutting in between the ACC car and the current lead car. With the modified target sets $\bar{G}_1 = \{(v,h,F_w) : |v - v^{des}| \le \varepsilon\}$ and $\bar{G}_2 = \{(v,h,F_w) : |v - h/\omega^{des}| \le \varepsilon\}$, an upper bound is formally guaranteed in steady state. These modified sets are depicted in Fig. 2b for $\varepsilon = 1$.

In summary, the set of atomic propositions is given by $AP = \{M_1, M_2, G_1, G_2, S_1, S_2, S_U\}$, where G_1 is either given as $\widetilde{G}_1$ or $\bar{G}_1$, and correspondingly for G_2.

The specifications considered in this paper can be expressed using the atomic propositions AP, the propositional logic conjunction "$\wedge$" and negation "$\neg$", and the temporal operator always "$\square$". We interpret LTL formulas over infinite sequences (ξ, ν) where the signal $\xi : \mathbb{N} \to \mathbb{R}^2$ is a sample-and-hold trajectory for a given sampling time τ, corresponding to the speed (v) and headway (h) states of the hybrid system in Fig. 1, given the input signal $\nu : \mathbb{N} \to \mathbb{R}$ generated by the ACC and some behavior of the lead car. We refer to the pair of such sequences (ξ, ν) as a *behavior of the closed-loop system*, i.e., the hybrid system controlled by the ACC.

We next describe the syntax and the semantics of the fragment of LTL that we use in this paper. We consider the LTL formulas that are constructed from atomic propositions AP according to the following grammar:

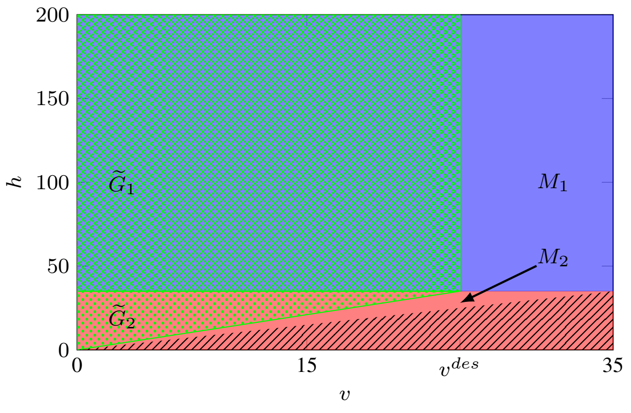
$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \square\varphi,$$

for $p \in AP$. We write $\varphi \wedge \psi$ and $\varphi \implies \psi$ as abbreviations for the formulas $\neg(\neg\varphi \vee \neg\psi)$ and $\neg\varphi \vee \psi$, respectively. We also use the short hand notation $\lozenge\varphi$, for $\neg\square\neg\varphi$.

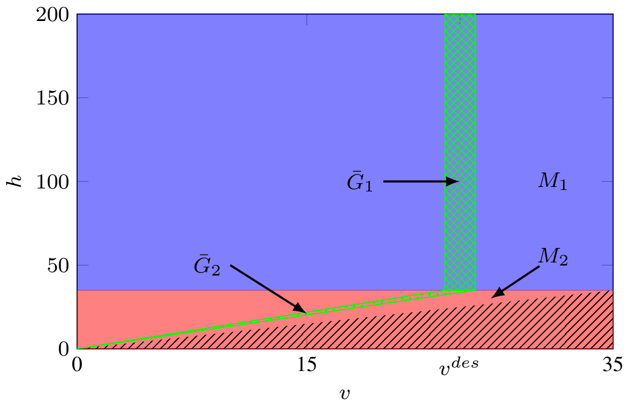
We give an inductive definition of when a behavior *satifies* a formula φ at time $i \in \mathbb{N}$, which we denote by $(\xi,\nu), i \models \varphi$:

- For $p \in AP$, $(\xi,\nu), i \models p$ iff $(\xi_i, \nu_i) \in p$,
- $(\xi,\nu), i \models \neg\varphi$ iff $(\xi,\nu), i \not\models \varphi$,
- $(\xi,\nu), i \models \varphi \vee \psi$ iff $(\xi,\nu), i \models \varphi$ or $(\xi,\nu), i \models \psi$,

[2]Since $\omega = h/v$, an upper bound on v is a lower bound on ω, and vice versa.



(a)



(b)

Fig. 2. Illustrations (best viewed in color) of projections of target sets $\widetilde{G}_1, \widetilde{G}_2$ (top), $\bar{G}_1, \bar{G}_2$ (bottom) and specification mode sets M_1 (blue), M_2 (red). The black dashed part is the unsafe region.

- $(\xi,\nu), i \models \square\varphi$ iff $(\xi,\nu), k \models \varphi$, for all $k \ge i$.

For example, consider the simple formulas $\varphi_1 = \square(M_1 \wedge G_1 \wedge S_U)$ and $\varphi_2 = \lozenge(M_2 \wedge G_2)$. A behavior satisfies φ_1 if $(\xi_t, \nu_t) \in M_1 \cap G_1 \cap S_U$ holds for *all* $t \in \mathbb{N}$. A behavior satisfies φ_2 if $(\xi_t, \nu_t) \in M_2 \cap G_2$ holds for *some* time $t \in \mathbb{N}$.

A closed-loop system satisfies a LTL formula φ if every system behavior (ξ,ν) satisfies $(\xi,\nu), 0 \models \varphi$.

Having introduced the semantics of LTL, the ACC specification can be described by the LTL formula

$$\psi = \square S_U \wedge \square \bigwedge_{i=1}^{2} \begin{pmatrix} (M_i \implies S_i) \\ \wedge\, (\square M_i \implies \lozenge\square G_i) \end{pmatrix}. \tag{5}$$

The first term $\square S_U$ dictates that the input constraint should always be met, while the second term specifies that the safe set S_i should be kept invariant when in mode M_i, and that the target set G_i should eventually be invariant, when indefinitely in mode M_i.

Notice that the specification gives no guarantees of reaching the desired time headway and velocity, when the system switches between M_1 and M_2 infinitely often. This is in compliance with the ISO standard [2], which does not specify what the system should guarantee in such a scenario.

D. Problem statement

The goal of this paper is to synthesize a controller for the hybrid system in Fig. 1 with reset maps given by (4) so that every behavior of the closed-loop system satisfies the specification ψ given in (5). However, if the lane-change behavior of other cars is not properly constrained, e.g., if a car with low speed may cut right in front of the ACC-equipped when it is traveling at a high speed, no controller can prevent collisions, hence ψ cannot be satisfied.

Motivated by this fact, the formal controller synthesis problem can now be stated as follows:

Problem 1: Assume (A.1) and (A.2) hold. Synthesize a controller and a control domain $\mathcal{D} \subseteq \mathcal{V} \times \mathcal{H} \times \mathcal{V}_L$ so that every behavior (ξ, ν) of the closed-loop system satisfies ψ if the values of $R_{1,2}(v, h)$ are restricted to $\mathcal{D}$.

In what follows, we present two approaches to this problem. Both approaches seek to synthesize controllers with control domains that are as large as possible. In general, finding a maximal $\mathcal{D}$ can be hard [23]. The presented approaches are approximate in the sense that although the synthesized controllers are provably-correct, the synthesized control domains are not necessarily maximal. In light of the discussion in Section II-C, we give solutions for different interpretations of the target sets and also show how different assumptions on the environment affect the solutions.

