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.2018.2804860, IEEE Transactions on Automatic Control

Composition of least restrictive controllers, with application to collision avoidance in multiagent systems.

Alessandro Colombo, *Member, IEEE,* Fabio Della Rossa

***Abstract*—A supervisor (of a continuous-time or hybrid system) is a controller in charge of modifying the input assigned by a user or set of users to a system, in order to enforce a given specification. This paper describes conditions under which multiple supervisors, designed to enforce different specifications, can be composed to obtain a supervisor enforcing the union of those specifications. As an application, we propose the composition of two supervisors, one enforcing collision avoidance of a large multiagent system, the other enforcing a second property, called sparsity, that allows efficient computation of the collision avoidance conditions.**

I. Introduction

Multiple methods for the computation of reachable sets for linear, nonlinear, and hybrid systems have been developed over the past twenty years [1]–[5]. In the context of mobile robotics, intelligent transportation systems, and multiagent systems, a particularly interesting application is the design of *least restrictive* controllers, or *supervisors* (by analogy with the framework defined in [6]), which are controllers in charge of restricting the set of available actions of a system to a subset guaranteed to avoid a given *bad set*. In these applications, the task of the supervisor can be formulated in terms of a reachability problem through computation of a *capture set* [2], [7].

In the presence of two or more specifications, supervisors can in principle be combined to avoid multiple bad sets simultaneously. This finds application in the design of multiobjective controllers for multiagent systems, ensuring for instance collision avoidance and visibility or connectivity [8]. In this paper, we discuss the properties required to ensure that the series composition of supervisors result in an algorithm with the same guarantees of least restrictiveness, as well as other important properties that we define later.

As an application, we propose an example of supervisors composition based on results from [9]: we tackle the design of a collision-avoidance supervisor that is known to be intractable on systems of many vehicles, and compose it with a second supervisor, in charge of restricting the system state to a set of instances that can be solved quickly, according to a *sparsity* property defined in [9]. The main novelty here is the introduction the second supervisor, that enforces sparsity regardless of the behaviour of each vehicle. The result is sketched in Fig. 1, where the system is a set of 370 cars modelled as second order nonlinear systems and driven by independent controllers (human or automatic) along intersecting paths. The supervisor is in charge of correcting the inputs of the independent controllers to avoid collisions. As we better detail in Sec. III, we obtain an algorithm capable of supervising in real time the 740-dimensional nonlinear model.

Alessandro Colombo and Fabio Della Rossa are with DEIB, Politecnico di Milano, via Ponzio 34/5, 20133 Milano, Italy. {alessandro.colombo,fabio.dellarossa}@polimi.it

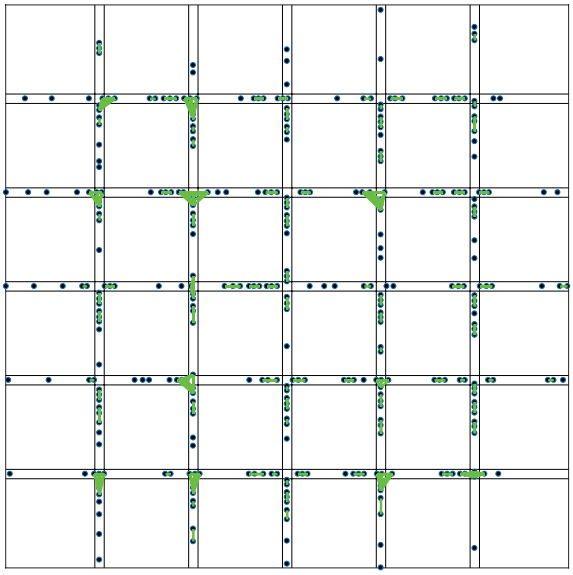


Fig. 1. 370 cars on paths intersecting in 25 locations. A video of this scenario is available at https://vimeo.com/212928016. Cars are supervised as detailed in Section III; cars in the same cluster are connected by a green segment.

All terminology mentioned above is precisely defined in the next section, together with the concept and requirements of a supervisor and the main theoretical result. The application is discussed in Section III.

II. Cascade Composition of Supervisors

The terminology introduced here is mostly derived from the formal verification and hybrid systems literature [2], [3], [10]. It will be used to formalize the properties of *least restrictiveness*, *nonblockingness*, and *correctness* of a supervisor. The main result is in Theorem 2, where we specify the conditions that a set of supervisors must satisfy for these properties to be preserved through supervisor composition.

Consider a model

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}),$$

where $\mathbf{x} \in X$ and $\mathbf{u} \in \mathcal{U}$ are state and input, while f is an arbitrary function. We use the symbol $\mathbf{x}$ to denote both the state value (an element of the set X) and a time signal (a vector function of time). When initial conditions can be ignored, we write $\mathbf{x}(t, \mathbf{u})$ to denote the state reached at time t with input $\mathbf{u}$, starting from the implicitly defined initial condition $\mathbf{x}(0)$, otherwise we specify the initial condition by writing $\mathbf{x}(t, \mathbf{u}, \mathbf{x}(0))$. We call *bad set* $B_S \subset X$ a set of states that the system should avoid. A supervisor enforcing $\mathbf{x} \notin B_S$ is said

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.2018.2804860, IEEE Transactions on Automatic Control

to satisfy the *specification* S. Given a bad set B_S, we define the *Maximal Controlled Invariant Set* (with respect to S):

$$\mathrm{MCIS}_S := \{\mathbf{x} \in X : \exists\, \mathbf{u} : \mathbf{x}(t, \mathbf{u}) \notin B_S,\, \forall\, t \geq 0\}.$$

The MCIS_S is the largest subset of X where an input $\mathbf{u}$ exists that can keep $\mathbf{x}$ out of the bad set for all positive times [11]. Finally, we call

$$\mathcal{U}_S(\mathbf{x}) := \{\mathbf{u} : \mathbf{x}(t, \mathbf{u}) \notin B_S,\, \forall\, t \geq 0\}$$

the set of inputs that guarantee that $\mathbf{x}$ remains outside of the bad set.

The supervisor is a map $\sigma : (\mathbf{x}, \bar{\mathbf{u}}) \mapsto \mathbf{u}$, where $\bar{\mathbf{u}}$ is a signal defined on a (short) time interval $[0, \tau]$, $\tau > 0$. In practice, it can be implemented with the discrete-time Algorithm 1 with time stepping τ. In the algorithm, which was originally

Algorithm 1 Supervisor for an arbitrary specification S

```
procedure σ(x(0), ū)
    x_next ← x(τ, ū, x(0))
    if x_next ∈ MCIS_S and x(t, ū) ∉ B_S for all t ∈ [0, τ] then
        return ū
    else
        return u_override
```

introduced in [12], $\mathbf{x}_{\mathrm{next}} = \mathbf{x}(\tau, \bar{\mathbf{u}}, \mathbf{x}(0))$ is the state that will be reached at time τ with input $\bar{\mathbf{u}}$ starting from initial condition $\mathbf{x}(0)$. Algorithm 1 uses the current state $\mathbf{x}(0)$, and the input $\bar{\mathbf{u}}$ (often assumed constant over the interval $[0, \tau]$), to predict a future state $\mathbf{x}_{\mathrm{next}}$. Then, it checks that the future state is in MCIS_S and that it can be reached without leaving MCIS_S. If this is not the case, the override input $\mathbf{u}_{\mathrm{override}}$ is returned, to be used instead of $\bar{\mathbf{u}}$. The problems of evaluating $\mathbf{x}_{\mathrm{next}} \in \mathrm{MCIS}_S$ and computing $\mathbf{u}_{\mathrm{override}}$ are not discussed here, but details for our example application are provided in Sec. III.

