**Noname manuscript No.**
(will be inserted by the editor)

# Supervisory Control for Collision Avoidance in Vehicular Networks Using Discrete Event Abstractions

**Eric Dallal · Alessandro Colombo · Domitilla Del Vecchio · Stéphane Lafortune**

Received: date / Accepted: date

**Abstract** We consider the problem of controlling a set of vehicles at an intersection, in the presence of uncontrolled vehicles and a bounded disturbance. We begin by discretizing the system in space and time to construct a suitable discrete event system (DES) abstraction, and formally define the problem to be solved as that of constructing a supervisor over the discrete state space that is safe (i.e., collision-free), non-deadlocking (i.e., the vehicles all cross the intersection eventually), and maximally permissive with respect to the chosen discretization. We show how to model the uncontrolled vehicles and the disturbance through uncontrollable events of the DES abstraction. We define two types of relations between systems and their abstraction: state reduction and exact state reduction. We prove that, when the abstraction is a state reduction of a continuous system, then we can obtain a safe, non-deadlocking, and maximally permissive memoryless supervisor. This is obtained by translating safety and non-deadlocking specifications to the abstract domain, synthesizing the supervisor in this domain, and finally translating the supervisor back to the concrete domain. We show that, when the abstraction is an exact state reduction, the resulting supervisor will be maximally permissive among the

E. Dallal
EECS Dept. at the University of Michigan, MI, USA
E-mail: edallal@umich.edu

A. Colombo
DEIB at Politecnico di Milano, Italy
E-mail: alessandro.colombo@polimi.it

D. Del Vecchio
ME Dept. at MIT, MA, USA
E-mail: ddv@mit.edu

S. Lafortune
EECS Dept. at the University of Michigan, MI, USA
E-mail: stephane@umich.edu

class of all supervisors, not merely memoryless ones. Finally, we provide a customized algorithm and demonstrate its scalability through simulation.

**Keywords** Discrete Event Systems · Abstraction · Vehicle Control · Supervisory Control

## 1 Introduction

We consider the problem of controlling a set of $n$ vehicles in the vicinity of an intersection. We assume that vehicles move along a set of $m$ intersecting two-way roads, $m \leq n$, and that the path that each vehicle will follow is known a priori (for example, by means of reading the turn signal of the vehicle), and we want to supervise the vehicles' behaviour to avoid a side impact of any two vehicles on intersecting paths, and a rear-end collision of any two vehicles on a common or on merging paths. See Fig. 1 for an example.

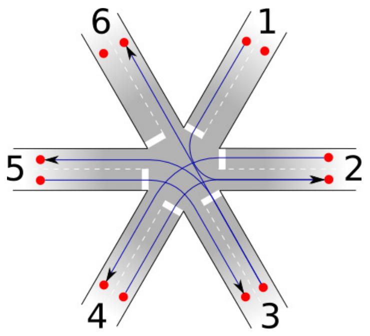


**Fig. 1** An example of the vehicle control problem.

We assume that a certain subset of the vehicles are uncontrolled, and that there is a disturbance on the vehicle dynamics with a known bound. The problem to be solved consists of designing a supervisor that restricts the actions of the controlled vehicles such that the system is safe (i.e., collision-free), non-deadlocking (i.e., the vehicles must eventually cross the intersection), and maximally permissive.

Three common approaches to this problem include: the computation of maximally controlled invariant sets; mapping the problem to that of scheduling; and abstraction/symbolic models. Among approaches falling in the first category, we mention, e.g., [18, 27, 19]. By explicitly computing the *capture set*, or set of states from which it is not possible to guarantee avoidance of the unsafe states, these approaches naturally satisfy safety, non-deadlockingness and maximal permissiveness, and can deal with sources of uncontrollability and also with measurement uncertainty. However, such approaches typically require conditions on the geometry of the unsafe set and on the structure of the dynamics, or else scale poorly to systems with more than a few dimensions. See also [26] for an example involving a flight management system. Scheduling approaches work by allocating time intervals during which the vehicles can be inside the intersection. The scheduling problem is generally NP-hard but takes polynomial time in the special case where all jobs require the same processing time. Reducing the vehicle control problem to the polynomial-time scheduling case amounts to either an assumption of certain symmetries in the vehicle control problem set-up, or a problem relaxation where such symmetries are not satisfied. Approaches in this category include [10], its extension to the case of dynamics with disturbances, [4], its extension to the case of uncontrolled vehicles [1], and its extension to the case of networks of sufficiently spaced intersections, [7]. See also [11], which uses precedence constraints to allow for vehicles on common or non-intersecting trajectories to use the intersection simultaneously. Another approach is to pre-compute fail-safe maneuvers as in [20], or evasion plans as in [3]. These last approaches deal with some types of environmental uncertainty, but do not guarantee maximal permissiveness.

Our approach falls in the category of abstraction/symbolic models. Abstraction based methods work by mapping the continuous system model and specifications to a finite model and solving for a supervisor on the finite model, in such a way that the obtained supervisor can be used on the original (continuous) system, while preserving safety and non-deadlocking properties. Work in this domain includes [2, 15] in the context of verification / model checking, as well as [8, 9, 12], which make use of differential flatness of dynamical systems to construct abstractions with provable errors bounds. Our work is most closely related to that of [16, 22, 30, 5], which construct symbolic models that satisfy simulation or alternating simulation relations with the original system. In particular, this work also makes use of alternating simulation relations, and variations thereof.

In this problem, the number of vehicles will typically be at least five (we provide simulation results for up to six vehicles) and the bad set has a non-convex shape, which makes exact computation of the capture set intractable. On the other hand, the scheduling methods of [10], [4], and [1] do not explicitly pre-compute sets of states from which there exist solutions to the corresponding scheduling problems, but instead perform verification on-line. Because the exact verification problem is NP-hard, only the polynomial-time problem relaxations are feasible in practice. While also suffering from problems related to state space explosion, abstraction based methods nevertheless offer more scalability than capture set computation and more flexibility than reductions to scheduling problems.

We proceed to solve the problem by discretizing the system in space and time, thus obtaining a finite solution space. Using this discretization as a basis, we construct a discrete-event system (DES) abstraction and model the two sources of uncontrollability (the uncontrolled vehicles and the disturbance) through uncontrollable events. By translating the safety and non-deadlocking specifications from the continuous to the discrete-event domain, we formulate the problem to be solved in the context of supervisory control theory of DES (see [23], [28], [6]). Specifically, we obtain a maximally permissive safe and non-deadlocking supervisor for the DES by solving the Basic Supervisor Control Problem in the Non-Blocking case (BSCP-NB). The resulting supervisor is then translated back to the original (continuous) problem domain, preserving safety, non-deadlockingness, and maximal permissiveness with respect to the discretization.

To prove that safety and non-deadlockingness are preserved when translating the obtained supervisor from the abstract back to the continuous problem domain and to characterize the sense in which the resulting solution is maximally permissive, we define two types of relations between systems and their abstractions: the state reduction and the exact state reduction. We prove that, when the abstraction is a state reduction of the original system, the obtained supervisor for the continuous domain problem will be safe, non-deadlocking, and maximally permissive among the class of memoryless supervisors. When the abstraction is an exact state reduction of the original system, the obtained supervisor will be maximally permissive among the class of all supervisors, not merely memoryless ones. In the context of the vehicles control problem, we show that our DES abstraction is a state reduction of the continuous system model. Additionally, we show that, if the bounds on the disturbance are an integral multiple of one of the discretization parameters, then our DES abstraction becomes an exact state reduction of the continuous system model.

Finally, we present a formulation of the control problem as a game against nature and show how this results in a categorization of the discrete states as winning for the controller, winning for nature, or losing for both. We then make use of this through a technique based on iterative refinement, which consists of computing the winning sets on an abstraction with a coarse discretization, and refining the abstraction at states found to be losing for both the controller and nature. By making use of iterative refinement and the problem's structure, we are able to obtain an algorithm that is faster than the standard DES supervisory control algorithms. We show through simulation that the algorithm is scalable in practice, with running times of under one minute for systems with tens of millions of states in the DES abstraction.

Our contributions are as follows. First, the translation of the system model and specifications to the domain of DES allows us to leverage methods from supervisory control theory, methods which are well-suited to finding maximally permissive supervisors in the presence of uncontrolled elements of the environment. Second, the notions of state reduction and exact state reduction are general notions that conserve maximal permissiveness, rather than merely safety and non-deadlockingness, when going from an abstraction back to the original system. To our knowledge, the construction of maximally permissive memoryless supervisors for DES specifications through abstractions has not been considered in other works. Finally, the iterative refinement algorithm presented in this work diminished running time by a factor of over 1000 in some cases and many of the techniques used in this algorithm could also generalize to other problems of interest. Preliminary versions of some of the results presented here have appeared in [13], [14].

The organization of this paper is as follows. In Sec. 2, we present the system model, its time/space discretization, and the problem to be solved. In Sec. 3, we describe the set of collision points to be avoided. In Sec. 4, we define the two modeling formalisms which are used in this paper, namely transition systems and discrete event systems, and present a fixed point algorithm for computing a maximally permissive safe and non-deadlocking supervisor. In Sec. 5, we present the state reduction, exact state reduction, and associated theorems. In Sec. 6, we define the DES abstraction of the system defined in Sec. 2, prove that this abstraction is a state reduction of the system defined in Sec. 2, and additionally prove under what conditions the abstraction is an exact state reduction. In Sec. 7, we present the problem formulation as a game against nature and describe the iterative refinement procedure. In Sec. 8, we present our algorithm for solving the vehicle control problem. In Sec. 9, we present simulation results for an implementation of our algorithm. Finally, we conclude in Sec. 10. We also include derivations of the equations used in our algorithms, which are contained in the appendix.

## 2 Model and Problem Definition

Consider a set of $n$ vehicles $\mathcal{N} = \{1, \ldots, n\}$ modeled as kinematic entities (integrators) and described by

$$\dot{x} = v + d \tag{1}$$

where $x \in X \subset \mathbb{R}^n$ is the state, $v \in V \subset \mathbb{R}^n$ is the control input, and $d \in D \subset \mathbb{R}^n$ is a disturbance input representing unmodeled dynamics (for instance, the dynamic response of the vehicle to the engine torque). That is, $d$ models the discrepancy between the full system model and the simple model $\dot{x} = v$. Assume that $X$ is compact (i.e., the vehicles are controlled in some neighbourhood of the intersection) and that $D = [d_{min}, d_{max}]^n$, with $d_{min} \leq 0 \leq d_{max}$. We take the set $V$ to be the (discrete) set of vectors with elements in the finite set $\{a\mu, (a+1)\mu, \ldots, b\mu\}$, with $a, b \in \mathbb{N}$ and $\mu \in \mathbb{R}_+$. The values $a\mu$ and $b\mu$ are denoted by $v_{min}$ and $v_{max}$, respectively. To allow for the possibility that a subset of the vehicles cannot be controlled, let $v$ be partitioned into two subvectors, $v_c \in V_c$ for the controlled vehicles, and $v_{uc} \in V_{uc}$ for the uncontrolled vehicles, so that $v = (v_c, v_{uc})$ and $V = V_c \times V_{uc}$. Assume also that $v_{min} + d_{min} \geq \mu$, so that $\mu$ constitutes a lower bound on the velocity of the vehicles. Finally, assume that the input $v$ is kept constant over time intervals $[k\tau, (k+1)\tau)$, $k \in \mathbb{N}$ and discretize the above system in time with step $\tau$, obtaining

$$x_{k+1} = x_k + u_k + \delta_k \tag{2}$$

with $x_k = x(k\tau)$, $u_k = v(k\tau)\tau$, $\delta_k = \int_{k\tau}^{(k+1)\tau} d(t)dt$. Calling $U = V\tau$ and $\Delta = D\tau$, we have that $u_k \in U$ and $\delta_k \in \Delta$. In the remainder of this paper, we will also use the notation $\delta_{\min} := d_{\min}\tau$ and $\delta_{\max} = d_{\max}\tau$. As with the set $V$, we use the notation $u = (u_c, u_{uc})$ to denote the controls of the controlled and uncontrolled vehicles and write $U = U_c \times U_{uc}$. Next, we discretize the system in space by defining a set of discrete states $\tilde{Q}$ and a mapping $\ell : X \to \tilde{Q}$ from continuous to discrete states as follows:

$$\ell_i(x_i) := \begin{cases} \begin{array}{c} (c+1/2)\tau\mu, \text{ for } c \in \mathbb{Z} \text{ s.t.} \\ c\tau\mu < x_i \leq (c+1)\tau\mu, \end{array} & \text{if } x_i \leq \alpha_k \\ \quad q_{i,m}, & \text{if } x_i > \alpha_k \end{cases} \tag{3}$$

where $k$ is the index of the road on which vehicle $i$ exits the intersection (i.e., after any turn) and $\alpha_k$ marks the end of the intersection on road $k$ (the shape of the intersection will be described in more detail in Sec. 3). Note that, if the vehicles are to be controlled beyond the end of the intersection, then a value greater than $\alpha_k$ could be used in Eq. (3). This could potentially result in more than one marked state in the definition of $G$ (see Sec. 6) and would not invalidate any results presented in this paper. Define $\ell(x)$ as the vector $(\ell_1(x_1), \ldots, \ell_n(x_n))$ and define the notation $\ell^{-1}(q) = \{x \in X : \ell(x) = q\}$. In words, the space $X$ is covered by a regular lattice with spacing $\tau\mu$. Vehicles before the end of the intersection are mapped to a point of this lattice whereas vehicles after the end of the intersection are mapped to "special" states $q_{i,m}$. The state $q_m = (q_{1,m}, \ldots, q_{n,m})$ is the (unique) discrete state where all vehicles have crossed the intersection. Assume that, for all $q \in \tilde{Q}$, there exists some $x \in X$ such that $\ell(x) = q$. Finally, assume that there is some set $B$ of bad states (representing collision points) and that we would like to define a supervisor so that $x(t) \notin B \, \forall \, t \geq 0$. We will describe the bad set in the following section. Specifically, we wish to solve the following problem:

**Problem 1** Let $X/\ell$ denote the quotient set of $X$ with respect to the equivalence relation $R_\ell \subseteq X \times X$ defined by $(x_1, x_2) \in R_\ell \Leftrightarrow \ell(x_1) = \ell(x_2)$. Given $\tilde{Q}$, define a supervisor $\sigma : X/\ell \to 2^{V_c}$ that assigns to each $x(k\tau) \in X$ a set of inputs $v_c \in V_c$ allowed for the interval $[k\tau, (k+1)\tau)$ and constant over this time interval, with the following properties:

- if $v_c(t) \in \sigma(x(\lfloor t/\tau \rfloor \tau))$ for $t \in [k\tau, (k+1)\tau)$, then $x(t) \notin B$ in the same time interval (safety)
- if $\sigma(x(k\tau)) \neq \emptyset$, $v_c(t) \in \sigma(x(\lfloor t/\tau \rfloor \tau))$ for $t \in [k\tau, (k+1)\tau)$, and $\ell(x((k+1)\tau)) \neq q_m$, then $\sigma(x((k+1)\tau)) \neq \emptyset$ (non-deadlockingness)
- if $\tilde{\sigma} \neq \sigma$ and $\tilde{\sigma}$ satisfies the two properties above, then $\tilde{\sigma}(x) \subseteq \sigma(x)$ for all $x \in X$ (maximal permissiveness). □

## 3 Bad Set Description

Let the set of roads in this system be denoted by $\mathcal{R} = \{1, \ldots, m\}$. Associated to each vehicle $i$ is a pair of roads $(r_{i,1}, r_{i,2})$, indicating that the vehicle starts on road $r_{i,1}$ and turns onto road $r_{i,2}$ at the intersection. Each road $r$ in this system is parametrized by the length $\alpha_r$ of the road that is inside the intersection. We assume that vehicles instantaneously switch from one road to another (i.e., when turning) at point 0. Thus, vehicle $i$ is on road $r_{i,1}$ when $x_i < 0$, inside the intersection when $x_i \in [-\alpha_{r_{i,1}}, \alpha_{r_{i,2}}]$, and on road $r_{i,2}$ when $x_i > 0$. We define any two pairs of roads $(r_{i,1}, r_{i,2})$ and $(r_{j,1}, r_{j,2})$ as *conflicting* in one of two cases:

Case 1 If two vehicles share the same start or end road, they must maintain a minimal separation distance of $\gamma > 0$ while on the same road. Mathematically, $r_{i,1} = r_{j,1} \Rightarrow (x_i, x_j \leq 0 \Rightarrow |x_i - x_j| \geq \gamma)$ and $r_{i,2} = r_{j,2} \Rightarrow (x_i, x_j \geq 0 \Rightarrow |x_i - x_j| \geq \gamma)$.

Case 2 If two vehicles are on trajectories that intersect inside the intersection region while turning, they are forbidden from being in the intersection region simultaneously. Mathematically, $(x_i, x_j) \notin (-\alpha_{r_{i,1}}, \alpha_{r_{i,2}}) \times (-\alpha_{r_{j,1}}, \alpha_{r_{j,2}})$.

It can be shown that, if $r_{i,1} \neq r_{j,1}$ and $r_{i,2} \neq r_{j,2}$, then case 2 occurs when an odd number of

$$r_{j,1} \geq r_{i,1}, \qquad r_{j,1} \geq r_{i,2}, \qquad r_{j,2} \leq r_{i,1}, \text{ and } \qquad r_{j,2} \leq r_{i,2} \tag{4}$$

are true (assuming vehicles driving on the right side of the road). We call the set of all forbidden points the bad set, and denote it by $B$. Note that we do not include collision points involving two uncontrolled vehicles in the bad set, since these cannot be prevented through any control action. If neither case 1 nor case 2 occur for a pair of vehicles $i$ and $j$ (ex: both vehicles turning right), then no constraints are placed on their joint behavior. See Fig. 2 for a pictorial example of cases 1 and 2.

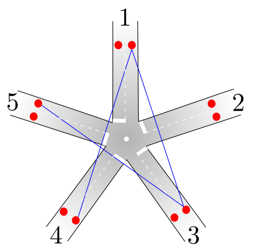


**Fig. 2** An example scenario involving three vehicles on five roads. Blue lines segments are drawn for each vehicle indicating starting road and ending road. Case 1 occurs when two line segments meet at an endpoint, and case 2 occurs when two line segments intersect.

## 4 Modelling Formalisms

This section defines the two types of system models that will be used in this work: transition systems and discrete-event systems. Relations between systems and abstractions are typically described in terms of transition systems. However, the specifications we consider and the solution computation are in the domain of discrete-event systems. Thus, this section will give brief overviews of both types of systems and finally unify the two of them.

## 4.1 Transition Systems

**Definition 1 (Transition System)** A transition system $S$ is defined as a tuple $S = (X, U, \rightarrow, Y, H)$, where $X$ is the set of states, $U$ is a set of control inputs, $\rightarrow \subseteq X \times U \times X$ is a transition relation, $Y$ is an output set, and $H : X \rightarrow Y$ is the output function. $\square$

Henceforth, we will usually refer to transition systems simply as systems. For a system $S = (X, U, \rightarrow, Y, H)$, we will use the notation $\mathbf{Post}_u(x) := \{x' \in X : (x, u, x') \in \rightarrow\}$ and $U(x) := \{u \in U : \mathbf{Post}_u(x) \neq \emptyset\}$. In the remainder of this paper, it will be assumed that all systems satisfy the property $H(x_1) = H(x_2) \Rightarrow U(x_1) = U(x_2)$, for all $x_1, x_2 \in X$. In words, this means that any two states with the same observation should not be distinguishable by their available set of inputs.

**Definition 2 (Run)** A run $\rho$ of length $n$ for a system $S = (X, U, \rightarrow, Y, H)$ is a sequence of past states and inputs $(x^0, u^0, \ldots, x^{n-1}, u^{n-1}, x^n)$, such that $u^i \in U(x^i)$ and $x^{i+1} \in \mathbf{Post}_{u^i}(x^i)$ for $i = 0, \ldots, n-1$. $\square$

The set of runs of length $n$ is denoted by $R_n(S)$ and the set of runs is $R(S) = \bigcup_{i=0}^{\infty} R_n(S)$. We use $R_n(S|x)$ and $R(S|x)$ to denote the set of runs of length $n$ starting from $x$ and the set of all runs starting from state $x$, respectively. For any $D \subseteq X$, also let $R_n(S|D) := \cup_{x \in D} R_n(S|x)$ and $R(S|D) := \cup_{x \in D} R(S|x)$. Given run $\rho = (x^0, u^0, \ldots, x^{n-1}, u^{n-1}, x^n)$, we define the notation $tgt(\rho) := x^n$ and $\rho(k) := (x^0, u^0, \ldots, x^{k-1}, u^{k-1}, x^k)$, called a *prefix* of $\rho$. We will also abuse notation and write $(x, u, x') \in \rho$ if $\rho = (x^0, u^0, \ldots, x^{n-1}, u^{n-1}, x^n)$ and there exists some $i = 0, \ldots, n-1$ such that $x^i = x$, $u^i = u$, and $x^{i+1} = x'$.

