This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2953213, IEEE Transactions on Automatic Control

Abstraction-based Safety Verification and Control of Cooperative Vehicles at Road Intersections

Heejin Ahn and Alessandro Colombo

***Abstract*—This paper considers the problem of designing a centralized controller for vehicle collision avoidance at road junctions and intersections. The controller supervises a set of vehicles, and overrides their inputs when necessary to prevent side and rear-end collisions. By supervising vehicles, rather than taking full control, we obtain a system that can work with semiautomated human-driven vehicles. The price to pay is in complexity: an override is only necessary if, without an intervention, all future input signals will result in a collision. Thus, deciding overrides requires verification of the full reachability set, rather than the computation of a single collision-free trajectory. Our approach to speeding this step up is to use an abstraction of the (concrete) system, which is suitably discretized to obtain a mixed-integer programming problem. We deduce the solution of the original verification problem from that of the abstraction-based verification problem by proving an approximate simulation relation between the abstract and concrete systems. The resulting supervisor provably guarantees safety of the concrete system. We also evaluate the approximation error of the supervisor due to the use of an abstraction. Computer simulations show that the supervisor exhibits computationally better performances than other existing controllers applicable to realistic intersection scenarios.**

***Index Terms*—safety verification; abstraction; collision avoidance; road intersection**

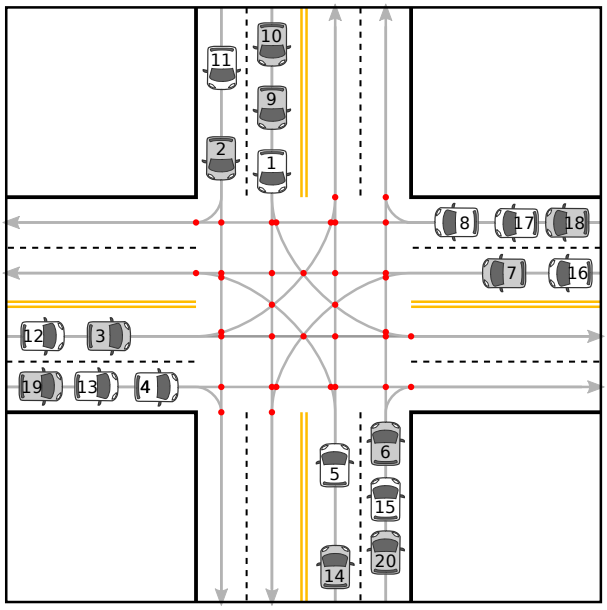


Fig. 1. Example of a large intersection.

I. Introduction

CONTROL design of cooperative and connected vehicles is receiving increasing attention, as a larger number of highly automated vehicles appear on roads [1]. In particular, much attention was recently devoted to the design of intersection management strategies for fully automated vehicles [2]–[8]. However, despite the speed at which vehicular technologies are evolving, human drivers are likely to remain a key factor in regular city traffic for years. In the shorter term, therefore, strategies that can reduce the chances of crashes at intersections, while leaving substantial control to the driver, constitute a more likely evolution of the existing advanced driver-assistance systems.

This paper presents the design of a centralized coordinator at busy road intersections (e.g., Fig. 1) that takes control of vehicles only when it detects unsafe inputs, with focus on computational complexity and how to manage it. We refer to such a coordinator as a *supervisor* [9], and to the task of detecting unsafe inputs as *safety verification*.

Several recent studies have focused on the design of supervisors at road intersections [10]–[18]. Mostly, the cited studies are based on restrictive assumptions of intersection complexity or vehicle dynamics. Earlier proposed methods [10]–[14] define a single conflict area within the intersection, such as the area enclosed by the dashed curve in Fig. 2, and avoid side collisions by allowing only one vehicle at a time in the area. With this simplified model, safety verification is translated into a relatively tractable single-machine scheduling problem. However, the resulting supervisor behaves conservatively in that, for example in Fig. 2, it prevents cars 1 and 4 from simultaneously being inside the intersection although their collision is geometrically impossible. More recent methods [15]–[18] handle a more realistic intersection model with multiple conflict areas, such as the shaded areas in Fig. 2, but fixed vehicles' paths. The works [15], [16] employ jobshop scheduling and present a computationally efficient approach by assuming simplified vehicle dynamics, with the primary focus on the avoidance of side collisions, ignoring rear-end collisions. The works [17], [18] solve a time-discretized version of the safety verification problem, considering both side and rear-end collision avoidance, under the assumption of linear vehicle dynamics.

The strategy we propose in this paper pushes the limits on the number of vehicles that can be simultaneously handled at an intersection, while relaxing the linearity assumptions on vehicle dynamics, handling side and rear-end collisions,

This work was done while Heejin Ahn was a Ph.D. Candidate at the Massachusetts Institute of Technology. (e-mail: heejin.ahn@alum.mit.edu)

Alessandro Colombo is with Department of Electronics, Information, and Bioengineering, Politecnico di Milano, Via Ponzio 34/5, 20133 Milano, Italy. (e-mail: alessandro.colombo@polimi.it)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2953213, IEEE Transactions on Automatic Control

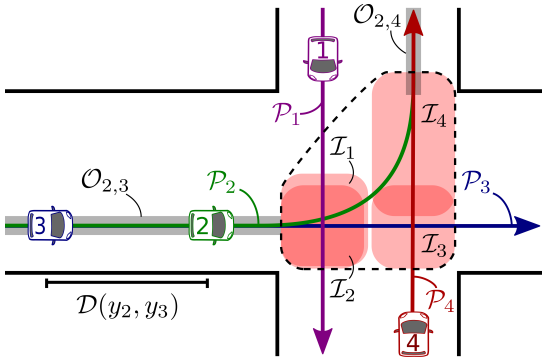
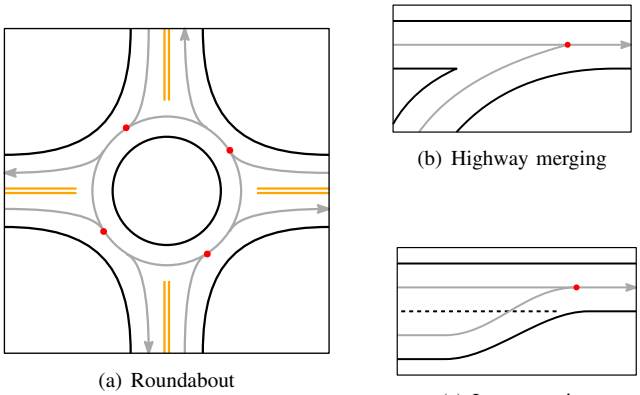


Fig. 2. Intersection with a set of distinct conflict areas (red-shaded regions for side conflict and gray-shaded segments for rear-end conflict). The notation defining the conflict areas is introduced in Section II.



(a) Roundabout

(b) Highway merging

(c) Lane merging

Fig. 3. These scenarios are classified as the same framework as the intersection depicted in Fig. 1.

and allowing to treat cases where vehicles' future paths are uncertain. To achieve this, our strategy is to solve the safety verification problem based on a simpler abstraction of the concrete vehicle model, and then use the relation between abstract and concrete models to synthesize overrides. In the following, we detail two peculiarities of our strategy.

First, we adopt a *space discretization* scheme to rewrite the abstraction-based verification problem as a mixed-integer programming (MIP) problem. The work [7] considered a space discretization scheme for coordinating vehicles through an intersection. However, the formulation was designed for fully automated intersection management with an assumption of a constant speed inside the intersection, and thus is not applicable to the safety verification of the unpredicted inputs generated by a human driver. A rather commonly used scheme is to base the coordination problem on a discrete-time model of the vehicles [17]–[20]. While formally correct, the solutions are computationally demanding due to the structure of ensuing optimization problems: we will explain this issue in detail in Section VII-C.

Second, to prove that the resulting supervisor guarantees safety of the concrete system, we define an *approximate simulation relation* between abstract and concrete systems [21], [22]. Based on the solution of the abstraction-based verification problem, which in turn implies the solution of the original verification problem, we design the supervisor. The approximate simulation relation implies that the supervisor prevents any side and rear-end collisions among vehicles, but in the meantime, results in a restrictive supervisor (i.e., one that is not minimal in term of number of interventions). We numerically estimate the approximation error of the supervisor. We also perform computational experiments to evaluate the performances of our approach, in terms of computation time and restrictiveness, and compare it with results in the literature.

Similar approaches to intersection management, which employ both discretization and abstraction, were presented in [23]–[25]. The work [23] solved verification based on abstraction in conjunction with the discretization of time and input, and exploited differential flatness to compute the solution of the original verification problem. The works [24], [25] constructed a discrete event system (DES) abstraction based on the discretization of time and space, and applied supervisory control theory of DES. However, the approaches [23], [24] targeted a simplified intersection scenario with a single conflict area, and the approaches [24], [25] considered a first-order vehicle dynamical model. Our approach can handle nonlinear second-order dynamics at a complicated intersection scenario with multiple conflict areas.

The rest of the paper is structured as follows. In Section II, we formally state the safety verification problem, following the introduction of vehicle dynamics and collision sets. In Section III, we outline our strategy to solve the problem based on a simpler abstraction of the concrete system. In Section IV, we rewrite the abstraction-based verification problem as a mixed-integer optimization problem via the discretization of vehicles' paths. In Section V, we prove the approximate simulation relation of the abstraction by the concrete system, and provide an algorithm that implements a supervisor. In Section VI, we estimate the approximation errors of the supervisor algorithm based on another abstraction of the concrete system. We present numerical simulation results in Section VII.

II. Problem Setup

We consider n vehicles moving on a planar road network near an intersection, a road junction, or a lane merging, such as in Figs. 1 and 3. Vehicles may be human-driven or autonomous, but we assume that all vehicles are equipped with control and communication modules that allow a central coordinator (the supervisor) to take control of their longitudinal dynamics, within a predefined *controlled region* around the intersection.

We assume that vehicles move through the controlled region approximately following a finite number of predetermined paths (light gray lines in Figs. 1 and 3). The approximation errors between predetermined and actual paths correspond to uncertainty in the dynamics. We can handle such uncertainty by for instance suitably enlarging conflict areas or robustifying the supervisor tracking controller responsible for generating the override input. We assume, at first, that the approximate path each vehicle will follow is known before it enters the controlled region. This can be a reasonable assumption, when the path is fixed by road regulation (e.g., when vehicles are on a lane that allows only right turn) or can be inferred through

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2953213, IEEE Transactions on Automatic Control

suitable algorithms [26]. The supervisor presented in this paper can however handle, with little additional complexity, multiple possible future paths of each vehicle. This extension is discussed in Section VII-D.

With the assumption of predetermined paths, the objective of this work is to design the supervisor that takes control of vehicles' longitudinal dynamics along the paths only when necessary to avoid an imminent rear-end or side collision. We now introduce the notation necessary to discuss the objective, and then, formulate the problem statement.

A. Vehicle Dynamics

We call $x_j \in X_j := [x_{j,\min}, x_{j,\max}] \subset \mathbb{R}$ the curvilinear coordinate of vehicle j's geometric center along its path, and $\dot{x}_j \in \dot{X}_j := [\dot{x}_{j,\min}, \dot{x}_{j,\max}]$ its velocity, with $\dot{x}_{j,\min} \geq 0$. The interval $[x_{j,\min}, x_{j,\max}]$ is the intersection of the vehicle's path with the controlled region, and the non-negative interval $[\dot{x}_{j,\min}, \dot{x}_{j,\max}]$ represents the set of velocities that the vehicle is allowed to take due to physical or legal limitations, assuming that reversing is not allowed.

With the state $s_j = (x_j, \dot{x}_j)^\intercal \in S_j := X_j \times \dot{X}_j$, we assume input-affine vehicle dynamics:

$$\Sigma_c : \left\{ \quad \dot{s}_j = \begin{bmatrix} \dot{x}_j \\ f(s_j) + b(s_j)\, u_j \end{bmatrix}, \forall j \in \{1, \ldots, n\} \right. \tag{1}$$

where $u_j \in U_j := [u_{j,\min}, u_{j,\max}]$ is the input, $f(s_j)$ and $b(s_j)$ are nonlinear functions of the state, and n is the number of vehicles inside the controlled region. We define the output function

$$h_j(s_j) := x_j.$$

We assume that (1) has a unique solution that continuously depends on the input signal, and that $-f(s_j)/b(s_j) \in (u_{j,\min}, u_{j,\max})$ for all $s_j \in S_j$ to ensure that any constant velocity in $\dot{X}_j$ is attainable. For example, in the simulations presented in Section VII, we use the dynamics

$$\ddot{x}_j = -c_1(\dot{x}_j)^2 + c_2 + c_3 u_j \tag{2}$$

with parameters $c_1 > 0, c_2$, and $c_3 > 0$. This can be written in the form of (1) with $f(s_j) = -c_1(\dot{x}_j)^2 + c_2$ and $b(s_j) = c_3$.

The function $s_j(t, u_j, s_j(0))$ denotes the state reached after time t according to the dynamics (1) with input signal $u_j \in \mathcal{U}_j$ starting from an initial state $s_j(0)$. Here, $\mathcal{U}_j$ is the set of piecewise continuous functions of time with a countable number of discontinuities whose values lie in the input space U_j. For notational simplicity, we write $s_j(t, u_j)$ if the initial state $s_j(0)$ is not relevant.