III. Solution by PCIS

The goal of this section is to describe how a set in which it is possible to enforce the specification (5) can be obtained through Polyhedral Controlled Invariant Set (PCIS) computations. For the discrete state $q = 1$, satisfying the specification (5) is not challenging. The system is always in specification mode M_1 which only requires the upper bound on velocity to be maintained. For this discrete state, we use a simple MPC controller that reaches and maintains $v = v^{des}$, and focus our formal synthesis efforts on synthesizing a controller for the state $q = 2$. Later in this section, we argue why the composition of these controllers solve Problem 1, assuming certain restrictions on the reset maps.

We require the ability to perform (robust) reachability computations, which is difficult for general non-linear systems. There are however known reachability algorithms for discrete-time affine systems, which we leverage by linearizing equations (1) and (3) in a correctness-preserving way. Below we describe the linearization step and recall robust reachability computations for affine systems. We then introduce set-valued mappings whose fixed-points provide a solution to Problem 1, and show how we can find a fixed-point through approximate computations.

A. Linearization

We wish to obtain an affine representation of (1)+(3) such that correctness in the linearized system implies correctness in (1)+(3). This can be achieved by assuring that, for any admissible control action applied to the linearized system, there should be a corresponding admissible control action for the original system such that the states of the two systems track each other.

To this end, consider the affine system

$$\begin{aligned} \dot{v} &= \frac{1}{m}\left(\bar{F}_w - \bar{f}_0 - \bar{f}_1 v\right), \\ \dot{h} &= v_L - v, \\ \dot{v}_L &= a_L, \end{aligned} \tag{6}$$

where $\bar{f}_0 = f_0 - f_2\bar{v}^2$ and $\bar{f}_1 = f_1 + 2f_2\bar{v}$ for some nominal speed $\bar{v} \in \mathcal{V}$, together with the restrictions on input and disturbance

$$\begin{aligned} \bar{F}_w &\in [-0.3mg, 0.2mg - \gamma], \\ a_L &\in [a_L^{min}, a_L^{max}], \end{aligned}$$

for $\gamma = \max_{v\in\mathcal{V}} f_2(v - \bar{v})^2$.

Proposition 1: For any $\tau > 0$ and $a_L : [0,\tau] \to [a_L^{min}, a_L^{max}]$, if there exists an input $\bar{F}_w : [0,\tau] \to [-0.3mg, 0.2mg - \gamma]$ that steers the state of (6) from (v^0, h^0, v_L^0) to (v^1, h^1, v_L^1), then there exists an input $F_w : [0,\tau] \to [-0.3mg, 0.2mg]$ that steers the state of (1)+(3) from (v^0, h^0, v_L^0) to (v^1, h^1, v_L^1).

Proof: The systems (6) and (1)+(3) only differ in the dynamics of v, so it is enough to show that the evolution of v is identical between the two. Given $\bar{F}_w$, consider the modified control signal $F_w = \bar{F}_w + f_2(v - \bar{v})^2$ applied to (1)+(3). By definition of γ, it satisfies $F_w(t) \in [-0.3mg, 0.2mg]$ for all $t \in [0,\tau]$. Furthermore, the closed-loop dynamics of v become

$$\begin{aligned} m\dot{v} &= F_w - f_0 - f_1 v - f_2 v^2 \\ &= \bar{F}_w + f_2(v-\bar{v})^2 - f_0 - f_1 v - f_2 v^2 \\ &= \bar{F}_w - \underbrace{(f_0 - f_2\bar{v}^2)}_{\bar{f}_0} - \underbrace{(f_1 + 2f_2\bar{v})}_{\bar{f}_1} v, \end{aligned}$$

i.e. identical to the linearized system. ■

B. Robust reachability

Given a general affine system $\dot{x} = Ax + Bu + Ed + K$, we consider the time-discretized version

$$\begin{aligned} x(t+\tau) = A_\tau x(t) + B_\tau u(t) + E_\tau d(t) + K_\tau, \\ x(t) \in \mathbb{R}^n, u(t) \in \mathbb{R}^m, d(t) \in \mathbb{R}^p. \end{aligned} \tag{7}$$

Here, $A_\tau = e^{A\tau}$ and, formally for $\Theta \in \{B, E, K\}$, we have $\Theta_\tau = \int_0^\tau e^{As}\mathrm{d}s\ \Theta$. Input restrictions and disturbance bounds are assumed to be of the following state-dependent linear forms:

$$H_{ux}x + H_{uu}u \leq h_u, \tag{8}$$

$$H_{dx}^- x + h_d^- \leq d \leq H_{dx}^+ x + h_d^+. \tag{9}$$

Let Ξ denote the dynamics (7) together with (8) - (9). Then, given Ξ and a final set X, we can characterize the *backwards reachable set* from X defined as

$$\begin{aligned} Pre_\tau^\Xi(X) = \{x_0 : \exists u_0 \text{ s.t. } (x_0, u_0) \text{ sat. (8)}, \\ A_\tau x_0 + B_\tau u_0 + E_\tau d + K_\tau \in X \\ \text{for all } d \text{ s.t. } (x_0, d) \text{ sat. (9)}\}, \end{aligned} \tag{10}$$

as a projection of a higher-dimensional polyhedron. Indeed, it turns out that for a final set $X = \{x : H_x x \leq h_x\}$ (i.e., a polyhedron), $Pre_{\tau}^{\Xi}(X)$ is the projection of a polyhedron in $x-u$-space onto its x-coordinates, that is,

$$Pre_{\tau}^{\Xi}(X) = \{x \mid \exists\, u \text{ s.t } (x,u) \in P\}, \tag{11}$$

for

$$P = \left\{ (x_0, u) : \begin{bmatrix} H_x(A_\tau + E_\tau H_{dx}^{+}) & H_x B_\tau \\ H_x(A_\tau + E_\tau H_{dx}^{-}) & H_x B_\tau \\ H_{ux} & H_{uu} \end{bmatrix} \begin{bmatrix} x_0 \\ u \end{bmatrix} \right. \\ \left. \leq \begin{bmatrix} h_x - H_x(E_\tau h_d^{+} + K_\tau) \\ h_x - H_x(E_\tau h_d^{-} + K_\tau) \\ h_u \end{bmatrix} \right\}.$$

The proof of (11) is a standard argument (e.g. [24, Chapter 11]) slightly adapted to accommodate state-dependent bounds on input and disturbance as in (8)-(9). There is software available for numerically computing polyhedron projections, such as the Multi-Parametric Toolbox [25] for MATLAB.

C. Set-valued mappings

In order to compute sets in which the specification can be enforced, we introduce two set-valued mappings. For a system Ξ given as above, we introduce the *robust, safe reachability* operator $\text{Rch}_{\tau,S}^{\infty,\Xi}(X)$ and the *robust controlled invariance* operator $\text{Inv}_{\tau}^{\infty,\Xi}(X)$.

Definition 1: $\text{Rch}_{\tau,S}^{\infty,\Xi}(X)$ is the set of all $x_0 \in S$ such that Ξ in finite time can reach $X \subset S$ from x_0 without exiting S, regardless of disturbance:

$$\text{Rch}_{\tau,S}^{\infty,\Xi}(X) = \bigcup_{k \geq 0} \underbrace{Pre_{\tau,S}^{\Xi} \circ \ldots \circ Pre_{\tau,S}^{\Xi}}_{k \text{ times}}(X),$$

where

$$Pre_{\tau,S}^{\Xi}(X) = S \cap Pre_{\tau}^{\Xi}(X).$$

Definition 2: $\text{Inv}_{\tau}^{\infty,\Xi}(X)$ is the set of all states $x_0 \in X$ starting from which Ξ can be controlled to remain in X for all future times:

$$\text{Inv}_{\tau}^{\infty,\Xi}(X) = \lim_{k \to \infty} \underbrace{Pre_{\tau,X}^{\Xi} \circ \ldots \circ Pre_{\tau,X}^{\Xi}}_{k \text{ times}}(X).$$

This invariance operator, and whether it is determined by a finite k, is discussed in the literature [26], [27], [28].