**Definition 3 (History)** A history $\theta$ of length $n$ for a system $S = (X, U, \rightarrow, Y, H)$ is a sequence of past outputs and inputs $(y^0, u^0, \ldots, y^{n-1}, u^{n-1}, y^n)$, such that there exists a run $\rho = (x^0, u^0, \ldots, x^{n-1}, u^{n-1}, x^n) \in R_n(S)$ that is consistent with $\theta$, in the sense that $y^i = H(x^i)$ for $i = 0, \ldots, n$. $\square$

The set of histories of length $n$ is denoted by $\Theta_n(S)$ and the set of histories is $\Theta(S) = \bigcup_{i=0}^{\infty} \Theta_n$. We will also write $\theta(\rho)$ to mean the unique history produced by a run $\rho \in R$. We use $\Theta_n(S|x) = \{\theta(\rho) | \rho \in R_n(S|x)\}$ and $\Theta(S|x) = \{\theta(\rho) | \rho \in R(S|x)\}$ to denote the set of histories of length $n$ starting from $x$ and the set of all histories starting from state $x$, respectively. For any $D \subseteq X$, also let $\Theta_n(S|D) := \cup_{x \in D} \Theta_n(S|x)$ and $\Theta(S|D) := \cup_{x \in D} \Theta(S|x)$. Given history $\theta = (y^0, u^0, \ldots, y^{n-1}, u^{n-1}, y^n)$, we define the notation $\theta(k) := (y^0, u^0, \ldots, y^{k-1}, u^{k-1}, y^k)$ and $tgt(\theta) := y^n$, as was the case with runs.

**Definition 4 (Specification)** A safety specification for a system $S = (X, U, \rightarrow, Y, H)$ is a subset $Safe \subseteq \rightarrow$ of transitions that we would like the system $S$ to be restricted to. A marking specification for $S$ is a set $X_m \subseteq X$ of "special" or marked states. We say that $S$ is deadlocking if there exists a run $\rho$ such that $U(tgt(\rho)) = \emptyset$ and $tgt(\rho) \notin X_m$. $\square$

**Definition 5 (Supervisor)** A supervisor $\sigma$ for a system $S = (X, U, \rightarrow, Y, H)$ is a function $\sigma : \Theta \rightarrow 2^U$ which chooses which control inputs to enable/disable after each history. A supervisor is called memoryless if it is of the form $\sigma : Y \rightarrow 2^U$. A run $\rho = (x^0, u^0, \ldots, x^{n-1}, u^{n-1}, x^n) \in R_n(S)$ is allowed by supervisor $\sigma$ if $u^i \in \sigma(\theta(\rho(i)))$, for $i = 0, \ldots, n-1$. $\square$

**Definition 6 (Specification Satisfaction)** A supervisor $\sigma$ for system $S = (X, U, \rightarrow, Y, H)$ is safe with respect to $Safe \subseteq \rightarrow$ if every run $\rho = (x^0, u^0, \ldots, x^{n-1}, u^{n-1}, x^n) \in R(S)$ allowed by $\sigma$ satisfies $(x^i, u^i, x^{i+1}) \in Safe$ for $i = 0, \ldots, n-1$. Supervisor $\sigma$ is non-deadlocking with respect to $X_m \subseteq X$ on domain $D \subseteq X$ if every run $\rho \in R(S|D)$ allowed by $\sigma$ satisfies $\sigma(\theta(\rho)) \neq \emptyset \vee tgt(\rho) \in X_m$. $\square$

**Definition 7 (Maximal Permissiveness)** Given a system $S = (X, U, \rightarrow, Y, H)$, a safety specification $Safe \subseteq \rightarrow$, and a marking specification $X_m \subseteq X$, supervisor $\sigma$ is maximally permissive on domain $D$ with respect to these safety and non-deadlocking conditions if there does not exist a supervisor $\sigma' \supset_D \sigma$ which also satisfies these conditions, where $\sigma' \supset_D \sigma$ signifies that $\sigma'(\theta) \supseteq \sigma(\theta)$ for all $\theta \in \Theta(S|D)$ and that there exists $\theta \in \Theta(S|D)$ such that $\sigma'(\theta) \supset \sigma(\theta)$. Finally, $\sigma$ is safe and non-deadlocking on a maximal domain $D$ if there does not exist a supervisor $\sigma'$ that is safe and non-deadlocking on a larger domain $D' \supset D$. $\square$

It should be noted that there exists a unique maximal domain on which a supervisor can be safe and non-deadlocking, and a unique maximally permissive supervisor on this domain. Furthermore, it is not possible to construct a supervisor that is safe, non-deadlocking, and strictly more permissive by considering a smaller domain. Thus, from this point forward we will refer to the conjunction of both the property of being maximally permissive and the property of being safe and non-deadlocking on a maximal domain simply as maximal permissiveness. These issues will become clearer in Sec. 4.4.

For any safe, non-deadlocking, and maximally permissive supervisor on domain $D$, we may assume (without loss of generality) $\sigma(\rho) = \emptyset$ for all $\rho = (x)$ such that $x \notin D$. Under this assumption, there exists a domain $D$ under which a supervisor $\sigma$ is non-deadlocking with respect to $X_m \subseteq X$ if and only if every *non-zero length* run $\rho \in R(S)$ allowed by $\sigma$ satisfies $\sigma(\theta(\rho)) \neq \emptyset \vee tgt(\rho) \in X_m$ (if a zero length run $\rho = (x)$ does not satisfy this condition, we may simply take $x \notin D$). Moreover, to verify that a *memoryless* supervisor is non-deadlocking on some domain $D$, it is sufficient to consider runs of length exactly one. To see this, consider any run $\rho = (x_0, \ldots, x_{n-1}, u_{n-1}, x_n) \in R_n(S)$ that is allowed by $\sigma$. Clearly, since $\rho$ is allowed by $\sigma$, it must be that $\sigma(\theta(\rho(k))) \neq \emptyset$, for any $k < n$. Hence, only the last state reached along $\rho$ may be deadlocked, and this may be precluded if we know that the run $(x_{n-1}, u_{n-1}, x_n) \in R_1(S)$ is non-deadlocking. This fact will be used in the proof of Thm. 3.

## 4.2 Discrete-Event Systems

**Definition 8 (Discrete Event System)** A (deterministic) discrete event system is a tuple $G = (X, E, \psi, x_0, X_m)$ where $X$ is a set of states, $E$ is a set of events, $\psi : X \times E \to X$ is a partial transition function, $x_0 \in X$ is the initial state, and $X_m \subseteq X$ is a set of marked states representing the completion of some behavior of interest. $\square$

Given a set of events $E$, $E^*$ denotes the set of finite strings of events in $E$. A set of strings $K \subseteq E^*$ is called a *language*. The *prefix-closure* of a language $K \subseteq E^*$, denoted by $\overline{K}$, is defined by $\overline{K} = \{s \in E^* : \exists t \in E^* \wedge st \in K\}$. Given a DES $G = (X, E, \psi, x_0, X_m)$, $\psi$ is extended from events to strings through $\psi(x, se) = \psi(\psi(x, s), e)$. The language *generated* by $G$, denoted by $\mathcal{L}(G)$, is defined as $\mathcal{L}(G) := \{s \in E^* : \psi(x_0, s)!\}$, where ! means "is defined". The language *marked* by $G$, denoted by $\mathcal{L}_m(G) \subseteq \mathcal{L}(G)$ is defined by $\mathcal{L}_m(G) := \{s \in \mathcal{L}(G) : \psi(x_0, s) \in X_m\}$. DES $G$ is *non-blocking* if $\overline{\mathcal{L}_m(G)} = \mathcal{L}(G)$, and blocking otherwise.

A specification for a DES $G$ is given by a second DES $H$ defined over the same event set and satisfying $\mathcal{L}(H) \subseteq \mathcal{L}(G)$ and $\mathcal{L}_m(H) \subseteq \mathcal{L}_m(G)$. Here, $\mathcal{L}(H)$ constitutes the legal sublanguage of $\mathcal{L}(G)$, representing safe system behavior. The language $\mathcal{L}_m(H)$ is usually assumed to satisfy the property $\mathcal{L}_m(H) = \mathcal{L}(H) \cap \mathcal{L}_m(G)$ (a technical condition called $\mathcal{L}_m(G)$-closure). In general, the event set of $G$ and $H$, denoted by $E$ is partitioned into controllable events $E_c$ and uncontrollable events $E_{uc}$. Controllable events are events which can be disabled (i.e., prevented), whereas uncontrollable events cannot be disabled. Control in the DES domain is concerned with obtaining a supervisor $S : \mathcal{L}(G) \to 2^E$ that is safe (i.e., $\mathcal{L}(S/G) \subseteq \mathcal{L}(H)$ and $\mathcal{L}_m(S/G) \subseteq \mathcal{L}_m(H)$), non-blocking (i.e., $\overline{\mathcal{L}_m(S/G)} = \mathcal{L}(S/G)$), and maximally permissive, where $S/G$ is the system $G$ controlled by $S$. Obtaining this supervisor consists of solving the basic supervisory control problem in the non-blocking case, or BSCP-NB, as described in [23], [6].

The solution to problem BSCP-NB is the language $(\mathcal{L}_m(H))^{\uparrow C}$, where $\uparrow C$ denotes the supremal controllable sublanguage operation. This is the largest sublanguage $K \subseteq \mathcal{L}_m(H)$ that is *controllable*, which means it satisfies the property $\overline{K}E_{uc} \cap \mathcal{L}(G) \subseteq \overline{K}$. Thus, $K$ is controllable if there exist no strings in $\overline{K}$ that can be extended by an uncontrollable event to a string in $\mathcal{L}(G) \setminus \overline{K}$. The standard algorithm which solves this problem is given in [29] and constructs a supervisor $S$ such that $\mathcal{L}_m(S/G) = (\mathcal{L}_m(H))^{\uparrow C}$ and $\mathcal{L}(S/G) = \overline{(\mathcal{L}_m(H))^{\uparrow C}}$. Notably, controllable sublanguages are closed under union, so that a unique maximal solution indeed exists.

## 4.3 Translating Between Transition Systems and Discrete Event Systems

The previous two sections describe models for systems and specifications using the two formalisms of transition systems and DES. In this section, we show how to translate a system model and specification from the domain of transition systems to the domain of DES, and unify notation between these. In what follows, the notation ! will be used to mean that a partial function is defined at a particular value. Given a system $S_a = (X_a, U_a, \rightarrow_a, Y_a, H_a)$, a safety specification $Safe_a \subseteq \rightarrow_a$ and a marking specification $X_{m,a} \subseteq X_a$, construct system automaton $G_a := (X_a \cup Z_a, E_c \cup E_{uc}, \psi_{G_a}, x_{a,0}, X_{m,a})$, and specification automaton $H_a := (X_a \cup Z_a, E_c \cup E_{uc}, \psi_{H_a}, x_{a,0}, X_{m,a})$ satisfying the following conditions:

$$E_c = U_a \tag{5}$$
$$\psi_{H_a} \subseteq \psi_{G_a} \subseteq (X_a \times E_c \times Z_a) \cup (Z_a \times E_{uc} \times (X_a \cup Z_a)) \tag{6}$$
$$\psi_{G_a}(x_a, u_a)! \Leftrightarrow \exists x'_a \in X_a : (x_a, u_a, x'_a) \in \rightarrow_a \tag{7}$$
$$\psi_{H_a}(x_a, u_a)! \Leftrightarrow \exists x'_a \in X_a : (x_a, u_a, x'_a) \in Safe_a \tag{8}$$
$$\exists t \in E^*_{uc} : \psi_{G_a}(x_a, u_a t) = x'_a \Leftrightarrow (x_a, u_a, x'_a) \in \rightarrow_a \tag{9}$$
$$\exists t \in E^*_{uc} : \psi_{H_a}(x_a, u_a t) = x'_a \Leftrightarrow (x_a, u_a, x'_a) \in Safe_a, \tag{10}$$

where $Z_a$ is a set of intermediate states. The above equations can be understood to mean that we use uncontrollable events in $H_a$ and $G_a$ to model non-determinism in the transition relation $\rightarrow_a$. In words, Eq. (5) signifies that the controllable events of $G_a$ and $H_a$ are the control inputs of $S_a$, whereas Eq. (6) signifies that controllable (resp. uncontrollable) events are defined only from states in $X_a$ (resp. $Z_a$) and lead only to states in $Z_a$ (resp. $X_a \cup Z_a$). Eqs. (7) and (9) signify that, for every $(x_a, u_a, x'_a) \in \rightarrow_a$, event $u_a$ is defined from state $x_a$ of $G_a$ and there exists some uncontrollable sequence of events following $u_a$ that takes $G_a$ from $\psi_{G_a}(x_a, u_a)$ to $x'_a$. The same interpretation holds for Eqs. (8) and (10) with respect to $H_a$.

We remark that we did not define what the initial state $x_{a,0}$ of $G_a$ and $H_a$ is. For now, we note that these will be dummy initial states without physical significance, but with transitions to some subset $X_{a,0}$ of initial states. We will return to this issue in Sec. 6.

To unify notation between systems as in Def. 1 and discrete event systems as described above, we will use the notation $U(x) := \{u \in E_c : \psi(x, u)!\}$ and $\mathbf{Post}_u(x) := \{x' \in X_a : (\exists t \in E^*_{uc})(x' = \psi(x, ut))\}$ for $x \in X_a$ and (in an abuse of notation) will write $(x, u, x') \in \psi$ if $x \in X_a$ and $x' \in \mathbf{Post}_u(x)$. This notation allows us to work with DES of the above form in the context of the state reductions and exact state reductions that will be presented in the following sections.

### 4.4 Supervisor Computation

In this section, we present the algorithm for solving problem BSCP-NB, recast as a state-based maximal fixed point computation, in a manner that is more akin to existing methods for control problems in the context of transition systems (see, e.g., [25]). Note that we use non-deadlocking specifications in this work, rather than the more general non-blocking specifications that are normally used in DES supervisory control problems.

Consider a transition system $S = (X, U, \rightarrow)$ (we have suppressed the last two arguments, as they are not relevant to the following discussion), a safety specification $Safe \subseteq \rightarrow$, and a marking specification $X_m \subseteq X$, as described in Sec. 4.1. We will define a function $F : 2^X \rightarrow 2^X$ whose greatest fixed point is used to obtain a maximally permissive supervisor with respect to the safety and non-deadlocking specifications.

Recall from Sec. 4.2 that the solution to the supervisory control problem in DES is the supremal controllable sublanguage, and that a language $K$ is controllable if there exist no strings in $\overline{K}$ that can be extended by an uncontrollable event to a string in $\mathcal{L}(G) \setminus \overline{K}$. Recall also from Sec. 4.3 that we use uncontrollable events to model non-determinism in the translation from transition systems to DES. Thus, given a state $x \in X$ and a current set of winning states $Z$ in the iteration of (the yet to be defined function) $F$, let $Cont(x|Z)$ denote the set of control inputs that do not violate controllability. Mathematically,

$$Cont(x|Z) = \{u \in U | \forall x' \in \mathbf{Post}_u(x), [(x, u, x') \in Safe \wedge x' \in Z]\}. \tag{11}$$

We now define $F : 2^X \rightarrow 2^X$ by

$$F(Z) = \{x \in Z | x \in X_m \vee Cont(x|Z) \neq \emptyset\}. \tag{12}$$

Now let $F^k(Z)$ denote the $k^{th}$ iteration of $F$ applied to $Z$, definable through the recursion $F^0(Z) = Z$ and $F^k(Z) = F(F^{k-1}(Z))$. Because $F$ is monotone, we obtain by the Knaster-Tarski theorem that the greatest fixed point $\nu Z.F(Z) = \lim_{k\rightarrow\infty} F^k(X)$ is well defined. A maximally permissive supervisor $\sigma : X \rightarrow 2^U$ for the safety and non-deadlocking specification exists, and is given by

$$\sigma(x) = Cont(x|\nu Z.F(Z)). \tag{13}$$

*Remark 1* Typically, DES have a fixed initial state, and the supervisor computation removes both states that are deadlocked, as well as states that are not accessible (not reachable from the initial state). We note here that inaccessible states may be removed at the end of the fixed point algorithm, as an inaccessible state is by definition unreachable from any accessible state. Thus, the inaccessible states cannot affect either deadlocking properties or controllability properties of any accessible state. As a consequence, the supervisor that is computed through Eq. (13) will be correct at all accessible states. □

## 5 State Reductions and Supervisory Control

In this section, we define two types of relations between systems: state reductions and exact state reductions, and prove theorems relating safety, non-deadlockness, and maximal permissiveness of supervisors for systems related through state reductions and exact state reductions. The state reduction and exact state reduction relations are based on the notions of alternating similarity relations, as defined in [25]. The theorems proven in this section will be used later in this paper to establish the correctness of our solution to Prob. 1.

We begin with a motivating example.

*Example 1* Consider the simple transition system $T = (X, U, \to, Y, H)$, with

$$\begin{aligned}
X &= [0,1) \\
U &= \{low, high\} \\
\to &= \left\{ (x,u,x') \in X \times U \times X \;\middle|\; \begin{array}{c} (u = low \wedge x' = x/2) \\ \vee(u = high \wedge x' = (1+x)/2) \end{array} \right\} \\
Y &= \{y\} \\
H(x) &= y, \quad \forall x \in X.
\end{aligned}$$

Let us suppose that there is no initial state information about the system. Given a state estimate of $[a,b)$, the state estimate following control input *low* will be $[a/2, b/2)$, whereas the state estimate following control input *high* will be $[(1+a)/2, (1+b)/2)$. In either case, the estimate has been reduced from an interval of size $b-a$ to one of size $(b-a)/2$. Thus, every single past input that is remembered by a controller yields exactly 1 bit of information about the current state. Notably, there is no finite number of past control decisions over a run beyond which no further information about the current state is obtained. For such an example, maximally permissive control could require not only memory, but infinite memory and hence an infinite state space. □

Because abstractions of systems typically have large state spaces, non-memoryless supervisors will typically be computationally infeasible. As the above example demonstrates, there exist very simple systems under which even finite memory supervisors are insufficient to the problem of obtaining maximally permissive supervisors. In this work, we therefore concentrate on determining conditions under which maximally permissive *memoryless* supervisors can be obtained, and also address the problem of finding conditions on system dynamics under which there is no loss by restricting attention to memoryless supervisors.

## 5.1 The State Reduction

**Definition 9 (State Reduction)** Given two systems $S_a$ and $S_b$ with $Y_a = Y_b = Y$, we say that $S_a$ is a state reduction of $S_b$ with state relation $R \subseteq X_a \times X_b$ and output dependent control relation $C : Y \to 2^{U_a \times U_b}$ (hereafter referred to only as control relation) if:

1. $R^{-1} = \{(x_b, x_a) \subseteq X_b \times X_a : (x_a, x_b) \in R\}$ is a function.
2. For every $y \in Y$, the relation $C(y) \subseteq U_a \times U_b$ is a bijection relation.
3. $H_a(x_a) = H_b(x_b)$ if and only if $(x_a, x_b) \in R$.
4. $\forall (x_a, u_a, x'_a) \in \to_a$, $\exists (x_b, u_b, x'_b) \in \to_b$ such that $(x_a, x_b) \in R$, $(u_a, u_b) \in C(H_a(x_a))$, and $(x'_a, x'_b) \in R$.

5. $\forall(x_b, u_b, x_b') \in \rightarrow_b$, $\exists(x_a, u_a, x_a') \in \rightarrow_a$ such that $(x_a, x_b) \in R$, $(u_a, u_b) \in C(H_b(x_b))$, and $(x_a', x_b') \in R$.

□

*Remark 2* The state reduction was first defined in [14], where we used slightly different conditions. In this work, we have changed notation for the control relation $C$ to resolve ambiguity. Furthermore, condition 5) was previously stated as: $\forall(x_a, x_b) \in R$, $(u_a, u_b) \in C$ and $x_b' \in \mathbf{Post}_{u_b}(x_b)$, $\exists x_a' \in \mathbf{Post}_{u_a}(x_a)$ such that $(x_a', x_b') \in R$. The two conditions can be shown to be equivalent under conditions 1) and 2). □

In words, condition 1) signifies that every $x_b \in X_b$ is in relation with exactly one $x_a \in X_a$, condition 5) signifies that, for every $(x_b, u_b, x_b') \in \rightarrow_b$, there exists $(x_a, u_a, x_a') \in \rightarrow_a$ which models $(x_b, u_b, x_b') \in \rightarrow_b$, and condition 4) signifies that every transition in $\rightarrow_a$ models *some* transition in $\rightarrow_b$. Significantly, conditions 4) and 5) can be achieved by construction for any system $S_b$, and relations $R$ and $C$ satisfying conditions 1), 2), and 3). Furthermore, the system $S_a$ is the quotient system of $S_b$ with respect to $R$ and $C$ in the case of alternating simulation, and is therefore uniquely defined.