We call $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbf{X} \subset \mathbb{R}^n$ the vector of the positions of all vehicles in the controlled region, $\dot{\mathbf{x}} = (\dot{x}_1, \dot{x}_2, \ldots, \dot{x}_n) \in \dot{\mathbf{X}} = [\dot{\mathbf{x}}_{\min}, \dot{\mathbf{x}}_{\max}] \subset \mathbb{R}^n$ the vector of corresponding velocities, $\mathbf{s} = (s_1, s_2, \ldots, s_n) \in \mathbf{S} = \mathbf{X} \times \dot{\mathbf{X}}$ the vector of the states, $\mathbf{u} = (u_1, u_2, \ldots, u_n) \in \mathbf{U} \subset \mathbb{R}^n$ the vector of inputs returned by the supervisor, and $\mathbf{u}_d = (u_{d,1}, u_{d,2}, \ldots, u_{d,n}) \in \mathbf{U} \subset \mathbb{R}^n$ the vector of inputs issued by the drivers. We call $\mathcal{U} := \mathcal{U}_1 \times \ldots \times \mathcal{U}_n$ the space of the input *signals* (i.e., of the vector function of time), and let $\mathcal{U}_{[0,\tau]}$ be the set of signals $\mathcal{U}$ restricted to the time interval $[0, \tau]$.

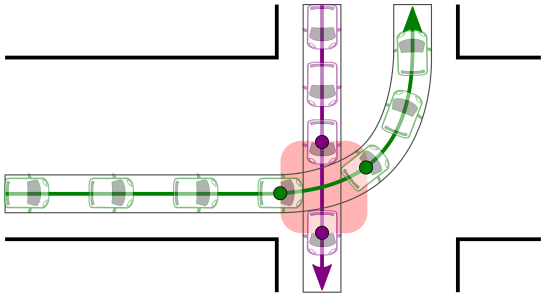


Fig. 4. Side conflict area $\mathcal{I}_1$ in Fig. 2 (red region) is determined such that side collisions occur when the geometric centers of the vehicles are simultaneously inside the area.

B. Bad Set

The supervisor is allowed to communicate with and take control of any vehicle within the controlled region. We assume that vehicles entering the region do not cause any immediate collision with other vehicles (this is achieved, for instance, by requiring that vehicles entering the controlled region be sufficiently distant from the vehicle in front and be able to fully stop before the intersection). This assumption enables the supervisor to guarantee that no conflict occurs as vehicles join or depart the region [2], [27]. We refer to a *bad set* as a set of position vectors that correspond to a rear-end or side collision between two vehicles in the controlled region.

To define the bad set B, let us call

$$\mathcal{P}_j(x_j) : \mathbb{R} \to \mathbb{R}^2,$$

the path followed by vehicle j inside the controlled region, and

$$y_j := \mathcal{P}_j(x_j)$$

the planar coordinate corresponding to x_j. Let $\mathbf{y} := (y_1, y_2, \ldots, y_n) \in \mathbb{R}^{2n}$. For two vehicles j, j' in the same lane, let a *rear-end conflict area* $\mathcal{O}_{j,j'} \subset \mathbb{R}^2$ be a closed set of points with nonempty interior where paths $\mathcal{P}_j$ and $\mathcal{P}_{j'}$ overlap. For $y_j, y_{j'} \in \mathcal{O}_{j,j'}$, call $\mathcal{D}(y_j, y_{j'})$ the distance between vehicles j and j' along their common path. Let a *side conflict area* $\mathcal{I}_i \subset \mathbb{R}^2$ be an open set of points where the vehicles' paths coming from different roads intersect or start to merge. The size of a side conflict area is determined such that if the geometric centers of vehicles are simultaneously inside the area, the vehicles collide with each other. This is illustrated in Fig. 4. The total number of side conflict areas is denoted by m. The above notation is represented in Fig. 2 in a scenario of four paths $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{P}_4$. The four red areas indicate side conflict areas $\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3, \mathcal{I}_4$, and the thick gray lines indicate rear-end conflict areas $\mathcal{O}_{2,3}$ and $\mathcal{O}_{2,4}$. The distance between vehicles 2 and 3 in the rear-end conflict area is denoted by $\mathcal{D}(y_2, y_3)$.

A rear-end collision is a configuration in which two vehicles in a rear-end conflict area get closer than a minimum safe distance d, which is larger than the length of vehicles. The set of points corresponding to rear-end collisions is

$$B_- := \{\mathbf{y} \in \mathbb{R}^{2n} : y_j, y_{j'} \in \mathcal{O}_{j,j'}, \mathcal{D}(y_j, y_{j'}) < d\}.$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2953213, IEEE Transactions on Automatic Control

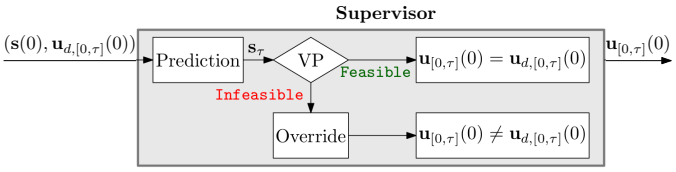


Fig. 5. If the Safety Verification Problem (VP) is infeasible given the predicted state $\mathbf{s}_\tau$, then the supervisor overwrites the requested input signal ($\mathbf{u}_{[0,\tau]}(0) \neq \mathbf{u}_{d,[0,\tau]}(0)$). Vehicles use the input signal returned by the supervisor during time interval $[0, \tau]$.

A side collision is a configuration in which two vehicles meet inside a side conflict area. The set of points corresponding to side collisions is

$$B_+ := \{\mathbf{y} \in \mathbb{R}^{2n} : y_j, y_{j'} \in \mathcal{I}_i \setminus \mathcal{O}_{j,j'}\}.$$

Definition 1 (Bad Set)**.** The bad set is the union $B := B_+ \cup B_-$.

The control method we propose in this paper utilizes an approximation of the dynamics of (1) to synthesize the supervisor. To account for the approximation error, while guaranteeing that the system remains out of the bad set, we will use an inflated version of the bad set, defined as the union of the two sets

$$B_-^*(\varepsilon) := \{\mathbf{y} \in \mathbb{R}^{2n} : y_j, y_{j'} \in \mathcal{O}_{j,j'}, \mathcal{D}(y_j, y_{j'}) < d^*\},$$

with $d^* := d + 2\varepsilon$, and

$$B_+^*(\varepsilon) := \{\mathbf{y} \in \mathbb{R}^{2n} : y_j, y_{j'} \in \mathcal{I}_i^* \setminus \mathcal{O}_{j,j'}\}.$$

with $\mathcal{I}_i^* := \mathcal{I}_i \oplus \{y_j \in \mathbb{R}^2 : \|y_j\|_2 \leq \varepsilon\}$, where the Minkowski sum of two sets A and B is $A \oplus B := \{a + b : a \in A, b \in B\}$.

Definition 2 (Inflated Bad Set)**.** The bad set inflated by ε is the union $B^*(\varepsilon) := B_-^*(\varepsilon) \cup B_+^*(\varepsilon)$.

C. Problem Statement

The supervisor is realized as a discrete-time algorithm running at sampling time τ. For instance, τ can be of the order of 0.1 s in Intelligent Transportation Systems applications based on 10 Hz communication [1]. For the sake of simplicity, we assume a synchronous algorithm and no communication delay. Asynchronous and delayed communication can be easily handled by incorporating an algorithm that estimates a set of possible states using delayed information [28]. In the notation we introduced above, the supervisor is a map

$$(\mathbf{s}(0), \mathbf{u}_{d,[0,\tau]}(0)) \mapsto \mathbf{u}_{[0,\tau]}(0),$$

where $\mathbf{u}_{d,[0,\tau]}(0) \in \mathcal{U}_{[0,\tau]}$ is the input signal requested at time 0 by drivers for time interval $[0, \tau]$, and $\mathbf{u}_{[0,\tau]}(0)$ is the input signal returned at time 0 by the supervisor for time interval $[0, \tau]$. At each generic time 0, the supervisor collects information on each vehicle's state and on the requested input signal, predicts the future state $\mathbf{s}_\tau$ reached by the requested input signal, and decides whether the input signal can be accepted (in which case $u_{[0,\tau],j}(0) = u_{d,[0,\tau],j}(0)$) or must be corrected (in which case $u_{[0,\tau],j}(0) \neq u_{d,[0,\tau],j}(0)$). The current state and requested input signal are measured through cameras, lidars, and radars on the roadside or through vehicles' embedded sensors. Note that the requested input signal after time τ is unknown to the supervisor. We illustrate the overview of the supervisor in Fig. 5.

The computational challenge of a supervisor resides chiefly in the design of the algorithm in charge of deciding whether an override is required for a given state-input pair $(\mathbf{s}(0), \mathbf{u}_{d,[0,\tau]}(0))$. As a consequence, our attention throughout the most of this paper will be devoted to solving this problem, formulated as follows.

Safety Verification Problem (VP). Given a state $\mathbf{s}_\tau$, determine the feasibility of the following:

$$\begin{aligned} &\min_{\mathbf{u} \in \mathcal{U}} && J(\mathbf{u}), \\ &\text{subject to} && \mathbf{y}(t, \mathbf{u}, \mathbf{s}_\tau) \notin B, \ \forall t \geq 0. \end{aligned} \tag{3}$$

Here, $\mathbf{y}(t, \mathbf{u}, \mathbf{s}_\tau)$ is the position vector in the planar coordinate reached after time t with input signal $\mathbf{u}$ from the state $\mathbf{s}_\tau$, where $\mathbf{s}_\tau$ itself is the state reached after time τ from $\mathbf{s}(0)$ using $\mathbf{u}_{d,[0,\tau]}(0)$. If the above problem is infeasible, then $\mathbf{u}_{d,[0,\tau]}(0)$ should be overwritten because this means that the state will inevitably enter the bad set at some time. If instead it is feasible, then $\mathbf{u}_{d,[0,\tau]}(0)$ is allowed to apply because there will be an input signal at the next time step that prevents any future collision. Note that, while this problem concerns the existence of a feasible solution $\mathbf{u}$, any choice of cost function $J(\mathbf{u})$ can be used to obtain an optimal input signal, for example, that minimizes the throughput of the system during overrides.

Given a state $\mathbf{s}_\tau$ and a real number $\varepsilon \geq 0$, let us define the set of input signals

$$\mathcal{U}_{\text{safe}}(\varepsilon) := \{\mathbf{u} \in \mathcal{U} : \mathbf{y}(t, \mathbf{u}, \mathbf{s}_\tau) \notin B^*(\varepsilon), \forall t \geq 0\},$$

and let $\mathcal{U}_{\text{safe},[0,\tau]}(\varepsilon)$ be the set of signals in $\mathcal{U}_{\text{safe}}(\varepsilon)$ restricted to time interval $[0, \tau]$. The set $\mathcal{U}_{\text{safe}}(\varepsilon)$ consists of an input signal that allows the state trajectory to maintain distance ε from the bad set. If the optimization problem (3) is feasible, then by definition $\mathbf{u}_{d,[0,\tau]}(0) \in \mathcal{U}_{\text{safe},[0,\tau]}(0)$.

We say that a supervisor is *ε-restrictive* if it has the following property.

Definition 3 (ε-restrictive Supervisor)**.** For all $\mathbf{s}(0) \in \mathbf{S}$,

$$\mathbf{u}_{[0,\tau]}(0) \neq \mathbf{u}_{d,[0,\tau]}(0) \implies \mathbf{u}_{d,[0,\tau]}(0) \notin \mathcal{U}_{\text{safe},[0,\tau]}(\varepsilon).$$

Similarly, we say that a supervisor is *least restrictive* if it is ε-restrictive with $\varepsilon = 0$.

III. Our Approach to Problem Solution

Our approach to solving the VP is to solve a computationally more tractable verification problem that is formulated based on an abstraction of the concrete system (1), and then deduce the solution of the VP by studying the relation between the two systems.

A. Abstraction-based Verification Problem

Let us first define an abstraction Σ_a of the concrete system Σ_c in (1). Call $\mathcal{V}_j$ the space of piecewise constant signals defined on time interval $[0, t_{f,j}]$ with N_j discontinuities. Here,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2953213, IEEE Transactions on Automatic Control

$t_{f,j}$ and N_j are the predefined final time and number of discontinuities, respectively, of the signals in $\mathcal{V}_j$. Let $t_j[k]$ for $k \in \{0, \ldots, N_j - 1\}$ be the times at which these discontinuities occur where $t_j[0] = 0$ and $t_j[N_j] = t_{f,j}$. Also, let $v_j[k]$ for $k \in \{1, \ldots, N_j\}$ be the value taken by the signal between $t_j[k-1]$ and $t_j[k]$ with $v_j[0]$ indicating an initial velocity at time $t_j[0]$. Let $\boldsymbol{\gamma}$ be a vector of functions γ_j. We define $\Sigma_a(\boldsymbol{\gamma})$ as the constrained system

$$\dot{x}_{a,j} = v_j, \quad v_j \in \mathcal{V}_j, \tag{4a}$$
$$\gamma_j(v_j[0], \ldots, v_j[N_j], t_j[0], \ldots, t_j[N_j]) \leq \mathbf{0}, \tag{4b}$$

for all $j \in \{1, \ldots, n\}$. In the above model, $x_{a,j} \in X_j$ is the state, and the set X_j is the same as the position set of (1). The values $v_j[k]$ of the input signal $v_j \in \mathcal{V}_j$ are bounded between $[v_{j,\min}[k], v_{j,\max}[k]]$ where $[v_{j,\min}[k], v_{j,\max}[k]]$ is a subset of $[\dot{x}_{j,\min}, \dot{x}_{j,\max}]$, the velocity set of (1). In Section IV, we will use (4b) to constrain the jumps in the velocity functions v_j, thereby introducing a form of inertia in the otherwise kinematic model (4a). We define the output function

$$h_{a,j}(x_{a,j}) := x_{a,j},$$

which has common domain with the output function of Σ_c. This allows us to define the distance

$$\|\mathbf{h}(\mathbf{s}) - \mathbf{h}_a(\mathbf{x}_a)\|_\infty$$

between the outputs of the two models.

Now, we formulate a safety verification problem on the abstraction $\Sigma_a(\boldsymbol{\gamma})$. Let $\mathbf{y}_a$ be the vector with elements

$$y_{a,j} = \mathcal{P}_j(x_{a,j})$$

for all j, where $y_{a,j} \in \mathbb{R}^2$ is the planar coordinate corresponding to the curvilinear coordinate $x_{a,j}$.