Given these two operators, consider the composed operators $\Gamma_i : 2^{\mathbb{R}^n} \times 2^{\mathbb{R}^n} \to 2^{\mathbb{R}^n}$ for $i = 1, 2$ defined by

$$\Gamma_i : (C_1, C_2) \mapsto C_i^{+}, \tag{12}$$

where

$$C_1^{+} = M_1 \cap \text{Rch}_{\tau,S_1}^{\infty,\Xi} \underbrace{\left(\text{Inv}_{\tau}^{\infty,\Xi}\left(G_1 \cap (C_1 \cup C_2)\right) \cup C_2\right)}_{D_1},$$

$$C_2^{+} = M_2 \cap \text{Rch}_{\tau,S_2}^{\infty,\Xi} \underbrace{\left(\text{Inv}_{\tau}^{\infty,\Xi}\left(G_2 \cap (C_1 \cup C_2)\right) \cup C_1\right)}_{D_2}.$$

Given safe sets S_1, S_2, mode sets M_1, M_2, and target sets G_1, G_2, which together define a specification of the form (5), we claim that any C_1, C_2 such that

$$\begin{aligned} C_1 &\subset \Gamma_1(C_1, C_2), \\ C_2 &\subset \Gamma_2(C_1, C_2), \end{aligned} \tag{13}$$

TABLE I
PARAMETER VALUES

m	1370 kg	v^{min}	0 m/s	v^{max}	35 m/s
f_0	51 N	h^{min}	0 m	h^{max}	200 m
f_1	1.2567 Ns/m	v^{des}	25 m/s	ε	1 m/s
f_2	0.4342 Ns^2/m^2	ω^{min}	1 s	ω^{des}	1.4 s
v_L^{min}	0 m/s	v_L^{max}	20 m/s	g	9.82 m/s^2
a_L^{min}	-0.97 m/s^2	a_L^{max}	0.65 m/s^2	τ	0.5 s

constitute a domain in which the specification can be fulfilled. Specifically, a correct control strategy for such sets is the following:

- When in C_1, make progress towards/remain in D_1 (such that it is reached in finite time),
- When in C_2, make progress towards/remain in D_2 (such that it is reached in finite time).

In order to implement such a strategy, a control method that can enforce set membership constraints is required. When the sets are polyhedra, a natural choice is model predictive control (MPC) which at a given time instant selects an optimal (with respect to MPC weights) input, among the set of inputs that are guaranteed to satisfy the specification. Provided that the sets satisfy (13), the MPC optimization problems are guaranteed to be feasible.

Proposition 2: Assume C_1, C_2 satisfy (13) and take $\mathcal{D} = C_1 \cup C_2$ as the restriction of the reset map $\mathcal{R}_{1,2}$. Then the proposed strategy for hybrid state $q = 2$ in conjunction with a correct[3] conventional cruise controller for hybrid state $q = 1$ solves Problem 1.

Proof: First remark that the strategy is feasible by definition and that the system will remain in $C_1 \cup C_2$ whenever $q = 2$. By definition of $\text{Rch}_{\tau,S}^{\infty,\Xi}$, the safety part of the specification is fulfilled. If the system is in C_1, the clause $\Box(\Box M_1 \to \Diamond\Box G_1)$ can be satisfied in two ways; either the system reaches G_1 and keeps it invariant, or it exits M_1. By definition of D_1, at least one of these alternatives will happen when D_1 is reached. The same holds for C_2 by symmetry. ■

The idea is to iterate (12) in order to find sets C_1 and C_2 that satisfy (13). For $C_1 \subset \tilde{C}_1$ and $C_2 \subset \tilde{C}_2$, these operators have the property that $\Gamma_i(C_1, C_2) \subset \Gamma_i(\tilde{C}_1, \tilde{C}_2)$. Therefore, the set valued function $\Gamma(C_1, C_2) \doteq (\Gamma_1(C_1, C_2), \Gamma_2(C_1, C_2))$ is monotone with respect to the set inclusion order. Thus, when initialized with $C_1 = M_1$ and $C_2 = M_2$, iterating (12) leads to a sequence of non-expanding sets that converges towards a fixed-point [29].

D. Synthesis of an ACC controller

We now make use of the fixed-point ideas presented in Section III-C together with the projection-based robust reachability computations for linear systems from Section III-B. The reachability computations allow us to implement $\text{Rch}_{\tau,S}^{\infty,\Xi}(X)$ and $\text{Inv}_{\tau}^{\infty,\Xi}(X)$ algorithmically, and by initializing $C_1 = M_1$ and $C_2 = M_2$, and iterating (12), we can look for fixed-points that give a valid controller domain.

[3]A correct conventional cruise controller should reach and maintain the desired speed without exceeding input bounds.

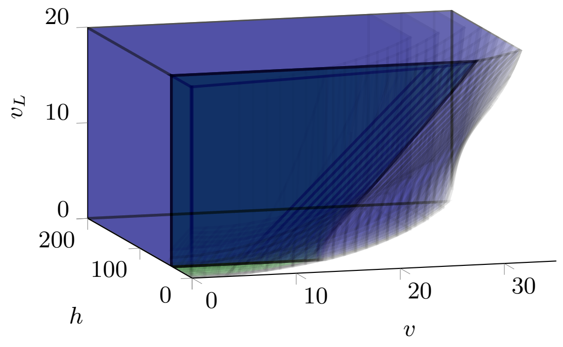


Fig. 3. Sets C_1 (rear part, blue) and C_2 (front part, green), together forming the controller domain.

Since non-convex sets must be represented by unions of convex polyhedra, some inner approximations are used to keep set representations simple. Crucially, correctness is preserved when inner approximations of the arguments to the Rch and Inv operators are used in (12).

In the rest of this paper, the parameters in Table I are used as nominal values. The parameters related to vehicle dynamics have been extracted from the "D-Class Sedan" model in CarSim. Using these parameters and target sets $\widetilde{G}_1$ and $\widetilde{G}_2$, the iterations (12) converge in one step and the resulting sets are illustrated in Fig. 3. The computations took 1 minute and 50 seconds on a 3.4 GHz iMac.

The final controller consists of a collection of convex polyhedra (i.e., linear inequalities) and user defined weights. At each time sample, this collection is queried to construct a set of linear inequality constraints that are fed into a quadratic program (QP) that solves for the optimal input. This controller can be implemented on any embedded platform that is equipped with a quadratic programming solver [30], [31].

E. Changing environment assumptions

The assumptions on environment (in this case lead car) behavior are crucial in formal synthesis methods. We demonstrate this by considering more "aggressive" assumptions on lead car acceleration capabilities and maximal speed. Using the modified parameters in Table II, we obtain a smaller controller domain which reflects the additional caution that is required when following a lead car that might decelerate quickly, as illustrated in Fig. 4.