We are now in the position to formalize the property of least restrictiveness of a supervisor $\sigma_S(\mathbf{x}(0), \mathbf{u})$.

Definition 1. *The supervisor of Algorithm 1 is least restrictive if* $(\sigma_S(\mathbf{x}(0), \bar{\mathbf{u}}) \neq \bar{\mathbf{u}}) \Leftrightarrow \big(\exists\, t \in [0, \tau] : \mathbf{x}(t, \bar{\mathbf{u}}) \in B_S$ *or* $\exists\, t \geq 0 : \mathbf{x}(t, \mathbf{u}, \mathbf{x}(\tau, \bar{\mathbf{u}})) \in B_S,\, \forall\, \mathbf{u} \in \mathcal{U}\big)$

In other words, a least restrictive supervisor returns $\bar{\mathbf{u}}$, unless using $\bar{\mathbf{u}}$ for $t \in [0, \tau]$ would eventually result in a violation of the specification S, no matter what input is used for $t \geq \tau$. A least restrictive supervisor is thus one that allows $\mathbf{x}$ to visit all and only the subset MCIS_S of X.

Now consider a vector of input signals $\bar{\mathbf{u}}_{[0,\tau]}$, issued at time 0 and valid for the time interval $[0, \tau]$, and a vector $\bar{\mathbf{u}}_{[\tau,2\tau]}$, issued at time τ and valid for the time interval $[\tau, 2\tau]$. The supervisor must possess the two properties of correctness and nonblockingness, defined as follows.

Definition 2. *We say that a supervisor* $\sigma_S : (\mathbf{x}(0), \bar{\mathbf{u}}) \mapsto \mathbf{u}$ *is correct if* $\mathbf{x}(t, \sigma_S(\mathbf{x}(0), \bar{\mathbf{u}})) \notin B_S$ *for all* $t \in [0, \tau]$, *nonblocking if* $\sigma_S\Big(\mathbf{x}\big(\tau, \sigma_S(\mathbf{x}(0), \bar{\mathbf{u}}_{[0,\tau]})\big), \bar{\mathbf{u}}_{[\tau,2\tau]}\Big) \neq \emptyset$.

Intuitively, correctness ensures that the supervisor keeps $\mathbf{x}$ out of the bad set B_S, nonblockingness ensures that it can return an input at each time step, i.e., that the map $\sigma_S(\mathbf{x}, \mathbf{u})$ can be iterated indefinitely.

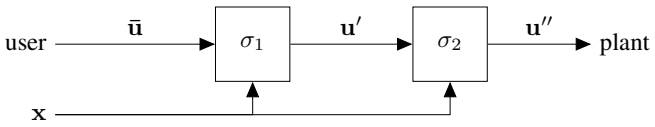


Fig. 2. Interconnection of two supervisors

In the following, we define the properties that allow to connect in series multiple supervisors, ensuring that the aggregate supervisor preserves correctness, nonblockingness, and least restrictiveness. This is the main result of this note. Let us write

$$\sigma_1 \dashrightarrow \sigma_2$$

when the output of a supervisor σ_1 is the input of σ_2 (as in Fig. 2). We write

$$\mathbf{u} \in \sigma(\mathbf{x}) \text{ if } \sigma(\mathbf{x}, \mathbf{u}) = \mathbf{u},$$

where $\mathbf{u} \in \sigma(\mathbf{x})$ reads '$\sigma(\mathbf{x})$ accepts the input $\mathbf{u}$'.

Definition 3. *A supervisor* $\sigma_2(\mathbf{x}, \mathbf{u})$ *is compatible with a supervisor* $\sigma_1(\mathbf{x}, \mathbf{u})$ *if*

$$\sigma_2(\mathbf{x}, \sigma_1(\mathbf{x}, \mathbf{u})) \in \sigma_1(\mathbf{x}),\, \forall\, \mathbf{x} \in \mathrm{MCIS}_1 \cap \mathrm{MCIS}_2,\, \forall\, \mathbf{u} \in \mathcal{U}$$

Notice that the relation $\sigma_2(\mathbf{x}, \sigma_1(\mathbf{x}, \mathbf{u})) \in \sigma_1(\mathbf{x})$ is not the same as $\sigma_2(\mathbf{x}, \sigma_1(\mathbf{x}, \mathbf{u})) = \sigma_1(\mathbf{x}, \mathbf{u})$; in the latter, the inputs returned by the maps $\sigma_2(\mathbf{x}, \sigma_1(\mathbf{x}, \mathbf{u}))$ and $\sigma_1(\mathbf{x}, \mathbf{u})$ must be identical, in the former the input returned by $\sigma_2(\mathbf{x}, \sigma_1(\mathbf{x}, \mathbf{u}))$ must simply be accepted by σ_1. Notice also that the above relation does not mean that $\sigma_2(\mathbf{x}, \mathbf{u}) \in \sigma_1(\mathbf{x}),\, \forall\, \mathbf{x} \in X,\, \forall\, \mathbf{u} \in \mathcal{U}$. In this case, the action of σ_2 would be sufficient to enforce the specification of σ_1, making σ_1 redundant.

We can provide a geometric condition for the satisfiability of Definition 3. Call $\partial\mathrm{MCIS}_S$ the boundary of MCIS_S, i.e., the set of $\mathbf{x} \in X$ that belong to the closure of both MCIS_S and of its complement in X.

Theorem 1. *The condition in Definition 3 can be satisfied if and only if*

$$\mathcal{U}_1(\mathbf{x}) \cap \mathcal{U}_2(\mathbf{x}) \neq \emptyset,\, \forall\, \mathbf{x} \in \partial MCIS_1 \cap \partial MCIS_2. \qquad (1)$$

Proof. The condition in Definition 3 is satisfiable, at a given $\mathbf{x}$, provided there exists $u \in \mathcal{U}$ accepted by both σ_1 and σ_2. This is possible if and only if $u \in \mathcal{U}_1(\mathbf{x}) \cap \mathcal{U}_2(\mathbf{x})$. This condition is trivially satisfied away from $\partial\mathrm{MCIS}_1 \cap \partial\mathrm{MCIS}_2$, since $\mathcal{U}_S(\mathbf{x}) = \mathcal{U}$ when $\mathbf{x}$ is in the interior of MCIS_S. Hence, condition (1) is necessary and sufficient. □

Theorem 2. *Consider two correct, nonblocking, and least restrictive supervisors* σ_1 *and* σ_2, *and the cascade composition* $\sigma_1 \dashrightarrow \sigma_2$. *If* σ_2 *is compatible with* σ_1, *then* $\sigma_1 \dashrightarrow \sigma_2$ *is correct, nonblocking, and least restrictive.*