Abstraction-based Verification Problem (AVP). Given a real number $\varepsilon \geq 0$ and a state $\mathbf{s}_\tau$, determine the feasibility of the following:

$$\begin{aligned} &\min_{\mathbf{v} \in \mathcal{V}} && J(\mathbf{v}), \\ &\text{subject to} && \mathbf{y}_a(t, \mathbf{v}, \mathbf{s}_\tau) \notin B^*(\varepsilon), \forall t \geq 0. \end{aligned} \tag{5}$$

Here the function $\mathbf{y}_a(t, \mathbf{v}, \mathbf{s}_\tau)$ is the position vector reached after time t according to the abstraction (4) with velocity signal $\mathbf{v} \in \mathcal{V} := \mathcal{V}_1 \times \ldots \times \mathcal{V}_n$ starting from an initial position $\mathbf{x}_\tau$ and velocity $\mathbf{v}[0] = \dot{\mathbf{x}}_\tau$ where $\mathbf{s}_\tau = (\mathbf{x}_\tau, \dot{\mathbf{x}}_\tau)$. We can solve the AVP by reformulating it as a mixed-integer program based on space discretization. This will be detailed in Section IV.

B. Approximate Simulation Relation

In order to deduce the solution of the VP from the solution of the AVP, we exploit an *approximate simulation relation* [21], [22] between the concrete system (1) and its abstraction (4). Let $\mathbf{s}_i \in \mathbf{S}_i$ denote the state, $\mathbf{u}_i \in \mathbf{U}_i$ the input, and $\mathbf{h}_i(\mathbf{s}_i)$ the output (e.g., position in this paper), for system Σ_i where $i \in \{1, 2\}$.

Definition 4 (ε-Approximate Simulation)**.** A relation $R_\varepsilon \subseteq \mathbf{S}_1 \times \mathbf{S}_2$ is called an approximate simulation relation of Σ_1 by Σ_2 of precision $\varepsilon \geq 0$, if for all $(\mathbf{s}_1, \mathbf{s}_2) \in R_\varepsilon$

- $\|\mathbf{h}_1(\mathbf{s}_1) - \mathbf{h}_2(\mathbf{s}_2)\|_\infty \leq \varepsilon$;
- for all $t \geq 0$, for all $\mathbf{u}_1 \in \mathbf{U}_1$, there exists an input $\mathbf{u}_2 \in \mathbf{U}_2$ such that $(\mathbf{s}_1(t, \mathbf{u}_1, \mathbf{s}_1), \mathbf{s}_2(t, \mathbf{u}_2, \mathbf{s}_2)) \in R_\varepsilon$.

Σ_2 *approximately simulates* Σ_1 with precision ε, denoted by $\Sigma_1 \preceq_\varepsilon \Sigma_2$, if there is R_ε such that for all $\mathbf{s}_1(0)$, there exists $\mathbf{s}_2(0)$ such that $(\mathbf{s}_1(0), \mathbf{s}_2(0)) \in R_\varepsilon$.

In the above definition, we intentionally use the same parameter ε that was used in the AVP. This is because in Section V, we prove that for any positive real number ε, a vector $\boldsymbol{\gamma}$ can always be chosen such that

$$\Sigma_a(\boldsymbol{\gamma}) \preceq_\varepsilon \Sigma_c, \tag{6}$$

which tells that for any position trajectory of $\Sigma_a(\boldsymbol{\gamma})$, there exists a position trajectory of Σ_c within maximum deviation ε. Thus, this indicates that a trajectory of $\Sigma_a(\boldsymbol{\gamma})$ avoiding the inflated bad set $B^*(\varepsilon)$ corresponds to a trajectory of Σ_c avoiding the bad set B.

IV. Reformulation of the AVP

Because the dynamics of $\Sigma_a(\boldsymbol{\gamma})$ in (4) are constrained to a piecewise constant velocity, they lend themselves to easy discretization, in time or space. We will follow this route to rewrite the abstraction-based verification problem as a mixed-integer programming (MIP) problem.

We discretize the longitudinal path of vehicle j into N_j short segments of possibly unequal length. The discretized longitudinal path is denoted by a finite sequence of intervals

$$\{[\xi_j[k-1], \xi_j[k]]\}_{k=1}^{N_j}$$

where $[\xi_j[k-1], \xi_j[k]] \subset X_j$, $\xi_j[k-1] < \xi_j[k]$, $\xi_j[0] = x_{\tau,j}$, and $\xi_j[N_j] = x_{j,\max}$. Let $\boldsymbol{\Delta}\xi_j : \{1, 2, \ldots, N_j\} \to \mathbb{R}_+$ denote a sequence of the segments' lengths, that is, $\boldsymbol{\Delta}\xi_j[k] = \xi_j[k] - \xi_j[k-1]$.

To enable simple and exact description of the collision avoidance constraints, we discretize the longitudinal path in such a way that all entries in each path segment belong either to a (inflated side or rear-end) conflict area or to the outside of the conflict area. This yields that the beginning and end points of conflict areas coincide with the end points of path segments. Also, longitudinal paths in the same rear-end conflict area share a common discretization grid. For each path segment k of vehicle j in rear-end conflict area $\mathcal{O}_{j,j'}$, there is a path segment $\delta(k)$ of the following vehicle j' in the rear-end conflict area that satisfies

$$\mathcal{P}_j(\xi_j[k] - d^*) = \mathcal{P}_{j'}(\xi_{j'}[\delta(k)]).$$

An example of a discretized path is depicted in Fig. 6.

The decision variables in the MIP problem are times necessary for a vehicle to cross each space segment. That is,

$$\boldsymbol{\Delta}\mathbf{t}_j[k] := \frac{\boldsymbol{\Delta}\xi_j[k]}{v_j[k]}$$

for all $j \in \{1, 2, \ldots, n\}$ and $k \in \{1, 2, \ldots, N_j\}$. The fact that $v_j[k]$ is bounded by $[v_{j,\min}[k], v_{j,\max}[k]]$ gives the bounds of time interval $\boldsymbol{\Delta}\mathbf{t}_j[k]$,

$$\frac{\boldsymbol{\Delta}\xi_j[k]}{v_{j,\max}[k]} \leq \boldsymbol{\Delta}\mathbf{t}_j[k] \leq \frac{\boldsymbol{\Delta}\xi_j[k]}{v_{j,\min}[k]}, \ \forall k \in \{1, 2, \ldots, N_j\}. \tag{7}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2953213, IEEE Transactions on Automatic Control

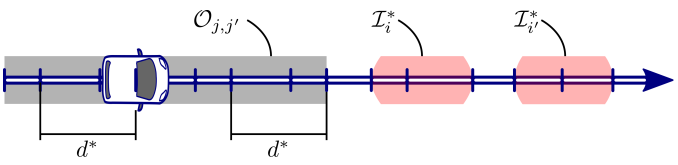


Fig. 6. The longitudinal path is discretized such that the end points of conflict areas ($\mathcal{O}_{j,j'}, \mathcal{I}_i^*$, and $\mathcal{I}_{i'}^*$) coincide with the end points of path segments and the length of some path segments in rear-end conflict areas is equal to the inflated safe distance d^*.

If $v_{j,\min}[k] = 0$, the constraint becomes $\boldsymbol{\Delta}\xi_j[k]/v_{j,\max}[k] \leq \boldsymbol{\Delta}\mathbf{t}_j[k] < \infty$.

Also, in order to encode constraint (4b), which aims to restrict velocity jumps between two consecutive segments, we introduce *smoothing function* $g_{j,k}$ and write the constraint as follows:

$$\left|\boldsymbol{\Delta}\xi_j[k-1] \cdot \boldsymbol{\Delta}\mathbf{t}_j[k] - \boldsymbol{\Delta}\xi_j[k] \cdot \boldsymbol{\Delta}\mathbf{t}_j[k-1]\right| \leq g_{j,k}(\boldsymbol{\Delta}\mathbf{t}_j[k]), \tag{8}$$

for all $j \in \{1, 2, \ldots, n\}$ and $k \in \{2, \ldots, N_j\}$. For $k = 1$, set $|\dot{x}_{\tau,j}\boldsymbol{\Delta}\mathbf{t}_j[1] - \boldsymbol{\Delta}\xi_j[1]| \leq g_{j,1}(\boldsymbol{\Delta}\mathbf{t}_j[1])$ where $\dot{\mathbf{x}}_\tau = (\dot{x}_{\tau,1}, \ldots, \dot{x}_{\tau,n})$ is a given initial velocity. We will provide a sufficient condition for the smoothing function in Appendix A to appropriately represent the dynamical behavior of the concrete system Σ_c. Since we want to have a linear constraint with respect to the decision variable, one favorable option is to set $g_{j,k}(\boldsymbol{\Delta}\mathbf{t}_j[k]) = c_1\boldsymbol{\Delta}\mathbf{t}_j[k] + c_2$ for some constants c_1 and c_2.

We can write the constraint (4b) as a set of

$$\gamma_{j,k}(v_j[k-1], v_j[k], t_j[k-2], t_j[k-1], t_j[k]) \leq 0$$

for $k \in \{2, \ldots, N_j\}$ that is the same inequality as (8) and $\gamma_{j,1}(v_j[0], v_j[1], t_j[0], t_j[1]) \leq 0$ for $k = 1$, with $\boldsymbol{\Delta}\mathbf{t}_j[k] = t_j[k] - t_j[k-1]$ and $\boldsymbol{\Delta}\xi_j[k] = v_j[k]\boldsymbol{\Delta}\mathbf{t}_j[k]$. Here, γ_j is a vector of $\gamma_{j,k}$ for all k, and $\gamma_{j,k}$ is a function of the smoothing function $g_{j,k}$.

We can now describe the constraint for rear-end collision avoidance. In a nonempty rear-end conflict area $\mathcal{O}_{j,j'}$, let $j \ll j'$ denote that vehicle j precedes vehicle j'. For all K where $\mathcal{P}_j(\xi_j[K]) \in \mathcal{O}_{j,j'}$ and $\mathcal{P}_{j'}(\xi_{j'}[\delta(K)]) \in \mathcal{O}_{j,j'}$, the rear-end collision avoidance constraint is as follows:

$$\text{if } j \ll j', \ \sum_{k=1}^{K} \boldsymbol{\Delta}\mathbf{t}_j[k] \leq \sum_{k=1}^{\delta(K)} \boldsymbol{\Delta}\mathbf{t}_{j'}[k]. \tag{9}$$

This means that after vehicle j reaches the position $\mathcal{P}_j(\xi_j[K])$, vehicle j' can reach $\mathcal{P}_j(\xi_j[K] - d^*)$. Thus, $x_{a,j}$ and $x_{a,j'}$ are distance d^* apart at the end points of each segment, and also between the end points because trajectories of the abstraction (4) are (piecewise) linear.

To describe the constraint for the side collision avoidance, let positive integers $K_{i,j}^{\text{in}}$ and $K_{i,j}^{\text{out}}$ denote

$$\left(\xi_j[K_{i,j}^{\text{in}}], \xi_j[K_{i,j}^{\text{out}}]\right) = \{x_j : \mathcal{P}_j(x_j) \in \mathcal{I}_i^* \setminus \mathcal{O}_{j,j'}\}.$$

This means that the path from the $K_{i,j}^{\text{in}}$-th segment to the $K_{i,j}^{\text{out}}$-th segment exactly overlaps with the inflated side conflict area

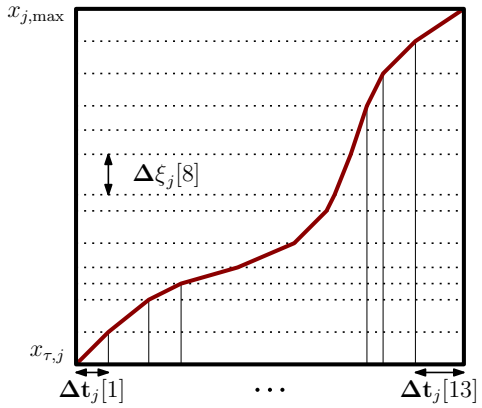


Fig. 7. A sequence of time intervals $\boldsymbol{\Delta}\mathbf{t}_j$ on the discretized path $\{[\xi_j[k-1], \xi_j[k]]\}_{k=1}^{13}$ defines a trajectory of the abstraction (red line) by the map (12).

$\mathcal{I}_i^* \setminus \mathcal{O}_{j,j'}$. Such integers exist by construction of the discretized paths. Vehicle j is inside the inflated side conflict area when

$$t \in \left(\sum_{k=1}^{K_{i,j}^{\text{in}}} \boldsymbol{\Delta}\mathbf{t}_j[k], \sum_{k=1}^{K_{i,j}^{\text{out}}} \boldsymbol{\Delta}\mathbf{t}_j[k]\right).$$

To avoid side collisions, two vehicles should never meet inside the same conflict area. The side collision avoidance constraint is as follows:

$$\sum_{k=1}^{K_{i,j}^{\text{out}}} \boldsymbol{\Delta}\mathbf{t}_j[k] \leq \sum_{k=1}^{K_{i,j'}^{\text{in}}} \boldsymbol{\Delta}\mathbf{t}_{j'}[k] \text{ or } \sum_{k=1}^{K_{i,j'}^{\text{out}}} \boldsymbol{\Delta}\mathbf{t}_{j'}[k] \leq \sum_{k=1}^{K_{i,j}^{\text{in}}} \boldsymbol{\Delta}\mathbf{t}_j[k]. \tag{10}$$

This means that either vehicle j' enters the side conflict area after vehicle j exits or the other way around. To encode this constraint, we introduce a binary variable $b_{ijj'}$, which takes 1 if vehicle j precedes vehicle j' inside side conflict area i and 0 otherwise. Let $\mathbf{b}$ be the vector of all such binary variables $b_{ijj'}$.