**Definition 10 (Induced Specification)** Given system $S_b$ with state reduction $S_a$, along with safety and marking specifications $Safe_b \subseteq \rightarrow_b$ and $X_{m,b} \subseteq X_b$ on system $S_b$, define the induced specification on $S_a$ as follows:

$$\begin{aligned} &(x_a, u_a, x_a') \in Safe_a \subseteq \rightarrow_a \\ \Leftrightarrow &\left\{ \begin{array}{c} (x_b, u_b, x_b') \in \rightarrow_b \text{ s.t. } (x_a, x_b) \in R \\ \wedge (u_a, u_b) \in C(H_a(x_a)) \wedge (x_a', x_b') \in R \end{array} \right\} \subseteq Safe_b \quad (14) \\ &x_a \in X_{m,a} \subseteq X_a \Leftrightarrow \{x_b \in X_b \text{ s.t. } (x_a, x_b) \in R\} \subseteq X_{m,b} \quad (15) \end{aligned}$$

□

The usefulness of Def. 9 is illustrated in the following theorem:

**Theorem 1** *Suppose that system $S_a$ is a state reduction of system $S_b$ with state relation $R$ and control relation $C$ and that we are given safety and marking specifications $Safe_b \subseteq \rightarrow_b$ and $X_{m,b} \subseteq X_b$ for system $S_b$. Let $Safe_a$ and $X_{m,a}$ be the corresponding induced specifications for system $S_a$ and suppose that we additionally have the property $(x_a, x_b) \in R \Rightarrow (x_a \in X_{m,a} \Leftrightarrow x_b \in X_{m,b})$. Finally, let $\sigma_a : Y \rightarrow 2^{U_a}$ be the maximally permissive, safe, and non-deadlocking supervisor, where $Y$ is the (common) output space, and define the supervisor $\sigma_b : Y \rightarrow 2^{U_b}$ by $u_b \in \sigma_b(y)$ iff $\exists u_a \in \sigma_a(y)$ such that $(u_a, u_b) \in C(y)$. Then $\sigma_b$ is safe, non-deadlocking, and maximally permissive among memoryless supervisors of the form $\sigma_b : Y \rightarrow 2^{U_b}$.*

*Proof.* We proceed in three claims. The first two claims show that $\sigma_a$ is non-deadlocking (resp., safe) if and only if $\sigma_b$ is non-deadlocking (resp., safe). The last claim uses monotonicity of the mapping from $\sigma_a$ to $\sigma_b$ to show that the

first two claims imply maximal permissiveness of $\sigma_b$.
**Claim 1**: $\sigma_a$ is non-deadlocking iff $\sigma_b$ is non-deadlocking.
As per the discussion following Def. 6, it suffices to consider runs of length one when verifying non-deadlockingness. Thus, the property to be proven is as follows

$$\begin{aligned} &\left\{ \begin{array}{c} \forall x_a \in X_a, \forall u_a \in \sigma_a(H_a(x_a)), \forall x'_a \in \mathbf{Post}_{u_a}(x_a), \\ [x'_a \in X_{m,a} \vee \sigma_a(H_a(x'_a)) \neq \emptyset] \end{array} \right\} \\ \Leftrightarrow &\left\{ \begin{array}{c} \forall x_b \in X_b, \forall u_b \in \sigma_b(H_b(x_b)), \forall x'_b \in \mathbf{Post}_{u_b}(x_b), \\ [x'_b \in X_{m,b} \vee \sigma_b(H_b(x'_b)) \neq \emptyset] \end{array} \right\}. \end{aligned} \tag{16}$$

($\Rightarrow$) Consider any $x_b \in X_b$, any $u_b \in \sigma_b(H_b(x_b))$, any $x'_b \in \mathbf{Post}_{u_b}(x_b)$, and let $y = H_b(x_b)$ and $y' = H_b(x'_b)$. By property (5) of Def. 9, there exist $x_a \in X_a$, $u_a \in U_a$, and $x'_a \in \mathbf{Post}_{u_a}(x_a)$ such that $(x_a, x_b) \in R$, $(u_a, u_b) \in C(y)$, and $(x'_a, x'_b) \in R$. By property (3) of Def. 9, $H_a(x_a) = H_b(x_b) = y$ and $H_a(x'_a) = H_b(x'_b) = y'$. Since $(u_a, u_b) \in C(y)$ and $u_b \in \sigma_b(y)$, we have that $u_a \in \sigma_a(y)$, and hence that $x'_a \in X_{m,a} \vee \sigma_a(y') \neq \emptyset$. By definition of $X_{m,a}$ in Eq. (15), we have that $x'_a \in X_{m,a} \wedge (x'_a, x'_b) \in R \Rightarrow x'_b \in X_{m,b}$. By definition of $\sigma_b$, we have that $\sigma_a(y') \neq \emptyset \Rightarrow \sigma_b(y') \neq \emptyset$. Thus, $x'_a \in X_{m,a} \vee \sigma_a(y') \neq \emptyset$ implies $x'_b \in X_{m,b} \vee \sigma_b(y') \neq \emptyset$ and we are done.
($\Leftarrow$) Suppose that there exist $x_a \in X_a$, $u_a \in \sigma_a(H_a(x_a))$, and $x'_a \in \mathbf{Post}_{u_a}(x_a)$ such that $x'_a \notin X_{m,a}$ and $\sigma_a(H_a(x'_a)) = \emptyset$. Let $y = H_a(x_a)$ and $y' = H_a(x'_a)$. By property (4) of Def. 9, there exist $x_b \in X_b$, $u_b \in U_b$, and $x'_b \in \mathbf{Post}_{u_b}(x_b)$ such that $(x_a, x_b) \in R$, $(u_a, u_b) \in C(y)$, and $(x'_a, x'_b) \in R$. Since $(u_a, u_b) \in C(y)$ and $u_a \in \sigma_a(y)$, we have that $u_b \in \sigma_b(y)$. By property (3) of Def. 9, $H_b(x_b) = H_a(x_a) = y$ and $H_b(x'_b) = H_a(x'_a) = y'$. By definition of $\sigma_b$, $\sigma_a(y') = \emptyset \Rightarrow \sigma_b(y') = \emptyset$. By assumption, $(x_a, x_b) \in R \Rightarrow (x_a \in X_{m,a} \Leftrightarrow x_b \in X_{m,b})$. Since $(x'_a, x'_b) \in R$ and $x'_a \notin X_{m,a}$, it follows that $x'_b \notin X_{m,b}$, and we are done.
**Claim 2**: $\sigma_a$ is safe iff $\sigma_b$ is safe.
Mathematically, this requires proving

$$\begin{aligned} &\{\forall x_a \in X_a, \forall u_a \in \sigma_a(H_a(x_a)), \forall x'_a \in \mathbf{Post}_{u_a}(x_a), (x_a, u_a, x'_a) \in Safe_a\} \\ \Leftrightarrow &\{\forall x_b \in X_b, \forall u_b \in \sigma_b(H_b(x_b)), \forall x'_b \in \mathbf{Post}_{u_b}(x_b), (x_b, u_b, x'_b) \in Safe_b\}. \end{aligned} \tag{17}$$

($\Rightarrow$) Consider any $x_b \in X_b$, any $u_b \in \sigma_b(H_b(x_b))$, any $x'_b \in \mathbf{Post}_{u_b}(x_b)$, and let $y$, $y'$, $x_a$, $u_a$, and $x'_a$ be as in ($\Rightarrow$) of Claim 1. Then $(x_a, u_a, x'_a) \in Safe_a$ and by Eq. (14), $(x_b, u_b, x'_b) \in Safe_b$.
($\Leftarrow$) Suppose that there exist $x_a \in X_a$, $u_a \in \sigma_a(H_a(x_a))$, and $x'_a \in \mathbf{Post}_{u_a}(x_a)$ such that $(x_a, u_a, x'_a) \notin Safe_a$. By Eq. (14), there exists $(x_b, u_b, x'_b) \in \rightarrow_b$ such that $(x_a, x_b) \in R$, $(u_a, u_b) \in C(y)$, $(x'_a, x'_b) \in R$, and $(x_b, u_b, x'_b) \notin Safe_b$. Since $(x_a, x_b) \in R$, we have, by property (3) of Def. 9, that $H_a(x_a) = H_b(x_b) = y$. By definition of $\sigma_b$ and the fact that $(u_a, u_b) \in C(y)$, we have that $u_b \in \sigma_b(y)$ and we are done.
**Claim 3**: $\sigma_b$ is maximally permissive.
Given any supervisor $\sigma'_b : Y \to 2^{U_b}$, let $\sigma'_a : Y \to 2^{U_a}$ be defined by $u_a \in \sigma'_a(y)$ iff $\exists u_b \in \sigma'_b(y)$ such that $(u_a, u_b) \in C(y)$ and let the function $\sigma_{b \to a}$ be the mapping which takes a supervisor $\sigma'_b$ for system $b$ to the supervisor $\sigma'_a$ for

system $a$ in this way. Clearly, $\sigma'_b \subseteq \sigma_b \Leftrightarrow \sigma_{b\to a}(\sigma'_b) \subseteq \sigma_{b\to a}(\sigma_b) = \sigma_a$. Thus, if there exists a safe and non-deadlocking supervisor $\sigma'_b \nsubseteq \sigma_b$ then it follows that $\sigma_a$ is not maximally permissive, a contradiction. □

The above theorem shows that it is possible to compute a supervisor for a system with a large or infinite state space by abstracting that system to one with a finite state space, computing a supervisor for the reduced system, and translating back. Furthermore, this process conserves not only safety and non-deadlockingness in the translation, but also maximal permissiveness.

*Remark 3* The above theorem characterizes a controller $\sigma$ as safe and non-deadlocking for system $S = (X, U, \to, Y, H)$, safety specification $Safe$, and marking specification $X_m$ if and only if $\forall x \in X$, $\forall u \in \sigma(H(x))$, $\forall x' \in \mathbf{Post}_u(x)$, we have that $(x, u, x') \in Safe \wedge (\sigma(H(x')) \neq \emptyset \vee x' \in X_m)$. This is a sufficient condition for a system to be safe and non-deadlocking, but it is not necessary if the supervisor can use initial state information, even if we restrict attention to memoryless supervisors. For an example of such a situation, see Example 2. □

*Example 2* Figure 3 shows an example of a system (left) and its corresponding state reduction (right). If we assume that there is only a marking specification and no safety specification, then the maximally permissive supervisor $\sigma_1$ for the state reduction would enable $\{a, b\}$ from state $\{1, 2\}$ and $\{a\}$ from state $\{3, 4, 5\}$. It can be seen that this would indeed be a maximally permissive memoryless solution for the left system if there were no initial state information. If, however, the initial state is known a priori to be one of $\{1, 2\}$, then there exists a strictly more permissive memoryless supervisor $\sigma_2$ for the left system which also enables $b$ from states $\{3, 4, 5\}$. It is possible to be more permissive from states $\{3, 4, 5\}$ by making use of the fact that the initial states are $\{1, 2\}$ and event $c$ was disabled from states $\{1, 2\}$, making state 5 unreachable. Another safe memoryless supervisor $\sigma_3$ enables $\{a\}$ from states $\{1, 2\}$ and $\{a, b, c\}$ from states $\{3, 4, 5\}$. Thus, it is possible to enable more from states $\{3, 4, 5\}$ by enabling less from states $\{1, 2\}$.

Consistent with the discussion of Remark 3, both of these supervisors violate the property of Eqs. (16) and (17), namely that $\forall x \in X$, $\forall u \in \sigma(H(x))$, $\forall x' \in \mathbf{Post}_u(x)$, we have that $(x, u, x') \in Safe \wedge (\sigma(H(x')) \neq \emptyset \vee x' \in X_m)$. In particular, $\sigma_2$ and $\sigma_3$ both allow $b$ from state 5, despite the fact that this allows $(5, b, 8)$, and state 8 is deadlocked. Furthermore, the union of $\sigma_2$ and $\sigma_3$ is deadlocking, since it allows the string $bc$, which leads to deadlocked state 8. Thus, there does not exist a maximally permissive safe and non-deadlocking supervisor which uses the initial state information in this case. The key point is that the property of Eqs. (16) and (17) is based on the discussion following Def. 6, which requires that the *suffix* of a run (including, in particular, suffixes of length 1) also be a run. Clearly, this is something which does not apply when there is initial state information.

Note that the system on the left is accessible, deterministic, and has both initial and marked states which respect the partition of states determined by

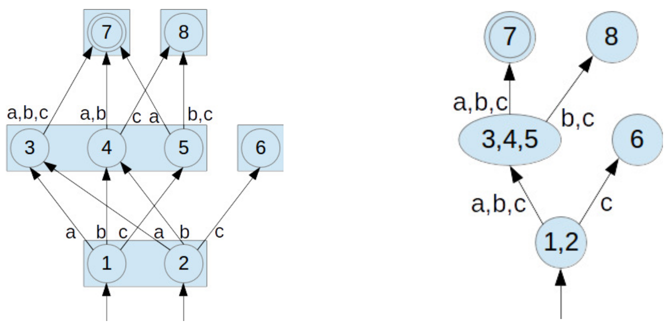


**Fig. 3** A system and its corresponding state reduction. States of the left system with the same output are placed in a common box. We use the usual DES convention of denoting marked states with a double circle and initial states with an incoming arrow that has no source state.

the output map. This example is very closely related to the problem of obtaining maximally permissive supervisors of the form $S : X_G \to 2^E$ for a discrete event system $G$, subject to specification automaton $H$, which would normally require the supervisor to be defined over the state space of the product automaton $G \times H$. □

## 5.2 The Exact State Reduction

**Definition 11 (Exact State Reduction (2))** Given two systems $S_a$ and $S_b$ with $Y_a = Y_b = Y$, we say that $S_a$ is an exact state reduction (2) of $S_b$ with state relation $R \subseteq X_a \times X_b$ and control relation $C : Y \to 2^{U_a \times U_b}$ if $S_a$ is a state reduction of $S_b$ with state and control relations $R$ and $C$ and:

6. $\forall (x_a, u_a, x'_a) \in \to_a$, $\forall x'_b \in X_b : (x'_a, x'_b) \in R$, $\exists (x_b, u_b, x'_b) \in \to_b$ such that $(x_a, x_b) \in R$ and $(u_a, u_b) \in C(H_a(x_a))$. □

The above condition is akin to a time-reversed alternating similarity condition, in the sense that it requires that every transition of $S_a$ have a corresponding transition in $S_b$, for every pair of related *target* states, rather than for every pair of related *source* states. Lemma 1 demonstrates its usefulness.

*Remark 4* The exact state reduction was first defined in [14], where we used a normal (i.e., non time-reversed) alternating similarity condition. We have added the “(2)” in this work to differentiate between these. □

**Lemma 1** *Suppose that system $S_b$ has an exact state reduction (2) $S_a$. Then, for any history $\theta_b$ for system $S_b$ and any $x_b \in X_b$ such that $H(x_b) = tgt(\theta_b)$, there exists a run $\rho_b$ such that $\theta_b = \theta(\rho_b)$ and $x_b = tgt(\rho_b)$.*

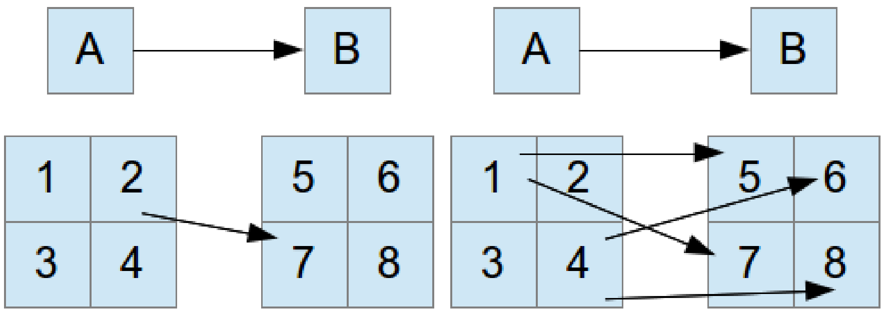
*Proof.* The proof is by induction on the length of $\theta_b$. The base case is trivially true. Assume that the lemma holds up to histories of length $n$ and consider a pair of histories $\theta_b \in \Theta_n(S_b)$ and $\theta'_b \in \Theta_{n+1}(S_b)$ such that $\theta_b$ is a prefix of $\theta'_b$. Also define $y = tgt(\theta_b)$, $y' = tgt(\theta'_b)$, and let $\rho'_b = (x_b^0, \ldots, x_b^n, u_b^n, x_b^{n+1}) \in R_{n+1}(S_b)$ be such that $\theta'_b = \theta(\rho'_b)$. Note that, in particular, this implies $H_b(x_b^n) = y$ and $H_b(x_b^{n+1}) = y'$. Since $(x_b^n, u_b^n, x_b^{n+1}) \in \rightarrow_b$, we have from property (5) that $\exists (x_a^n, u_a^n, x_a^{n+1}) \in \rightarrow_a$ such that $(x_a^n, x_b^n) \in R$, $(u_a^n, u_b^n) \in C(H_b(x_b^n)) = C(y)$ and $(x_a^{n+1}, x_b^{n+1}) \in R$. From property (3), we have $H_a(x_a^n) = H_b(x_b^n) = y$ and $H_a(x_a^{n+1}) = H_b(x_b^{n+1}) = y'$. Now consider any $x'_b \in X_b$ such that $H(x'_b) = tgt(\theta'_b) = y'$. Using property (3) again, we have that $(x_a^{n+1}, x'_b) \in R$. From property (6) we therefore have that $\exists (x_b, u_b, x'_b) \in \rightarrow_b$ such that $(x_a^n, x_b) \in R$ and $(u_a^n, u_b) \in C(H_a(x_a^n)) = C(y)$. From property (3), we have that $H_b(x_b) = H_a(x_a^n) = y$ and from property (2) we have that $u_b = u_b^n$. From the induction hypothesis, there exists a run $\rho_b$ such that $\theta_b = \theta(\rho_b)$ and $tgt(\rho_b) = x_b$. Thus we can form the run $\rho''_b := \rho_b.u_b.x'_b$ satisfying $\theta'_b = \theta(\rho''_b)$ and $tgt(\rho''_b) = x'_b$, which completes the proof. □

In words, the above lemma implies that, when there exists an exact state reduction (2) for system $S_b$, a history $\theta_b$ gives no more information about the current state of $S_b$ than does the last output $tgt(\theta_b)$.