IV. Solution by PESSOA

In this section we explain how we synthesize a control strategy which satisfies the LTL specification (5) using the MATLAB toolbox PESSOA [32]. First, we explain how PESSOA constructs a discrete abstraction of the system, and

TABLE II
Modified Parameter Values

v_L^{min}	0 m/s	v_L^{max}	35 m/s
a_L^{min}	-3 m/s^2	a_L^{max}	2 m/s^2

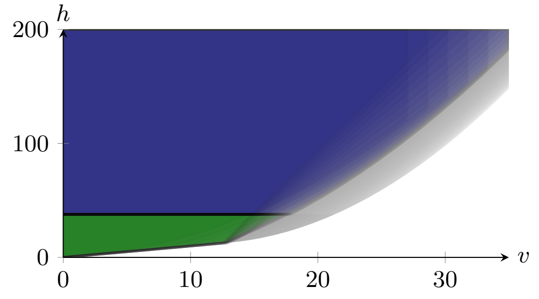


Fig. 4. More aggressive environment assumptions (see Table II) lead to a smaller controller domain. The figure shows cross sections at $v_L = 10$ m/s of sets C_1 (blue) and C_2 (green) which together form the controller domain. The gray part is the cross section of the set in Fig. 3.

then present the PESSOA algorithms that solve the synthesis problem on the abstraction.

A. Construction of a discrete abstraction

To obtain a discrete abstraction of the system given in Fig. 1, we follow the approach in [33] which is based on the discretization of the state space and the input space. Note that, this abstraction should be constructed in a way that enables us to refine a controller that is synthesized for the abstraction to a controller enforcing the same specification on the original system. It is shown in [34] that the existence of an ϵ-approximate alternating simulation relation from the abstraction to the original system guarantees that any controller synthesized for the abstraction can be refined to a controller enforcing the specification on the original system up to an error of ϵ. Further details on ϵ-approximate alternating simulation relations and controller refinements are provided in [35].

A *finite transition system* is a tuple $\Sigma = (Q, U, \delta)$ consisting of:

- A finite set Q of states;
- A finite set U of inputs;
- A *transition function* $\delta : Q \times U \to 2^Q$.

For a given set $Z \subseteq \mathbb{R}^n$ and discretization parameter $m > 0$, we define:

$$[Z]_m = \{x \in Z | \exists k \in \mathbb{Z}^n : x = km\},$$

to denote a uniform grid on Z. Consider a set of discretization parameters η, μ, and τ, which correspond to the state space discretization, the input space discretization, and the time discretization, respectively. We obtain the abstraction Σ by constructing the finite transition system (Q, U, δ), where:

- $Q = [\mathcal{V} \times \mathcal{H} \times \mathcal{V}_L]_\eta$;
- $U = [S_U]_\mu$;
- $q' \in \delta(q, u)$ if

$$\|\xi_{(q,u)}(\tau) - q'\| \leq \mathcal{R}_q(u, \tau) + \frac{\eta}{2},$$

where $\mathcal{R}_q(u, \tau)$ is an over-approximation of the reachable states from state $q \in Q$, when input $u \in U$ is applied for τ seconds. We compute $\mathcal{R}_q(u, \tau)$ using the Lipschitz constant of the dynamics given in Fig. 1. Note that the sets Q and U are uniform grids on compact sets and hence they are finite.

By construction, Σ is $\epsilon-$approximately alternatingly simulated by the original system, for any ϵ satisfying $\epsilon \geq \frac{\eta}{2}$. It can be shown that, for the specification given by (5), by shrinking the sets M_1, M_2, G_1, G_2, S_1, and S_2 by ϵ, we can guarantee that the refined controller will enforce the specification (5) on the original system in Fig. 1. We then compute an under-approximation of these shrunk sets by finding the grid points that lie inside these regions. We denote the corresponding sets in Q as $M_{1,a}$, $M_{2,a}$, $G_{1,a}$, $G_{2,a}$, $S_{1,a}$, and $S_{2,a}$.

B. Synthesis algorithms

The transition system Σ is nondeterministic. This means that for a given state q and input u, there might be multiple successor states in Σ, i.e., $|\delta(q,u)| \geq 1$. There are three sources of this nondeterminism in Σ. The first is the conservativeness in the reachable set estimates. The second is due to the fact that the lead car's acceleration is unknown. The last one results from the nondeterministic change of the states in the hybrid system model. Recall that, since the lead car behaves nondeterministically, the switching between the `no lead car` state and the `lead car` state can occur at any point in time.

For the rest of the discussion, $[\![\varphi]\!]$ denotes the states in Σ for which there exists a control strategy that enforces φ no matter what the successor state is in each transition of Σ. Note that, when $[\![\psi]\!]$ is obtained, a control strategy on Σ can be constructed by using the intermediate computations which led to $[\![\psi]\!]$. Thus, the controller domain $\mathcal{D} \doteq [\![\psi]\!]$ together with such a control strategy constitute a solution to Problem 1.

The specification given in ψ is equivalent to

$$\psi = \Box S_U \wedge \bigwedge_{i=1}^{2} \begin{pmatrix} \Box (M_i \implies S_i) \\ \wedge (\Box\Diamond\neg M_i \vee \Diamond\Box G_i) \end{pmatrix},$$

which can be rewritten in the following form:

$$\psi = \Box S_U \wedge \bigwedge_{i=1}^{2} \begin{pmatrix} \Box (M_i \implies S_i) \\ \wedge (\Diamond\Box M_i \implies \Diamond\Box G_i) \end{pmatrix}.$$

Note that we accounted for the input constraint $\Box S_U$ by defining $U = [S_U]_\mu$ in Σ. PESSOA handles the rest of the formula by first synthesizing a controller for the safety part of the specification, i.e., $\Box \wedge_{i=1}^{2} (M_i \implies S_i)$. It computes $[\![\Box \wedge_{i=1}^{2} (M_{i,a} \implies S_{i,a})]\!]$, using Algorithm 1. In this algorithm,

$$Pre_\Sigma(Q') = \{q \in Q | \exists u \in U \,:\, \delta(q,u) \in Q'\}$$

denotes the controllable predecessors of a set of states. The operator Pre_Σ is very similar to the operator Pre_τ^Ξ defined in equation (10). The main difference is that Pre_Σ is based on a discrete abstraction where as Pre_τ^Ξ is based on the continuous dynamics.

Algorithm 1: Computation of $[\![\Box K]\!]$.

```
$\tilde{X} := Q, \;\; X := \emptyset$
**while** $\tilde{X} \neq X$ **do**
    $X := \tilde{X}, \;\; \tilde{X} := Pre_\Sigma(X) \cap K$
**return** $X$
```

From the set $[\![\Box \wedge_{i=1}^{2} (M_{i,a} \implies S_{i,a})]\!]$ PESSOA constructs a set-valued controller that provides all possible inputs that can be applied at each state of Σ, while enforcing the safety specification. By composing the plant with the set valued controller enforcing the safety specification we obtain a new plant (finite transition system) that is guaranteed to satisfy the safety specification. The next step is then to synthesize a controller for this new plant that enforces the remaining part of the specification. The formula $\wedge_{i=1}^{2} (\Diamond\Box M_i \implies \Diamond\Box G_i)$, has the form of a *mode-target formula* which was introduced in [36], [37]. To construct the set

$$[\![\wedge_{i=1}^{2} (\Diamond\Box M_{i,a} \implies \Diamond\Box G_{i,a})]\!], \tag{14}$$

PESSOA uses Algorithm 2. Since the proof of correctness of this algorithm is out of the scope of this paper we refer the interested reader to [36] and only provide an intuitive explanation here.