Proof. Correctness with respect to specification 2 is guaranteed by assumption, since the action of σ_1 on the inputs does not change the properties of σ_2, while correctness with respect to specification 1 is guaranteed by compatibility, since $\sigma_2(\mathbf{x}, \sigma_1(\mathbf{x}, \mathbf{u})) \in \sigma_1(\mathbf{x})$. Nonblockingness follows by a similar argument, since $\sigma_2(\mathbf{x}, \sigma_1(\mathbf{x}, \mathbf{u})) \in \sigma_2(\mathbf{x})$ (trivially),

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.2018.2804860, IEEE Transactions on Automatic Control

$\sigma_2(\mathbf{x}, \sigma_1(\mathbf{x}, \mathbf{u})) \in \sigma_1(\mathbf{x})$ (by compatibility), and both σ_1 and σ_2 are nonblocking. Finally, least restrictiveness is a simple consequence of the structure of Algorithm 1, which defines the two supervisors: if $\sigma_2(\mathbf{x}, \sigma_1(\mathbf{x}, \mathbf{u})) \neq \mathbf{u}$, then either $\mathbf{u} \notin \sigma_1(\mathbf{x})$ or $\mathbf{u} \notin \sigma_2(\mathbf{x})$, and since they are both least restrictive, so must be $\sigma_1 \dashrightarrow \sigma_2$. □

It is a simple exercise to extend the above result to an arbitrary chain of supervisors by iterative reasoning.

Corollary 3. *Consider a chain $\sigma_1 \dashrightarrow \sigma_2 \dashrightarrow \cdots$ of correct, nonblocking, and least restrictive supervisors. If σ_i is compatible with σ_{i-1}, then $\sigma_1 \dashrightarrow \sigma_2 \dashrightarrow \cdots$ is correct, nonblocking, and least restrictive.*

III. Application: a semiautonomous vehicle network

Consider a group of cars moving through a road network, such as in Fig. 1. Each car's driver is responsible of choosing its car's input, and may occasionally commit a mistake. The task of a supervisor is to correct the wrong inputs to avoid collisions between any two cars in the system. Techniques to design a supervisor for vehicle collision avoidance have been discussed in [12], [13] and further expanded in [14]–[19]. Computation of the MCIS for a supervisor avoiding collisions of cars near a single intersection was proven to be NP-hard in [12], [20], so a monolithic approach to a scenario such as in Fig. 1 is in general intractable. Real-time computation is however achievable if the number of cars in the system is small enough [17], [19], [21]; this suggests, as a solution strategy, partitioning cars into small independently controllable groups. A means to partition a large set of vehicles into smaller subsets that can be independently supervised for collision avoidance was recently proposed in [9], but hinges on a sparsity hypothesis requiring vehicles to be 'not too packed together', (in a sense precisely defined in [9]), and sparsity may be violated as vehicles move about the network. Here we propose to construct a composite supervisor, consisting of a layer in charge of enforcing sparsity to allow vehicle partition into clusters of a predefined maximum size $N_{\max}$, and of another layer in charge of avoiding collisions.

The basic structure of the architecture shares similarities with [22]–[30]. However, contrary to these references, our design focus is on minimizing the amount of corrections to the drivers inputs, that is, to make the controller least restrictive. This is a desirable property to minimally perturb the driving strategy pursued by the human or automatic driver of each car.

In the following, we assume that the short-term paths of all cars are known. We model the motion of each car i along its path with the equation

$$\ddot{x}_i = u_i - 0.0005\dot{x}_i^2, \tag{2}$$

with input u_i and velocity $\dot{x}_i$ bounded in the intervals $[u_{\min}, u_{\max}]$, $[\dot{x}_{\min}, \dot{x}_{\max}]$, with $\dot{x}_{\min} \geq 0$. The quadratic term in (2) accounts for air drag. The position of the car along its path is $x_i \in \mathbb{R}$, and $u_i \in \mathbb{R}$ is the control input. We call $\mathbf{x}_i := (x_i, \dot{x}_i)$ the state of car i, and use the symbols $\mathbf{x} \in X$ and $\mathbf{u} \in \mathcal{U}$ without subscript to indicate the aggregate state and input of all cars.

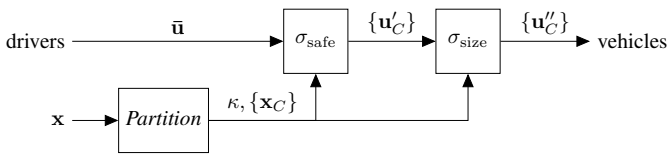


Fig. 3. Cascade of the supervisors σ_{safe} and σ_{size}. The partition κ is a set of clusters C, while $\{\mathbf{x}_C\}$ is the set of initial conditions of cars in each cluster.

We call $\mathcal{P}_i$ the path of vehicle i on a planar road network, and denote $\mathcal{P}_i(x_i) : \mathbb{R} \to \mathbb{R}^2$ the position of car i on the plane. We call $\mathcal{O}_{i,j}$ a subset of the plane where the paths of vehicles i and j overlap, $\mathcal{D}(\mathcal{P}_i(x_i), \mathcal{P}_j(x_j)) \in \mathbb{R}$ the distance between these vehicles measured along the paths, and define a minimum distance d, below which vehicles on overlapping paths have a rear-end collision. Vehicles on overlapping path intervals are assumed to move in the same direction. Finally, we call $\mathcal{I}_k$ an open region around the k-th intersection where the simultaneous presence of two vehicles on different paths would constitute a side collision. With this notation, we define the *conflict set*

$$\begin{aligned}\mathcal{C} := \\ \{\mathbf{x} \in X : \mathcal{P}_i(x_i), \mathcal{P}_j(x_j) \in \mathcal{O}_{i,j}, \mathcal{D}(\mathcal{P}_i(x_i), \mathcal{P}_j(x_j)) < d\} \cup \\ \left\{\mathbf{x} \in X : \mathcal{P}_i(x_i), \mathcal{P}_j(x_j) \in \mathcal{I}_k \setminus \mathcal{O}_{i,j}\right\}\end{aligned} \tag{3}$$

as the set of all configurations corresponding to a collision of two or more cars.

A. Time-τ independent partition

The partitions described below, and the partitioning algorithm described in the appendix, are taken from [9]. We report here only the details needed to follow the rest of the paper.

Call $\kappa := \{C_1, \ldots, C_c\}$ a partition of cars into clusters $C_1, \ldots, C_c$. We denote with a subscript C the restriction of a vector or set to the vehicles in cluster C: $\mathbf{x}_C$ is the aggregate state of all vehicles in cluster C, $\text{MCIS}_{S,C}$ is MCIS_S restricted to vehicles in C, $\mathcal{U}(\mathbf{x}_C)$ is the set $\mathcal{U}(\mathbf{x})$ restricted to vehicles in C.