**Theorem 2** *Suppose that system $S_a$ is an exact state reduction (2) of system $S_b$ and that all other conditions of Thm. 1 are satisfied, except for the requirement that $(x_a, x_b) \in R \Rightarrow (x_a \in X_{m,a} \Leftrightarrow x_b \in X_{m,b})$. Then the obtained supervisor $\sigma_b$ will be safe, non-deadlocking, and maximally permissive among supervisors of the form $\sigma_b : \Theta(S_b) \to 2^{U_b}$.*

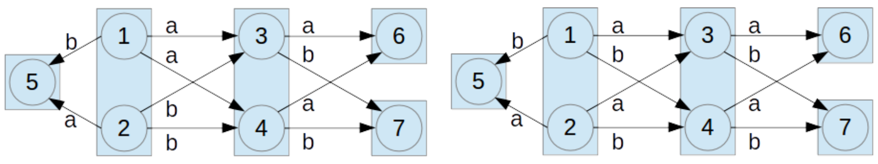
*Proof.* Lemma 1 shows that nothing is gained through a supervisor of the form $\sigma_b : \Theta(S_b) \to 2^{U_b}$ over a memoryless one. What remains to be proven is that the result of Thm. 1 holds true for the case of an exact state reduction (2), without the requirement that $(x_a, x_b) \in R \Rightarrow (x_a \in X_{m,a} \Leftrightarrow x_b \in X_{m,b})$. This requirement is used only in the proof of ($\Leftarrow$) in Claim 1. We therefore rewrite this part of the proof, using the exact state reduction (2).
($\Leftarrow$) Suppose that there exist $x_a \in X_a$, $u_a \in \sigma_a(H_a(x_a))$, and $x'_a \in \mathbf{Post}_{u_a}(x_a)$ such that $x'_a \notin X_{m,a}$ and $\sigma_a(H_a(x'_a)) = \emptyset$. Let $y = H_a(x_a)$ and $y' = H_a(x'_a)$. From Eq. (15) and $x'_a \notin X_{m,a}$, there must exist some $x'_b \in X_b$ such that $(x'_a, x'_b) \in R$ and $x'_b \notin X_{m,b}$. By property (6) of Def. 11, there exist $x_b \in X_b$ and $u_b \in U_b$ such that $(x_a, x_b) \in R$, $(u_a, u_b) \in C(y)$, and $x'_b \in \mathbf{Post}_{u_b}(x_b)$. Since $(u_a, u_b) \in C(y)$ and $u_a \in \sigma_a(y)$, we have that $u_b \in \sigma_b(y)$. By property (3) of Def. 9, $H_b(x_b) = H_a(x_a) = y$ and $H_b(x'_b) = H_a(x'_a) = y'$. By definition of $\sigma_b$, $\sigma_a(y') = \emptyset \Rightarrow \sigma_b(y') = \emptyset$, and we are done. □

*Remark 5* As in the case of (non-exact) state reductions, the obtained supervisor will not generally be maximally permissive if the supervisor can use initial state information. In particular, if the set of initial states $X_{0,b}$ gives more information than the initial output $y_0$, then there may exist more permissive supervisors. Note however that, if $H(x_{b,1}) = H(x_{b,2}) \Rightarrow [x_{b,1} \in X_{0,b} \Leftrightarrow x_{b,2} \in X_{0,b}]$, then $X_{0,b}$ gives no more information than the initial output $y_0$, and hence the resulting supervisor will still be maximally permissive. This is contrary to



(a) A State Reduction (b) An Exact State Reduction

**Fig. 4** A depiction of the state reduction (left) and exact state reduction (right) for a simple system $S_b = (\{1, \ldots, 8\}, \{u\}, \rightarrow_b, \{A, B\}, H_b)$, where $H_b(x) = A$ for $x \in \{1, \ldots, 4\}$ and $H_b(x) = B$ for $x \in \{5, \ldots, 8\}$. In both the left and right cases, there is a transition $(x, u, x') \in \rightarrow_b$ with $x \in H_b^{-1}(A)$ and $x' \in H_b^{-1}(B)$, and hence a transition from $A$ to $B$ in the corresponding state reduction. The system on the right contains some transition $(x, u, x') \in \rightarrow_b$ with $x \in H_b^{-1}(A)$, *for every* $x' \in H_b^{-1}(B)$. For the system on the left, the occurrence of a transition from $A$ to $B$ in the state reduction allows us to determine that $S_b$ is in state 7. For the system on the right, this transition only allows to determine that the system is some state in the set $H_b^{-1}(B)$.



(a) System $S_1$ (b) System $S_2$

**Fig. 5** Systems $S_1$ and $S_1$, demonstrating the difference between exact state reduction and alternating bisimulation.

the case of non-exact state reductions, in which case the above condition is still not sufficient to guarantee maximal permissiveness of the supervisor $\sigma_2$ obtained in Thm. 1, as is demonstrated in Ex. 2. □

Figure 4 depicts an example of a state reduction and an example of an exact state reduction.

## 5.3 Comparison of System Relations

Consider the three systems $S_1$, $S_2$, and $S_a$ of Figs. 5 and 6. The difference between $S_1$ and $S_2$ is the label on the transition between states 1 and 4 and on the transition between states 2 and 3. It can be verified that $S_a$ is an exact state reduction (2) of $S_1$, and that it alternatingly bisimulates $S_2$. Moreover, $S_a$ is not an exact state reduction of $S_2$, and does not alternatingly bisimulate $S_1$. Consider the safety specification $Safe_i = \rightarrow_i \setminus \{(3, a, 6)\}$. For system $S_1$, there

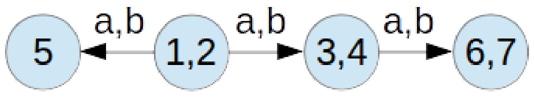


**Fig. 6** System $S_a$, which is both an exact state reduction (2) of system $S_1$ of Fig. 5 and alternatingly bisimulates system $S_2$ of Fig. 5. This system is not an exact state reduction (2) of $S_2$, and does not alternatingly bisimulate system $S_1$. Finally, $S_a$ is also bisimilar to both $S_1$ and $S_2$. It follows that the three relations do not coincide.

exists a unique maximally permissive memoryless supervisor $\sigma$ which achieves this specification, namely the supervisor which disables event $a$ on output $H(3) = H(4)$. As per Thm. 2, this supervisor is also maximally permissive among supervisors with memory. On the other hand, there does not exist a maximally permissive memoryless supervisor for system $S_2$. If event $a$ is enabled upon output $H(1) = H(2)$, then $a$ must be disabled upon output $H(3) = H(4)$. On the other hand, if $a$ is disabled upon output $H(1) = H(2)$, then state 3 becomes unreachable and $a$ can be enabled upon output $H(3) = H(4)$. This occurs because the abstraction is not "aligned" (normally referred to as proposition preserving) with the specification, in the sense that there exist states $x_{i,1}, x_{i,2}$ and control input $u_i$ of each system $S_i$, $i = 1, 2$, such that $H_i(x_{i,1}) = H_i(x_{i,2})$, but $\{x_{i,1}\} \times \{u_i\} \times \mathbf{Post}_{u_i}(x_{i,1}) \subseteq Safe_i \not\Leftrightarrow \{x_{i,2}\} \times \{u_i\} \times \mathbf{Post}_{u_i}(x_{i,2}) \subseteq Safe_i$.

This example demonstrates a key point in abstraction based synthesis. To obtain maximally permissive supervisors with respect to the abstraction, it is typically required that the abstraction be aligned not only with the dynamics of the system to be abstracted, but also with the specification. In particular, this means that a change of specification requires reconstructing the abstraction if one wishes to maintain maximal permissiveness. This is not the case with exact state reductions, since *exact state reductions produce maximally permissive solutions without requiring that abstractions be aligned with specifications.* Note, however, that abstraction techniques that produce maximally permissive supervisors when the abstractions are aligned with the specifications produce solutions that are maximally permissive with respect to the original system, not merely with respect to the chosen discretization.

## 6 Discrete Abstraction

Returning to the vehicle control problem of Sec. 2, we construct a DES $G$ that models the behavior of the continuous time system, using the lattice $\tilde{Q}$ as the set of discrete states.

To construct a DES abstraction of the continuous-time system, we use a three-layered transition function $\psi$. The first layer consists of events in the set $U_c$, for the actions of the controlled vehicles. The second layer consists of events in the set $U_{uc}$, for the actions of the uncontrolled vehicles. It remains to model the disturbance $d$. We achieve this by discretizing the set $\Delta$ to obtain a set of "discretized disturbances" $W$. Specifically, let

$$W = \{k\tau\mu : k \in \mathbb{Z} \wedge \lfloor \delta_{min}/(\tau\mu) \rfloor \leq k \leq \lceil \delta_{max}/(\tau\mu) \rceil\}^n. \tag{18}$$

This set $W$ makes up the third layer of $G$'s transition structure. For any $q \in \tilde{Q}$, $u_c \in U_c$, $u_{uc} \in U_{uc}$, and $w \in W$, we define

$$\psi(q, u_c u_{uc} w) := q + u + w, \tag{19}$$

where $u = (u_c, u_{uc})$. In Sec. 6.1, we will show that $\psi(q, u_c u_{uc} w) = q'$ if and only if there exist $x \in X$, $\delta \in \Delta$, and $x' \in X$ such that $x' = x + u + \delta$, $q = \ell(x)$, and $q' = \ell(x')$ (see Prop. 1). To define the discrete system state in between the occurrence of events in $U_c$ and $U_{uc}$ and in between the occurrence of events in $U_{uc}$ and $W$ (all of which occur simultaneously in the continuous-time system), we introduce two sets of "intermediate" states $Q_{I1}$ and $Q_{I2}$ (disjoint from each other and from $\tilde{Q}$ and with no physical meaning), and three intermediate transition functions: $\psi_1 : \tilde{Q} \times U_c \to Q_{I1}$, $\psi_2 : Q_{I1} \times U_{uc} \to Q_{I2}$, and $\psi_3 : Q_{I2} \times W \to \tilde{Q}$, defined only by $\psi(q, u_c, u_{uc}, w) = \psi_3(\psi_2(\psi_1(q, u_c), u_{uc}), w)$. See Fig. 7 for a depiction of the transition function $\psi$. We take the set of marked states to be the set $Q_m = \{q_m\}$. Finally, we define a set $Q_0$ of possible initial states, which we model by introducing a dummy initial state $q_0$ and having transitions from $q_0$ to each state $q \in Q_0$ with event label $e_q$. We denote this set of events by $E_Q := \{e_q : q \in Q_0\}$ and define $\psi(q_0, e_q) := q$. The final DES is defined as:

$$G := (Q, E_Q \cup U_c \cup U_{uc} \cup W, \psi, q_0, Q_m) \tag{20}$$

where $Q = \{q_0\} \cup \tilde{Q} \cup Q_{I1} \cup Q_{I2}$. The sets of events $U_c$ is taken to be controllable, whereas the sets of events $U_{uc}$ and $W$ are taken to be uncontrollable. Note that, in the context of supervisory control problems of DES, a supervisor is obtained which does not choose a particular event from any given state, but rather chooses which events to enable (allow) and which ones to disable (prevent). An uncontrollable event is an event that cannot be disabled.

*Remark 6* Although the initial state can not be chosen by the system, we take the set of events $E_Q$ to also be controllable. In the following section, we will use $G$ in a supervisory control problem. If $E_Q$ were defined as uncontrollable, we would obtain an empty solution to the supervisory control problem whenever there was *any* initial state from which there was no solution, even if there existed solutions from some of them. By defining the set $E_Q$ as controllable, the computed supervisor will contain a transition from $q_0$ to $q$ for every $q \in Q_0$ from which there exists a solution to the supervisory control problem. □

### 6.1 Relations Between the Time-discretized and Discrete Event Systems

**Proposition 1** *Define the observation maps $H_{\tilde{Q}}(q) := q$, $H_X(x) := \ell(x)$, the relation $R := \{(q, x) \in \tilde{Q} \times X : \ell(x) = q\}$, and the control relation*

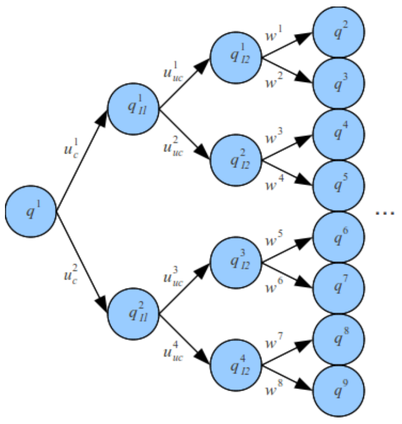


**Fig. 7** The transition function $\psi$.

*$C(q) := \{(u_c, v_c) : v_c\tau = u_c \in U_c\}$, for all $q \in \tilde{Q}$. Then DES $G$ of Sec. 6 is a state reduction of system* (2)*.*

*Proof.* Properties (1), (2), and (3) follow immediately from the definitions of $H_X$, $H_Q$, $\ell$, $R$, and $C$.
Property (4): Consider any $q \in \tilde{Q}$, $u_c \in U_c$, $u_{uc} \in U_{uc}$, and $w \in W$, with $q' = \psi(q, u_c, u_{uc}, w) = q + u + w$ (where $u = (u_c, u_{uc})$). We construct $x \in X$, $x' \in X$ and $\delta \in \Delta$ such that $\ell(x) = q$, $\ell(x') = q'$, and $x + u + \delta = x'$ by considering each co-ordinate in turn. There are three cases, depending on where $w_i$ lies with respect to the interval $[\delta_{\text{min}}, \delta_{\text{max}}]$ (recall from Eq. (18) that $w_i$ may be smaller than $\delta_{\text{min}}$ or larger than $\delta_{\text{max}}$ when these values are not integer multiples of $\mu\tau$, because of the floor and ceiling operations).

**Case 1:** $\delta_{min} \le w_i \le \delta_{max}$. Take $x_i = q_i$, $\delta_i = w_i$, and $x'_i = q'_i$.
**Case 2:** $w_i > \delta_{max}$. Take $x_i = q_i + \mu\tau/2$, $\delta_i = \delta_{max}$, and $x'_i = x_i + u_i + \delta_{max}$. From the definition of $\ell$, we have that $\ell_i(x_i) = q_i$. With these values, we obtain $q'_i - x'_i = (q_i + u_i + w_i) - (x_i + u_i + \delta_{max}) = w_i - \delta_{max} - \mu\tau/2$. From the definition of $W$, we know that $\delta_{max} < w_i < \delta_{max} + \mu\tau$, or equivalently that $0 < w_i - \delta_{max} < \mu\tau$. From this and the previous statement, we obtain $-\mu\tau/2 < q'_i - x'_i < \mu\tau/2$, from which it follows that $\ell(x'_i) = q'_i$.
**Case 3:** $w_i < \delta_{min}$. Take $x'_i = q'_i + \mu\tau/2$, $\delta_i = \delta_{min}$, and $x_i = x'_i - u_i - \delta_{min}$. The same reasoning as in the previous case shows that $\ell(x) = q$ and that $\ell(x') = q'$.

Property (5): Consider any $x \in X$, $u_c \in U_c$, $u_{uc} \in U_{uc}$, and $\delta \in \Delta$, with $x' = x+u+\delta$ (where $u = (u_c, u_{uc})$). Take $q = \ell(x)$, $q' = \ell(x')$, and $w = q'-q-u$. It suffices to show that $w \in W$. From $q = \ell(x)$ and $q' = \ell(x')$, we have $-\mu\tau/2 < x - q \le \mu\tau/2$ and $-\mu\tau/2 < x' - q' \le \mu\tau/2$ (component-wise). Combining these inequalities with $w = q' - q - u$ and $\delta = x' - x - u$, we obtain $w = \delta + (x - q) - (x' - q')$ and hence:

$$-\tau\mu + \delta < w < \delta + \tau\mu.$$

It follows that $w$ is a vector whose components are all integer multiples of $\tau\mu$ and in the interval $(\delta_{min} - \mu\tau, \delta_{max} + \mu\tau)$. But from Eq. 18, this set of vectors is precisely equal to $W$, proving that $w \in W$. $\square$

**Proposition 2** *Define $H_X(\cdot)$, $H_{\tilde{Q}}(\cdot)$, $R$, and $C$ as in Prop. 1. If $\delta_{min}$ and $\delta_{max}$ are both integer multiples of $\tau\mu$, then DES $G$ of Sec. 6 is an exact state reduction (2) of system* (2).

*Proof.* Property (6): Consider any $q \in \tilde{Q}$, $u_c \in U_c$, $u_{uc} \in U_{uc}$, $w \in W$, and $x'$ such that $q' = q + u + w = \ell(x')$, where $u = (u_c, u_{uc})$. We construct $x \in X$ and $\delta \in \Delta$ such that $q = \ell(x)$ and $x' = x + u + \delta$. Simply take $\delta = w$ and $x = x' - u - \delta$. As remarked in the proof of Prop. 1, $w$ must be a vector whose components are integer multiples of $\tau\mu$ and in the interval $(\delta_{min} - \mu\tau, \delta_{max} + \mu\tau)$. If $\delta_{min}$ and $\delta_{max}$ are multiples of $\tau\mu$, then it follows that the components of $w$ are in the (closed) interval $[\delta_{min}, \delta_{max}]$. Thus $\delta \in \Delta$. Furthermore, $x' - x = q' - q = u + w$, so that $x - q = x' - q'$, from which it follows that $q' = \ell(x') \Rightarrow q = \ell(x)$. $\square$

Given the above results, we can solve Prob. 1. Before presenting the relevant theorems, we define the notations $A_{x,v_c,x'}(t)$ and $A_{q,u_c,q'}(t)$ as follows:

$$A_{x,v_c,x'}(t) = \left\{ x'' \in X \;\middle|\; \begin{array}{c} \exists v_{uc} \in V_{uc}, \exists d \in D^{[0,\tau]} : \\ x + v\tau + \int_0^\tau d(s)ds = x' \\ \wedge x'' = x + vt + \int_0^t d(s)ds \end{array} \right\}, \tag{21}$$

$$A_{q,u_c,q'}(t) = \left\{ x'' \in X \;\middle|\; \begin{array}{l} \exists x \in \ell^{-1}(q), \exists v_{uc} \in V_{uc}, \exists d \in D^{[0,\tau]} : \\ \quad x + v\tau + \int_0^\tau d(s)ds \in \ell^{-1}(q') \\ \quad \wedge x'' = x + vt + \int_0^t d(s)ds \end{array} \right\}, \tag{22}$$

where $v = (v_c, v_{uc}) = (u_c/\tau, v_{uc})$. In words, $A_{x,v_c,x'}(t)$ is the set of possible vehicle positions at time $t$ when they are at $x$ at time 0, at $x'$ at time $\tau$, and control input $v_c$ is chosen. Similarly, $A_{q,u_c,q'}(t)$ is the set of possible vehicle positions at time $t$ when they are at some $x \in \ell^{-1}(q)$ at time 0, at some $x' \in \ell^{-1}(q')$ at time $\tau$, and control input $u_c$ is chosen.

**Theorem 3** *Define the automaton $H := (Q, E_Q \cup U_c \cup U_{uc} \cup W, \psi_{safe}, q_0, Q_m) \sqsubseteq G$, where $\psi_{safe} \subseteq \psi$ is defined by:*

$$(q, u_c, q') \in \psi_{safe} \Leftrightarrow \nexists t \in [0, \tau] : A_{q,u_c,q'}(t) \cap B \neq \emptyset. \tag{23}$$

*Solve for the supremal controllable sublanguage $(\mathcal{L}_m(H))^{\uparrow C}$ of $\mathcal{L}_m(H)$ with respect to $\mathcal{L}(G)$ and uncontrollable event set $E_{uc} = U_{uc} \cup W$, obtaining a maximally permissive safe and non-deadlocking supervisor $S : \tilde{Q} \to 2^{U_c}$. Then the supervisor $\sigma : X/\ell \to 2^{V_c}$ defined by $v_c \in \sigma(x) \Leftrightarrow u_c = \tau v_c \in S(\ell(x))$ solves Prob. 1.*

*Proof.* Solving Prob. 1 requires finding the maximally permissive safe and non-deadlocking supervisor $\sigma$ for System $S_b = (X, V_c, \to_b, \tilde{Q}, \ell)$ subject to safety specification $Safe_b$ and marking $X_{m,b}$, where:

$$(x, v_c, x') \in \to_b \Leftrightarrow (\exists v_{uc} \in V_{uc})(\exists \delta \in \Delta) : x + \tau v + \delta = x', \quad v = (v_c, v_{uc}), \tag{24}$$

$$(x, v_c, x') \in Safe_b \Leftrightarrow \nexists t \in [0, \tau] : A_{x,v_c,x'}(t) \cap B \neq \emptyset, \tag{25}$$

and $X_{m,b} = \ell^{-1}(q_m)$. Thus, it suffices to apply Thm. 1, and we proceed to verify its conditions. Proposition 1 shows that $G$ is a state reduction of $S_b$, with the state and control relations $R = \{(q, x) \in \tilde{Q} \times X : q = \ell(x)\}$ and $C(q) = \{(u_c, v_c) \in U_c \times V_c : u_c = v_c\tau\}$. Comparing Eqs. (21) and (22), we see that the safety specification $Safe_a$ defined by equation (23) does indeed satisfy the condition $(q, u_c, q') \in Safe_a$ if and only if, for all $(x, v_c, x') \in \rightarrow_b$ such that $(q, x) \in R$, $(u_c, v_c) \in C(q)$ and $(q', x') \in R$, we have that $(x, v_c, x') \in Safe_b$. Finally, the set $Q_m$ of marked states for $G$ and $H$ obviously satisfies the condition $q \in Q_m$ if and only if $x \in X_{m,b}$ for all $x \in X_b$ such that $(q, x) \in R$, since $Q_m = \{q_m\}$, $X_{m,b} = \ell^{-1}(q_m)$, and $(q, x) \in R \Leftrightarrow q = \ell(x)$. Thus, $G$ is a state reduction of $S_b$, and $Safe_a$ and $X_{m,a} = Q_m$ are induced specifications, satisfying the conditions of Thm. 1. □

**Theorem 4** *If $\delta_{min}$ and $\delta_{max}$ are both integer multiples of $\tau\mu$, then the supervisor $\sigma$ of Thm. 3 solves Prob. 1, and is maximally permissive among the class of all supervisors, not merely memoryless ones.*

*Proof.* Immediate from Prop. 2, Thm. 2, and the proof of Thm. 3. □

## 7 Iterative Refinement

In this section, we describe a procedure for iterative refinement of the discrete state space of the abstraction. At a high level, the iterative refinement procedure consists of constructing an abstraction with a coarse discretization, categorizing each state of the abstraction as either winning, losing, or undetermined, refining the abstraction with a finer discretization at the undetermined states, and repeating. We present this method in the context of the vehicle control problem, but the method can be extended to other problems with little modification. The method is similar to those of [21, 24].