MIP Formulation of the AVP (MIP-AVP). Given a state $\mathbf{s}_\tau$, determine the feasibility of the following problem:

$$\begin{aligned} &\min_{\boldsymbol{\Delta}\mathbf{t}_j, \mathbf{b}} && J(\boldsymbol{\Delta}\mathbf{t}_j, \forall j), \\ &\text{subject to} && (7), (8), (9), (10). \end{aligned} \tag{11}$$

In the problem, constraints (9) and (10) depend on parameter ε, due to the inflated side conflict areas $\mathcal{I}_i^*$ and inflated safe distance d^*. The constraints (7), (9), (10) are linear with respect to $\boldsymbol{\Delta}\mathbf{t}_j[k]$, but the MIP-AVP is a mixed-integer linear programming (MILP) problem only if the smoothing constraint (8) is also linear, with a linear cost function $J(\boldsymbol{\Delta}\mathbf{t}_j, \forall j)$. We discuss how to compute a smoothing function $g_{j,k}(\boldsymbol{\Delta}\mathbf{t}_j[k])$ that is linear with respect to $\boldsymbol{\Delta}\mathbf{t}_j[k]$ in Appendix A.

Given a state $\mathbf{s}_\tau$ and a feasible solution $(\boldsymbol{\Delta}\mathbf{t}_j, \forall j)$ to the MIP-AVP, a feasible solution $\mathbf{v} \in \mathcal{V}$ to the AVP is obtained

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2953213, IEEE Transactions on Automatic Control

from the following map:

$$\text{for } t \in \left[\sum_{i=1}^{k-1} \boldsymbol{\Delta}\mathbf{t}_j[i], \sum_{i=1}^{k} \boldsymbol{\Delta}\mathbf{t}_j[i]\right), v_j(t) = \frac{\boldsymbol{\Delta}\xi_j[k]}{\boldsymbol{\Delta}\mathbf{t}_j[k]},$$
$$x_{a,j}(t) = \xi_j[k-1] + v_j(t)\left(t - \sum_{i=1}^{k-1} \boldsymbol{\Delta}\mathbf{t}_j[i]\right), \quad (12)$$

for all $k \in \{1, \dots, N_j\}$, where $\mathbf{x}_a = (x_{a,1}, \dots, x_{a,n})$ is the position trajectory of the abstraction $\Sigma_a(\boldsymbol{\gamma})$. An example of a trajectory $x_{a,j}$ is illustrated in Fig. 7.

V. Problem Solution

Now, we design the supervisor algorithm and prove that it prevents the dynamical state $\mathbf{s}$ from entering the bad set B.

Algorithm 1 Implementation of the supervisor at step k

```
1:  procedure Supervisor(s(0), u_{d,[0,τ]}(0))
2:      Prediction: s_τ ← s(τ, u_{d,[0,τ]}(0))
3:      Verification: Solve the MIP-AVP given s_τ
4:      if Feasible then
5:          Store a new safe input signal u^k
6:          return u_{d,[0,τ]}(0)
7:      else
8:          Prediction: s_τ ← s(τ, u^{k-1}_{[0,τ]})
9:          Verification: Solve the MIP-AVP given s_τ
10:         if Feasible then
11:             Store a new safe input signal u^k
12:         else
13:             Store a previous input signal u^k ← u^{k-1}_{[τ,∞)}
14:         return u^{k-1}_{[0,τ]}
```

The supervisor verifies whether an override is required given a current state-input pair $(\mathbf{s}(0), \mathbf{u}_{d,[0,\tau]}(0))$ (line 3). If there is a feasible solution in the MIP-AVP (line 4), then the supervisor allows the requested input (line 6) and stores a safe input signal $\mathbf{u}^k$ (i.e., an input signal in $\mathcal{U}_{\text{safe}}(0)$) for a possible use at future steps (line 5). Otherwise, the supervisor returns the safe input signal stored at the previous step restricted to time $[0, \tau]$ (line 14). To update the safe input signal, the supervisor computes a new input signal by solving the MIP-AVP again (lines 9 and 11), or reuses the previous safe input signal by translating it by τ (line 13). The MIP-AVP in line 9 is not always feasible given $\mathbf{s}(\tau, \mathbf{u}^{k-1}_{[0,\tau]})$ because the formulation is based on the abstraction $\Sigma_a(\boldsymbol{\gamma})$.

Note that the algorithm computes a new safe input signal $\mathbf{u}^k$ when the MIP-AVP has a feasible solution. Such a safe input signal can be obtained as a byproduct during the proof of the approximate simulation relation. In the following theorem, we state the approximate simulation relation of $\Sigma_a(\boldsymbol{\gamma})$ by Σ_c.

Theorem 1. *For any $\varepsilon > 0$, there exists $\boldsymbol{\gamma}$ such that $\Sigma_a(\boldsymbol{\gamma}) \preceq_\varepsilon \Sigma_c$.*

Proof. See Appendix A. □

As a consequence of the above theorem, Algorithm 1 always returns an input that belongs to $\mathcal{U}_{\text{safe},[0,\tau]}(0)$.

Corollary 1. *Algorithm 1 guarantees safety, that is, it keeps the system state $\mathbf{s}$ outside the bad set B.*

Proof. The algorithm allows the requested input if there is a trajectory $\mathbf{x}_a$ of $\Sigma_a(\boldsymbol{\gamma})$ starting from $\mathbf{s}_\tau = \mathbf{s}(\tau, \mathbf{u}_{d,[0,\tau]}(0))$ that avoids the inflated bad set $B^*(\varepsilon)$ (i.e., the MIP-AVP is feasible). This case implies, by Theorem 1, the existence of a trajectory of Σ_c that avoids the bad set B because the approximate simulation relation tells that $|x_j(t) - x_{a,j}(t)| \le \varepsilon$ for all $t \ge 0$. This assures that the requested input does not lead to any unavoidable future collision, that is, it belongs to $\mathcal{U}_{\text{safe},[0,\tau]}(0)$. Note that, when the MIP-AVP is feasible, the algorithm stores a safe input signal $\mathbf{u}^k$ of Σ_c that makes a trajectory that avoids B, for example, by using the feedback control law (21) in Appendix A. When the MIP-AVP is not feasible, the algorithm overwrites the requested input with a safe input signal $\mathbf{u}^{k-1}_{[0,\tau]}$ computed at a previous step and restricted to $[0, \tau]$, which belongs to $\mathcal{U}_{\text{safe},[0,\tau]}(0)$. Therefore, the algorithm always returns an input that belongs to $\mathcal{U}_{\text{safe},[0,\tau]}(0)$, thereby preventing the system state $\mathbf{s}$ from entering the bad set B. □

Corollary 2. *Algorithm 1 is free of deadlock, that is, it guarantees that vehicles eventually exit the controlled region.*

Proof. A safe input signal $\mathbf{u}^k$ makes a trajectory of Σ_c that avoids the bad set B and leaves the controlled region. Because Algorithm 1 keeps the system in a state where there exists at least one safe input signal, it is free of deadlock. □

VI. Estimation of the Approximation Error

In this section, we prove that the supervisor (Algorithm 1) is at most ϕ-restrictive for some finite value ϕ. To do this, we introduce a second abstraction $\Sigma'_a(\boldsymbol{\gamma}')$ of the concrete system Σ_c.

Call $\mathcal{V}'_j$ the space of piecewise constant signals defined on time interval $[0, t'_{f,j}]$ with N'_j discontinuities. Here, $t'_{f,j}$ and N'_j are the predefined final time and number of discontinuities, respectively, of the signals in $\mathcal{V}'_j$. Let $t'_j[0], \dots, t'_j[N'_j - 1]$ be the times at which these discontinuities occur with $t'_j[0] = 0$ and $t'_j[N'_j] = t'_{f,j}$. Also, let $v'_j[k]$ for $k \in \{1, \dots, N'_j\}$ be the value taken by the signal between $t'_j[k-1]$ and $t'_j[k]$ with $v'_j[0]$ indicating an initial velocity at time $t'_j[0]$. Let $\boldsymbol{\gamma}'$ be a vector of functions γ'_j. We define $\Sigma'_a(\boldsymbol{\gamma}')$ as the constrained system

$$\dot{x}'_{a,j} = v_j, \quad v_j \in \mathcal{V}'_j, \quad (13a)$$
$$\gamma'_j(v'_j[0], \dots, v'_j[N'_j], t'_j[0], \dots, t'_j[N'_j]) \le \mathbf{0}, \quad (13b)$$

for all $j \in \{1, \dots, n\}$. In this model, the values $v'_j[k]$ of the input signal $v'_j \in \mathcal{V}'_j$ are bounded between $[\dot{x}_{j,\min}, \dot{x}_{j,\max}]$ for all k, the velocity set of Σ_c. This abstract system is the same as $\Sigma_a(\boldsymbol{\gamma})$ except that γ'_j represents a more relaxed constraint on velocity jumps than γ_j. We will prove in this section that there exist a vector $\boldsymbol{\gamma}'$ and a finite ϕ such that

$$\Sigma_c \preceq_{\phi/2} \Sigma'_a(\boldsymbol{\gamma}'). \quad (14)$$

This relation leads to the proof that the supervisor is (at most) ϕ-restrictive.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2953213, IEEE Transactions on Automatic Control

Again, we formulate a safety verification problem based on the abstraction $\Sigma'_a(\boldsymbol{\gamma}')$ as a mixed-integer programming (MIP) problem. The safety verification on $\Sigma'_a(\boldsymbol{\gamma}')$ is the problem of determining the existence of $\mathbf{v} \in \mathcal{V}' := \mathcal{V}'_1 \times \ldots \times \mathcal{V}'_n$ such that

$$\mathbf{y}'_a(t, \mathbf{v}, \mathbf{s}_\tau) \notin B^*(\varepsilon), \forall t \geq 0$$

given a real number $\varepsilon \geq 0$ and a state $\mathbf{s}_\tau$. Here, $\mathbf{y}'_a(t, \mathbf{v}, \mathbf{s}_\tau)$ is the position vector in the planar coordinate reached after time t according to the dynamics (13) with the velocity signal $\mathbf{v}$ from the state $\mathbf{s}_\tau$.

For the MIP reformulation, we uniformly discretize the longitudinal path of vehicle j into N'_j segments of length $\boldsymbol{\Delta}\xi'_j[k] = \Delta\xi'_j$ for all k. Due to the uniform discretization, the definition of parameters $\delta(K), K^{\text{in}}_{i,j}$, and $K^{\text{out}}_{i,j}$ change to the values satisfying

$$\mathcal{D}(\mathcal{P}_j(\xi_j[K]), \mathcal{P}_{j'}(\xi_{j'}[\delta(K)])) \geq d^* \text{ and}$$
$$(\xi_j[K^{\text{in}}_{i,j}], \xi_j[K^{\text{out}}_{i,j}]) \supseteq \{x_j : \mathcal{P}_j(x_j) \in \mathcal{I}^* \setminus \mathcal{O}_{j,j'}\}.$$

We write the constraint (13b) as a set of

$$c_1 \leq \boldsymbol{\Delta}\mathbf{t}_j[k] - \boldsymbol{\Delta}\mathbf{t}_j[k-1] \leq c_2 \tag{15}$$

for all $k \in \{1, \ldots, N'_j\}$, and $c_1 \leq \boldsymbol{\Delta}\mathbf{t}_j[1] - \Delta\xi'_j/\dot{x}_{\tau,j} \leq c_2$ for $k = 1$. Here, real numbers c_1 and c_2 are numerically quantified as the minimum and maximum time changes allowed over distance $\Delta\xi'_j$ by the dynamics of the concrete system Σ_c. The values of c_1 and c_2 depend on the discrete step size $\Delta\xi'_j$ but do not depend on k due to the uniform discretization.

The verification based on $\Sigma'_a(\boldsymbol{\gamma}')$ is formulated as the MIP problem as follows.

MIP Formulation of the $\Sigma'_a(\boldsymbol{\gamma}')$-based Verification (MIP-AVP'). Given a state $\mathbf{s}_\tau$, determine the feasibility of the following problem:

$$\begin{aligned} &\min_{\boldsymbol{\Delta}\mathbf{t}_j, \mathbf{b}} && J(\boldsymbol{\Delta}\mathbf{t}_j, \forall j), \\ &\text{subject to} && (7), (9), (10), (15). \end{aligned} \tag{16}$$

The above problem is also MILP given a linear cost function $J(\boldsymbol{\Delta}\mathbf{t}_j, \forall j)$ and is used to estimate an upper bound of the restrictiveness.

Now, we show that the supervisor is ϕ-restrictive for some finite real number ϕ. We define ϕ as the smallest positive number that satisfies the following two conditions. First,

$$\frac{\phi}{2} \geq \max_j \max_{t \in [0, t^*_j]} \max_{u_j \in \mathcal{U}_j, s_{\tau,j} \in 0 \times \dot{X}_j} |x_j(t, u_j, s_{\tau,j}) - v^*_j t|, \tag{17}$$

where $t^*_j = \{t : x_j(t, u_j, s_{\tau,j}) = \Delta\xi'_j\}$ and $v^*_j = \Delta\xi'_j/t^*_j$. Second, for any $\mathbf{s}_\tau$, when the MIP-AVP is infeasible with $\varepsilon > 0$, we have

$$\mathbf{y}'_a(t, \mathbf{v}, \mathbf{s}_\tau) \in B^*(\phi/2), \ \forall \mathbf{v} \in \mathcal{V}', \ \forall t \geq 0. \tag{18}$$

We can numerically find the value of ϕ that satisfies (18) by solving the MIP-AVP' using a large set of randomly generated initial states.