Definition 4. *A partition $\kappa := \{C_1, \ldots, C_c\}$ is time-τ independent if, for all $t \in [0, \tau]$ and for all $\mathbf{u} \in \mathcal{U}$,*

$$\Big(\mathbf{x}_C(t, \mathbf{u}) \in \text{MCIS}_{S,C}, \forall\, C \in \kappa\Big) \Leftrightarrow \Big(\mathbf{x}(t, \mathbf{u}) \in \text{MCIS}_S\Big).$$

In other words, a partition is time-τ independent if, for all states $\mathbf{x}(t, \mathbf{u})$ reachable in a time τ with an arbitrary input $\mathbf{u} \in \mathcal{U}$, a set of parallel supervisors (with time stepping τ, as in Algorithm 1), each one acting on a different cluster C by enforcing $\mathbf{x}_C(t, \mathbf{u}) \in \text{MCIS}_{S,C}$, jointly result in a supervisor enforcing specification S exactly, that is, in a least restrictive centralized supervisor. An algorithm to compute a time-τ independent partition is reported in Appendix A-A

B. Implementation of the two-layer supervisor

The information flow in our control architecture is represented in Fig. 3: a partitioning function *partition*$(\mathbf{x})$ separates

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.2018.2804860, IEEE Transactions on Automatic Control

cars into clusters, then the inputs of cars in each cluster are filtered by a supervisor ($\sigma_{\text{safe}}(\mathbf{x}_C, \mathbf{u}_C)$) in charge of ensuring safety, and one ($\sigma_{\text{size}}(\mathbf{x}, \mathbf{u})$) enforcing that, at the following time step, there will not form clusters greater than a given maximum size $N_{\max}$. The supervisor σ_{safe}, which is the most computationally intensive task, computes on each cluster of the partition independently, hence it can be run in parallel over clusters. The complexity of the distributed σ_{safe} is thus capped through the choice of $N_{\max}$, while the complexity of partitioning vehicles in clusters is, as explained in the appendix, $O(nN_{\max})$, where n is the total number of cars in the system. In what follows, we describe how each of the three elements of Fig. 3 is constructed.

The function *partition*$(\mathbf{x})$ is defined in Algorithm 2, in the appendix.

We can construct σ_{safe} using Algorithm 1, with the specification 'safe' defined by its bad set $B_{\text{safe}} := \mathcal{C}$, where the right-hand side is the conflict set (3). Ways to compute $\text{MCIS}_{\text{safe}}$ and the override input, and therefore to implement the supervisor in different road scenarios, are discussed for instance in [12]–[19].

Finally, to construct σ_{size} we first define its bad set

$$B_{\text{size}} := \{\mathbf{x} \in X : \max_{C_i \in \kappa} \text{size}(C_i) > N_{\max}\}, \qquad (4)$$

with $\kappa = \text{partition}(\mathbf{x})$. The parameter $N_{\max}$ is the maximum size of a cluster that the supervisor should allow to form. To define the supervisor, we need to identify $\text{MCIS}_{\text{size}}$. We do this through the following results.

Theorem 4. *partition*$(\mathbf{x})$ *can be defined so that, for all* $\mathbf{x}(0) \in X$, $\exists \mathbf{u} \in \mathcal{U}$ *such that* *partition*$(\mathbf{x}(0)) =$ *partition*$(\mathbf{x}(\tau, \mathbf{u}, \mathbf{x}(0)))$

The above theorem is proved in the Appendix.

Corollary 5 (Corollary of Theorem 4)**.** $\text{MCIS}_{\text{size}} = \neg B_{\text{size}}$.

Proof. Theorem 4 asserts the existence of an input that preserves the partition; therefore, any $\mathbf{x} \notin B_{\text{size}}$ belongs to $\text{MCIS}_{\text{size}}$. □

Finally, to prove compatibility of the two supervisors, we use the following theorem, proved in the appendix.

Theorem 6. *partition*$(\mathbf{x})$ *can be defined so that, for all* $\mathbf{x}(0) \in \text{MCIS}_{\text{safe}} \cap \text{MCIS}_{\text{size}}$*, there exists a* $\mathbf{u} \in \mathcal{U}_{\text{safe}}(\mathbf{x})$ *such that* *partition*$(\mathbf{x}(0)) =$ *partition*$(\mathbf{x}(\tau, \mathbf{u}, \mathbf{x}(0)))$

Corollary 7 (Corollary of Theorem 6)**.** *The supervisor* σ_{size} *can be constructed compatible with* σ_{safe}.

Proof. By Theorem 6, with suitable definition of partition$(\mathbf{x})$, for all $\mathbf{x}(0) \in \text{MCIS}_{\text{safe}} \cap \text{MCIS}_{\text{size}}$ there exists a $\mathbf{u} \in \mathcal{U}_{\text{safe}}(\mathbf{x})$ that preserves the partition. This implies $\mathbf{u} \in \mathcal{U}_{\text{size}}(\mathbf{x})$, hence $\mathbf{u} \in \mathcal{U}_{\text{size}}(\mathbf{x}) \cap \mathcal{U}_{\text{safe}}(\mathbf{x})$, so that

$$\mathcal{U}_{\text{size}}(\mathbf{x}) \cap \mathcal{U}_{\text{safe}}(\mathbf{x}) \neq \emptyset, \ \forall \mathbf{x} \in \text{MCIS}_{\text{safe}} \cap \text{MCIS}_{\text{size}}.$$

By Theorem 1, σ_{size} is thus compatible with σ_{safe}. □

By the above corollary, $\mathbf{x} \in \text{MCIS}_{\text{size}}$ can be evaluated simply by checking whether $\mathbf{x}$ admits a partition with elements not larger than $N_{\max}$. Computation of $\text{MCIS}_{\text{size}}$ therefore shares the same complexity as Algorithm 2.

C. Simulations

Using the above results, we can construct the cascade composition of σ_{safe} and σ_{size} to supervise cars in a road network. We performed simulations on the network in Fig. 4, where the distance between intersections along any path is 100m. Vehicles have length $d = 5$m, with $u_i \in (-5, 2)$m/s and $\dot{x}_i \in (0, 13.9)$m/s^2. The bad set B_{size} is defined as in (4) with $N_{\max} = 4$. The supervisor runs with step 0.1s.

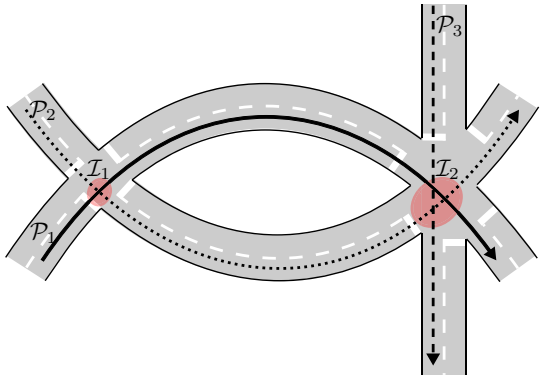
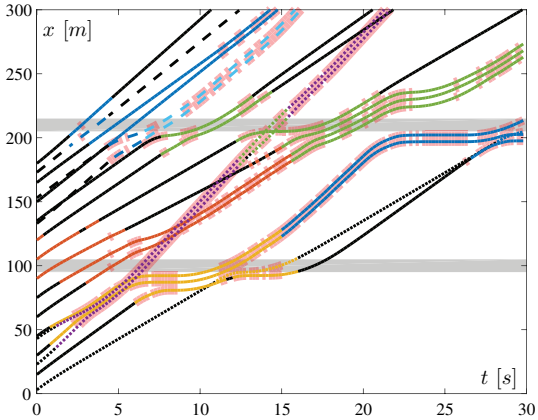


Fig. 4. Two intersections of three paths, the conflict regions around each intersection are shaded red.