Algorithm 2: Computation of (14).

```
$X' := Q$, $X := \emptyset$, $Y' := \emptyset$
$Y := Q$, $Z' := V$, $Z := \emptyset$
**while** $Z' \neq Z$ **do**
    $Z := Z'$
    **for all** $i \in \{1,2\}$ **do**
        **while** $Y' \neq Y$ **do**
            $Y := Y'$
            **while** $X' \neq X$ **do**
                $X := X'$
                $X' := Pre(X) \cap [\![M_{i,a} \wedge G_{i,a}]\!]$
                $X' := X' \cup ([\![\neg M_{i,a}]\!] \cap Pre(Z))$
                $X' := X' \cup Pre(Y)$
            $Y' := X'$, $X' = Q$
        $Y_i = Y'$, $Y' = \emptyset$
    $Z' = \bigcap_{i \in I} Y_i$
**return** $Z$
```

Algorithm 2 performs a triple nested fixed-point computation. The inner double fixed-point computation gives the set of states from which the controller can enforce the specification $\Diamond\Box(M_i \wedge G_i) \vee \Diamond\neg M_i$. This is the set of states from which the controller can force the system to either eventually settle down in a set of states are in the target region G_i for the current mode M_i, i.e., $\Diamond\Box(M_i \wedge G_i)$, or change its mode at some time in the future, i.e., $\Diamond\neg M_i$. The outermost loop, i.e., the largest fixed-point over Z, makes sure that once the system switches to another mode it still ends up in a state that can either reach and stay in the corresponding target region or go through another mode change.

Note that the convergence of all the fixed-point algorithms presented in this section is guaranteed due to the finiteness of Q [36].

C. Solving the ACC problem

We synthesized a controller for the ACC problem using PESSOA. For a desired precision $\epsilon = 0.5$, the state space and input space discretization parameters were chosen to be $\eta = 0.5$ and $\mu = 0.3$, respectively. We used $\tau = 0.5$ for the

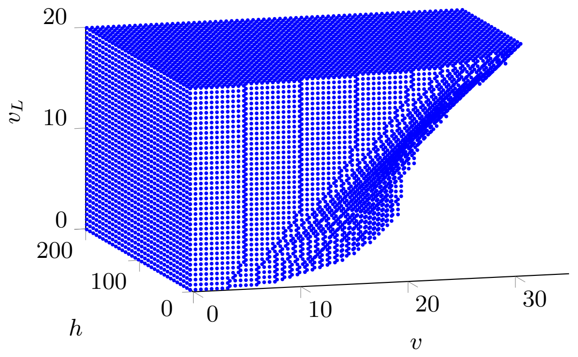


Fig. 5. Domain $\mathcal{D}\dot{=}[\![\varphi]\!]$ of the controller synthesized by PESSOA using $\widetilde{G}_1$, $\widetilde{G}_2$.

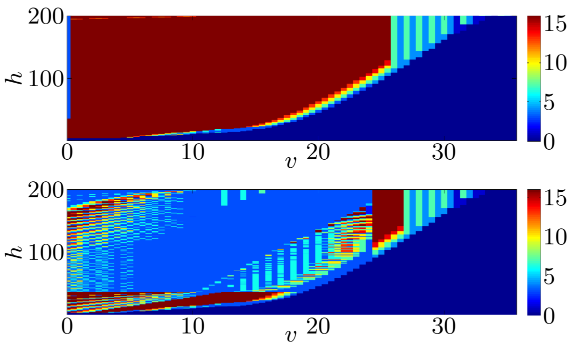


Fig. 6. Different interpretations of target sets lead to different controllers as can be appreciated by considering the number of available inputs in each state. The plots show the number of valid inputs at the cross section $v_L = 10$ using $\widetilde{G}_1$, $\widetilde{G}_2$ (top) and $\bar{G}_1$, $\bar{G}_2$ (bottom).

sampling time and the same problem parameter values as in Table I. We ran Algorithm 1 followed by Algorithm 2 to obtain a controller that enforces the formula ψ given in (5) on the abstraction Σ using $\widetilde{G}_1$, $\widetilde{G}_2$. The abstraction was computed in 70 hours while algorithms 1 and 2 terminated after 5 minutes and 18 hours, respectively. All computations were done on a 3.4 GHz iMac with 32GB of RAM. Fig. 5 illustrates the domain of the controller synthesized in PESSOA.

The controller synthesized by PESSOA is given in the form of an ordered binary decision diagram (OBDD), a memory efficient data structure. We query the OBDD at each sampling time to get the set of valid inputs which enforces the specification on the closed-loop system. Hence, implementation of the controller on an embedded device is straightforward.

D. Smaller target sets

In Section III-E we showed how assumptions on the environment affected the synthesized controller domain. We now explore how the modified target sets $\bar{G}_1$ and $\bar{G}_2$ result in different controllers. Fig. 6 shows the number of available inputs when using the target sets $\widetilde{G}_1$ and $\widetilde{G}_2$ versus the target sets $\bar{G}_1$ and $\bar{G}_2$. Here, we illustrate that the synthesized controller for the target sets $\widetilde{G}_1$ and $\widetilde{G}_2$ is less restrictive in terms of the number of valid inputs in most, but not all, of the states in the domain. This is because the sets that result from the two different interpretations of the specification are not comparable.

V. Results

A. Evaluation on high-order model

We simulated both controllers in Simulink and CarSim for the following scenario: At time $t = 0$s, a lead car is present driving below the desired speed $v^{des} = 25$m/s of the ACC car, but leaves the lane at $t = 3$s, allowing the ACC car to reach and attain its desired speed[4]. At $t = 13$s a new lead car cuts in 30m in front of the ACC car and starts decelerating. The ACC car is forced to slow down in order to increase the time headway. Fig. 7 shows how both controllers perform. Notably, all constraints, which are indicated by green lines, are satisfied throughout the simulations.

[4]With $\tilde{G}_1$, desired speed attainment is not guaranteed by construction, but is assured in steady state by suitable choices of MPC weights.

B. Supervision of a legacy controller

Next, we demonstrate how synthesized ACC software can be used to guarantee correctness of a black-box legacy controller Ψ by acting as a supervisor. In order to retain as much of the original performance as possible, the supervision is done in a *minimally intrusive way* and we focus just on satisfying the safety specification

$$\Box S_U \wedge \Box(M_2 \implies S_2). \tag{15}$$

Both synthesis methods provide a set-valued mapping P that maps the system state x to a set of valid inputs. For PCIS, $P(x)$ is given in terms of linear inequalities that the input must satisfy, while PESSOA provides $P(x)$ as a finite set. Given a synthesized function P, for an arbitrary control system $\dot{x} = g(x, u)$, we use the supervisory scheme

$$F_w = \begin{cases} \min\limits_u \|g(x,u) - g(x, \Psi(t,x))\|, \\ \text{s.t. } u \in P(x), \end{cases}$$

that at each sample instant selects the safe input $u \in P(x)$ that minimizes the difference in system behavior (by minimizing difference between the resulting vector fields). Fig. 8 shows the effect of PCIS and PESSOA supervision of the naive proportional ACC controller Ψ given as

$$(v, h) \mapsto F_r(v) - k(v - \min(v^{des}, h/\omega_{des})). \tag{16}$$

Without supervision and with a gain $k = 500$, the controller violates both the input bound as well as the safety specification in (15). However, the supervisory framework is able to correct this behavior during the critical phase, depicted in Fig. 8 with grayed areas, and then hand back control to Ψ. In Fig. 9 it is shown how PCIS-based supervision forces the state to remain within the safe set.