The categorization of states into winning, losing, and unknown requires considering the control problem of Prob. 1 as a game against nature. The control problem for nature is to cause the vehicles to enter the bad set $B$, before they have all crossed the intersection. Thus, the set of "good" states for nature are the bad states of the controller and vice-versa. Additionally, the control properties of the various inputs are also reversed. That is, nature chooses the inputs for the uncontrolled vehicles and the disturbance, but does not choose the inputs of the controlled vehicles.

In what follows, let $G^C_\tau$ and $H^C_\tau$ respectively denote the DES abstraction $G$ of Sec. 6 and the DES abstraction $H$ defined in the statement of Thm. 3, parametrized by $\tau$. Similarly, let $\ell_\tau$, $\tilde{Q}_\tau$, $Q_\tau$, $E_{Q_\tau}$, $\psi_\tau$, $q_{\tau,0}$, $Q_{\tau,m}$, and $\psi_{\tau,safe}$ be $\tau$ parametrized versions of the relevant functions, events, relations, states, or sets. We define DES $G^N_\tau$ and $H^N_\tau$, which are the relevant automata of the control problem for nature that consists of forcing vehicles into the bad set. Because only the specification changes, and not the dynamics, $G^N_\tau$ is defined almost identically to $G_\tau^C$. Given $G_\tau^C = (Q_\tau, E_{Q_\tau} \cup U_c \cup U_{uc} \cup W, \psi_\tau, q_{\tau,0}, Q_{\tau,m})$ as in Eq. (20), define

$$G_\tau^N = (Q_\tau, E_{Q_\tau} \cup U_c \cup U_{uc} \cup W, \psi_\tau, q_{\tau,0}, Q_{\tau,m}^N), \tag{26}$$

with $Q_{\tau,m}^N$ given by

$$Q_{\tau,m}^N = \{q \in \tilde{Q} : \ell^{-1}(q) \subseteq B\}. \tag{27}$$

Thus, a discrete state is marked for nature only if it is entirely contained in the bad set. A discrete state that only partially intersects the bad set may be split into some combination of states that are winning for nature and states that are winning for the controller. Similarly,

$$H_\tau^N = (Q_\tau, E_{Q_\tau} \cup U_c \cup U_{uc} \cup W, \psi_{\tau,safe}^N, q_{\tau,0}, Q_{\tau,m}^N), \tag{28}$$

where $\psi_{\tau,safe}^N$ is defined by

$$(q, u_c, q') \in \psi_{\tau,safe}^N \Leftrightarrow \begin{pmatrix} \exists v_{uc} \in V_{uc}, \exists d \in D^{[0,\tau]}, \forall x \in \ell^{-1}(q) \\ \text{s.t. } x' = x + v\tau + \int_0^\tau d(t)dt \in \ell^{-1}(q'), \\ x(t) = x + vt + \int_0^t d(t')dt' \notin \ell^{-1}(Q_{\tau,m}),\ \forall t \in [0,\tau] \end{pmatrix}, \tag{29}$$

where $v = (u_c/\tau, v_{uc})$. As noted above, $U_c$ is taken to be uncontrollable whereas $U_{uc}$ and $W$ are taken to be controllable in $G_\tau^N$ and $H_\tau^N$. Note that, in Eq. (29), the set of safe transitions are still parametrized by $u_c$, which is not controlled by nature, and not on $u_{uc}$ and $w$. This is because the DES model allows for nature to choose $u_{uc}$ and $W$ *in response* to $u_c$. Thus, a state is winning for the controller if there exists a control input $u_c$ that is safe for all possible $u_{uc}$ and $w$ chosen by nature. On the other hand, a state is winning for nature if, for all possible $u_c$ chosen by the controller, there exists a safe choice of $u_{uc}$ and $w$.

Given a particular time discretization $\tau$, the set of winning states for the controller are obtained through the fixed point of Eqs. (12) and (11). The set of winning states for nature can be characterized analogously. Given transition system $S = (X, U, \to)$ and safety and marking specifications $Safe^N$ and $X_m^N$, define

$$Cont^N(x|Z) = \{u \in U | \exists x' \in \mathbf{Post}_u(x), [(x,u,x') \in Safe^N \wedge x' \in Z]\}, \tag{30}$$

and let $F^N : 2^X \to 2^X$ be defined by

$$F^N(Z) = \{x \in Z | x \in X_m^N \vee Cont^N(x|Z) = U\}. \tag{31}$$

Consistent with the discussion above regarding the reversal of universal and existential quantifiers, Eq. (30) is identical to Eq. (11), except that $\forall$ becomes $\exists$, and Eq. (31) is identical to Eq. (12), except that $Cont(x|Z) \neq \emptyset$ becomes $Cont^N(x|Z) = U$.

Thus, for a given time discretization $\tau$, it is possible to categorize the discrete states as winning (for the controller), losing (i.e., winning for nature), and undetermined (i.e., losing for both). Refinement is performed by refining the discretization at the undetermined states. This is done by taking $\tau' = \tau/2$, which results in a lattice of discrete states $\tilde{Q}'$ with space discretization of $\mu\tau' = \mu\tau/2$. It can be shown that $G_\tau$ will be a state reduction of $G_{\tau/2}$ (with $C$ being the identity map and $R$ defined in the obvious way), or an exact state reduction if $d_{\min}$ and $d_{\max}$ are integer multiples of $\mu$. The proof is very similar to that of Props. 1 and 2, and is therefore omitted.

Importantly, the fact that $\tau$ is diminished by half at each iteration means that all control inputs of the refined supervisor $\sigma$ for the continuous time system are still feasible for $\tau' = \tau/2$. That is, a control input of $v_c$ that is held for time $\tau$ is identical to two consecutive control inputs $v_c$, each of which is held for time $\tau/2$. Thus, if $\ell_\tau(x)$ is winning for a time discretization of $\tau$, then $\ell_{\tau/2}(x)$ is winning for a time discretization of $\tau/2$.

The process of refinement is continued up until some desired stopping condition has been reached (e.g., a minimal value of $\tau$), at which point the set of allowed control inputs is determined by Eq. (13) and the set of states that are winning for the controller. Any indeterminate states remaining at this point must be treated as losing states by the controller.

*Remark 7* For non-deadlocking specifications, it is possible in general to have discrete states that are winning for both the controller and nature (e.g., in the case of a livelock that does not violate either safety specification). In the vehicle control problem under consideration, however, the fact that vehicles have strictly positive velocity implies that the vehicles will eventually cross the intersection if they do not collide first. Thus, either the controller or nature will eventually lose. $\square$

## 8 Algorithmic Implementation

In past work, [13], we provided an algorithm for computing the DES supervisor $S$ of Thm. 3 that is based on a depth-first search (DFS) and has a lower asymptotic complexity than the standard algorithm. This customized algorithm was based on the following three observations: the vehicle's velocities are bounded by $\mu > 0$; the specification automaton $H$ is a sub-automaton of $G$; and each pair of events $u_{uc}w \in U_{uc}W$ is feasible after each event $U_c$ from each state $q \in \tilde{Q}$. The first observation implies that the system is acyclic, and hence livelock-free. This allows for solving problem BSCP-NB in time linear in the size of $G \times H$, rather than quadratic (see, e.g. [17]). The second observation implies that the product automaton $H \times G$ is isomorphic to $H$ which, combined with the first observation, allows for the problem to be solved through a DFS on $G$. Finally, the third observation implies that there is no need to determine the safety of each string $u_c u_{uc} w \in U_c U_{uc} W$ from each state $q$. Instead, a single test of safety for each $u_c \in U_c$ and state $q \in \tilde{Q}$ suffices. The algorithm's running time was shown to be $O\left(|\tilde{Q}||U_c|\left[|\mathbf{Post}_{u_c}(q)| + n^2\right]\right)$.

*Remark 8* In fact, the DES supervisor $S$ of Thm. 3 could be computed in time linear in the size of $G \times H$, even if the system were not acyclic. This can be seen from Eqs. (11) and (12), which are very similar to the well known controllable predecessor operator whose iteration reaches a fixed point in linear time for safety properties. The formulation of the algorithm for computing *S as a DFS* is, however, reliant on the system being acyclic. □

In this work, we adapt the algorithm of [13] to the problem of computing the set of discrete states that are winning for the controller, using the iterative refinement technique described in Sec. 7.

In what follows, let $\ell_\tau$ be the discretization function of Eq. (3), parametrized by $\tau$, and let $\tilde{Q}_\tau$ be the resulting set of discrete states. As per the discussion of Sec. 7, we can model the control problem to be solved as a game against nature. To that end, let $\phi_\tau : \tilde{Q}_\tau \cup (\tilde{Q}_\tau \times U_c) \cup (\tilde{Q}_\tau \times U_c \times \tilde{Q}_\tau) \to \{-1, 0, 1\}$ denote the value (to the controller) of states, control inputs, or transitions. More specifically, $\phi_\tau(q)$, $\phi_\tau(q, u_c)$, and $\phi_\tau(q, u_c, q')$ each take a value of -1, 1, or 0 to denote a victory for nature, a victory for the controller, or a loss for both. These are defined through the following equations:

$$\phi_\tau(q, u_c, q') = \begin{cases} 1, & \text{if } (q, u_c, q') \in \psi_{\tau,safe} \wedge \phi_\tau(q') = 1 \\ -1, & \text{if } (q, u_c, q') \in \psi^N_{\tau,safe} \wedge \phi_\tau(q') = -1 \\ 0, & \text{else} \end{cases} \tag{32}$$

$$\phi_\tau(q, u_c) = \min_{q' \in \mathbf{Post}_{u_c}(q)} \phi_\tau(q, u_c, q') \tag{33}$$

$$\phi_\tau(q) = \begin{cases} 1, & \text{if } q \in Q_{\tau,m} \\ -1, & \text{if } q \in Q^N_{\tau,m} \\ \max_{u_c \in U_c} \phi_\tau(q, u_c), & \text{else} \end{cases} \tag{34}$$

**Theorem 5** *In Eqs. (11)-(12), take $X$, $U$, $\to$, $Safe$, and $X_m$ to be $\tilde{Q}_\tau$, $U_c$, $\psi_\tau$, $\psi_{\tau,safe}$, and $Q_{\tau,m}$. Similarly, in Eqs. (30)-(31), take $X$, $U$, $\to$, $Safe$, and $X_m$ to be $\tilde{Q}_\tau$, $U_c$, $\psi^N_\tau$, $\psi^N_{\tau,safe}$, and $Q^N_{\tau,m}$. Then there exists a unique solution to Eqs. (32)-(34), and this solution satisfies $\phi_\tau(q) = 1 \Leftrightarrow q \in \nu Z.F(Z)$, $\phi_\tau(q) = -1 \Leftrightarrow q \in \nu Z.F^N(Z)$, $\phi_\tau(q, u_c) = 1 \Leftrightarrow u_c \in Cont(q|\nu Z.F(Z))$, and $\phi_\tau(q, u_c) = -1 \Leftrightarrow u_c \in Cont^N(q|\nu Z.F^N(Z))$, for all $q \in \tilde{Q}_\tau$ and $u_c \in U_c$.*

*Proof.* We proceed in two claims. The first claim shows that there is a solution for $\phi_\tau$ satisfying the required constraints. The second claim shows that $\phi_\tau$ has a unique solution.

**Claim 1:** Suppose that some sets $M \subseteq \tilde{Q}_\tau$ and $N \subseteq \tilde{Q}_\tau$ are fixed points of Eq. (12) and Eq. (31), respectively. Then

$$\phi_\tau(q) = \begin{cases} 1, & \text{if } q \in M \\ -1, & \text{if } q \in N \\ 0, & \text{else} \end{cases} \tag{35}$$

$$\phi_\tau(q, u_c) = \begin{cases} 1, & \text{if } u_c \in Cont(q|M) \\ -1, & \text{if } u_c \in Cont^N(q|N) \\ 0, & \text{else} \end{cases} \tag{36}$$

$$\phi_\tau(q, u_c, q') = \begin{cases} 1, & \text{if } (q, u_c, q') \in \psi_{\tau,safe} \wedge q' \in M \\ -1, & \text{if } (q, u_c, q') \in \psi^N_{\tau,safe} \wedge q' \in N \\ 0, & \text{else} \end{cases} \tag{37}$$

constitutes a solution to Eqs. (32)-(34).
Clearly, Eqs. (35) and (37) imply Eq. (32). Recall from Eq. (11) that

$$\begin{aligned}Cont(q|M) &= \{u_c \in U_c | \forall q' \in \mathbf{Post}_{u_c}(q), [(q, u_c, q') \in \psi_{\tau,safe} \wedge q' \in M]\} \\ &= \{u_c \in U_c | \forall q' \in \mathbf{Post}_{u_c}(q), \phi_\tau(q, u_c, q') = 1\},\end{aligned}$$

from which it follows that $\phi_\tau(q, u_c) = 1$ if and only if $\min_{q' \in \mathbf{Post}_{u_c}(q)} \phi_\tau(q, u_c, q') = 1$, as required by Eq. (33). It can similarly be shown from Eq. (30) defining $Cont^N(q|N)$ that $\phi_\tau(q, u_c) = -1$ if and only if $\min_{q' \in \mathbf{Post}_{u_c}(q)} \phi_\tau(q, u_c, q') = -1$. Thus, Eq. (33) is satisfied. Now, since $M$ is a fixed point of Eq. (12), we have that

$$\begin{aligned}M &= \{q \in \tilde{Q}_\tau | q \in \tilde{Q}_{\tau,m} \vee Cont(q|M) \neq \emptyset\} \\ &= \{q \in \tilde{Q}_\tau | q \in \tilde{Q}_{\tau,m} \vee \exists u_c \in U_c : \phi_\tau(q, u_c) = 1\}\end{aligned}$$

It follows that $\phi_\tau(q) = 1$ if and only if either $q \in \tilde{Q}_{\tau,m}$ or $\max_{u_c \in U_c} \phi_\tau(q, u_c) = 1$ holds. It can similarly be shown from the fact that $N$ is a fixed point of Eq. (31) that $\phi_\tau(q) = -1$ if and only if either $q \in \tilde{Q}^N_{\tau,m}$ or $\max_{u_c \in U_c} \phi_\tau(q, u_c) = -1$.
**Claim 2:** There is a unique solution to Eqs. (32)-(34).
The set $\tilde{Q}_\tau$ is finite, since it is the discretization of a compact space. Furthermore, the requirement that vehicles have a positive velocity implies that there can be no cycles of states, and the vehicles must eventually cross the intersection (if there is no collision first). Thus, there are no cyclical dependencies in Eqs. (32)-(34), and one may solve them by backwards induction starting from $\tilde{Q}_{\tau,m} \cup \tilde{Q}^N_{\tau,m}$. □

Algorithm 2 below is based on Eqs. (32)-(34), but with the following optimizations. For each one, we provide a description and applicable line numbers for Alg. 2. Note that some lines have multiple optimizations used simultaneously.

1. The algorithm uses parameters $\tau_{\max}$ and $\tau_{\min}$, where it is assumed that $\tau_{\max} = 2^r \tau_{\min}$, for some non-negative integer $r$. When working at any discretization level $\tau > \tau_{\min}$, the algorithm correctly classifies states as winning for the controller, winning for nature, or losing for both. When $\tau = \tau_{\min}$, the algorithm does not differentiate between states which are winning for nature and states which are losing for both nature and the controller, terminating at state $q \in \tilde{Q}_{\tau_{\min}}$ as soon as it is determined that $\phi_{\tau_{\min}}(q) \leq 0$. Lines 7, 15-18, 32.
2. The algorithm uses a version of $\alpha - \beta$ pruning, a technique for accelerating computation of winning strategies in min-max games by not exploring game subtrees that can be determined to not have any bearing on the value of a state. Thus, once it has been determined that $\phi_\tau(q, u_c) \leq \phi_\tau(q)$, then the algorithm ceases to examine any other successors $q' \in \mathbf{Post}_{u_c}(q)$, as these can only result in a smaller value of $\phi_\tau(q, u_c)$. Line 32.

3. Other than verifying if states are contained in the winning sets for the controller and nature, we also verify if states have a non-empty intersection with these sets. This allows for the determination that a state is losing for the controller and/or losing for nature, restricting the possible values for $\phi_\tau(q)$. When used with $\alpha-\beta$ pruning, this can potentially diminish the number of control inputs $u_c$ or successor states $q'$ that must be examined. Lines 11-22.
4. From Eqs. (32) and (33), we have that $\phi_\tau(q,u_c)=1$ requires that $(q,u_c,q')\in\psi_{\tau,safe}$, for each $q'\in\mathbf{Post}_{u_c}(q)$. Rather than checking if $(q,u_c,q')\in\psi_{\tau,safe}$, for each $q'\in\mathbf{Post}_{u_c}(q)$, we aggregate all these tests into a single test on $q$ and $u_c$. By Eq. (23), $(q,u_c,q')\in\psi_{\tau,safe}$ if and only if $\nexists t\in[0,\tau]:A_{q,u_c,q'}(t)\cap B\neq\emptyset$. Thus we define $A_{q,u_c}(t)=\cup_{q'\in\mathbf{Post}_{u_c}(q)}A_{q,u_c,q'}(t)$ and instead verify if $\nexists t\in[0,\tau]:A_{q,u_c}(t)\cap B\neq\emptyset$. Parametrized by $\tau$, the set $A_{\tau,q,u_c}(t)$ is given by

$$A_{\tau,q,u_c}(t)=(q-\mathbf{1}\mu\tau/2+\underline{v}_{u_c}t,q+\mathbf{1}\mu\tau/2+\overline{v}_{u_c}t], \tag{38}$$

where $\mathbf{1}$ denotes the $n=|\mathcal{N}|$ dimensional vector $(1,\dots,1)$,

$$\underline{v}_{u_c,i}=\begin{cases}u_{c,i}/\tau+d_{min}, & \text{vehicle } i \text{ is controlled}\\ v_{min}+d_{min}, & \text{vehicle } i \text{ is uncontrolled}\end{cases} \tag{39}$$

$$\overline{v}_{u_c,i}=\begin{cases}u_{c,i}/\tau+d_{max}, & \text{vehicle } i \text{ is controlled}\\ v_{max}+d_{max}, & \text{vehicle } i \text{ is uncontrolled}\end{cases} \tag{40}$$

and, for any $a,b\in\mathbb{R}^n$, $(a,b]:=\{c\in\mathbb{R}^n|a_i<c_i\leq b_i,i=1,\dots,n\}$. Equations for verifying the condition $\nexists t\in[0,\tau]:A_{\tau,q,u_c}(t)\cap B\neq\emptyset$ are given in the Appendix. The general idea is to check intersection with the bad set for each pair of vehicles, so that the test takes $O(n^2)$ time. Lines 28-30.
5. Recall that $\phi_\tau(q')=-1$ if it is possible for nature to force the vehicles to enter the bad set (for any strategy of the controller), and that $(q,u_c,q')\in\psi^N_{\tau,safe}$ if there is at least one vehicle that does not cross the intersection for this transition. Because vehicles always move at strictly positive velocity, it is not possible for all the vehicles to cross the intersection and then enter the bad set. Thus $\phi_\tau(q')=-1\Rightarrow(q,u_c,q')\in\psi^N_{\tau,safe}$ in Eq. (32). It follows that it is possible to conclude that $\phi_\tau(q,u_c,q')=-1$ if $\phi_\tau(q')=-1$, without additionally verifying if $(q,u_c,q')\in\psi^N_{\tau,safe}$. Line 35.

Our algorithm uses a number of subroutines, which we explain below.

- ContVic$(q,\tau)$ returns true if $\ell_\tau^{-1}(q)$ consists entirely of states where all vehicles have crossed the intersection (i.e., it checks if $q\in Q_{\tau,m}$).
- NatVic$(q,\tau)$ returns true if $\ell_\tau^{-1}(q)\subseteq B$ (i.e., it checks if $q\in Q^N_{\tau,m}$).
- ContLoss$(q,\tau)$ returns true if $\ell_\tau^{-1}(q)\cap B\neq\emptyset$.
- NatLoss$(q,\tau)$ returns true if $\ell_\tau^{-1}(q)$ contains any states where all vehicles have crossed the intersection.
- EnqueueRefined$(q,\tau,queue)$ is called when $\phi_\tau(q)=0$ and $\tau>\tau_{\min}$, in which case the set of refined states $\{q'\in\tilde{Q}_{\tau/2}|\ell^{-1}_{\tau/2}(q')\subseteq\ell_\tau^{-1}(q)\}$ are added to *queue*.