Theorem 2. *There exists $\boldsymbol{\gamma}'$ such that $\Sigma_c \preceq_{\phi/2} \Sigma'_a(\boldsymbol{\gamma}')$.*

Proof. Suppose a position trajectory of Σ_c, denoted by $x_j(t, u_j)$, is given, starting from the initial state $s_j(0) = (x_j(0), \dot{x}_j(0))$. We define $T_j[k]$ as time for the position trajectory to reach $\xi'_j[k]$ where $\xi'_j[k] := x_j(0) + k\Delta\xi'_j$. That is, $T_j[k] := \{t : x_j(t, u_j) = \xi'_j[k]\}$. Let $x'_{a,j}(0) = x_j(0)$ and

$$v_j(t) := v^*_j[k] = \frac{\xi'_j[k] - \xi'_j[k-1]}{T_j[k] - T_j[k-1]}, \forall t \in [T_j[k-1], T_j[k]], \tag{19}$$

for $k \in \{2, \ldots, N'_j\}$ with $v^*_j[0] = \dot{x}_j(0)$. Then, a position trajectory $x'_{a,j}(t, v_j)$ satisfies $x'_{a,j}(T_j[k], v_j) = \xi'_j[k]$ for all k. Note that v_j is piecewise constant, $v_j(t) \in [\dot{x}_{j,\min}, \dot{x}_{j,\max}]$ because $\dot{x}_j(t) \in \dot{X}_j = [\dot{x}_{j,\min}, \dot{x}_{j,\max}]$, and $\gamma'_j(v_j(0)', \ldots, T_j[0], \ldots) \leq \mathbf{0}$ if γ'_j represents a set of inequalities (15) for all k. Thus, $x'_{a,j}$ is a trajectory of $\Sigma'_a(\boldsymbol{\gamma}')$. For each path segment, $|x_j(t, u_j) - x'_{a,j}(t, v_j)| = |x_j(t, u_j, (0, \dot{x}_j(T_j[k-1]))) - v^*_j[k]t)| \leq \phi/2$ by (17). This proves that $\Sigma'_a(\boldsymbol{\gamma}')$ approximately simulates Σ_c with precision $\phi/2$. □

Corollary 3. *Algorithm 1 is ϕ-restrictive.*

Proof. When the MIP-AVP in line 3 of Algorithm 1 is infeasible (i.e., $\mathbf{u}_{[0,\tau]}(0) \neq \mathbf{u}_{d,[0,\tau]}(0)$), by construction of ϕ in (18), there is no trajectory of $\Sigma'_a(\boldsymbol{\gamma}')$ that avoids the inflated bad set $B^*(\phi/2)$. This implies, by Theorem 2, that there is no trajectory of Σ_c that can avoid the inflated bad set $B^*(\phi)$ (i.e., $\mathbf{u}_{d,[0,\tau]}(0) \notin \mathcal{U}_{\text{safe},[0,\tau]}(\phi)$). By Definition 3, this concludes the proof. □

VII. Supervisor Performance

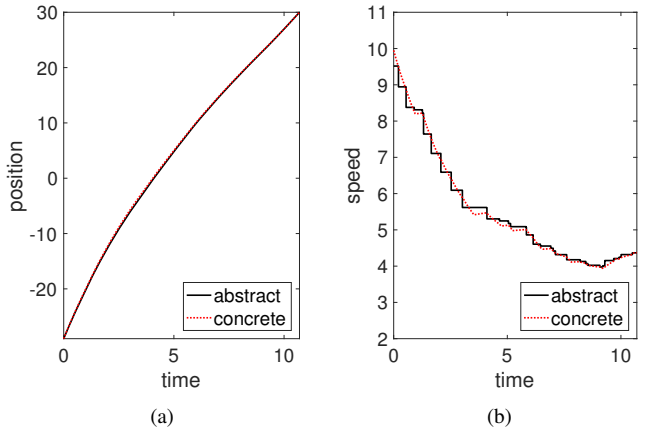
In this section, we implement the supervisor algorithm (Algorithm 1) using MATLAB on a personal computer with Intel Core i7 processor at 3.10 GHz and 8 GB RAM. We use CPLEX [29] to solve mixed-integer programming problems. We use as our test case the traffic scenario depicted in Fig. 1, which involves the same ingredients (intersecting, merging and splitting paths) as the scenarios in Fig. 3, but an overall higher computational complexity (more distinct conflict regions).

We present the simulation of the supervisor algorithm in Section VII-A. In Section VII-B, we discuss the computation time and restrictiveness of the algorithm in terms of the number of vehicles and the step size of the MIP-AVP. In Section VII-C, we compare the performance of our approach with other previous work, especially with one based on time discretization (provided in [18]). In Section VII-D, we provide an extension of the algorithm, which relaxes the assumption that the longitudinal paths of vehicles are determined before they enter the controlled region.

A. Implementation of the Supervisor Algorithm

We simulate the supervisor algorithm in the scenario illustrated in Fig. 1, with 20 vehicles whose longitudinal dynamics are modeled by (2) with parameters $c_1 = 0.005, c_2 = 0$, and $c_3 = 1$. We use the parameters $[v_{j,\min}[k], v_{j,\max}[k]] = [\dot{x}_{j,\min}, \dot{x}_{j,\max}] = [1, 15]$, $[u_{j,\min}, u_{j,\max}] = [-3, 3]$, $x_{j,\max} = 30$ for all j and k, and $\tau = 0.1$, $\varepsilon = 1$. We divide the longitudinal paths uniformly into segments of length 3 and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2953213, IEEE Transactions on Automatic Control



(a) (b)

Fig. 8. By Theorem 1, any position trajectory of $\Sigma_a(\gamma)$ can be tracked by a position trajectory of Σ_c within $\varepsilon = 1$.

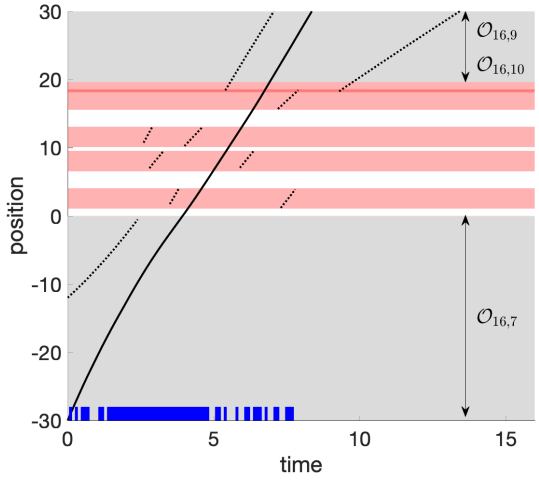


Fig. 9. Trajectory of vehicle 16 through two rear-end conflict areas ($\mathcal{O}_{16,9}$, coinciding with $\mathcal{O}_{16,10}$, and $\mathcal{O}_{16,7}$) and five side conflict areas. The dotted lines represent the trajectories of other vehicles that share the same rear-end or side conflict area. The time axis is colored blue at times when the supervisor is overriding the drivers, so as to ensure that vehicles maintain a safe distance of 4 on the rear-end conflict areas and are not simultaneously present inside each side conflict area.

then refine them so that the discretized paths satisfy the specifications given in Section IV. Trajectories of one of the vehicles through the intersection are illustrated in Figs. 8 and 9, for the purpose of illustrating Theorem 1 and Corollary 1, respectively.

In Fig. 8, we present a single trajectory of the abstraction $\Sigma_a(\gamma)$ computed at time step 0 by solving the MIP-AVP, and the corresponding trajectory of the concrete system Σ_c. Notice that the position trajectory of the concrete system can track the position trajectory of the abstraction within $\varepsilon = 1$, which confirms Theorem 1.

In Fig. 9, we run the supervisor algorithm until all vehicle exit the controlled region and present the resulting trajectory (solid black line) of vehicle 16 through two rear-end conflict areas (gray regions) and five side conflict areas (red regions between position 0 and 20). The dotted lines represent segments of trajectories of other vehicles that share the same conflict areas. Notice that due to the overrides of the supervisor (marked in blue on the time axis), vehicle 16 slows down before entering the intersection until around time 4.5 and then accelerates inside the intersection, thereby avoiding collisions inside the conflict areas; the trajectory would be a concave line without any overrides of the supervisor because the requested input at each time step is set to be 0. This confirms Corollary 1.

B. Computation Time and Restrictiveness

1) Effect of the Number of Vehicles: Fig. 10 shows the maximum computation time required to solve the MIP-AVP in the scenario of Fig. 1, as the number of vehicles is increased from 8 to 20. The scenario of 8 vehicles represents, for example, the arrangement of vehicles 1 through 8 in Fig. 1. The total number of segments used in each scenario is between 208 and 769 at the first iteration of Algorithm 1. These values vary at each iteration, depending on a given state of vehicles. The MIP-AVP takes less than 0.1 s for 15 vehicles and less than 0.46 s for 20 vehicles. The computation time can possibly be reduced by using more powerful machines and more efficient programming language, and by using a larger discretization size $\mathbf{\Delta}\xi_j[k]$ (detailed in the next section).

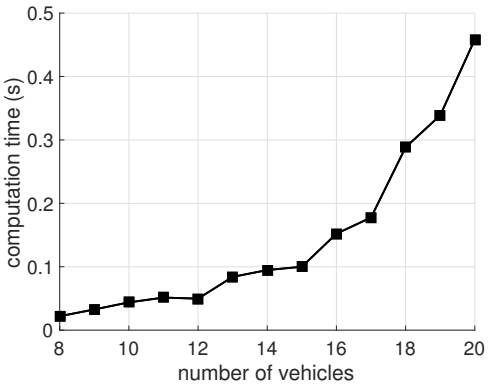


Fig. 10. Maximum computation time for solving the MIP-AVP in Algorithm 1 as the number of vehicles increases.

Also, a supervisor running at a longer time step of 0.2 to 0.5 s is usually acceptable in many scenarios, although 0.1 s is the typical time step in Intelligent Transportation System applications [1].

2) Effect of the Design Parameters : In the design of the supevisor, designers can freely choose the step sizes $\mathbf{\Delta}\xi_j[k]$ and ε depending on required system performances. The other parameters, such as the smoothing function $g_{j,k}$ and $\gamma_{j,k}$, are then determined as functions of $\mathbf{\Delta}\xi_j[k]$ and ε. Here, we show the effects of these independent design parameters on the restrictiveness and computation time of the supervisor.

Let $\mathbf{\Delta}\xi_{\max}$ be the maximum size of the step sizes $\mathbf{\Delta}\xi_j[k]$ in the MIP-AVP. The choice of $\mathbf{\Delta}\xi_{\max}$ significantly affects the computation time and the restrictiveness of the supervisor. To numerically show the effects of step sizes on the supervisor performances, we select a set of 1,000 random initial states of 8 vehicles (arranged as vehicles 1 through 8 in Fig. 1) and solve the MIP-AVP for each initial state with various step

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2953213, IEEE Transactions on Automatic Control

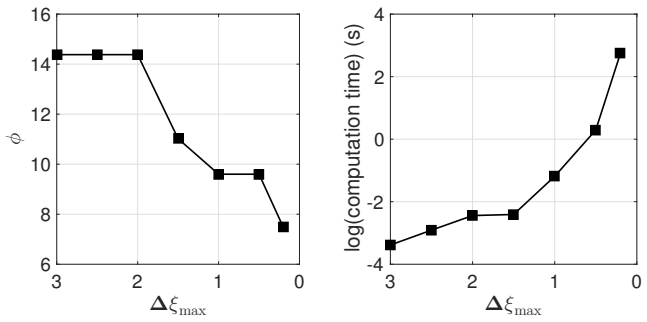


Fig. 11. As the maximum discrete step size ($\mathbf{\Delta}\xi_{\max}$) is reduced, the upper bound ϕ of the restrictiveness of the supervisor decreases while computation time increases.

sizes from 0.2 to 3. The maximum computation time and an upper bound of restrictiveness are illustrated in Fig. 11 as a function of $\mathbf{\Delta}\xi_{\max}$. The maximum number of segments used in the MIP-AVP is in total 254, 310, 352, 434, 646, 1206, 2898 when $\mathbf{\Delta}\xi_{\max} = 3, 2.5, 2, 1.5, 1, 0.5, 0.2$, respectively.

Fig. 11(a) shows an upper bound ϕ of the restrictiveness of the supervisor. Here, the upper bound ϕ decreases as $\mathbf{\Delta}\xi_{\max}$ decreases. Recall that by Definition 3, the supervisor is at most ϕ-restrictive if its override implies that there is no input signal that avoids entering the inflated bad set $B^*(\phi)$. For example, $\phi = 8$ tells that when the supervisor intervenes, vehicles in the same lane will get closer than $16 + d_{\min}$ in the future or will not be able to maintain distance 8 away from any side conflict area that is currently occupied by a conflicting vehicle. The decreasing ϕ indicates the reduced inflation of the bad set and thus, a less restrictive supervisor.

In Fig. 11(b), the computation time increases (note that y-axis is in a logarithmic scale) as the step size decreases. Notice that Fig. 11 enables the selection of the step size based on performance requirements; for example, if an alloted time for computation is shorter than $\exp(-2) = 0.13\,\mathrm{s}$, a step size larger than 1.5 can be chosen to obtain fast computation time at the expense of restrictiveness.

We observe that ε has a milder effect on the computation time and ϕ of the supervisor. This is because small ε leads to the reduced inflation of the bad set $B^*(\varepsilon)$, but at the same time, affects the smoothing function $g_{j,k}$ (discussed in Appendix A), thereby restricting the allowed velocity changes of $\Sigma_a(\boldsymbol{\gamma})$. In the simulations, we choose the value of ε that is of the same order of magnitude as the step sizes; we use $\varepsilon = 1$ for $\mathbf{\Delta}\xi_{\max} = 3, 2.5, 2, 1.5$, $\varepsilon = 0.25$ for $\mathbf{\Delta}\xi_{\max} = 1, 0.5$, and $\varepsilon = 0.1$ for $\mathbf{\Delta}\xi_{\max} = 0.2$.

C. Performance Comparison

In this section, we compare the computational complexity with other approaches, in terms of the number of binary variables appearing in the mixed-integer programming (MIP) formulations. In particular, we focus on discussing the advantages and disadvantages of time discretization and space discretization to justify our choice of the solution strategy. We also present the simulation results of the comparison of the two discretization schemes.