VI. Hardware implementation

The simulations in the previous section exhibited good performance of both formally synthesized controllers when implemented on a high-dimensional industry standard model. In particular, the trajectories of the complex CarSim dynamics and those of the simplified model (1) tracked each other

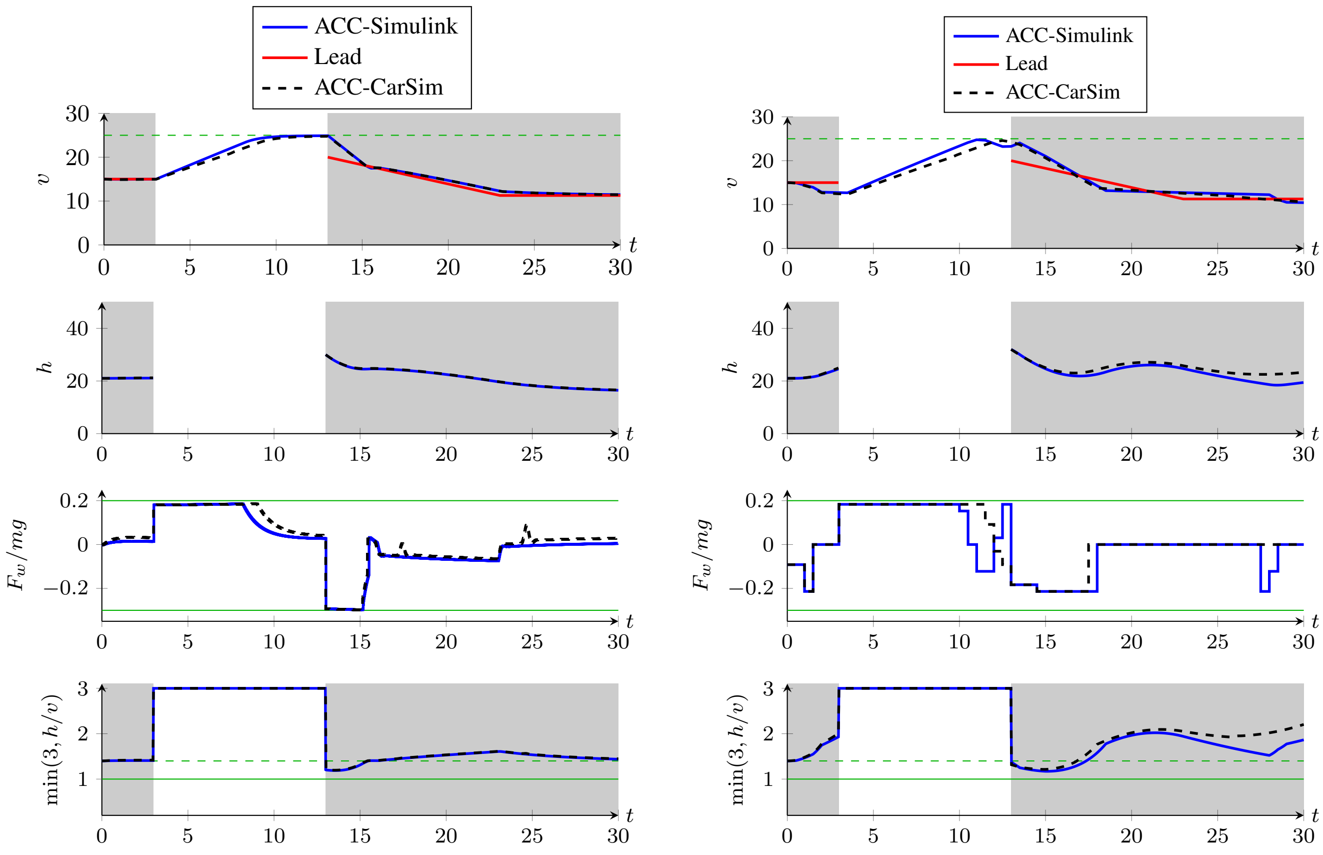


Fig. 7. Simulations in Simulink and in CarSim of the PCIS (left) and PESSOA (right) controllers. The plots show, from top to bottom, velocities, headway, applied control input, and time headway. Grayed areas indicate that the system is in specification mode M_2. Dashed green lines indicate target sets, solid green indicate safety sets.

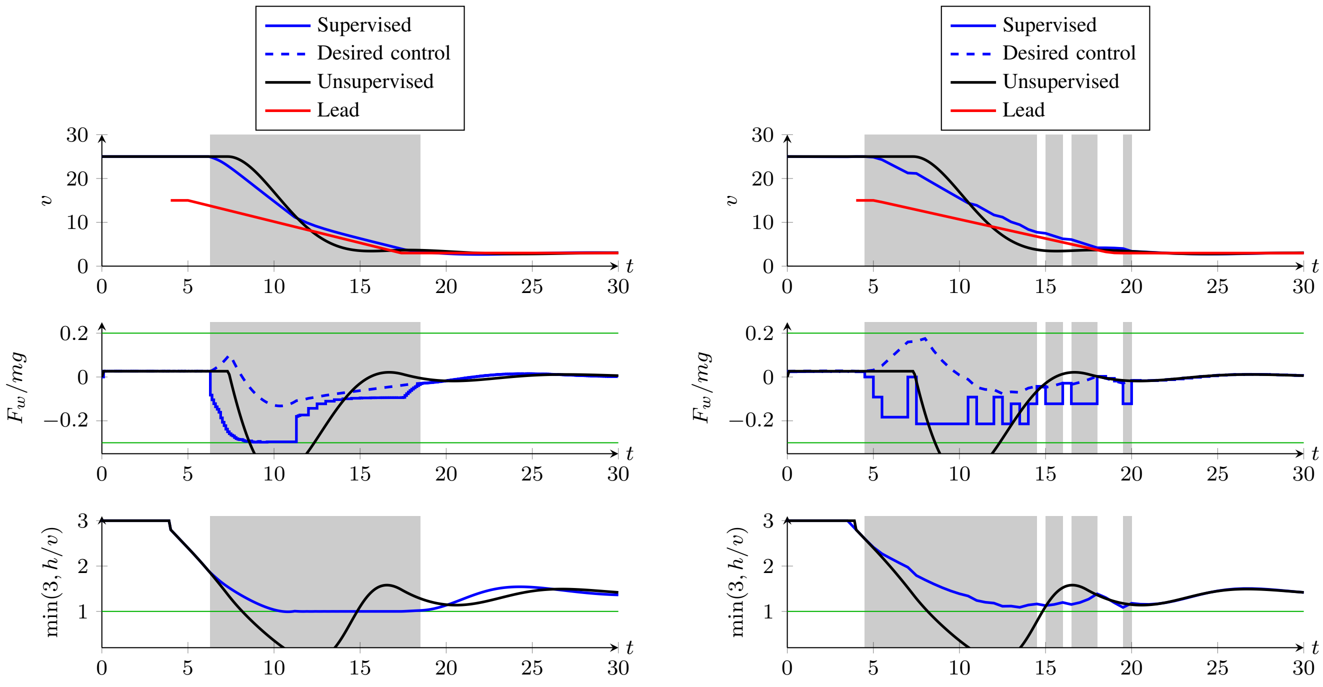


Fig. 8. Supervision of a legacy controller Ψ using PCIS (left) and PESSOA (right) are shown in blue. The plots show velocities, applied control input, and time headway. For comparison, the corresponding unsupervised trajectory when using Ψ is plotted in black. The action of Ψ is overridden in the grayed areas in order to guarantee satisfaction of (15), which is indicated with green lines. The dashed blue line shows the value of the desired (but overridden) control (16) during supervision phases. Since the system states change as a result of supervision, the desired control during supervision is different from the control that is applied in the unsupervised case. For PESSOA, starting from $t = 14$s the framework hands control back to Ψ. This is immediately followed by PESSOA supervision to prevent violation of the requirement on the time headway. Switching between PESSOA and Ψ stops at $t = 20$s.