**Algorithm 1** Initialization

```
1: procedure DoInit($\tau_{\min}$, $\tau_{\max}$)
2:     $queue \leftarrow \tilde{Q}_{\tau_{\max}} \times \{\tau_{\max}\}$
3:     while $queue \neq \emptyset$ do
4:         $(q, \tau) \leftarrow$ Dequeue(queue)
5:         DoDFS$(q, \tau, \tau_{\min}, queue)$
6:     end while
7: end procedure
```

**Algorithm 2** DFS Computation

```
1: procedure DoDFS($q$, $\tau$, $\tau_{\min}$, $queue$)
2:     if !$\phi_\tau(q)$ then
3:         return $\phi_\tau(q)$
4:     else if ContVic$(q, \tau)$ then
5:         $\phi_\tau(q) \leftarrow 1$
6:         return 1
7:     else if $\tau > \tau_{\min} \wedge$ NatVic$(q, \tau)$ then
8:         $\phi_\tau(q) \leftarrow -1$
9:         return -1
10:     end if
11:     $phimin(q) \leftarrow -1$
12:     $phimax(q) \leftarrow 1$
13:     if ContLoss$(q, \tau)$ then
14:         $phimax(q) \leftarrow 0$
15:         if $\tau = \tau_{\min}$ then
16:             $\phi_\tau(q) \leftarrow 0$
17:             return 0
18:         end if
19:     end if
20:     if NatLoss$(q, \tau)$ then
21:         $phimin(q) \leftarrow 0$
22:     end if
23:     for all $u_c \in U_c$ do
24:         if $phimin(q) = phimax(q)$ then
25:             break
26:         end if
27:         $phimax(q, u_c) \leftarrow 1$
28:         if $\exists t \in [0, \tau] : A_{\tau,q,u_c}(t) \cap B \neq \emptyset$ then
29:             $phimax(q, u_c) \leftarrow 0$
30:         end if
31:         for all $q' \in \mathbf{Post}_{u_c}(q)$ do
32:             if $phimax(q, u_c) \leq phimin(q) \vee (\tau = \tau_{\min} \wedge phimax(q, u_c) \leq 0)$ then
33:                 break
34:             end if
35:             $phimax(q, u_c) \leftarrow \min\{phimax(q, u_c), \text{DoDFS}(q', \tau, \tau_{\min}, queue)\}$
36:         end for
37:         $phimin(q) \leftarrow \max\{phimin(q), phimax(q, u_c)\}$
38:     end for
39:     $\phi_\tau(q) \leftarrow phimin(q)$
40:     if $\phi_\tau(q) = 0 \wedge \tau = \tau_{\min}$ then
41:         EnqueueRefined$(q, \tau, queue)$
42:     end if
43:     return $\phi_\tau(q)$
44: end procedure
```

**Proposition 3** *Let $\tau_{\max} = 2^r \tau_{\min}$ and let $n = |\mathcal{N}|$, the number of vehicles. Then the running time of Algorithms 1 and 2 is in*
$O\left(\frac{1-2^{-(r+1)n}}{1-2^{-n}} |\tilde{Q}_{\tau_{\min}}||U_c| \left[|\mathbf{Post}_{u_c}(q)| + n^2\right]\right)$.

*Proof.* At a particular level of discretization $\tau$, Alg. 2 is executed at most $|\tilde{Q}_\tau|$ times, once for each examined $q \in \tilde{Q}_\tau$. All subroutines of lines 2-22 run in time at most $O(n^2)$. The outer for loop (lines 23-38) is executed $|U_c|$ times and consists of verifying the condition $[A_{q,u_c}(t) \cap B \neq \emptyset]$ and executing the inner for loop. Verifying the condition $\nexists t \in [0, \tau] : A_{q,u_c}(t) \cap B \neq \emptyset$ (line 28) takes $O(n^2)$ time (see Appendix). The inner for loop (lines 31-36) is executed $|\mathbf{Post}_{u_c}(q)|$ times, each of which takes $O(1)$ time beyond that of the recursive call. The total running time at discretization level $\tau$ is therefore $O(|\tilde{Q}_\tau||U_c| \left[|\mathbf{Post}_{u_c}(q)| + n^2\right])$. Taking $\tau' = \tau/2$ means refining each state into two, along each of $n$ dimensions. Thus, $|\tilde{Q}_\tau| = 2^{-n}|\tilde{Q}_{\tau/2}|$. The total running time is therefore in

$$O\left(\sum_{i=0}^{r} 2^{-ni}|\tilde{Q}_{\tau_{\min}}||U_c| \left[|\mathbf{Post}_{u_c}(q)| + n^2\right]\right)$$
$$= O\left(\frac{1-2^{-(r+1)n}}{1-2^{-n}}|\tilde{Q}_{\tau_{\min}}||U_c| \left[|\mathbf{Post}_{u_c}(q)| + n^2\right]\right).$$

□

*Remark 9* Because the particular state $q \in \tilde{Q}$ and control action $u_c \in U_c$ do not restrict the set of possible actions of the uncontrolled vehicles $U_{uc}$ or the set of possible disturbance events $W$, the value $|\mathbf{Post}_{u_c}(q)|$ is independent of the particular $q \in \tilde{Q}$ and $u_c \in U_c$. This value is, however, dependent on the number of vectors of actions of the uncontrolled vehicles (which determines $|U_{uc}|$), as well as on the bounds of the disturbance (which determines $|W|$). □

We note that the expression $\frac{1-2^{-(r+1)n}}{1-2^{-n}}$ will typically be quite small, meaning that even if iterative refinement yields no benefit (i.e., if $\phi_\tau(q) = 0$ for all $q \in \tilde{Q}_\tau$, for all $\tau > \tau_{\min}$), there will be little overhead. In the worst case, $r \to \infty$ and $n = 2$, yielding $\lim_{r\to\infty} \frac{1-2^{-(r+1)n}}{1-2^{-n}} = \frac{4}{3}$.

## 9 Simulation Results

In this section, we present results from simulations run in C++. Simulations sought to compare running time for an algorithm using iterative refinement compared to one which does not; and for an algorithm which uses an optimization based on capture set computation (described in the appendix), to one which does not. Thus four simulation were run for each problem instance, consisting of the four possible combinations. Briefly, the capture set optimization consists of computing the capture set (the complement of the maximal controlled invariant set) for each pair of vehicles that cannot simultaneously be inside the intersection. This can be done easily for such pairs of vehicles, since the bad set is bounded and convex in this case (N.B., for more than two vehicles, the bad set is a union of inverse projections of sets, which is neither bounded nor convex). The changes to Alg. 2 from using the capture set optimization consist of replacing the two subroutines NatVic and ContLoss. Recall that NatVic (resp., ContLoss) tests whether a discrete state is contained in (resp., intersects) the *bad* set. In the capture set version, NatVic (resp., ContLoss) tests whether a discrete state is contained in (resp., intersects) the *capture* set. We focus on this optimization in particular since it is the only one which can affect the level of discretization at which some part of the state space is classified as winning for nature. That is, it is possible that $\phi_\tau(q)$ will be evaluated to be -1 when the capture set optimization is used, but evaluated to be 0 without the capture set optimization (perhaps only to have all $\phi_{\tau/2}(q')$ evaluated to be -1 for the refined states). Thus, one would expect that the use of the capture set optimization might increase the benefit of using iterative refinement. The five optimizations of the previous section, on the other hand, may affect running time, but will have no effect on the value of $\phi_\tau(q)$ that is computed for any $\tau$ and any $q \in \tilde{Q}_\tau$.

### 9.1 Simulation Descriptions

In each case, we used $\mu = 1$ and $\tau_{\min} = 1$ for the space and time discretization. For simulations which used iterative refinement, $\tau_{\max}$ was chosen automatically at run time, and was determined so that the entire state space was covered with a single discrete state. We consider three different scenarios: the first has no disturbance and no uncontrolled vehicles; the second has uncontrolled vehicles but no disturbance; the third has no uncontrolled vehicles but has a disturbance. We do not present a scenario which includes both uncontrolled vehicles and a disturbance, since these often result in empty solutions. For each scenario, we used four different problem instances, where we varied the number of vehicles among 2, 3, 4, and 6. For the six vehicle cases, the intersection consisted of six roads arranged in a regular hexagonal pattern. Vehicles cross from one road to the road opposite their starting road. Specifically, if the set of vehicles is $\mathcal{N} = \{1, \ldots, 6\}$, then vehicle $i \in \mathcal{N}$ starts on road $r_{i,1} = i$ and ends on road $r_{i,2} = 1 + [(i+2) \mod 6]$. Thus, the three pairs of vehicles $(1,4)$, $(2,5)$, and $(3,6)$ can occupy the intersection simultaneously, but vehicles from different pairs cannot (see Fig. 8). Problem instances with 2, 3, and 4 vehicles used the same intersection, but restricted to the sets of vehicles $\{1,2\}$, $\{1,2,3\}$, and $\{1,2,4,5\}$, respectively. In problem instances with uncontrollable vehicles, the uncontrollable vehicles were chosen to be vehicles 1 and 4. In problem instances with a disturbance, we used $d_{\min} = -1$ and $d_{\max} = 1$. Different problem instances used different road lengths and values of $\alpha_r$ (recall that $\alpha_r$ is the length of road $r$ that is inside the intersection), but these were constant for all roads in a particular problem instance. As an example, a road length of $l = 20$ with $\alpha = 2$ would mean that a vehicle starting at the beginning of the

**Table 1** List of Problem Instances

| Instance | Road Length | $\alpha$ | Total States | Total Transitions | Safe States |
|---|---|---|---|---|---|
| $\bar{U}\bar{D}2$ | 6870 | 1374 | $6.80 \times 10^7$ | $2.72 \times 10^8$ | $5.29 \times 10^7$ |
| $\bar{U}\bar{D}3$ | 357 | 51 | $6.84 \times 10^7$ | $5.47 \times 10^8$ | $4.69 \times 10^7$ |
| $\bar{U}\bar{D}4$ | 82.5 | 7.5 | $6.86 \times 10^7$ | $1.10 \times 10^9$ | $5.55 \times 10^7$ |
| $\bar{U}\bar{D}6$ | 19 | 1 | $8.58 \times 10^7$ | $5.49 \times 10^9$ | $6.31 \times 10^7$ |
| $U\bar{D}2$ | 6870 | 1374 | $6.80 \times 10^7$ | $2.72 \times 10^8$ | $3.02 \times 10^7$ |
| $U\bar{D}3$ | 357 | 51 | $6.84 \times 10^7$ | $5.47 \times 10^8$ | $1.60 \times 10^7$ |
| $U\bar{D}4$ | 82.5 | 7.5 | $6.86 \times 10^7$ | $1.10 \times 10^9$ | $1.59 \times 10^7$ |
| $U\bar{D}6$ | 19 | 1 | $8.58 \times 10^7$ | $5.49 \times 10^9$ | $9.56 \times 10^6$ |
| $\bar{U}D2$ | 2500 | 500 | $9.01 \times 10^6$ | $1.30 \times 10^9$ | $5.34 \times 10^6$ |
| $\bar{U}D3$ | 178.5 | 25.5 | $8.62 \times 10^6$ | $1.49 \times 10^{10}$ | $3.31 \times 10^6$ |
| $\bar{U}D4$ | 49.5 | 4.5 | $9.15 \times 10^6$ | $1.90 \times 10^{11}$ | $5.36 \times 10^6$ |
| $\bar{U}D6$ | 9.5 | 0.5 | $1.77 \times 10^6$ | $5.29 \times 10^{12}$ | $1.83 \times 10^5$ |

Problem instances are denoted as $\{U, \bar{U}\}\{D, \bar{D}\}\{2, 3, 4, 6\}$ where: $U$ or $\bar{U}$ denotes the presence or absence of uncontrollable vehicles; $D$ or $\bar{D}$ denotes the presence or absence of a disturbance; and $\{2, 3, 4, 6\}$ denotes the number of vehicles. Parameters were chosen so as to make the number of states approximately the same for simulations with 2, 3, or 4 vehicles, for each of the three scenarios considered. Simulations with 6 vehicles had more states in the scenarios without a disturbance, and far fewer states in the scenario with a disturbance. All instances with two vehicles had values for $l$ (road length) and $\alpha$ chosen so that vehicles would be inside the intersection for 1/3 of their path (i.e., $2\alpha/(l + \alpha) = 1/3$). For 3, 4, and 6 vehicles the corresponding ratios were chosen to be 1/4, 1/6, and 1/10, respectively.

road would travel a distance of $l - \alpha = 18$ to reach the intersection, a further distance of $\alpha = 2$ to reach the center of the intersection, and a final distance of $\alpha = 2$ to exit the intersection.

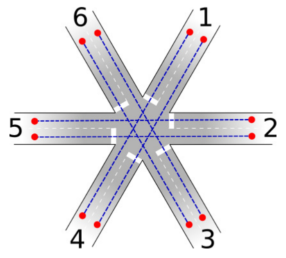


**Fig. 8** The intersection and vehicle paths used in each of the simulations of this section. Blue lines are drawn for each vehicle indicating starting road and ending road.

For each problem instance, we provide the following data: road length, $\alpha$, total number of discrete states, total number of transitions, and number of winning states for the controller. The last three values are determined at the $\tau = 1$ level of discretization. See Table 1.

### 9.2 Results & Analysis

For each simulation, we provide the running time in seconds, the total number of discrete states examined and the number of these found to be winning for the controller. For simulations not using iterative refinement, these last two values will be as in Table 1. For simulations using iterative refinement, these values are summed over all levels of discretization. Thus, for the total number of discrete states, we include *all* states, including those which were classified as losing for both the controller and nature and were later refined. See Table 2.

The results of Table 2 show that iterative refinement usually improves running time. Furthermore, the improvement in running time was large for simulations with two vehicles (ranging from a factor of 79 to a factor of 1150 without the capture set optimization, and from a factor of 160 to a factor of 1932 with the capture set optimization), and diminishing as the number of vehicles increased. The reason for this is not a lack of scalability of the method, but a consequence of the fact that road lengths were shorter in simulations with more vehicles. Intuitively, a finer discretization is needed when closer to the boundary between winning states for the controller and winning states for nature. Thus, iterative refinement works best when there is a significant portion of the state set that is "far" from the bad set. Problem instances with more vehicles used shorter road lengths, and hence states were generally closer to the boundary between winning sets in these instances. Indeed, simulations (not shown here) conducted with few vehicles and small road lengths showed improvement by a much smaller factor than for the same number of vehicles and long road lengths.

There are two problem instances (out of 12) where iterative refinement did not improve running time. In both cases, these instances are with six vehicles and with a source of non-determinism. Predictably, the number of states examined through iterative refinement in these two problem instances was high relative to the number of states examined without iterative refinement. The relevant proportions were 94.8% for instance $U\bar{D}6$ and 101% for instance $\bar{U}D6$. In no other instance was this proportion greater than 57.1%.

As per the discussion at the beginning of this section, the use of the capture set optimization did indeed increase the benefit of using iterative refinement. In all but two problem instances, the ratio of running time without iterative refinement to running time with iterative refinement was higher with the capture set optimization than without. The exceptions were instance $\bar{U}D3$, where the relevant ratios were 2.90 and 4.01, and instance $\bar{U}D6$, where the relevant ratios were 0.790 and 0.825.

## 10 Conclusion

We considered the problem of supervising a set of vehicles approaching an intersection so as to avoid collisions, in the presence of environmental un-

**Table 2** List of Simulations

| Simulation | Running Time | Examined States | Winning States |
|---|---|---|---|
| $\bar{U}\bar{D}2\bar{C}\bar{R}$ | 82.6 | $6.80 \times 10^7$ | $5.29 \times 10^7$ |
| $\bar{U}\bar{D}2\bar{C}R$ | 0.0962 | $1.27 \times 10^5$ | $1.37 \times 10^4$ |
| $\bar{U}\bar{D}2C\bar{R}$ | 73.8 | $6.80 \times 10^7$ | $5.29 \times 10^7$ |
| $\bar{U}\bar{D}2CR$ | 0.0382 | $8.57 \times 10^4$ | $1.37 \times 10^4$ |
| $\bar{U}\bar{D}3\bar{C}\bar{R}$ | 90.1 | $6.84 \times 10^7$ | $4.69 \times 10^7$ |
| $\bar{U}\bar{D}3\bar{C}R$ | 3.65 | $2.94 \times 10^6$ | $3.00 \times 10^5$ |
| $\bar{U}\bar{D}3C\bar{R}$ | 68.9 | $6.84 \times 10^7$ | $4.69 \times 10^7$ |
| $\bar{U}\bar{D}3CR$ | 1.01 | $2.01 \times 10^6$ | $3.00 \times 10^5$ |
| $\bar{U}\bar{D}4\bar{C}\bar{R}$ | 69.7 | $6.86 \times 10^7$ | $5.55 \times 10^7$ |
| $\bar{U}\bar{D}4\bar{C}R$ | 14.8 | $7.56 \times 10^6$ | $9.64 \times 10^5$ |
| $\bar{U}\bar{D}4C\bar{R}$ | 52.3 | $6.86 \times 10^7$ | $5.55 \times 10^7$ |
| $\bar{U}\bar{D}4CR$ | 2.84 | $6.41 \times 10^6$ | $9.64 \times 10^5$ |
| $\bar{U}\bar{D}6\bar{C}\bar{R}$ | 153 | $8.58 \times 10^7$ | $6.31 \times 10^7$ |
| $\bar{U}\bar{D}6\bar{C}R$ | 128 | $3.30 \times 10^7$ | $9.65 \times 10^6$ |
| $\bar{U}\bar{D}6C\bar{R}$ | 81.5 | $8.58 \times 10^7$ | $6.31 \times 10^7$ |
| $\bar{U}\bar{D}6CR$ | 41.8 | $3.30 \times 10^7$ | $9.65 \times 10^6$ |
| $U\bar{D}2\bar{C}\bar{R}$ | 122 | $6.80 \times 10^7$ | $3.02 \times 10^7$ |
| $U\bar{D}2\bar{C}R$ | 0.106 | $2.03 \times 10^7$ | $1.40 \times 10^4$ |
| $U\bar{D}2C\bar{R}$ | 57.0 | $6.80 \times 10^7$ | $3.02 \times 10^7$ |
| $U\bar{D}2CR$ | 0.0488 | $1.08 \times 10^5$ | $1.40 \times 10^4$ |
| $U\bar{D}3\bar{C}\bar{R}$ | 109 | $6.84 \times 10^7$ | $1.60 \times 10^7$ |
| $U\bar{D}3\bar{C}R$ | 2.83 | $3.69 \times 10^7$ | $2.82 \times 10^5$ |
| $U\bar{D}3C\bar{R}$ | 42.8 | $6.84 \times 10^7$ | $1.60 \times 10^7$ |
| $U\bar{D}3CR$ | 0.993 | $3.95 \times 10^6$ | $2.82 \times 10^5$ |
| $U\bar{D}4\bar{C}\bar{R}$ | 98.0 | $6.86 \times 10^7$ | $1.59 \times 10^7$ |
| $U\bar{D}4\bar{C}R$ | 13.0 | $3.92 \times 10^7$ | $1.95 \times 10^6$ |
| $U\bar{D}4C\bar{R}$ | 45.4 | $6.86 \times 10^7$ | $1.59 \times 10^7$ |
| $U\bar{D}4CR$ | 4.18 | $9.20 \times 10^6$ | $1.95 \times 10^6$ |
| $U\bar{D}6\bar{C}\bar{R}$ | 257 | $8.58 \times 10^7$ | $9.56 \times 10^6$ |
| $U\bar{D}6\bar{C}R$ | 271 | $8.13 \times 10^7$ | $3.43 \times 10^6$ |
| $U\bar{D}6C\bar{R}$ | 59.1 | $8.58 \times 10^7$ | $9.56 \times 10^6$ |
| $U\bar{D}6CR$ | 39.5 | $8.13 \times 10^7$ | $3.43 \times 10^6$ |
| $\bar{U}D2\bar{C}\bar{R}$ | 21.6 | $9.01 \times 10^6$ | $5.34 \times 10^6$ |
| $\bar{U}D2\bar{C}R$ | 0.272 | $9.95 \times 10^4$ | $8.23 \times 10^3$ |
| $\bar{U}D2C\bar{R}$ | 10.5 | $9.01 \times 10^6$ | $5.34 \times 10^6$ |
| $\bar{U}D2CR$ | 0.0657 | $4.50 \times 10^4$ | $8.23 \times 10^3$ |
| $\bar{U}D3\bar{C}\bar{R}$ | 61.4 | $8.62 \times 10^6$ | $3.31 \times 10^6$ |
| $\bar{U}D3\bar{C}R$ | 15.3 | $1.67 \times 10^6$ | $2.08 \times 10^5$ |
| $\bar{U}D3C\bar{R}$ | 17.9 | $8.62 \times 10^6$ | $3.31 \times 10^6$ |
| $\bar{U}D3CR$ | 6.17 | $1.20 \times 10^6$ | $2.08 \times 10^5$ |
| $\bar{U}D4\bar{C}\bar{R}$ | 154 | $9.15 \times 10^6$ | $5.36 \times 10^6$ |
| $\bar{U}D4\bar{C}R$ | 138 | $4.30 \times 10^6$ | $8.19 \times 10^5$ |
| $\bar{U}D4C\bar{R}$ | 73.5 | $9.15 \times 10^6$ | $5.36 \times 10^6$ |
| $\bar{U}D4CR$ | 38.9 | $3.10 \times 10^6$ | $8.19 \times 10^5$ |
| $\bar{U}D6\bar{C}\bar{R}$ | 288 | $1.77 \times 10^6$ | $1.83 \times 10^5$ |
| $\bar{U}D6\bar{C}R$ | 349 | $1.79 \times 10^6$ | $1.55 \times 10^5$ |
| $\bar{U}D6C\bar{R}$ | 128 | $1.77 \times 10^6$ | $1.83 \times 10^5$ |
| $\bar{U}D6CR$ | 162 | $1.79 \times 10^6$ | $1.55 \times 10^5$ |

Simulations are denoted as $\{U, \bar{U}\}\{D, \bar{D}\}\{2, 3, 4, 6\}\{C, \bar{C}\}\{R, \bar{R}\}$ where: $C$ or $\bar{C}$ denotes the use or non-use of the capture set optimization and $R$ or $\bar{R}$ denotes the use or non-use of iterative refinement. The first three elements of this notation were defined in the description of Table 1.

certainty in the form of uncontrolled vehicles and a disturbance. We solved this problem by constructing a DES abstraction and leveraging supervisory control methods of DES, a natural formulation for problems involving uncontrolled elements in which it is desired to obtain maximally permissive safe and non-deadlocking supervisors. We described the state reduction and exact state reduction relations between systems and abstractions, and used these to show that translating the supervisor for the abstraction back to the original problem domain preserves not only safety and non-deadlockingness, but also maximal permissiveness. Finally, we presented an algorithm for solving this supervisory control problem, based on a technique called iterative refinement, and demonstrated its scalability through simulation. This works extends the range of applications of DES. Moreover, to the best of our knowledge, it is the first DES application where the discrete event model is obtained by building a state reduction abstraction of the underlying continuous system model. Future work includes the extension of this work to the case of measurement uncertainty, second order dynamics, and stochastic problem formulations.