Other previous works [10]–[14] require $O(n(n-1))$ binary variables to write the coordination problem as an MIP formulation, whereas our approach requires $O(mn(n-1))$ binary variables. This is because the previous works concern collision avoidance at a single conflict area, which means $m = 1$. Notice that the number of binary variables appearing in our MIP formulation is different only by the factor of the number of conflict areas m. Moreover, the methods presented in the previous works do not apply to the more complex scenario of this paper unless the whole intersection is lumped into a single large conflict area, which can be occupied by only one vehicle at a time. The more complex scenario with multiple conflict areas is considered in [25] and [18]. However, [25] handles only first-order vehicle dynamics, and [18] adopts a time discretization scheme which results in an MIP formulation that requires a much larger number of binary variables. In the following, we explain why the time discretization-based MIP formulation requires more binary variables and what the advantage of time discretization is. Also, via computer simulation, we compare the performance of our space discretization-based approach with the performance of the time discretization-based approach.

In the case of time discretization, let $\mathbf{x}[k]$ be a decision variable where $\mathbf{x}[k]$ is the position at time step k (i.e., at time $k\Delta t$). The constraints of rear-end collision avoidance and side collision avoidance respectively take the following form:

- if $\mathcal{P}_j(x_j[k]), \mathcal{P}_{j'}(x_{j'}[k]) \in \mathcal{O}_{j,j'}$, then $x_j[k] - x_{j'}[k] \geq d$;
- if vehicle j precedes vehicle j' at side conflict area i and if $\mathcal{P}_j(x_j[k]) \in \mathcal{I}_i \backslash \mathcal{O}_{j,j'}$, then $\mathcal{P}_{j'}(x_{j'}[k+1]) \notin \mathcal{I}_i \backslash \mathcal{O}_{j,j'}$.

The first constraint tells that, if two vehicles are in the rear-end conflict area, the distance between them at $k\Delta t$ must be no smaller than the minimum safe distance d. The second one tells that if vehicle j is inside the side conflict area at time step k, then vehicle j' must not be inside the side conflict area at the next time step.

Note that the conditional statements in both constraints require the introduction of binary variables. The truth of "if $\mathcal{P}_j(x_j[k]) \in \mathcal{O}_{j,j'}$" or "if $\mathcal{P}_j(x_j[k]) \in \mathcal{I}_i \backslash \mathcal{O}_{j,j'}$" is represented by a binary variable for each vehicle j at each time step k, for each conflict area. We can expect $O(Nnm)$ binary variables, where N is the number of discretization steps, n the number of vehicles, and m the number of side conflict areas. Also, the truth of "if vehicle j precedes vehicle j' at side conflict area i" is represented by a binary variable for each pair of vehicles at each side conflict area, which results in a total of $O(mn(n-1))$ binary variables. In particular, the total number of binary variables is nearly proportional to the length of the time horizon N, whose lower bound is set by the minimum velocity allowed for vehicles in the controlled region, and may go to infinity if vehicles are allowed to stop within the controlled region. On the other hand, the space discretization scheme requires a total of $O(mn(n-1))$ binary variables which is independent of the horizon length.

Time discretization has some advantages over space discretization in terms of modeling flexibility. Indeed, for time discretization, linear dynamics are enough to insure linear constraints in the optimization problem, and therefore to admit a formulation as a mixed-integer *linear* program, with a linear

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2953213, IEEE Transactions on Automatic Control

TABLE I
PERFORMANCE OF THE MIP-AVP AND THE TIME DISCRETIZATION-BASED APPROACH.

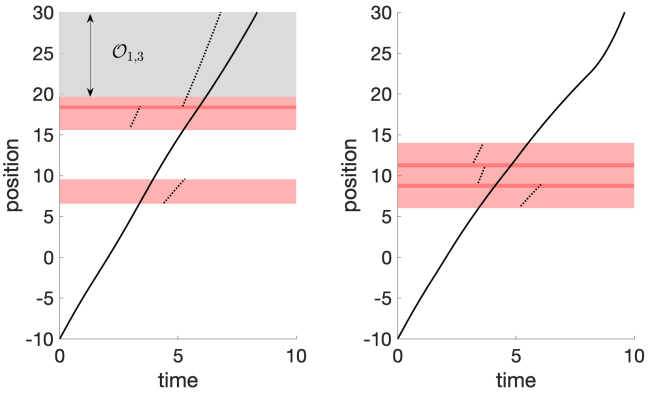
	$\boldsymbol{\Delta}\xi_{\max}=3$	$\boldsymbol{\Delta}\xi_{\max}=1$	$\boldsymbol{\Delta}\xi_{\max}=0.5$	$\Delta t=1$	$\Delta t=0.5$
Accepted States (%)	79.8	95	96	90.6	96.2
CPU Time (s)	0.22	0.36	1.70	1.93	5.15

cost function. Space discretization, on the other hand, only allows a linear program formulation for first order dynamics. In the light of our need for a computationally light MIP formulation, this may appear as a deal-breaker. However, by discretizing the abstraction $\Sigma_a(\boldsymbol{\gamma})$ rather than the concrete system Σ_c, and then using the approximate simulation relation to link the resulting trajectories with trajectories of the non-linear concrete system Σ_c, we can enjoy the computational advantages of the space discretization, without significant limitations on the model dynamics.

We compare the performances of the MIP-AVP and the safety verification formulation given in [18], in terms of two measures: computation time and ratio of accepted states (i.e., leading to feasible solutions). To evaluate the ratio, we estimate the fraction of a given set of 500 random initial states that is classified as the accepted states by solving the MIP-AVP and the time discretization-based verification. The original verification problem (VP) yields the ratio of accepted states that is larger than that of discretization-based verification problems, and thus, how large this measure is indicates how close each verification problem is to the VP. For computation time, we measure the maximum CPU time taken to solve each verification problem.

In the comparison, we consider linear vehicle dynamics $\ddot{x}_j = u_j$, which is the same as (2) with $c_1 = c_2 = 0$ and $c_3 = 1$. This is because the time discretization approach requires the assumption of linear vehicle dynamics, while our approach can handle nonlinearity of vehicle dynamics by using the abstraction and sliding mode control, which is detailed in Appendix A.

In Table I, we present the results of our approach (MIP-AVP) on three different discrete step sizes and the results of the time discretization-based approach on two different step sizes. When solving the time discretization-based verification problem, we select $N = \lceil \max_j (x_{j,\max} - x_j(0))/(\dot{x}_{j,\min}\Delta t) \rceil$ and impose the constraints described in [18] with a zero cost function. The zero cost function is used because CPLEX tends to solve mixed integer problems with constant cost functions more quickly than those with nonconstant cost function. Notice from the comparison between space discretization with $\boldsymbol{\Delta}\xi_{\max} = 0.5$ and time discretization with $\Delta t = 0.5$ that the two verification approaches find feasible solutions on the similar number of states (96% and 96.2% of the total number of states in the random set, respectively), while time discretization takes significantly longer to complete. This large computation time is mainly due to the number of required binary variables; in these simulations, 24 binary variables are involved in our approach, independent of step sizes, whereas 3160 and 6296 binary variables are involved in the time discretization-based approach with $\Delta t = 1$ and 0.5, respectively. The performance comparison shows that our approach is more than 3 times computationally faster than the time discretization-based approach, while similarly close to the VP in terms of the ratio of accepted states.



(a) Path of turning left (b) Path of going straight

Fig. 12. When the path of vehicle 1 is undetermined before entering the controlled region, we solve Problem 1 so that its solution ensures collision avoidance in rear-end and side conflict areas of all possible future paths. The same coloring is used as in Fig.9.

D. Extension to Undetermined Vehicle Paths

Our approach can handle multiple possible future paths, when a unique path cannot be identified before a vehicle enters the controlled region. In the simulations, we consider the scenario of 8 vehicles, with the same arrangement of vehicles 1-8 in Fig. 1.

When the path of vehicle 1 is unknown, we simultaneously consider all possible paths (two paths in the scenario), as if there are two vehicles starting from the same state but traversing the intersection along different paths. To do this, we solve the following problem:

Problem 1 (MIP-AVP with Undetermined Paths)**.** Given a state $\mathbf{s}_\tau$, determine the feasibility of the following problem:

$$\begin{aligned} \min \quad & J(\boldsymbol{\Delta}\mathbf{t}_1^1, \boldsymbol{\Delta}\mathbf{t}_1^2, \boldsymbol{\Delta}\mathbf{t}_j, \forall j \in \{2,\dots,n\}), \\ \text{subject to} \quad & \boldsymbol{\Delta}\mathbf{t}_1^1[k] = \boldsymbol{\Delta}\mathbf{t}_1^2[k], \forall k \in \{1,\dots,\bar{K}\} \\ & (7),(8),(9),(10). \end{aligned} \tag{20}$$

Here, we discretize each path of vehicle 1 and find feasible time sequences $\boldsymbol{\Delta}\mathbf{t}_1^1$ for path option 1 and $\boldsymbol{\Delta}\mathbf{t}_1^2$ for path option 2. A feasible solution of (20) should satisfy $\boldsymbol{\Delta}\mathbf{t}_1^1[k] = \boldsymbol{\Delta}\mathbf{t}_1^2[k]$ until vehicle 1 reaches the intersection, which is represented by path segments k for $k \in \{1,\dots,\bar{K}\}$, because the two paths are identical before the intersection. This modification is accompanied by additional computational complexity; that is, considering two possible paths for a vehicle increases computational complexity as if one additional vehicle is considered.

In Fig. 12, the position trajectories of vehicle 1 (solid lines) along the two different paths are shown. The left-turn path consists of one rear-end conflict area (gray region $\mathcal{O}_{1,3}$) and three side conflict areas (red regions), and the straight path consists of three side conflict areas. Note that in both options,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2953213, IEEE Transactions on Automatic Control

vehicles are not simultaneously inside the side conflict area and maintain a safe distance of 4 in the rear-end conflict area. Also, the position trajectories of vehicle 1 along the two paths are identical until they reach the intersection located between position 0 and 20. By finding a solution to Problem 1, we ensure that starting from a state $\mathbf{s}_\tau$, there is an input signal that allows all vehicles to cross the intersection without conflict, no matter what path vehicle 1 will choose in the future.

VIII. Conclusion

We designed a supervisor that takes control of vehicles when necessary to prevent side and rear-end collisions at busy road intersections. This supervisor significantly improves our previous results [10]–[16] in that it now handles more complicated intersection scenarios including multiple conflict areas, road junctions, and merging lanes. To make it computationally acceptable, we introduced an abstraction that is approximately simulated by the concrete system with precision ε, and determined the timing of overrides by solving the abstraction-based verification problem. We showed that the supervisor guarantees no conflict within the controlled region, and we found an upper bound ϕ to supervisor restrictiveness. Through the computational experiments, we validated that our approach is computationally practical for busy intersection scenarios. The correctness of our result relies on the correct prediction of vehicles' states and perfect path following. Integrating robust prediction and path following, along the line of the result in [12], remains as future work.

Appendix A
Proof of Theorem 1

To prove Theorem 1, we start with designing a feedback law for the input of Σ_c, so as to track, with precision ε, a trajectory $(\mathbf{x}_a, \mathbf{v})$ given by (12). Though different feedback laws could be used, our choice is for sliding mode control, due to its robustness to model uncertainties, and error bound guarantees [30]. With reference to the functions f and b in (1), we set

$$u_j = \frac{1}{b(x_j,\dot{x}_j)}\left(-f(x_j,\dot{x}_j) - \eta \text{sat}\left(\frac{s(x_j,\dot{x}_j)}{\Phi}\right) - \lambda(\dot{x}_j - v_j)\right) \tag{21}$$

if $\dot{x}_j \in (\dot{x}_{j,\min}, \dot{x}_{j,\max})$, and $u_j = -f(x_j,\dot{x}_j)/b(x_j,\dot{x}_j)$ if $\dot{x}_j = \dot{x}_{j,\min}$ or $\dot{x}_j = \dot{x}_{j,\max}$. Here, the design parameters λ, η, and Φ are positive real numbers, and a scalar function s is

$$s(x_j,\dot{x}_j) := (\dot{x}_j - v_j) + \lambda\,(x_j - x_{a,j})\,.$$

The saturation function $\text{sat}\,(s/\Phi)$ is 1 if $s/\Phi > 1$, -1 if $s/\Phi < -1$, and s/Φ otherwise. With the input (21), if $\dot{x}_j \in (\dot{x}_{j,\min}, \dot{x}_{j,\max})$, $|s(x_j,\dot{x}_j)|$ decreases in time because

$$\frac{1}{2}\frac{d}{dt}\,|s(x_j,\dot{x}_j)|^2 \le -\eta|s(x_j,\dot{x}_j)| \tag{22}$$

where we let $\dot{v}_j = 0$ by neglecting a finite number of points at which v_j jumps. Since we focus on a trajectory of an individual vehicle, we omit the subscript j for the sake of notational simplicity, which has been used to indicate vehicle j.

Using sliding mode control, we can easily quantify upper bounds of tracking errors $|x - x_a|$ and $|\dot{x} - v|$. If $|s(x(t),\dot{x}(t))| \le S$ for all $t \ge 0$, we have

$$|x(t) - x_a(t)| \le \frac{S}{\lambda}, \qquad |\dot{x}(t) - v(t)| \le 2S, \tag{23}$$

for all $t \ge 0$. If there is an input signal for the concrete system Σ_c that $|x(t) - x_a(t)| \le \varepsilon$ for all t, for any x_a, then we prove the ε-approximate simulation relation of $\Sigma_a(\boldsymbol{\gamma})$ by Σ_c. To enable the existence of such an input signal, we provide some conditions of λ, η, Φ and the smoothing function $g_{j,k}$.