The results of the simulations are in Fig. 5. The action of σ_{safe} is apparent around $t = 5$s, when it avoids two rear collisions between cars 10-11 and 13-14, or in the first 10s of simulation, when a sequence of interventions prevents a side collision between agents 1-5 and 16-18. The action of σ_{size} can be seen e.g., around $t = 20$s, where the blue cluster is slowed to avoid merging with the green 4-vehicle cluster.

The same supervisor was then used to control the road network in Fig. 1, with 370 cars on 10 intersecting paths at a distance of 90m from each other. For the sake of simplicity,



i	$x_i(0)$	$\bar{v}_i$	$\mathcal{P}$
1	180	11.4	1
2	165	9.2	1
3	150	10.2	1
4	135	9	1
5	120	8.6	1
6	105	7.4	1
7	90	6.6	1
8	75	7	1
9	60	8.3	1
10	45	5.9	1
11	30	11.7	1
12	15	9.2	1
13	43	6.1	2
14	23	12.9	2
15	3	7.5	2
16	173	10.4	3
17	153	9.2	3
18	133	11.3	3

Fig. 5. 18 cars in the road network of Fig. 4. Trajectories are solid, dotted, and dashed for cars on the solid, dotted, or dashed path of Fig. 4. The intersections, represented by the shaded bands, are at the coordinates $(95, 105)$ and $(205, 215)$ of each path. Same-colour trajectories at a given time instant identify cars temporarily grouped in the same cluster, while black trajectories represent cars in a singleton cluster. Trajectories are highlighted in red at times when a supervisor is overriding the driver input. Agents are numbered as in the table on the right, which also reports their initial position (initial velocity $\dot{x}_i(0) = 5$m/s, $\forall i$), and their target velocity $\bar{v}_i$. The desired input of each car is computed through a proportional negative feedback with unit gain on the difference between target velocity and current velocity.

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.2018.2804860, IEEE Transactions on Automatic Control

for the simulation linked in the figure we have programmed all cars to follow straight paths, but our results hold regardless of the paths geometry. The supervisor had step $\tau = 0.2s$ and $N_{\max} = 6$. Fig. 6 reports the time taken to run the partitioning algorithm and the two supervisors on each of the 370 agents. The maximum time never exceeds 0.175s, this suggest that a real-time execution of the two algorithms is plausible (for instance running the supervisors with time stepping $\tau = 0.2s$, as we did in the simulation), though communication delay was not considered in the simulations.

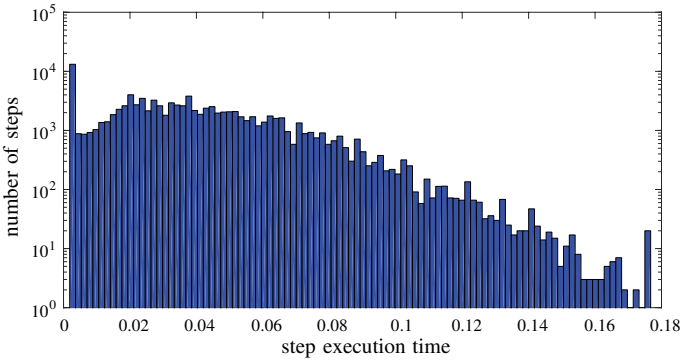


Fig. 6. Distribution of the time length of a single step of the partitioning and supervisory algorithms, across all vehicles, during the simulation of Fig. 1.

IV. Conclusion

We have discussed a general framework to design multiple supervisors that can operate simultaneously, while retaining their fundamental properties of correctness, nonblockingness, and least restrictiveness. Under this framework supervisors for different specifications can be designed separately, then applied sequentially in a modular fashion, without losing their performance guarantees.

As an example, and building on results in [9], we have used these ideas to design a two-layer supervisor that can prevent collisions between cars, while ensuring that the largest set of cars that must be simultaneously controlled to avoid a collision never exceeds a set size. This allowed us to overcome the computational complexity barrier typically affecting collision-avoidance architectures for multiagent systems.

While here we explored an application of cascaded supervisor to a vehicle collision avoidance problem, the same approach could be used to merge supervisors with other specifications, such as visibility, connectivity, or energy efficiency.

Appendix A
Implementation details

In Appendix A-A and A-B we report and adapt to satisfy the presented theory the results presented in [9]. Then, we prove Theorems 4 and 6 in Appendix A-C and A-D.

A. Definition of partition$(\mathbf{x})$

Consider a set-valued function $I_i^\tau(t) : \mathbb{R}_+ \to 2^{\mathbb{R}^2}$, which attaches to a time t a subset of $\mathbb{R}^2$ with $\mathbf{x}_i \in I_i^\tau(0)$. Call $I_C^\tau(t) := \{I_i^\tau(t)\}$, $i \in C$.

Definition 5. *$I_C^\tau(t)$ is a time-τ guaranteed hull for cluster C if, whenever $\mathcal{U}_S(\mathbf{x}_C) \neq \emptyset$, for all $\mathbf{u}_{C,[0,\tau]} \in \mathcal{U}_S(\mathbf{x}_C)$ there exists at least one $\mathbf{u}_{C,(\tau,\infty)} \in \mathcal{U}_S(\mathbf{x}_C(\tau, \mathbf{u}_{C,[0,\tau]}))$ such that $\mathbf{x}_i(t, u_i) \in I_i^\tau(t)$, for all $t \geq 0$ and for all $i \in C$.*

In simple terms, a time-τ guaranteed hull is a set of states which is guaranteed to contain at least one safe trajectory, when such a trajectory exists.

Algorithm 2 defines a procedure to partition a set of cars. At line 6, the symbols $I^\tau_{\{i,j\}}(t)$ and $B_{\{i,j\}}$ denote the time-τ guaranteed hull and the bad set for the set of cars $\{i, j\}$.

Algorithm 2 Partitioning algorithm

1: **procedure** *Partition*$(\mathbf{x})$
2: Define a partition $\kappa := \{C\}$ where all vehicles are singleton clusters
3: Tag all clusters *not done*
4: **while** there exists a cluster *not done* **do**
5: Compute I_C for all $C \in \kappa$
6: Form a new partition κ' by merging all $C_a, C_b \in \kappa$ such that $\exists\, t, i \in C_a, j \in C_b : I^\tau_{\{i,j\}}(t) \cap B_{\{i,j\}} \neq \emptyset$
7: Tag as *done* all clusters that were not merged (i.e., those that were not modified between κ and κ')
8: Set $\kappa = \kappa'$
9: **return** $\kappa, \{\mathbf{x}_C\}$

Theorem 8 (Proved in [9])**.** *Algorithm 2 terminates and finds a time-τ independent partition.*

A note on the complexity of Algorithm 2 is now due. Assume that, when executing Algorithm 2, the maximum size of a cluster is $N_{\max}$. Then, the while loop was iterated at most $N_{\max}$ times. Assuming a worst-case scenario where, at line 6, all guaranteed hulls must be mutually compared, and assuming Line 6 is executed in parallel on each car, the algorithm has running time $O(nN_{\max})$ for a problem with n cars. In practice, as discussed in [9], the running time is much faster, since only guaranteed hulls of neighbouring cars need be compared.