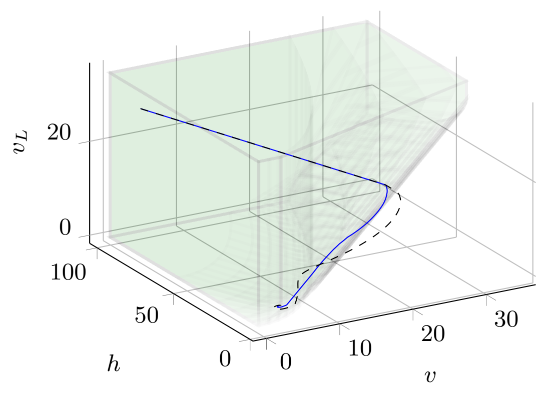


Fig. 9. The supervised trajectory (blue) is forced to stay in safe set, whereas the unsupervised trajectory (dashed black) exits.

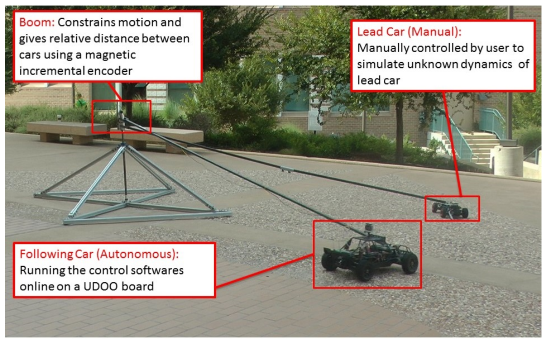

Boom: Constrains motion and gives relative distance between cars using a magnetic incremental encoder
Lead Car (Manual): Manually controlled by user to simulate unknown dynamics of lead car
Following Car (Autonomous): Running the control softwares online on a UDOO board

Fig. 10. Hardware testbed

closely when subject to input provided by the synthesized controllers.

To also demonstrate practical feasibility of the proposed controllers, we implemented them in real hardware and conducted several experiments. Our testbed, shown in Fig. 10, consists of a remote controlled car connected to a UDOO board that gives velocity commands. On the UDOO board, the controllers were executed at 100 Hz, which demonstrates real-time feasibility of the proposed controllers. The same testbed was used in [38], where further details about the hardware are presented. A video is also available at http://youtu.be/i1w6BWMnTDE.

VII. Discussion and conclusion

In this paper we formalized the ACC problem using a hybrid dynamical system model and an LTL specification. Then, we presented two solution approaches to synthesize correct-by-construction control software for ACC. Both approaches rely on fixed-point-based computation of a controller domain from where the LTL specification can be enforced, one computing such fixed-points directly on the continuous state-space, the other on a finite-state abstraction of the nonlinear dynamics. Each approach has certain advantages and disadvantages: (i) Termination of fixed-point-based algorithms on continuous state-spaces is not guaranteed whereas termination is always guaranteed when working with finite-state abstractions. (ii) Approximate finite-state abstractions can be computed directly for nonlinear dynamics, whereas the current implementation of the PCIS approach requires linearization of the dynamics and error bounds between the actual model and the linearized model. (iii) Changes in the system model parameters require recomputing the finite-state abstractions in PESSOA, which is the main computational bottleneck. Since PCIS does not require computations of abstractions, overall computation time for a controller is very short and is independent of different model parameters. On the other hand, PESSOA can handle arbitrary sets describing the atomic propositions in the specification; while intermediate steps in PCIS become computationally challenging when the atomic propositions correspond to non-convex sets. (iv) It is currently not possible to handle quantitative objectives in PESSOA, whereas the invariant-set based approach readily gives the designer the flexibility to shape the transient behavior and encode "soft" constraints through optimization. In this work, tracking objectives (i.e., velocity lower bounds in goal sets) were encoded as soft constraints in PCIS.

We made some simplifying assumptions with regard to the lead car behavior (A.1) and the number of cars within the radar range (A.2). The assumptions in (A.1) cover a wide variety of driving behaviors, therefore we believe they are not particularly restrictive. Moreover, as demonstrated using PCIS, it is possible to synthesize different controllers to see the effects of the assumptions on the lead car behavior. On the other hand, the assumption (A.2) is rather technical and was adopted to avoid the pathological cases where lead cars cut into the lane infinitely often in a way to prevent maintaining the desired time headway while always staying in mode M_2. The synthesized controllers are still correct when this assumption is relaxed to either (i) $R_{2,2}(v,h,v_L) = \mathcal{D}$ for all $(v,h,v_L) \in \mathcal{V} \times \mathcal{H} \times \mathcal{V}_L$ and there are finitely many resets, or (ii) $R_{2,2}(v,h,v_L) = G_2$ for all $(v,h,v_L) \in \mathcal{V} \times \mathcal{H} \times \mathcal{V}_L$. The other way to eliminate this assumption is to properly adjust the LTL formula (5) so that reaching and maintaining a time headway is required only until the next reset.

For both approaches, we used discrete-time semantics of LTL. This may lead to "small" safety violations in between sampling times. However for both approaches, it is possible to guarantee correctness with respect to continuous-time semantics of LTL by properly shrinking the goal sets and safe sets [39], [40]. Similarly, for simplicity, we ignored delays in the measurements and possible sensor noise, which can be incorporated into both approaches using ideas from [40].

Finally, it is worth noting that textual specifications are often ambiguous, leading to multiple interpretations. Formalizing specifications using temporal logics can help resolve such ambiguity as it requires precise mathematical descriptions of desired behavior and assumptions. We presented multiple interpretations of the ACC specification and demonstrated, using PESSOA, how the solution changes with different interpretations of the specification.

It can happen that an ACC system cannot maintain safety within specified limits on braking (e.g., (2)), such as in the event of a car cutting dangerously close into a lane. In this case, the ACC system is expected to alert the driver through automated emergency braking and/or other feedback cues, and hand control of the vehicle back to the driver. Another advantage of the presented correct-by-construction controllers is that the safe set (i.e., control domain $\mathcal{D}$) is explicitly computed and thus conditions for passing control to an emergency braking module are clearly defined.

REFERENCES

[1] P. Nilsson, O. Hussien, Y. Chen, A. Balkan, M. Rungger, A. D. Ames, J. W. Grizzle, N. Ozay, H. Peng, and P. Tabuada, "Preliminary results on correct-by-construction control software synthesis for adaptive cruise control," in *Proc. of IEEE CDC*, 2014, pp. 816 – 823.

[2] ISO 15622:2010 (E), "Intelligent transport systems – adaptive cruise control systems – performance requirements and test procedures," International Organization for Standardization, Tech. Rep., 2010.

[3] SAE J2399, "Adaptive cruise control (ACC) human factors: Operating characteristics and user interface," SAE, Tech. Rep., 2003.