## References

1. Ahn H, Colombo A, Del Vecchio D (2014) Supervisory Control for Intersection Collision Avoidance in the Presence of Uncontrolled Vehicles. In: American Control Conference (ACC), 2014
2. Alur R, Henzinger T, Lafferriere G, Pappas G (2000) Discrete abstractions of hybrid systems. Proceedings of the IEEE 88(7):971–984
3. Au TC, Fok CL, Vishwanath S, Julien C, Stone P (2012) Evasion planning for autonomous vehicles at intersections. In: Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, pp 1541–1546
4. Bruni L, Colombo A, Del Vecchio D (2013) Robust multi-agent collision avoidance through scheduling. In: Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on
5. Camara J, Girard A, Gossler G (2011) Safety controller synthesis for switched systems using multi-scale symbolic models. In: Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on, pp 520–525
6. Cassandras CG, Lafortune S (2008) Introduction to Discrete Event Systems, 2nd edn. Springer
7. Colombo A (2014) A mathematical framework for cooperative collision avoidance of human-driven vehicles at intersections. In: International Symposium on Wireless Communication Systems
8. Colombo A, Del Vecchio D (2011) Enforcing Safety of Cyberphysical Systems Using Flatness and Abstraction. SIGBED Rev 8(2):11–14
9. Colombo A, Del Vecchio D (2011) Supervisory control of differentially flat systems based on abstraction. In: Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on, pp 6134–6139

10. Colombo A, Del Vecchio D (2012) Efficient Algorithms for Collision Avoidance at Intersections. In: Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control, ACM, New York, NY, USA, HSCC '12, pp 145–154
11. Colombo A, Del Vecchio D (2015) Least restrictive supervisors for intersection collision avoidance: A scheduling approach. IEEE Transactions on Automatic Control DOI 10.1109/TAC.2014.2381453
12. Colombo A, Girard A (2013) An approximate abstraction approach to safety control of differentially flat systems. In: Control Conference (ECC), 2013 European, pp 4226–4231
13. Dallal E, Colombo A, Del Vecchio D, Lafortune S (2013) Supervisory control for collision avoidance in vehicular networks using discrete event abstractions. In: American Control Conference (ACC), 2013, pp 4380–4386
14. Dallal E, Colombo A, Del Vecchio D, Lafortune S (2013) Supervisory control for collision avoidance in vehicular networks with imperfect measurements. In: Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on, pp 6298–6303
15. Daws C, Tripakis S (1998) Model checking of real-time reachability properties using abstractions. In: Steffen B (ed) Tools and Algorithms for the Construction and Analysis of Systems, Lecture Notes in Computer Science, vol 1384, Springer Berlin Heidelberg, pp 313–329
16. Girard A, Pola G, Tabuada P (2010) Approximately Bisimilar Symbolic Models for Incrementally Stable Switched Systems. Automatic Control, IEEE Transactions on 55(1):116–126
17. Hadj-Alouane NB, Lafortune S, Lin F (1994) Variable lookahead supervisory control with state information. Automatic Control, IEEE Transactions on 39(12):2398–2410
18. Hafner M, Del Vecchio D (2011) Computational Tools for the Safety Control of a Class of Piecewise Continuous Systems with Imperfect Information on a Partial Order. SIAM Journal on Control and Optimization 49(6):2463–2493
19. Hafner M, Cunningham D, Caminiti L, Del Vecchio D (2013) Cooperative collision avoidance at intersections: Algorithms and experiments. Intelligent Transportation Systems, IEEE Transactions on 14(3):1162–1175
20. Kowshik H, Caveney D, Kumar P (2011) Provable Systemwide Safety in Intelligent Intersections. Vehicular Technology, IEEE Transactions on 60(3):804–818
21. Nilsson P, Ozay N (2014) Incremental synthesis of switching protocols via abstraction refinement. In: Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on, IEEE, pp 6246–6253
22. Pola G, Tabuada P (2009) Symbolic Models for Nonlinear Control Systems: Alternating Approximate Bisimulations. SIAM Journal on Control and Optimization 48(2):719–733
23. Ramadge PJ, Wonham WM (1987) Supervisory control of a class of discrete event processes. SIAM journal on control and optimization 25(1):206–230

24. Shoham S, Grumberg O (2003) A game-based framework for ctl counterexamples and 3-valued abstraction-refinement. In: Computer Aided Verification, Springer, pp 275–287
25. Tabuada P (2009) Verification and control of hybrid systems: a symbolic approach. Springer
26. Tomlin CJ, Mitchell I, Bayen AM, Oishi M (2003) Computational techniques for the verification of hybrid systems. Proceedings of the IEEE 91(7):986–1001
27. Verma R, Del Vecchio D (2011) Semiautonomous multivehicle safety. Robotics & Automation Magazine, IEEE 18(3):44–54
28. Wonham W (2013) Supervisory Control of Discrete-Event Systems, http://www.control.toronto.edu/people/profs/wonham/wonham.html
29. Wonham W, Ramadge P (1987) On the Supremal Controllable Sublanguage of a Given Language. SIAM Journal on Control and Optimization 25(3):637–659
30. Zamani M, Pola G, Mazo M, Tabuada P (2012) Symbolic Models for Nonlinear Control Systems Without Stability Assumptions. Automatic Control, IEEE Transactions on 57(7):1804–1809

**Acknowledgements**

Research supported in part by NSF grant CNS-0930081 and by the NSF Expeditions in Computing project ExCAPE: Expeditions in Computer Augmented Program Engineering (grant CCF-1138860).

## Appendix : Equations for Checking Safety

This appendix provides the equations that were used in the simulations of Sec. 9 for verifying the safety of DES transitions (Part 1), and the equations for the pair-wise capture sets for vehicles that cannot simultaneously be inside the intersection (Part 2).

**Part 1**: Verifying if $A_{q,u_c}(t) \cap B = \emptyset$ for all $t \in [0, \tau]$.

In part 1 of this appendix, we prove the equations used for verifying the safety of transitions. As stated in Sec. 8, there are equations for each pair of vehicles $i, j \in \mathcal{N}$, and verifying the safety of a DES transition for some initial state $q \in \tilde{Q}$ and $u_c \in U_c$ is done by verifying the corresponding equations for each pair of vehicles. We consider three cases (see Sec. 3): $x_i, x_j \leq 0$, $|x_i - x_j| < \gamma$ (case 1a), $x_i, x_j \geq 0$, $|x_i - x_j| < \gamma$ (case 1b), and $[-\alpha_{r_{i,1}} < x_i < \alpha_{r_{i,2}}] \wedge [-\alpha_{r_{j,1}} < x_j < \alpha_{r_{j,2}}]$ (case 2). The equations for these cases are provided in Props. (5)-(7), respectively. Note that there is no "case 1c" when $x_i \leq 0$ and $x_j \geq 0$, since the vehicles would then be on different roads.

We begin by defining the set $A_{q,u_c}([0, \tau]) := \bigcup_{t \in [0,\tau]} A_{q,u_c}(t)$. Because the bad set is defined as a union of sets of linear inequalities, with one set for each pair of vehicles, we verify $A_{q,u_c}([0, \tau]) \cap B = \emptyset$ by considering each pair of vehicles in turn. For any vehicle $i \in \mathcal{N}$ and any set $P \subseteq X$, let $\pi_i(P)$ denote the projection of $P$ onto the $x_i$ axis. Similarly, for any pair of vehicles $i, j \in \mathcal{N}$ and a set $P \subseteq X$, let $\pi_{i,j}(P)$ denote the projection of $P$ onto the $x_i - x_j$ plane. Also recall the notation $\underline{v}_{u_c,i}$ and $\overline{v}_{u_c,i}$ defined in Eqs. (39) and (40).

**Proposition 4** $(x_i, x_j) \in \pi_{i,j}(A_{q,u_c}([0,\tau]))$ *iff all of the following inequalities hold:*

$$x_i > q_i - \mu\tau/2 \tag{41}$$
$$x_j > q_j - \mu\tau/2 \tag{42}$$
$$x_i \leq q_i + \mu\tau/2 + \overline{v}_{u_c,i}\tau \tag{43}$$
$$x_j \leq q_j + \mu\tau/2 + \overline{v}_{u_c,j}\tau \tag{44}$$
$$\overline{v}_{u_c,i}(x_j - q_j + \mu\tau/2) - \underline{v}_{u_c,j}(x_i - q_i - \mu\tau/2) > 0 \tag{45}$$
$$\overline{v}_{u_c,j}(x_i - q_i + \mu\tau/2) - \underline{v}_{u_c,i}(x_j - q_j - \mu\tau/2) > 0 \tag{46}$$

*Proof.* From Eqs. (39), (40) and the assumption that $v_{min} + d_{min} \geq \mu > 0$, we have that $\pi_i(A_{q,u_c}(t)) = (q_i - \mu\tau/2 + \underline{v}_{u_c,i}t, q_i + \mu\tau/2 + \overline{v}_{u_c,i}t]$ is an interval whose lower and upper bounds are increasing in time, for every $i \in \mathcal{N}$. It follows that, for any $x_i$, the set $\{t \in \mathbb{R} : x_i \in \pi_i(A_{q,u_c}(t)\}$ will have the form $[t_{i,min}, t_{i,max})$, where $t_{i,min} := \inf\{t \in \mathbb{R} : x_i \in \pi_i(A_{q,u_c}(t))\}$ and $t_{i,max} := \sup\{t \in \mathbb{R} : x_i \in \pi_i(A_{q,u_c}(t))\}$ are given by:

$$t_{i,min} = \frac{x_i - q_i - \mu\tau/2}{\overline{v}_{u_c,i}} \tag{47}$$
$$t_{i,max} = \frac{x_i - q_i + \mu\tau/2}{\underline{v}_{u_c,i}} \tag{48}$$

Now define $t_{j,min}$ and $t_{j,max}$ analogously to $t_{i,min}$ and $t_{i,max}$. Then:

$$\begin{aligned} & \exists t \in [0,\tau] \text{ s.t. } [x_i \in \pi_i(A_{q,u_c}(t))] \wedge [x_j \in \pi_j(A_{q,u_c}(t))] \\ \Leftrightarrow\ & [0,\tau] \cap [t_{i,min}, t_{i,max}) \cap [t_{j,min}, t_{j,max}) \neq \emptyset \\ \Leftrightarrow\ & \begin{array}{l} [t_{i,max} > 0] \wedge [t_{j,max} > 0] \wedge [t_{i,min} \leq \tau] \wedge [t_{j,min} \leq \tau] \\ \quad \wedge [t_{j,max} > t_{i,min}] \wedge [t_{i,max} > t_{j,min}] \end{array} \end{aligned}$$

and these last six inequalities give Eqs. (41)-(46), in order. □

As stated above, we can check if $A_{q,u_c}([0,\tau]) \cap B = \emptyset$ by considering each pair of vehicles in turn. There are three types of constraints to consider:

**Case 1a**: $x_i, x_j \leq 0$, $|x_i - x_j| < \gamma$.

**Lemma 2** *Consider any* $\underline{x}_i, \overline{x}_i, \underline{x}_j, \overline{x}_j \in \mathbb{R}$. *Then:*

$$\begin{aligned} & (\exists x_i \in (\underline{x}_i, \overline{x}_i])(\exists x_j \in (\underline{x}_j, \overline{x}_j])(x_i \leq 0 \wedge x_j \leq 0 \wedge |x_i - x_j| < \gamma) \\ \Leftrightarrow\ & [\underline{x}_i < \overline{x}_i \wedge \underline{x}_i < 0 \wedge \underline{x}_j < \overline{x}_j \wedge \underline{x}_j < 0 \wedge \underline{x}_i - \overline{x}_j < \gamma \wedge \underline{x}_j - \overline{x}_i < \gamma] \end{aligned} \tag{49}$$

*Proof.* ($\Rightarrow$):

$$\begin{aligned}
x_i \in (\underline{x}_i, \overline{x}_i] \Rightarrow \underline{x}_i < x_i \leq \overline{x}_i \Rightarrow \underline{x}_i < \overline{x}_i \\
\underline{x}_i < x_i \wedge x_i \leq 0 \Rightarrow \underline{x}_i < 0 \\
x_j \in (\underline{x}_j, \overline{x}_j] \Rightarrow \underline{x}_j < x_j \leq \overline{x}_j \Rightarrow \underline{x}_j < \overline{x}_j \\
\underline{x}_j < x_j \wedge x_j \leq 0 \Rightarrow \underline{x}_j < 0 \\
x_i - x_j < \gamma \wedge \underline{x}_i < x_i \wedge x_j \leq \overline{x}_j \Rightarrow \underline{x}_i - \overline{x}_j < \gamma \\
x_j - x_i < \gamma \wedge \underline{x}_j < x_j \wedge x_i \leq \overline{x}_i \Rightarrow \underline{x}_j - \overline{x}_i < \gamma
\end{aligned}$$

($\Leftarrow$) It cannot be that both $\underline{x}_i - \underline{x}_j \geq \gamma$ and $\underline{x}_j - \underline{x}_i \geq \gamma$, as this would imply $0 \geq 2\gamma > 0$. Thus, at least one of $\underline{x}_i - \underline{x}_j < \gamma$, or $\underline{x}_j - \underline{x}_i < \gamma$ holds. If they both hold, we may take $x_i = \underline{x}_i + \epsilon$ and $x_j = \underline{x}_j + \epsilon$ for some sufficiently small $\epsilon > 0$ and we are done. Suppose without loss of generality then that $\underline{x}_i - \underline{x}_j < \gamma$ but $\underline{x}_j - \underline{x}_i \geq \gamma$. Let $\overline{x}_i = \underline{x}_j - \gamma$. Thus, $\underline{x}_j - \overline{x}_i = \gamma$, $\overline{x}_i - \underline{x}_j = -\gamma < \gamma$ and $\overline{x}_i < 0$ (since $\underline{x}_j < 0$). We may therefore take $x_i = \overline{x}_i + 2\epsilon$ and $x_j = \underline{x}_j + \epsilon$ for some sufficiently small $\epsilon > 0$ and we are done. $\square$

**Proposition 5** *The set* $\{(x_i, x_j) \in \pi_{i,j}(A_{q,u_c}([0,\tau])) : x_i, x_j \leq 0 \wedge |x_i - x_j| < \gamma\}$ *is non-empty iff all of the following inequalities hold:*

$$\begin{aligned}
& q_i < \mu\tau/2 && (50) \\
& q_j < \mu\tau/2 && (51) \\
& \underline{v}_{u_c,j}(q_i + \mu\tau/2 + \gamma) - \max\{\overline{v}_{u_c,i}, \underline{v}_{u_c,j}\}(q_j - \mu\tau/2) > 0 && (52) \\
& \underline{v}_{u_c,i}(q_j + \mu\tau/2 + \gamma) - \max\{\overline{v}_{u_c,j}, \underline{v}_{u_c,i}\}(q_i - \mu\tau/2) > 0 && (53) \\
& [q_i + \mu\tau/2 + \gamma + \tau \max\{\overline{v}_{u_c,i}, \underline{v}_{u_c,j}\}] - [q_j - \mu\tau/2 + \tau\underline{v}_{u_c,j}] > 0 && (54) \\
& [q_j + \mu\tau/2 + \gamma + \tau \max\{\overline{v}_{u_c,j}, \underline{v}_{u_c,i}\}] - [q_i - \mu\tau/2 + \tau\underline{v}_{u_c,i}] > 0 && (55)
\end{aligned}$$

*Proof.* Let $\pi_i(A_{q,u_c}(t)) = (\underline{x}_i(t), \overline{x}_i(t)]$ and $\pi_j(A_{q,u_c}(t)) = (\underline{x}_j(t), \overline{x}_j(t)]$. By Lemma 2, it is necessary and sufficient to find some $t \in [0,\tau]$ such that $\underline{x}_i(t) < 0$, $\underline{x}_j(t) < 0$, $\underline{x}_i(t) - \overline{x}_j(t) < \gamma$, and $\underline{x}_j(t) - \overline{x}_i(t) < \gamma$. Now define $t_{i,max}$, $t_{j,max}$, $t_{i-j}$, and $t_{j-i}$ by $\underline{x}_i(t_{i,max}) = 0$, $\underline{x}_j(t_{j,max}) = 0$, $\underline{x}_i(t_{i-j}) - \overline{x}_j(t_{i-j}) = \gamma$, and $\underline{x}_j(t_{j-i}) - \overline{x}_i(t_{j-i}) = \gamma$. These are given by:

$$t_{i,max} = -\frac{q_i - \mu\tau/2}{\underline{v}_{u_c,i}} \quad (56)$$

$$t_{j,max} = -\frac{q_j - \mu\tau/2}{\underline{v}_{u_c,j}} \quad (57)$$

$$t_{i-j} = \frac{(q_i - \mu\tau/2) - (q_j + \mu\tau/2 + \gamma)}{\overline{v}_{u_c,j} - \underline{v}_{u_c,i}} \quad (58)$$

$$t_{j-i} = \frac{(q_j - \mu\tau/2) - (q_i + \mu\tau/2 + \gamma)}{\overline{v}_{u_c,i} - \underline{v}_{u_c,j}} \quad (59)$$

Obviously, $t_{i-j}$ is only well defined when $\overline{v}_{u_c,j} \neq \underline{v}_{u_c,i}$ and $t_{j-i}$ is only well defined when $\overline{v}_{u_c,i} \neq \underline{v}_{u_c,j}$. Because $\underline{x}_i(t)$ and $\underline{x}_j(t)$ are increasing in time, we have that:

$$\underline{x}_i(t) < 0 \Leftrightarrow t < t_{i,max} \tag{60}$$
$$\underline{x}_j(t) < 0 \Leftrightarrow t < t_{j,max} \tag{61}$$

On the other hand, $\underline{x}_i(t) - \overline{x}_j(t)$ is increasing in time if $\overline{v}_{u_c,j} < \underline{v}_{u_c,i}$, decreasing in time if $\overline{v}_{u_c,j} > \underline{v}_{u_c,i}$, and constant if $\overline{v}_{u_c,j} = \underline{v}_{u_c,i}$. It therefore follows that:

$$\underline{x}_i(t) - \overline{x}_j(t) < \gamma \Leftrightarrow \begin{cases} t < t_{i-j}, & \overline{v}_{u_c,j} < \underline{v}_{u_c,i} \\ t > t_{i-j}, & \overline{v}_{u_c,j} > \underline{v}_{u_c,i} \\ (q_j + \mu\tau/2 + \gamma) > (q_i - \mu\tau/2), & \overline{v}_{u_c,j} = \underline{v}_{u_c,i} \end{cases} \tag{62}$$