B. Definition of the guaranteed hull

We first provide the definition in the simplified scenario of an isolated intersection $\mathcal{I}$. The result is extended to an arbitrary road network in the next section.

For a path $\mathcal{P}_h$, call (a_h, b_h) the interval $\{x_i : \mathcal{P}_h(x) \in \mathcal{I}\}$. Assume that cluster C has n vehicles on path $\mathcal{P}_h$, and number them in order so that $x_1 > x_2 > \dots$ Call $C_{\mathcal{P}_h}$ the subcluster formed by these vehicles. Assume $\mathcal{U}_{\text{safe}}(\mathbf{x}_{C_{\mathcal{P}_h}}) \neq \emptyset$. Fixed $\mathbf{x}$, consider the preorder $\mathbf{u} \preceq \mathbf{u}'$ if $\mathbf{x}(t, \mathbf{u}) \leq \mathbf{x}(t, \mathbf{u}')$, $\forall\, t$. It is proved in [17] that $\mathcal{U}_{\text{safe}}(\mathbf{x}_{C_{\mathcal{P}_h}})$ has a minimum $\underline{\mathbf{u}}_{\mathbf{x}_{C_{\mathcal{P}_h}}}$ in this preorder. Let us use the symbol $\underline{|\mathbf{u}_{[t_1,t_2)}, \mathbf{u}_{[t_2,t_3)}, \dots|}$ to write the concatenation of multiple input signals defined over nonoverlapping time intervals. Call

$$D_{\text{stop}} := \lim_{t\to\infty} x_i(t, u_{\min}) \text{ with } x_i(0) = 0, \dot{x}_i(0) = \dot{x}_{\max},$$

$$\Delta_{i,\text{stop}}(\tau) := \max_{\dot{x}_i(0)\in[0,\dot{x}_{\max}]} \lim_{t\to\infty} x_i\left(t, \underline{|u_{\max,[0,\tau]}, u_{\min,(\tau,\infty)}|}\right) - x_i(t, u_{\min}).$$

and $\Delta_{\text{stop}}(\tau) := \max_i \Delta_{i,\text{stop}}(\tau)$.

Define $I_i^\tau(t) := [\mathbf{x}_i(t, u_{\min}), \mathbf{x}_i(t, \bar{u}_i)]$,where $\bar{u}_i$ is a bang-bang input defined as follows. Consider the two cases:

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.2018.2804860, IEEE Transactions on Automatic Control

(i) $[x_n(0), x_1(0) + D_{\text{stop}} + \Delta_{\text{stop}}] \cap (a_h, b_h) = \emptyset$, i.e., vehicles in cluster C can safely stop before entering the intersection, regardless of their current velocity and of their input in the time interval $[0, \tau]$, or they are all past the intersection;

(ii) $[x_n(0), x_1(0) + D_{\text{stop}} + \Delta_{\text{stop}}] \cap (a_h, b_h) \neq \emptyset$, i.e., some vehicles within the cluster may not stop before the intersection.

In case (i), define $\bar{u}_i$ as

$$\bar{u}_i(t) := \underline{\lfloor u_{\max,[0,t_i^*]}, u_{\min,(t_i^*,\infty)} \rfloor}, \tag{5}$$

switching at a time $t_i^* \geq 0$ so as to satisfy

$$\lim_{t\to\infty} x_n(t, \bar{\mathbf{u}}) = \lim_{t\to\infty} x_n(t, u_{\min}) + \Delta_{\text{stop}}$$

for vehicle n, and

$$\lim_{t\to\infty} x_i(t, \bar{u}_i) = \max\Big\{ \lim_{t\to\infty} x_i(t, \underline{u}_i) + \Delta_{\text{stop}}, \lim_{t\to\infty} x_{i+1}(t, \bar{u}_{i+1}) + d \Big\} \tag{6}$$

for all other vehicles.

In case (ii), for all vehicles i such that $x_i(0) + D_{\text{stop}} + \Delta_{\text{stop}} \leq a_h$, define $\bar{u}_i$ as in (5) Assume there are $n - m$ such vehicles. For the remaining m vehicles, define $\bar{u}_i$ as a bang-bang input

$$\bar{u}_i(t) := \underline{\lfloor u_{\max,[0,t_i^*]}, u_{\min,(t_i^*,\infty)} \rfloor}, \tag{7}$$

switching at a time $t_i^* \geq 0$ so as to satisfy $\lim_{t\to\infty} x_i(t, \bar{u}_i) = \max\Big\{ \lim_{t\to\infty} x_i(t, \underline{u}_i) + \Delta_{\text{stop}}, b_h + D_{\text{stop}} + \Delta_{\text{stop}} + (m - i)d \Big\}$.

Lemma 9. *$I_C^\tau(t)$ defined above satisfies the property in Definition 5.*

Proof. The definition of guaranteed hull given above is a slight modification of the same definition in [9]. The proof follows straight from the proof of Theorem 8 in [9], noting that $t_i^* \geq \tau$ and therefore the guaranteed hull defined here contains that defined in [9]. We can prove $t_i^* \geq \tau$ as follows. We have, by definition of Δ_{stop},

$$\Delta_{\text{stop}} + \lim_{t\to\infty} x_i(t, \underline{u}_i) \geq \lim_{t\to\infty} x_i\Big(t, \underline{\lfloor u_{\max,[0,\tau]}, u_{\min,(\tau,\infty)} \rfloor}\Big),$$

and from definitions (i)-(ii) t_i^* must satisfy

$$\Delta_{\text{stop}} + \lim_{t\to\infty} x_i(t, \underline{u}_i) \leq \lim_{t\to\infty} x_i\Big(t, \underline{\lfloor u_{\max,[0,t_i^*]}, u_{\min,(t_i^*,\infty)} \rfloor}\Big);$$

therefore $t_i^* \geq \tau$. □

C. Proofs of Theorems 4 and 6 for an isolated intersection

We now write $\mathcal{I}_C^\tau(\mathbf{x}, t)$ for the set $\mathcal{I}_C^\tau(t)$ computed from state $\mathbf{x}$.

Proof of Theorem 4. It is sufficient to take $u_{i,\text{override}} := u_{\min}$. Given our definition of I_C^τ, this ensures that $I_i^\tau(x_i(\tau, u_{i,\text{override}}), t) \subseteq I_i^\tau(x_i(0), t_\tau)$, for all i and for all t. As a consequence, the partition computed at time τ is the same as that computed at time 0. □

Proof of Theorem 6. We prove the theorem by constructing the required $\mathbf{u}_{\text{override}}$.