[4] P. Fancher, H. Peng, Z. Bareket, C. Assaf, and R. Ervin, "Evaluating the influences of adaptive cruise control systems on the longitudinal dynamics of strings of highway vehicles," *Vehicle Syst. Dyn.*, vol. 37, pp. 125–136, 2003.

[5] DOT HS 808 849, "Intelligent cruise control field operational test," US DOT, Tech. Rep., 1998.

[6] J. Zhou and H. Peng, "String stability conditions of adaptive cruise control algorithms," in *IFAC Symp. on Advances in Automotive Control*, 2004.

[7] P. A. Ioannou and C.-C. Chien, "Autonomous intelligent cruise control," *IEEE Trans. Veh. Tehnol.*, vol. 42, no. 4, pp. 657–672, 1993.

[8] B. Van Arem, C. J. van Driel, and R. Visser, "The impact of cooperative adaptive cruise control on traffic-flow characteristics," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 4, pp. 429–436, 2006.

[9] C.-Y. Liang and H. Peng, "Optimal adaptive cruise control with guaranteed string stability," *Vehicle Syst. Dyn.*, vol. 32, no. 4-5, pp. 313–330, 1999.

[10] J. Zhou and H. Peng, "Range policy of adaptive cruise control vehicles for improved flow stability and string stability," *IEEE Trans. Intell. Transp. Syst.*, vol. 6, no. 2, pp. 229–237, 2005.

[11] X. Lu and J. Hedrick, "Practical string stability," in *IAVSD Symp. on Dynamics of Vehicles on Roads and Tracks*, 2003, pp. 25–29.

[12] Y. Yamamura, Y. Seto, H. Nishira, and T. Kawabe, "An ACC design method for achieving both string stability and ride comfort," *J. System Design and Dynamics*, vol. 2, pp. 979–990, 2008.

[13] A. Kesting, M. Treiber, M. Schönhof, and D. Helbing, "Adaptive cruise control design for active congestion avoidance," *Transpn Res.-C: Emerging Technologies*, vol. 16, no. 6, pp. 668–683, 2008.

[14] S. Li, K. Li, R. Rajamani, and J. Wang, "Model predictive multi-objective vehicular adaptive cruise control," *IEEE Trans. Control Syst. Technol.*, vol. 19, no. 3, pp. 556–566, 2011.

[15] S. Moon, I. Moon, and K. Yi, "Design, tuning, and evaluation of a full-range adaptive cruise control system with collision avoidance," *Control Eng. Pract.*, vol. 17, no. 4, pp. 442–455, 2009.

[16] A. Vahidi and A. Eskandarian, "Research advances in intelligent collision avoidance and adaptive cruise control," *IEEE Trans. Intell. Transp. Syst.*, vol. 4, no. 3, pp. 143–153, 2003.

[17] G. Naus, J. Ploeg, M. Van de Molengraft, W. Heemels, and M. Steinbuch, "Design and implementation of parameterized adaptive cruise control: An explicit model predictive control approach," *Control Eng. Pract.*, vol. 18, no. 8, pp. 882–892, 2010.

[18] M. Althoff, D. Althoff, D. Wollherr, and M. Buss, "Safety verification of autonomous vehicles for coordinated evasive maneuvers," in *IEEE Intelligent Vehicles Symposium*, 2010, pp. 1078–1083.

[19] O. Stursberg, A. Fehnker, Z. Han, and B. H. Krogh, "Verification of a cruise control system using counterexample-guided search," *Control Eng. Pract.*, vol. 12, no. 10, pp. 1269–1278, 2004.

[20] S. M. Loos, A. Platzer, and L. Nistor, "Adaptive cruise control: Hybrid, distributed, and now formally verified," in *FM 2011: Formal Methods*. Springer, 2011, pp. 42–56.

[21] M. Asplund, A. Manzoor, M. Bouroche, S. Clarke, and V. Cahill, "A formal approach to autonomous vehicle coordination," in *FM 2012: Formal Methods*. Springer, 2012, pp. 52–67.

[22] M. R. Hafner and D. Del Vecchio, "Computational tools for the safety control of a class of piecewise continuous systems with imperfect information on a partial order," *SIAM J. Control Optim.*, vol. 49, no. 6, pp. 2463–2493, 2011.

[23] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata?" in *Proc. of the ACM Symp. on Theory of Computing*, 1995, pp. 373–382.

[24] F. Borrelli, A. Bemporad, and M. Morari, *Model Predictive Control*, 2015.

[25] M. Herceg, M. Kvasnica, C. N. Jones, and M. Morari, "Multi-Parametric Toolbox 3.0," in *Proc. of the ECC*, 2013.

[26] D. Bertsekas, "Infinite time reachability of state-space regions by using feedback control," *IEEE Trans. Autom. Control*, vol. 17, no. 5, pp. 604–613, 1972.

[27] E. De Santis, M. D. Di Benedetto, and L. Berardi, "Computation of Maximal Safe Sets for Switching Systems," *Automatic Control, IEEE Transactions on*, vol. 49, no. 2, pp. 184–195, 2004.

[28] E. G. Gilbert and K. T. Tan, "Linear systems with state and control constraints: the theory and application of maximal output admissible sets," *IEEE Trans. Autom. Control*, vol. 36, no. 9, pp. 1008–1020, 1991.

[29] A. Tarski, "A lattice-theoretical fixpoint theorem and its applications," *Pacific J. Math*, vol. 5, pp. 285–309, 1955.

[30] J. Mattingley and S. Boyd, "Cvxgen: a code generator for embedded convex optimization," *Optimization and Engineering*, vol. 13, no. 1, pp. 1–27, 2012.

[31] K. Galloway, K. Sreenath, A. Ames, and J. Grizzle, "Torque saturation in bipedal robotic walking through control lyapunov function-based quadratic programs," *IEEE Access*, vol. 3, pp. 323–332, 2015.

[32] M. Mazo Jr, A. Davitian, and P. Tabuada, "Pessoa: A tool for embedded controller synthesis," in *Computer Aided Verification*. Springer, 2010, pp. 566–569.

[33] M. Zamani, G. Pola, M. Mazo, and P. Tabuada, "Symbolic models for nonlinear control systems without stability assumptions," *IEEE Trans. Autom. Control*, vol. 57, no. 7, pp. 1804–1809, 2012.

[34] G. Pola and P. Tabuada, "Symbolic models for nonlinear control systems: Alternating approximate bisimulations," *SIAM J. Control Optim.*, vol. 48, no. 2, pp. 719–733, 2009.

[35] P. Tabuada, *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer, 2009.

[36] A. Balkan, M. Vardi, and P. Tabuada, "Controller Synthesis for Mode-Target Games," arXiv:1504.07702v1 [math.OC], 2015.

[37] A. Balkan, M. Vardi, and P. Tabuada, "Controller Synthesis for Mode-Target Games," in *Proc. of ADHS*, 2015, pp. 342–350.

[38] A. Mehra, W.-L. Ma, F. Berg, P. Tabuada, J. Grizzle, and A. Ames, "Adaptive cruise control: Experimental validation of advanced controllers on scale-model cars," in *Proc. of ACC*, 2015, pp. 1411–1418.

[39] A. Girard, "Reachability of Uncertain Linear Systems Using Zonotopes," in *Proc. of HSCC*, 2005, pp. 291–305.

[40] J. Liu and N. Ozay, "Abstraction, discretization, and robustness in temporal logic control of dynamical systems," in *Proc. of HSCC*, 2014, pp. 293–302.