Let parameters Φ and λ be constant over all segments, and parameter η be constant at η_k on the k-th segment and vary between segments. Let $\boldsymbol{\Delta}\mathbf{v}[k]$ denote the velocity difference between two consecutive segments $k-1$ and k of the abstraction trajectory, that is,

$$\boldsymbol{\Delta}\mathbf{v}[k] := v[k-1] - v[k] = \frac{\boldsymbol{\Delta}\xi[k-1]}{\boldsymbol{\Delta}\mathbf{t}[k-1]} - \frac{\boldsymbol{\Delta}\xi[k]}{\boldsymbol{\Delta}\mathbf{t}[k]}.$$

For $k = 1$, set $\boldsymbol{\Delta}\mathbf{v}[1] := \dot{x}_\tau - \boldsymbol{\Delta}\xi[1]/\boldsymbol{\Delta}\mathbf{t}[1]$.

Condition 1. The design parameters η and λ satisfy the following:

- $\eta = \eta_k \ge \max(|\boldsymbol{\Delta}\mathbf{v}[k-1]|, |\boldsymbol{\Delta}\mathbf{v}[k]|)/\boldsymbol{\Delta}\mathbf{t}[k]$ on k-th segment;
- $\lambda \ge (\Phi + \max_k |\boldsymbol{\Delta}\mathbf{v}[k]|)/\varepsilon$.

For $k = 1$, the condition is that $\eta_1 \ge |\boldsymbol{\Delta}\mathbf{v}[1]|/\boldsymbol{\Delta}\mathbf{t}[1]$.

We prove the lemma that Condition 1 ensures the position tracking error bounded by ε.

Lemma 1. *If η and λ satisfy Condition 1, the sliding mode control input* (21) *makes tracking error $|x(t) - x_a(t)|$ bounded by ε for all $t \in [0, \sum_{k=1}^{N} \boldsymbol{\Delta}\mathbf{t}[k]]$.*

Proof. We will prove that $|s(x,\dot{x})| \le \Phi + \max(|\boldsymbol{\Delta}\mathbf{v}[k-1]|, |\boldsymbol{\Delta}\mathbf{v}[k]|)$ on each k-th segment by mathematical induction on k. On the first segment, $|s(x,\dot{x})| \le |\boldsymbol{\Delta}\mathbf{v}[1]|$ because $x_a(0) = x_\tau, |\dot{x}_\tau - v(0)| = |\boldsymbol{\Delta}\mathbf{v}[1]|$, and $|s(x,\dot{x})|$ does not increase in time on the segment by (22). Suppose $|s(x,\dot{x})| \le \Phi + \max(|\boldsymbol{\Delta}\mathbf{v}[k-2]|, |\boldsymbol{\Delta}\mathbf{v}[k-1]|)$ on the $(k-1)$-th segment. We consider three cases depending on whether $\dot{x}$ reaches the minimum or maximum velocity.

If $\dot{x} \in (\dot{x}_{\min}, \dot{x}_{\max})$, since $\eta_{k-1} \ge \max(|\boldsymbol{\Delta}\mathbf{v}[k-2]|, |\boldsymbol{\Delta}\mathbf{v}[k-1]|)/\boldsymbol{\Delta}\mathbf{t}[k-1]$ and $|s(x,\dot{x})|$ linearly decreases in time at rate η_{k-1}, we have $|s(x,\dot{x})| \le \Phi$ at the end of the $(k-1)$-th segment. Because between the $(k-1)$-th and k-th segments, the velocity of the abstraction trajectory changes by $\boldsymbol{\Delta}\mathbf{v}[k]$ and the position does not change instantaneously, $|s(x,\dot{x})|$ increases by at most $|\boldsymbol{\Delta}\mathbf{v}[k]|$, that is, $|s(x,\dot{x})| \le \Phi + |\boldsymbol{\Delta}\mathbf{v}[k]|$. Since $|s(x,\dot{x})|$ does not increase in time on a segment, $|s(x,\dot{x})| \le \Phi + |\boldsymbol{\Delta}\mathbf{v}[k]|$ on the k-th segment.

Suppose $v[k-1] = \boldsymbol{\Delta}\xi[k-1]/\boldsymbol{\Delta}\mathbf{t}[k-1] = \dot{x}_{\min}$ and $\dot{x}$ starts to reach $\dot{x}_{\min}$ in the middle of the $(k-1)$-th segment. In this case, $\boldsymbol{\Delta}\mathbf{v}[k-1] \ge 0$ and $\boldsymbol{\Delta}\mathbf{v}[k] \le 0$. Also, we have $|s(x,\dot{x})| \le \Phi$ at the end of the $(k-2)$-th segment because of the same reason above for the case of $\dot{x} \in (\dot{x}_{\min}, \dot{x}_{\max})$. Thus, at the beginning of the $(k-1)$-th segment, we have

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2953213, IEEE Transactions on Automatic Control

either $-\Phi + \mathbf{\Delta v}[k-1] < s(x,\dot{x}) < 0$ or $0 < s(x,\dot{x}) < \Phi + \mathbf{\Delta v}[k-1]$. In the former, $|-\Phi + \mathbf{\Delta v}[k-1]| \le \Phi$ and thus $|s(x,\dot{x})| \le \Phi$ over the segment regardless of $\dot{x}$ because the magnitude $|s(x,\dot{x})|$ is not increasing in time. Thus, in this case, $|s(x,\dot{x})| \le \Phi + |\mathbf{\Delta v}[k]|$ on the k-th segment. In the latter, $s(x,\dot{x})$ decreases until $\dot{x}$ reaches $\dot{x}_{\min}$ and remains constant at $\bar{s}$. At the beginning of the k-th segment, since $0 \le \bar{s} \le \Phi + \mathbf{\Delta v}[k-1]$,

$$\mathbf{\Delta v}[k] \le s(x,\dot{x}) \le \Phi + \mathbf{\Delta v}[k-1] + \mathbf{\Delta v}[k].$$

Therefore, on the k-th segment, $|s(x,\dot{x})| \le \Phi + \max(|\mathbf{\Delta v}[k]|, |\mathbf{\Delta v}[k-1]|)$ because $|\mathbf{\Delta v}[k-1] + \mathbf{\Delta v}[k]| \le \max(|\mathbf{\Delta v}[k]|, |\mathbf{\Delta v}[k-1]|)$ and $|s(x,\dot{x})|$ does not increase in time.

When $v[k-1] > \dot{x}_{\min}$ and $\dot{x} = \dot{x}_{\min}$ in the middle of the $(k-1)$-th segment, $s(x,\dot{x})$ does not remain constant at $\bar{s}$ but decreases further because $\dot{s} = \lambda(\dot{x} - v) < 0$. Thus, the same conclusion as above can be made. The proof when $\mathbf{\Delta}\xi[k-1]/\mathbf{\Delta t}[k-1] = \dot{x}_{\max}$ follows the same procedure.

This concludes the proof because by (23)

$$|x(t) - x_a(t)| \le \frac{\Phi + \max(|\mathbf{\Delta v}[k-1]|, \mathbf{\Delta v}[k])}{\lambda}$$

for $t \in \left[\sum_{i=1}^{k-1} \mathbf{\Delta t}[i], \sum_{i=1}^{k} \mathbf{\Delta t}[i]\right)$, which becomes $|x(t) - x_a(t)| \le \varepsilon$ since $\lambda \ge (\Phi + \max_k |\mathbf{\Delta v}[k]|)/\varepsilon$ by Condition 1. □

However, parameters η and λ cannot be made arbitrarily large to satisfy Condition 1 because the magnitude of control input (21) increases with respect to η and λ, and should be bounded by $[u_{\min}, u_{\max}]$. By appropriately choosing smoothing function g_k to restrict velocity changes of x_a, we can enforce the input bound constraint. We provide a condition for smoothing function g_k.

Condition 2. Smoothing function g_k at the k-th segment is a map from $[\mathbf{\Delta}\xi[k]/v_{\max}[k], \mathbf{\Delta}\xi[k]/v_{\min}[k]]$ to $\mathbb{R}_+$ and satisfies

$$\frac{1}{b(x,\dot{x})}\left(-f(x,\dot{x}) \pm \frac{\max(\bar{g}_{k-1,\max}, \bar{g}_k)}{\mathbf{\Delta t}[k]} \pm \frac{2(\Phi + \bar{g}_{\max})^2}{\varepsilon}\right) \in [u_{\min}, u_{\max}]$$

for all possible $x \in X$ and $\dot{x} \in \dot{X}$ and for all $\mathbf{\Delta t}[k] \in [\mathbf{\Delta}\xi[k]/v_{\max}[k], \mathbf{\Delta}\xi[k]/v_{\min}[k]]$ where

$$\bar{g}_k := \frac{g_k(\mathbf{\Delta t}[k])}{\mathbf{\Delta t}[k-1]\mathbf{\Delta t}[k]}, \quad \bar{g}_{k-1,\max} := \max_{\mathbf{\Delta t}[k-1], \mathbf{\Delta t}[k-2]} \bar{g}_{k-1},$$

and $\bar{g}_{\max} = \max_k \bar{g}_{k,\max}$, for all $\mathbf{\Delta t}[k-1]$ satisfying $|\mathbf{\Delta}\xi[k-1] \cdot \mathbf{\Delta t}[k] - \mathbf{\Delta}\xi[k] \cdot \mathbf{\Delta t}[k-1]| \le g_k(\mathbf{\Delta t}[k])$ and $\mathbf{\Delta t}[k-2]$ satisfying $|\mathbf{\Delta}\xi[k-2] \cdot \mathbf{\Delta t}[k-1] - \mathbf{\Delta}\xi[k-1] \cdot \mathbf{\Delta t}[k-2]| \le g_{k-1}(\mathbf{\Delta t}[k-1])$. For $k = 1$, set $\bar{g}_1$ as $g_1(\mathbf{\Delta t}[1])/\mathbf{\Delta t}[1]$ and $\bar{g}_{0,\max}$ as 0.

Since the smoothing function g_k appears in the MIP-AVP, we should determine g_k before solving the problem. This is why g_k should satisfy the inequalities in Condition 2 for all possible solutions. The set of smoothing functions g_k that satisfy Condition 2 is nonempty because if $g_k \equiv 0$ for all k,

$$u_{\min} \le \frac{1}{b(x,\dot{x})}\left(-f(x,\dot{x}) - \frac{2\Phi^2}{\varepsilon}\right),$$
$$u_{\max} \ge \frac{1}{b(x,\dot{x})}\left(-f(x,\dot{x}) + \frac{2\Phi^2}{\varepsilon}\right),$$

which are true because ε is nonzero, Φ is arbitrarily small, and $-f(x,\dot{x})/b(x,\dot{x}) \in (u_{\min}, u_{\max})$ for all $x \in X, \dot{x} \in \dot{X}$ by assumption. Therefore, $g_k \equiv 0$ is in the set.

We prove in the following lemma that if g_k satisfies Condition 2, we can find η and λ satisfying Condition 1 for any solution $(\mathbf{\Delta t}_j, \forall j)$ such that the sliding mode input (21) lies in $\mathcal{U}$.

Lemma 2. *Suppose smoothing function g_k satisfies Condition 2. For any solution satisfying the dynamics constraints* (7) *and* (8) *in the MIP-AVP, there exist design parameters η and λ that satisfy Condition 1 and make the sliding mode control input* (21) *bounded by* $[u_{\min}, u_{\max}]$.

Proof. Given any solution $\mathbf{\Delta t}$ satisfying (7) and (8) of the MIP-AVP, Condition 1 is satisfied if

$$\eta_k = \max\left(\frac{g_{k-1}(\mathbf{\Delta t}[k-1])}{\mathbf{\Delta t}[k-2]\mathbf{\Delta t}[k-1]}, \frac{g_k(\mathbf{\Delta t}[k])}{\mathbf{\Delta t}[k-1]\mathbf{\Delta t}[k]}\right) \cdot \frac{1}{\mathbf{\Delta t}[k]}$$

and

$$\lambda = \frac{1}{\varepsilon}\left(\Phi + \max_k \frac{g_k(\mathbf{\Delta t}[k])}{\mathbf{\Delta t}[k-1]\mathbf{\Delta t}[k]}\right)$$

because from smoothing constraint (8),

$$|\mathbf{\Delta v}[k]| \le \frac{g_k(\mathbf{\Delta t}[k])}{\mathbf{\Delta t}([k-1])\mathbf{\Delta t}[k]}.$$

Thus, by Lemma 1, the tracking error $|x - x_a|$ is bounded by ε, and $|\dot{x} - v| \le 2\varepsilon\lambda$. Then the control input (21) is bounded by the expression $1/b(-f \pm \max(\bar{g}_{k-1,\max}, \bar{g}_k)/\mathbf{\Delta t}[k] \pm 2(\Phi + \bar{g}_{\max})^2/\varepsilon)$ in Condition 2, which implies that the input is bounded by $[u_{\min}, u_{\max}]$. □

We can numerically obtain smoothing function g_k satisfying Condition 2 before solving the MIP-AVP. The following example shows a process to obtain a sequence of smoothing functions g_k.

Example 1. Consider the vehicle dynamics $\ddot{x} = -0.005\dot{x}^2 + u$, which is the same as (2) with $c_1 = 0.005, c_2 = 0, c_3 = 1$. Suppose $\mathbf{\Delta}\xi[k] = 3$ and $[v_{\min}[k], v_{\max}[k]] = [1, 15]$ for all k, $\varepsilon = 1$, $[u_{\min}, u_{\max}] = [-3, 3]$, and $\Phi = 0.001$. We want to find linear smoothing functions $g_k(\mathbf{\Delta t}[k]) = c_1\mathbf{\Delta t}[k] + c_2$ that satisfy Condition 2.