Consider first a cluster of cars $1, \ldots, n$ on the same path, with state $\mathbf{x}$. Assume $\mathcal{U}_{\text{safe}}(\mathbf{x}) \neq \emptyset$. Take $\mathbf{u}_{\text{override}} = \underline{\mathbf{u}}$, and take $\mathbf{x}' := \mathbf{x}(\tau, \mathbf{u}_{\text{override}})$. Since all cars are on the same path, $\underline{\mathbf{u}} \in \mathcal{U}_{\text{safe}}(\mathbf{x})$. We must prove that $I^\tau(\mathbf{x}', t) \subseteq I^\tau(\mathbf{x}, t+\tau)$, $\forall\, t$.

Let us call $\bar{\mathbf{u}}$ the input giving the upper bound of $I^\tau(\mathbf{x}, t)$, and $\bar{\mathbf{u}}'$ the same input for $I^\tau(\mathbf{x}', t)$. Since $\mathbf{u}_{\min} \preceq \mathbf{u}_{\text{override}}$, $\mathbf{x}(t + \tau, \mathbf{u}_{\min}) \leq \mathbf{x}'(t, \mathbf{u}_{\min})$, $\forall\, t \geq 0$, the lower bound of $I^\tau(\mathbf{x}, t)$ is smaller than the lower bound of $I^\tau(\mathbf{x}', t)$. We have

$$x_i(t, \bar{u}_i) \geq x_i(t, u_{i,\text{override}}),\ \forall\, t \in [0, t_i^*],\ \forall\, i. \tag{8}$$

We also have

$$\lim_{t\to\infty} \mathbf{x}(t, \bar{\mathbf{u}}) = \lim_{t\to\infty} \mathbf{x}'(t, \bar{\mathbf{u}}'), \tag{9}$$

since by choosing $\mathbf{u}_{\text{override}} = \underline{\mathbf{u}}$ the two terms in the max in (6) remain unchanged. This, with (8) and by monotonicity, insures $\mathbf{x}(t + \tau, \bar{\mathbf{u}}) \geq \mathbf{x}'(t, \bar{\mathbf{u}}')$, $\forall\, t \geq 0$, i.e., the upper bound of $I^\tau(\mathbf{x}, t)$ is greater than the upper bound of $I^\tau(\mathbf{x}', t)$. Therefore, $I^\tau(\mathbf{x}', t) \subseteq I^\tau(\mathbf{x}, t + \tau)$, $\forall\, t$.

Consider now the more general case of a cluster of cars on different paths, when for some cars I^τ is defined as in case (ii). For all cars falling under case (i) we can proceed with the same proof as above. For the remaining ones, it is shown in [9] (Lemma 9) that there exists an input $u_i \in \mathcal{U}_{\text{safe}}$ giving $\lim_{t\to\infty} x_i(t, u_i) = \max\Big\{ \lim_{t\to\infty} x_i(t, \underline{u}_i), b_h + D_{\text{stop}} + (m - i)d \Big\}$. Using such an input as $u_{i,\text{override}}$, we show with the following reasoning that (9) still holds. Assume $\lim_{t\to\infty} x_i(t, \underline{u}_i) \geq b_h + D_{\text{stop}} + (m - i)d$; then $u_{i,\text{override}} = \underline{u}_i$, and (9) follows as above. If instead $\lim_{t\to\infty} x_i(t, \underline{u}_i) < b_h + D_{\text{stop}} + (m - i)d$, $u_{i,\text{override}}$ is constructed so that $\lim_{t\to\infty} x_i(t, u_{i,\text{override}}) = b_h + D_{\text{stop}} + (m - i)d$. This implies $\lim_{t\to\infty} x_i'(t, \underline{u}_i') + \Delta_{\text{stop}} \leq b_h + D_{\text{stop}} + \Delta_{\text{stop}} + (m - i)d$, (where $\underline{\mathbf{u}}'$ is the minimum of $\mathcal{U}_{\text{safe}}(\mathbf{x}')$). Therefore, $\lim_{t\to\infty} x_i'(t, \bar{u}') = b_h + D_{\text{stop}} + \Delta_{\text{stop}} + (m - i)d = \lim_{t\to\infty} x_i(t, \bar{u})$. Given (9), the proof follows as for case (i). □

D. Extension to arbitrary network topology

We now extend the above results to work on a network with multiple nearby intersections.

Consider a road network with multiple paths and intersections, and assume that Algorithm 2 returns clusters with at most $N_{\max}$ agents on each path. Given a path $\mathcal{P}$ crossing, in this order, two intersections $\mathcal{I}_h$ and $\mathcal{I}_k$, let $[a_h, b_h]$ and $[a_k, b_k]$ be the length of the intersections along $\mathcal{P}$.

Theorem 10. *If $a_k - b_h > D_{\text{stop}} + \Delta_{\text{stop}} + (N_{\max} - 1)(\Delta_{\text{stop}} + d)$ for any path and pair of intersections crossed by the path, then Lemma 9 and Theorem 6 hold.*

Proof. From (5) and (7), we have

$$\lim_{t\to\infty} x_i(t, \bar{u}_i) \leq \max\Big\{ \lim_{t\to\infty} x_i(t, \underline{u}_i) + \Delta_{\text{stop}}, b_h + D_{\text{stop}} + \Delta_{\text{stop}} + (n - i)d, \lim_{t\to\infty} x_{i+1}(t, \bar{u}_{i+1}) + d \Big\} \tag{10}$$

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.2018.2804860, IEEE Transactions on Automatic Control

and, in particular for car n,

$$\lim_{t\to\infty} x_n(t,\bar{u}_n) \leq \max\Big\{ \lim_{t\to\infty} x_n(t,\underline{u}_n) + \Delta_{\text{stop}}, b_h + D_{\text{stop}} + \Delta_{\text{stop}}\Big\}. \quad (11)$$

From the proof of Theorem 1 in [17], we also have

$$\lim_{t\to\infty} x_i(t,\underline{u}_i) \leq \max\Big\{ \lim_{t\to\infty} x_i(t,u_{\min}), \lim_{t\to\infty} x_{i+1}(t,\underline{u}_{i+1}) + d\Big\}. \quad (12)$$

Intuitively, the above equation says that, if a car in a line must stop without collisions, the lower bound to its stopping position depends on its stopping distance, and on the lower bound to the position at which will stop the car behind it. From Algorithm 2 we can deduce that

$$\lim_{t\to\infty} x_i(t,u_{\min}) \leq \lim_{t\to\infty} x_{i+1}(t,\bar{u}_{i+1}) + d, \quad (13)$$

otherwise cars i and $i+1$ would not be clustered together. Using (13) in (12) and the result in (10) we obtain $\lim_{t\to\infty} x_1(t,\bar{u}_1) \leq \max\Big\{ \lim_{t\to\infty} x_2(t,\bar{u}_2) + \Delta_{\text{stop}} + d, b_h + D_{\text{stop}} + \Delta_{\text{stop}} + (n-1)d\Big\}$. Iterating the above inequality we obtain

$$\lim_{t\to\infty} x_1(t,\bar{u}_1) \leq \max\Big\{ \lim_{t\to\infty} x_n(t,\bar{u}_n) + (n-1)(\Delta_{\text{stop}}+d), b_h + D_{\text{stop}} + \Delta_{\text{stop}} + (n-1)(\Delta_{\text{stop}}+d)\Big\}. \quad (14)$$

By the definition of D_{stop}

$$\lim_{t\to\infty} x_n(t,\underline{u}_i) \leq x_n(0) + D_{\text{stop}}, \quad (15)$$

Using (11) and (15) in (14), we finally obtain $\lim_{t\to\infty} x_1(t,\bar{u}_i) \leq \max\Big\{x_n(0) + D_{\text{stop}} + \Delta_{\text{stop}} + (n-1)(\Delta_{\text{stop}}+d), b_h + D_{\text{stop}} + \Delta_{\text{stop}} + (n-1)(\Delta_{\text{stop}}+d)\Big\}$. As a consequence, if $a_k - b_h > D_{\text{stop}} + \Delta_{\text{stop}} + (n-1)(\Delta_{\text{stop}}+d)$ the guaranteed hull will intersect, for $t \in [0,\infty)$, only one intersection along each path, so that the results obtained above for an isolated intersection hold. □

Note that, with the parameter in the simulation of Fig. 1, we have $D_{\text{stop}} = 19.2$m and $\Delta_{\text{stop}} = 2.8$m, verifying the assumption of Theorem 10.

REFERENCES

[1] O. Shakernia, G. J. Pappas, and S. Sastry, "Decidable controller synthesis for classes of linear systems," in *Hybrid Systems: Computation and Control*, 2000.

[2] C. J. Tomlin, I. Mitchell, A. M. Bayen, and M. Oishi, "Computational techniques for the verification of hybrid systems," *Proc. IEEE*, vol. 91, pp. 986–1001, 2003.

[3] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin, "A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games," *IEEE Trans. Autom. Control*, vol. 50, pp. 947–957, 2005.

[4] F. Blanchini and S. Miani, *Set-Theoretic Methods in Control*. Birkhauser, 2008.

[5] M. Herceg, M. Kvasnica, C. Jones, and M. Morari, "Multi-Parametric Toolbox 3.0," in *European Control Conf.*, Zürich, Switzerland, July 17–19 2013, pp. 502–510, http://control.ee.ethz.ch/~mpt.

[6] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control Optim.*, vol. 25, pp. 206–230, 1987.

[7] T. A. Henzinger, B. Horowitz, R. Majumdar, and H. Wong-Toi, "Beyond HyTech: Hybrid systems analysis using interval numerical methods," in *Hybrid Systems: Computation and Control*, 2000.

[8] M. M. Zavlanos and G. J. Pappas, "Potential fields for maintaining connectivity of mobile networks," *IEEE Trans. Robot*, vol. 23, pp. 812–816, 2007.

[9] A. Colombo, G. Rodrigues De Campos, and F. Della Rossa, "Control of a city road network: Distributed exact verification of traffic safety," *IEEE Trans. Autom. Control*, vol. 62, pp. 4933–4948, 2017.

[10] S. M. Loos and A. Platzer, "Safe intersections: At the crossing of hybrid systems and verification," in *IEEE Conf. on Intelligent Transportation Systems*, 2011.

[11] J. Lygeros, C. Tomlin, and S. Sastry, "Controllers for reachability specifications for hybrid systems," *Automatica*, vol. 35, pp. 349–370, 1999.

[12] A. Colombo and D. Del Vecchio, "Efficient algorithms for collision avoidance at intersections," in *Hybrid Systems: Computation and Control*, 2012.

[13] M. Hafner, D. Cunningham, L. Caminiti, and D. Del Vecchio, "Cooperative collision avoidance at intersections: Algorithms and experiments," *IEEE Trans. Intell. Transp. Syst*, 2013.

[14] L. Bruni, A. Colombo, and D. Del Vecchio, "Robust multi-agent collision avoidance through scheduling," in *IEEE Conf. on Decision and Control*, 2013.

[15] H. Ahn, A. Rizzi, A. Colombo, and D. Del Vecchio, "Experimental testing of semi-autonomous multi-vehicle control for collision avoidance at intersections," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2015.

[16] G. Rodrigues de Campos, F. Della Rossa, and A. Colombo, "Optimal and least restrictive supervisory control: safety verification methods for human-driven vehicles at traffic intersections," in *IEEE Conf. on Decision and Control*, 2015.

[17] A. Colombo and D. Del Vecchio, "Least restrictive supervisors for intersection collision avoidance: A scheduling approach," *IEEE Trans. Autom. Control*, vol. 60, pp. 1515–1527, 2015.

[18] H. Ahn and D. Del Vecchio, "Semi-autonomous intersection collision avoidance through job-shop scheduling," in *Hybrid Systems: Computation and control*, 2016.

[19] F. Altché, X. Quian, and A. De La Fortelle, "Least restrictive and minimally deviating supervisor for safe semi-autonomous driving at an intersection: An MIQP approach," in *Int. Conf. on Intelligent Transportation Systems*, 2016.

[20] S. A. Reveliotis and E. Roszkowska, "On the complexity of maximally permissive deadlock avoidance in multi-vehicle traffic systems," *IEEE Trans. Autom. Control*, vol. 55, pp. 1646–1651, 2010.

[21] H. Ahn and D. Del Vecchio, "Safety verification and control for collision avoidance at road intersections," *IEEE Trans. Autom. Control*, accepted.

[22] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," *J. Artif. Intell. Res.*, vol. 31, pp. 591–656, 2008.

[23] A. De La Fortelle, "Analysis of reservation algorithms for cooperative planning at intersections," in *Int. Conf. on Intelligent Transportation Systems*, 2010, pp. 445–449.

[24] T.-C. Au, C.-L. Fok, S. Vishwanath, C. Julien, and P. Stone, "Evasion planning for autonomous vehicles at intersections," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2012.

[25] J. Lee and B. Park, "Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment," *IEEE Trans. Intell. Transp. Syst*, vol. 13, pp. 81–90, 2012.

[26] J. Gregoire, S. Bonnabel, and A. De La Fortelle, "Priority-based intersection management with kinodynamic constraints," in *European Control Conf.*, 2014, pp. 2901–2907.

[27] R. Hult, G. R. Campos, P. Falcone, and H. Wymeersch, "An approximate solution to the optimal coordination problem for autonomous vehicles at intersections," in *American Control Conference*, 2015, pp. 763–768.

[28] M. A. S. Kamal, J.-I. 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, pp. 1136–1147, 2015.

[29] N. Murgovski, G. Rodrigues de Campos, and J. Sjöberg, "Convex modeling of conflict resolution at traffic intersections," in *IEEE Conf. on Decision and Control*, 2016, pp. 4708–4713.

[30] G. R. de Campos, P. Falcone, R. Hult, H. Wymeersch, and J. Sjöberg, "Traffic coordination at road intersections: Autonomous decision-making algorithms using model-based heuristics," *IEEE Intell. Transp. Syst. Mag*, 2017.