Suppose that the velocity decreases between the $(k-1)$-th and the k-th segments, in which case the lower bound of the control input is important to ensure. We first find g_k that satisfies

$$-3 \le 0.005\dot{x}^2 - \frac{g_k(\mathbf{\Delta t}[k])}{\mathbf{\Delta t}[k-1]\mathbf{\Delta t}[k]^2} - 2\left(0.001 + \bar{g}_{\max}\right)^2,$$

where the comparison with $\bar{g}_{k-1,\max}$ is not yet considered. From smoothing constraint (8), $\mathbf{\Delta}\xi[k-1] \cdot \mathbf{\Delta t}[k] - \mathbf{\Delta}\xi[k] \cdot \mathbf{\Delta t}[k-1] \le g_k(\mathbf{\Delta t}[k])$, thereby implying $\mathbf{\Delta t}[k-1] \ge$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2953213, IEEE Transactions on Automatic Control

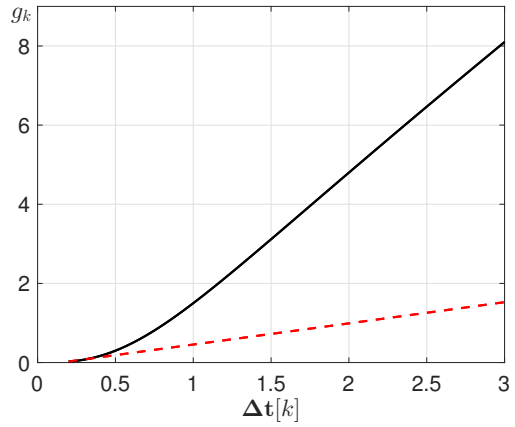


Fig. 13. With $\bar{g}_{\max} = 0$, the smoothing function that satisfies (24) with equality is represented by the black solid line, and a linear smoothing function that satisfies (24) by the red dotted line.

$\Delta\mathbf{t}[k] - \frac{1}{3}g_k(\Delta\mathbf{t}[k])$ in the decelerating case. Also, we have $\dot{x} \geq 3/\Delta\mathbf{t}[k] - 2(\Phi + \bar{g}_{\max})$ on the k-th segment because $\dot{x}$ decreases over the segment and satisfies $|\dot{x} - v| \leq 2(\Phi + \bar{g}_{\max})$ where $v = 3/\Delta\mathbf{t}[k]$. In the above inequality, we substitute $\dot{x}$ with $3/\Delta\mathbf{t}[k] - 2(\Phi + \bar{g}_{\max})$, and $\Delta\mathbf{t}[k-1]$ with $\Delta\mathbf{t}[k] - \frac{1}{3}g_k(\Delta\mathbf{t}[k])$:

$$-3 \leq 0.005\left(\frac{3}{\Delta\mathbf{t}[k]} - 2(0.001 + \bar{g}_{\max})\right)^2 - \frac{g_k}{(\Delta\mathbf{t}[k] - \frac{1}{3}g_k)\Delta\mathbf{t}[k]^2} - 2\left(0.001 + \bar{g}_{\max}\right)^2. \quad (24)$$

Let $\bar{g}_{\max} = 0$. The smoothing function g_k that satisfies (24) with equality is depicted as a solid black line in Fig. 13. Any function that lies below this black line satisfies (24). In the figure, the red dotted line represents the linear smoothing function $0.4761\Delta\mathbf{t}[k] - 0.0751$, which satisfies (24), is non-negative for all $\Delta\mathbf{t}[k] \in [\Delta\xi[k]/v_{\max}[k], \Delta\xi[k]/v_{\min}[k]]$, and maximizes the trapezoidal area below the linear function. Similarly, for an increasing velocity, $g_k(\Delta\mathbf{t}[k]) = 0.4560\Delta\mathbf{t}[k] - 0.0777$. With these two linear smoothing functions, the maximum velocity change is $\bar{g}_{k,\max} = 0.8217$. With $\bar{g}_{\max} = 0.8217$, we again find $g_k = 0.2708\Delta\mathbf{t}[k] - 0.0429$ for the decelerating case and $0.1958\Delta\mathbf{t}[k] - 0.0354$ for the accelerating case. These new smoothing functions yield the maximum velocity change of 0.4474, which is smaller than the value (0.8217) used to find the functions. Using these smoothing functions, we check if

$$-3 \leq 0.005\left(\frac{3}{\Delta\mathbf{t}[k]} - 2(0.001 + \bar{g}_{\max})\right)^2 - \frac{\bar{g}_{k-1,\max}}{\Delta\mathbf{t}[k]} - 2\left(0.001 + \bar{g}_{\max}\right)^2 \quad (25)$$

for the decreasing case, and a similar inequality for the increasing case. In this example, (25) holds, and thus, the constraint (8) becomes

$$\begin{aligned}\Delta\xi[k-1]\cdot\Delta\mathbf{t}[k] - \Delta\xi[k]\cdot\Delta\mathbf{t}[k-1] &\leq 0.2708\Delta\mathbf{t}[k] - 0.0429,\\ -\Delta\xi[k-1]\cdot\Delta\mathbf{t}[k] + \Delta\xi[k]\cdot\Delta\mathbf{t}[k-1] &\leq 0.1958\Delta\mathbf{t}[k] - 0.0354.\end{aligned}$$

If (25) does not hold, lower the value of $\bar{g}_{k-1,\max}$ by recomputing smoothing function g_{k-1} (e.g., finding $\epsilon \in [0,1]$ such that ϵg_{k-1} yields the appropriate value of $\bar{g}_{k-1,\max}$).

Using the lemmas developed in this appendix, we now prove Theorem 1.

Theorem 1. *For any $\varepsilon > 0$, there exists $\boldsymbol{\gamma}$ such that $\Sigma_a(\boldsymbol{\gamma}) \preceq_\varepsilon \Sigma_c$.*

Proof. For any $\varepsilon > 0$, we can find smoothing function $g_{j,k}$ satisfying Condition 2 (e.g., following the procedure given in Example 1), and this gives a vector of function sequences $\boldsymbol{\gamma}$ as in (8). Because $g_{j,k}$ satisfies Condition 2, by Lemma 2, for any $(\Delta\mathbf{t}_j, \forall j)$ that satisfies the dynamical constraints (7) and (8) of the MIP-AVP, there exist η and λ that satisfy Condition 1 and makes the sliding mode feedback control input (21) lie in the set $\mathcal{U}$. By Lemma 1, this input signal corresponds to the trajectory $\mathbf{x}$ of Σ_c such that $|x_j(t) - x_{a,j}(t)| \leq \varepsilon$ for all j for all t, where $\mathbf{x}_a$ is the position trajectory of $\Sigma_a(\boldsymbol{\gamma})$ represented by $(\Delta\mathbf{t}_j, \forall j)$ that satisfies (7) and (8) by the map (12). By Definition 4, this completes the proof. □

ACKNOWLEDGMENT

The authors would like to thank Professor Domitilla Del Vecchio at MIT for her constructive comments and persistent support in developing the approach presented in this paper.

REFERENCES

[1] U.S. Department of Transportation, "ITS Strategic research plan 2015-2019," http://www.its.dot.gov/strategicplan.pdf, 2014.
[2] D. Miculescu and S. Karaman, "Polling-systems-based control of high-performance provably-safe autonomous intersections," in *Proc. IEEE Conf. on Decision and Control*, Dec. 2014, pp. 1417–1423.
[3] L. Chen and C. Englund, "Cooperative intersection management: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 2, pp. 570–586, Feb. 2016.
[4] R. Tachet, P. Santi, S. Sobolevsky, L. I. Reyes-Castro, E. Frazzoli, D. Helbing, and C. Ratti, "Revisiting street intersections using slot-based systems," *PLoS ONE*, vol. 11, no. 3, p. e0149607, Mar. 2016.
[5] E. R. Müller, R. C. Carlson, and W. K. Junior, "Intersection control for automated vehicles with MILP," *IFAC-PapersOnLine*, vol. 49, no. 3, pp. 37–42, 2016.
[6] A. I. Morales Medina, N. van de Wouw, and H. Nijmeijer, "Cooperative intersection control based on virtual platooning," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 6, pp. 1727–1740, Jun. 2018.
[7] M. W. Levin and D. Rey, "Conflict-point formulation of intersection control for autonomous vehicles," *Transportation Research Part C: Emerging Technologies*, vol. 85, pp. 528–547, Dec. 2017.
[8] F. Belkhouche, "Collaboration and optimal conflict resolution at an unsignalized intersection," *IEEE Trans. Intell. Transp. Syst.*, Oct. 2018.
[9] T. B. Sheridan, *Telerobotics, automation, and human supervisory control*. MIT press, 1992.
[10] A. Colombo and D. Del Vecchio, "Efficient algorithms for collision avoidance at intersections," in *Proc. Int. Conf. Hybrid Syst.: Computation and Control (HSCC)*, Apr. 2012, pp. 145–154.
[11] A. Colombo and D. Del Vecchio, "Least restrictive supervisors for intersection collision avoidance: A scheduling approach," *IEEE Trans. Autom. Control*, vol. 60, no. 6, pp. 1515–1527, Jun. 2015.

[12] L. Bruni, A. Colombo, and D. Del Vecchio, "Robust multi-agent collision avoidance through scheduling," in *Proc. IEEE Conf. Decision and Control (CDC)*, Dec. 2013, pp. 3944–3950.

[13] H. Ahn, A. Colombo, and D. Del Vecchio, "Supervisory control for intersection collision avoidance in the presence of uncontrolled vehicles," in *Proc. American Control Conf. (ACC)*, Jun. 2014, pp. 867–873.

[14] H. Ahn, A. Rizzi, A. Colombo, and D. Del Vecchio, "Experimental testing of a semi-autonomous multi-vehicle collision avoidance algorithm at an intersection testbed," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots and Syst. (IROS)*, Sep. 2015, pp. 4834–4839.

[15] H. Ahn and D. Del Vecchio, "Semi-autonomous intersection collision avoidance through job-shop scheduling," in *Proc. Int. Conf. Hybrid Syst.: Computation and Control (HSCC)*, Apr. 2016, pp. 185–194.

[16] H. Ahn and D. Del Vecchio, "Safety verification and control for collision avoidance at road intersections," *IEEE Trans. Autom. Control*, vol. 63, no. 3, pp. 630–642, Mar. 2018.

[17] F. Altché, X. Qian, and A. de La Fortelle, "Least restrictive and minimally deviating supervisor for safe semi-autonomous driving at an intersection: An MIQP approach," in *Proc. IEEE Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2016.

[18] ——, "An algorithm for supervised driving of cooperative semi-autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 12, pp. 3527–3539, Dec. 2017.

[19] M. A. S. Kamal, J. Imura, T. Hayakawa, A. Ohata, and K. Aihara, "A vehicle-intersection coordination scheme for smooth flows of traffic without using traffic lights," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 3, Sep. 2014.

[20] F. Altché, X. Qian, and A. de La Fortelle, "Time-optimal coordination of mobile robots along specified paths," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots and Syst. (IROS)*, Oct. 2016, pp. 5020–5026.

[21] A. Girard and G. J. Pappas, "Approximation metrics for discrete and continuous systems," *IEEE Trans. Autom. Control*, vol. 52, no. 5, pp. 782–798, May 2007.

[22] A. Girard, A. A. Julius, and G. J. Pappas, "Approximate simulation relations for hybrid systems," *Discrete Event Dynamic Syst.*, vol. 18, no. 2, pp. 163–179, Oct. 2007.

[23] A. Colombo and D. Del Vecchio, "Supervisory control of differentially flat systems based on abstraction," in *Proc. IEEE Conf. Decision and Control and European Control Conf. (CDC-ECC)*, Dec. 2011, pp. 6134–6139.

[24] E. Dallal, A. Colombo, D. Del Vecchio, and S. Lafortune, "Supervisory control for collision avoidance in vehicular networks using discrete event abstractions," in *Proc. American Control Conf. (ACC)*, Jun. 2013, pp. 4380–4386.

[25] ——, "Supervisory control for collision avoidance in vehicular networks using discrete event abstractions," *Discrete Event Dynamic Syst.*, vol. 27, no. 1, pp. 1–44, Mar. 2017.

[26] P. Lytrivis, G. Thomaidis, M. Tsogas, and A. Amditis, "An advanced cooperative path prediction algorithm for safety applications in vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 3, pp. 669–679, Sep. 2011.

[27] A. Colombo, G. R. de Campos, and F. D. Rossa, "Control of a city road network: Distributed exact verification of traffic safety," *IEEE Trans. Autom. Control*, vol. 62, no. 10, pp. 4933–4948, Oct 2017.

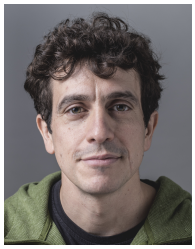
[28] D. Bresch-Pietri and D. Del Vecchio, "Estimation for decentralized safety control under communication delay and measurement uncertainty," *Automatica*, vol. 62, pp. 292–303, Dec. 2015.

[29] IBM Corporation, "CPLEX User's Manual," 2015. [Online]. Available: http://www.ibm.com/support/knowledgecenter/SSSA5P_12.6.3/ilog.odms.studio.help/pdf/usrcplex.pdf

[30] J. E. Slotine and W. Li, "Applied nonlinear control," *Prentice-Hall*, 1991.

Heejin Ahn received the B.S. degree in mechanical and aerospace engineering from Seoul National University, Seoul, South Korea, in 2012, and the S.M. and Ph.D. degrees in mechanical engineering from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2014 and 2018, respectively.

She was with Mitsubishi Electric Research Laboratories, Cambridge, MA, USA, as a Visiting Research Scientist from 2018 for two years. Her research interests include the analysis and control of hybrid dynamical systems for transportation and biomedical applications.



Alessandro Colombo received the Diplôme D'Ingénieur from ENSTA in Paris in 2005, and the Ph.D. from Politecnico di Milano in 2009. He was Postdoctoral Associate at the Massachusetts Institute of Technology in 2010-2012, and is currently Associate Professor in the Department of Electronics, Information and Bioengineering at Politecnico di Milano. His research interests are in the analysis and control of discontinuous, hybrid systems, and networked systems.