Similarly,

$$\underline{x}_j(t) - \overline{x}_i(t) < \gamma \Leftrightarrow \begin{cases} t < t_{j-i}, & \overline{v}_{u_c,i} < \underline{v}_{u_c,j} \\ t > t_{j-i}, & \overline{v}_{u_c,i} > \underline{v}_{u_c,j} \\ (q_i + \mu\tau/2 + \gamma) > (q_j - \mu\tau/2), & \overline{v}_{u_c,i} = \underline{v}_{u_c,j} \end{cases} \tag{63}$$

This would give nine cases to consider, but three are impossible, since $\overline{v}_{u_c,j} < \underline{v}_{u_c,i} \Rightarrow \underline{v}_{u_c,j} \leq \overline{v}_{u_c,j} < \underline{v}_{u_c,i} \leq \overline{v}_{u_c,i} \Rightarrow \underline{v}_{u_c,j} < \overline{v}_{u_c,i}$ and similarly, $\overline{v}_{u_c,i} < \underline{v}_{u_c,j} \Rightarrow \underline{v}_{u_c,i} < \overline{v}_{u_c,j}$. We will consider each of the six remaining cases in turn, but first prove the following claims:

$$t_{j-i} < t_{j,max} \wedge t_{i,max} > 0 \Rightarrow t_{j-i} < t_{i,max} \tag{64}$$
$$t_{i-j} < t_{i,max} \wedge t_{j,max} > 0 \Rightarrow t_{i-j} < t_{j,max} \tag{65}$$
$$t_{i-j} > 0 \wedge \overline{v}_{u_c,j} < \underline{v}_{u_c,i} \Rightarrow t_{j-i} < t_{i-j} \tag{66}$$
$$t_{j-i} > 0 \wedge \overline{v}_{u_c,i} < \underline{v}_{u_c,j} \Rightarrow t_{i-j} < t_{j-i} \tag{67}$$

Clearly, Eq. (64) holds if $t_{j-i} \leq 0$. If $t_{j-i} > 0$, then $\underline{x}_i(t_{j-i}) < \overline{x}_i(t_{j-i}) = \underline{x}_j(t_{j-i}) - \gamma < \underline{x}_j(t_{j-i})$. From Eq. (61), we have that $t_{j-i} < t_{j,max} \Leftrightarrow \underline{x}_j(t_{j-i}) < 0$. Hence, $\underline{x}_i(t_{j-i}) < \underline{x}_j(t_{j-i}) < 0$ and therefore $t_{j-i} < t_{i,max}$ follows from Eq. (60), proving Eq. (64). Eq. (65) is proven similarly. To prove Eq. (66), suppose to the contrary that $t_{j-i} \geq t_{i-j} > 0$. As before, $t_{j-i} > 0 \Rightarrow \underline{x}_i(t_{j-i}) < \underline{x}_j(t_{j-i})$. From $\overline{v}_{u_c,j} < \underline{v}_{u_c,i}$, $t_{j-i} \geq t_{i-j}$, and Eq. (62), we have that $\underline{x}_i(t_{j-i}) \geq \overline{x}_j(t_{j-i}) + \gamma > \overline{x}_j(t_{j-i})$. Thus we have $\underline{x}_j(t_{j-i}) > \underline{x}_i(t_{j-i}) > \overline{x}_j(t_{j-i})$, which is a contradiction since it cannot be that $\underline{x}_j(t_{j-i}) > \overline{x}_j(t_{j-i})$ for $t_{j-i} > 0$, proving Eq. (66). Eq. (67) is proven similarly. We now proceed with the six cases. In what follows, note that Eqs. (52) and (54) both reduce to $(q_i + \mu\tau/2 + \gamma) > (q_j - \mu\tau/2)$ when $\overline{v}_{u_c,i} \leq \underline{v}_{u_c,j}$ and that Eqs. (53) and (55) similarly both reduce to $(q_j + \mu\tau/2 + \gamma) > (q_i - \mu\tau/2)$ when $\overline{v}_{u_c,j} \leq \underline{v}_{u_c,i}$.

<u>Case (i)</u>: $\overline{v}_{u_c,j} = \underline{v}_{u_c,i}$ and $\overline{v}_{u_c,i} = \underline{v}_{u_c,j}$.

$$\begin{aligned}
&\exists t \in [0,\tau] \text{ s.t. } [\underline{x}_i(t) < 0] \wedge [\underline{x}_j(t) < 0] \\
&\quad \wedge [\underline{x}_i(t) - \overline{x}_j(t) < \gamma] \wedge [\underline{x}_j(t) - \overline{x}_i(t) < \gamma] \\
\Leftrightarrow\ & \begin{array}{l} [0,\tau] \cap (-\infty, t_{i,max}) \cap (-\infty, t_{j,max}) \neq \emptyset \\ \quad \wedge[(q_j + \mu\tau/2 + \gamma) > (q_i - \mu\tau/2)] \\ \quad \wedge[(q_i + \mu\tau/2 + \gamma) > (q_j - \mu\tau/2)] \end{array} && \text{(Eqs. (60)-(63))} \\
\Leftrightarrow\ & \begin{array}{l} [0 < t_{i,max}] \wedge [0 < t_{j,max}] \\ \quad \wedge[(q_j + \mu\tau/2 + \gamma) > (q_i - \mu\tau/2)] \\ \quad \wedge[(q_i + \mu\tau/2 + \gamma) > (q_j - \mu\tau/2)] \end{array} \\
\Leftrightarrow\ & [(50)] \wedge [(51)] \wedge [(53) \wedge (55)] \wedge [(52) \wedge (54)]
\end{aligned}$$

Case (ii): $\overline{v}_{u_c,j} > \underline{v}_{u_c,i}$ and $\overline{v}_{u_c,i} = \underline{v}_{u_c,j}$.

$$\begin{aligned}
&\exists t \in [0,\tau] \text{ s.t. } [\underline{x}_i(t) < 0] \wedge [\underline{x}_j(t) < 0] \\
&\quad \wedge [\underline{x}_i(t) - \overline{x}_j(t) < \gamma] \wedge [\underline{x}_j(t) - \overline{x}_i(t) < \gamma] \\
\Leftrightarrow\ & \begin{array}{l} [0,\tau] \cap (-\infty, t_{i,max}) \cap (-\infty, t_{j,max}) \cap (t_{i-j}, \infty) \neq \emptyset \\ \quad \wedge[(q_i + \mu\tau/2 + \gamma) > (q_j - \mu\tau/2)] \end{array} && \text{(Eqs. (60)-(63))} \\
\Leftrightarrow\ & \begin{array}{l} [0 < t_{i,max}] \wedge [0 < t_{j,max}] \wedge [t_{i-j} < \tau] \wedge [t_{i-j} < t_{i,max}] \wedge [t_{i-j} < t_{j,max}] \\ \quad \wedge[(q_i + \mu\tau/2 + \gamma) > (q_j - \mu\tau/2)] \end{array} \\
\Leftrightarrow\ & \begin{array}{l} [0 < t_{i,max}] \wedge [0 < t_{j,max}] \wedge [t_{i-j} < \tau] \wedge [t_{i-j} < t_{i,max}] \\ \quad \wedge[(q_i + \mu\tau/2 + \gamma) > (q_j - \mu\tau/2)] \end{array} && \text{(Eq. (65))} \\
\Leftrightarrow\ & [(50)] \wedge [(51)] \wedge [(55)] \wedge [(53)] \wedge [(52) \wedge (54)]
\end{aligned}$$

Case (iii): $\overline{v}_{u_c,j} = \underline{v}_{u_c,i}$ and $\overline{v}_{u_c,i} > \underline{v}_{u_c,j}$.
This is case is symmetrical to Case (ii).

Case (iv): $\overline{v}_{u_c,j} < \underline{v}_{u_c,i}$ and $\overline{v}_{u_c,i} > \underline{v}_{u_c,j}$.

$$\begin{aligned}
&\exists t \in [0,\tau] \text{ s.t. } [\underline{x}_i(t) < 0] \wedge [\underline{x}_j(t) < 0] \\
&\quad \wedge [\underline{x}_i(t) - \overline{x}_j(t) < \gamma] \wedge [\underline{x}_j(t) - \overline{x}_i(t) < \gamma] \\
\Leftrightarrow\ & [0,\tau] \cap (-\infty, t_{i,max}) \cap (-\infty, t_{j,max}) \cap (-\infty, t_{i-j}) \cap (t_{j-i}, \infty) \neq \emptyset && \text{(Eqs. (60)-(63))} \\
\Leftrightarrow\ & \begin{array}{l} [0 < t_{i,max}] \wedge [0 < t_{j,max}] \wedge [0 < t_{i-j}] \wedge [t_{j-i} < \tau] \\ \quad \wedge[t_{j-i} < t_{i,max}] \wedge [t_{j-i} < t_{j,max}] \wedge [t_{j-i} < t_{i-j}] \end{array} \\
\Leftrightarrow\ & [0 < t_{i,max}] \wedge [0 < t_{j,max}] \wedge [0 < t_{i-j}] \wedge [t_{j-i} < \tau] \wedge [t_{j-i} < t_{j,max}] && \text{(Eqs. (64), (66))} \\
\Leftrightarrow\ & [(50)] \wedge [(51)] \wedge [(53) \wedge (55)] \wedge [(54)] \wedge [(52)]
\end{aligned}$$

Case (v): $\overline{v}_{u_c,j} > \underline{v}_{u_c,i}$ and $\overline{v}_{u_c,i} < \underline{v}_{u_c,j}$.
This is case is symmetrical to Case (iv).

Case (vi): $\overline{v}_{u_c,j} > \underline{v}_{u_c,i}$ and $\overline{v}_{u_c,i} > \underline{v}_{u_c,j}$.

$$\begin{aligned}
&\exists t \in [0,\tau] \text{ s.t. } [\underline{x}_i(t) < 0] \wedge [\underline{x}_j(t) < 0] \\
&\quad \wedge [\underline{x}_i(t) - \overline{x}_j(t) < \gamma] \wedge [\underline{x}_j(t) - \overline{x}_i(t) < \gamma] \\
\Leftrightarrow\ & [0,\tau] \cap (-\infty, t_{i,max}) \cap (-\infty, t_{j,max}) \cap (t_{i-j}, \infty) \cap (t_{j-i}, \infty) \neq \emptyset \quad \text{(Eqs. (60)-(63))} \\
&\quad [0 < t_{i,max}] \wedge [0 < t_{j,max}] \\
\Leftrightarrow &\quad \wedge [t_{j-i} < \tau] \wedge [t_{j-i} < t_{i,max}] \wedge [t_{j-i} < t_{j,max}] \\
&\quad \wedge [t_{i-j} < \tau] \wedge [t_{i-j} < t_{i,max}] \wedge [t_{i-j} < t_{j,max}] \\
&\quad [0 < t_{i,max}] \wedge [0 < t_{j,max}] \\
\Leftrightarrow &\quad \wedge [t_{j-i} < \tau] \wedge [t_{j-i} < t_{j,max}] \qquad \text{(Eqs. (64), (65))} \\
&\quad \wedge [t_{i-j} < \tau] \wedge [t_{i-j} < t_{i,max}] \\
\Leftrightarrow\ & [(50)] \wedge [(51)] \wedge [(54)] \wedge [(52)] \wedge [(55)] \wedge [(53)]
\end{aligned}$$

□

**Case 1b**: $x_i, x_j \geq 0$, $|x_i - x_j| < \gamma$.

**Proposition 6** *The set* $\{(x_i, x_j) \in \pi_{i,j}(A_{q,u_c}([0,\tau])) : x_i, x_j \geq 0 \wedge |x_i - x_j| < \gamma\}$ *is non-empty iff all of the following inequalities hold:*

$$q_i \geq -\mu\tau/2 - \overline{v}_{u_c,i}\tau \tag{68}$$

$$q_j \geq -\mu\tau/2 - \overline{v}_{u_c,j}\tau \tag{69}$$

$$\begin{aligned}&\max\{\overline{v}_{u_c,i}, \underline{v}_{u_c,j}\}(q_i + \mu\tau/2 + \tau\overline{v}_{u_c,i}) \\ &\quad -\overline{v}_{u_c,i}(q_j - \mu\tau/2 - \gamma + \tau\underline{v}_{u_c,j}) > 0\end{aligned} \tag{70}$$

$$\begin{aligned}&\max\{\overline{v}_{u_c,j}, \underline{v}_{u_c,i}\}(q_j + \mu\tau/2 + \tau\overline{v}_{u_c,j}) \\ &\quad -\overline{v}_{u_c,j}(q_i - \mu\tau/2 - \gamma + \tau\underline{v}_{u_c,i}) > 0\end{aligned} \tag{71}$$

$$\begin{aligned}&(q_i + \mu\tau/2 + \tau\max\{\underline{v}_{u_c,j}, \overline{v}_{u_c,i}\}) \\ &\quad -(q_j - \mu\tau/2 - \gamma + \tau\underline{v}_{u_c,j}) > 0\end{aligned} \tag{72}$$

$$\begin{aligned}&(q_j + \mu\tau/2 + \tau\max\{\underline{v}_{u_c,i}, \overline{v}_{u_c,j}\}) \\ &\quad -(q_i - \mu\tau/2 - \gamma + \tau\underline{v}_{u_c,i}) > 0\end{aligned} \tag{73}$$

*Proof.* The proof is similar to that of Prop. 5, and is omitted. □

**Case 2**: $[-\alpha_{r_{i,1}} < x_i < \alpha_{r_{i,2}}] \wedge [-\alpha_{r_{j,1}} < x_j < \alpha_{r_{j,2}}]$.

**Proposition 7** *The set* $\{(x_i, x_j) \in \pi_{i,j}(A_{q,u_c}([0,\tau])) : [-\alpha_{r_{i,1}} < x_i < \alpha_{r_{i,2}}] \wedge [-\alpha_{r_{j,1}} < x_j < \alpha_{r_{j,2}}]\}$ *is non-empty iff all of the following inequalities hold:*

$$q_i < \alpha_{r_{i,2}} + \mu\tau/2 \tag{74}$$

$$q_j < \alpha_{r_{j,2}} + \mu\tau/2 \tag{75}$$

$$q_i > -\alpha_{r_{i,1}} - \mu\tau/2 - \overline{v}_{u_c,i}\tau \tag{76}$$

$$q_j > -\alpha_{r_{j,1}} - \mu\tau/2 - \overline{v}_{u_c,j}\tau \tag{77}$$

$$\underline{v}_{u_c,j}(q_i + \mu\tau/2 + \alpha_{r_{i,1}}) - \overline{v}_{u_c,i}(q_j - \mu\tau/2 - \alpha_{r_{j,2}}) > 0 \tag{78}$$

$$\underline{v}_{u_c,i}(q_j + \mu\tau/2 + \alpha_{r_{j,1}}) - \overline{v}_{u_c,j}(q_i - \mu\tau/2 - \alpha_{r_{i,2}}) > 0 \tag{79}$$

*Proof.* We proceed similarly to the proof of Prop. 4. From Eqs. (39), (40) and the assumption that $v_{min} + d_{min} \geq \mu > 0$, we have that $\pi_i(A_{q,u_c}(t)) = (q_i - \mu\tau/2 + \underline{v}_{u_c,i}t, q_i + \mu\tau/2 + \overline{v}_{u_c,i}t]$ is an interval whose lower and upper bounds are increasing in time, for every $i \in \mathcal{N}$. It follows that the set $\{t \in \mathbb{R} : (-\alpha_{r_{i,1}}, \alpha_{r_{i,2}}) \cap \pi_i(A_{q,u_c}(t)) \neq \emptyset\}$ will have the form $(t^2_{i,min}, t^2_{i,max})$, where $t^2_{i,min} := \inf\{t \in \mathbb{R} : (-\alpha_{r_{i,1}}, \alpha_{r_{i,2}}) \cap \pi_i(A_{q,u_c}(t)) \neq \emptyset\}$ and $t^2_{i,max} := \sup\{t \in \mathbb{R} : (-\alpha_{r_{i,1}}, \alpha_{r_{i,2}}) \cap \pi_i(A_{q,u_c}(t)) \neq \emptyset\}$ are given by:

$$t^2_{i,min} = \frac{-q_i - \alpha_{r_{i,1}} - \mu\tau/2}{\overline{v}_{u_c,i}} \tag{80}$$

$$t^2_{i,max} = \frac{-q_i + \alpha_{r_{i,2}} + \mu\tau/2}{\underline{v}_{u_c,i}} \tag{81}$$

Now define $t^2_{j,min}$ and $t^2_{j,max}$ analogously to $t^2_{i,min}$ and $t^2_{i,max}$. Then:

$$\begin{aligned} &\exists t \in [0,\tau] \text{ s.t. } [(-\alpha_{r_{i,1}}, \alpha_{r_{i,2}}) \cap \pi_i(A_{q,u_c}(t))] \wedge [(-\alpha_{r_{j,1}}, \alpha_{r_{j,2}}) \cap \pi_j(A_{q,u_c}(t))] \\ \Leftrightarrow\ & [0,\tau] \cap (t^2_{i,min}, t^2_{i,max}) \cap (t^2_{j,min}, t^2_{j,max}) \neq \emptyset \\ \Leftrightarrow\ & \begin{array}{l} [t^2_{i,max} > 0] \wedge [t^2_{j,max} > 0] \wedge [t^2_{i,min} < \tau] \wedge [t^2_{j,min} < \tau] \\ \quad \wedge t^2_{j,max} > t^2_{i,min} \wedge t^2_{i,max} > t^2_{j,min} \end{array} \end{aligned}$$

and these last six inequalities give Eqs. (74)-(79), in order. □

**Part 2**: The Capture Set Optimization

Here we describe the capture set optimization which replaces subroutines NatVic and ContLoss in Alg. 2. The optimization is based on the observation that the bad set is convex (rectangular) for a pair of vehicles which cannot simultaneously be inside the intersection (Case 2 of Part 1). Thus it is straight-forward to compute the *capture set* of states from which no supervisor can ensure avoidance of the bad set for such a pair of vehicles. Before stating the theorem, we define the minimal and maximal velocities which can be forced by the supervisor, given that it does not control the uncontrolled vehicles or the disturbance:

$$\underline{v}^c_i = \begin{cases} v_{min} + d_{max}, & \text{vehicle } i \text{ is controlled} \\ v_{max} + d_{max}, & \text{vehicle } i \text{ is uncontrolled} \end{cases} \tag{82}$$

$$\overline{v}^c_i = \begin{cases} v_{max} + d_{min}, & \text{vehicle } i \text{ is controlled} \\ v_{min} + d_{min}, & \text{vehicle } i \text{ is uncontrolled} \end{cases} \tag{83}$$

**Proposition 8** *Given two vehicles $i$ and $j$ on different roads, there does not exist any safe and non-deadlocking supervisor $\sigma : \tilde{Q} \to 2^{U_c}$ with $\sigma(q) \neq \emptyset$, for any $q \in \tilde{Q}$ such that $\exists x \in \ell^{-1}(q)$ satisfying all of the following equations:*

$$x_i < \alpha_{r_{i,2}} \tag{84}$$

$$x_j < \alpha_{r_{j,2}} \tag{85}$$

$$\overline{v}^c_i(x_j + \alpha_{r_{j,1}}) - \underline{v}^c_j(x_i - \alpha_{r_{i,2}}) > 0 \tag{86}$$

$$\overline{v}^c_j(x_i + \alpha_{r_{i,1}}) - \underline{v}^c_i(x_j - \alpha_{r_{j,2}}) > 0 \tag{87}$$

*Proof.* First, it follows from the definitions of $\underline{v}_i^c$ and $\overline{v}_i^c$ that, for any $x$ satisfying Eqs. (84)-(87) and $u_c \in U_c$, there exists some $u_{uc} \in U_{uc}$ and $d : [0, \tau] \to D$ such that $x(t) = x + u(t/\tau) + d(t)$ either remains inside the set given by Eqs. (84)-(87) for $t \in [0, \tau]$, or enters the bad set for some $t \in [0, \tau]$ (see Fig. 9). Second, it follows from $v_{min} + d_{min} > 0$ that no control strategy can prevent the vehicles from eventually entering the set $x_i > -\alpha_{r_{i,1}} \wedge x_j > -\alpha_{r_{j,1}}$. Thus either the system eventually reaches some state $q' \in \tilde{Q}$ such that $\sigma(q') = \emptyset$, or $\sigma$ allows the system to enter the bad set. $\square$

To obtain the set of states $q$ for which $\ell^{-1}(q)$ is contained in the set of Eqs. (84)-(87), we can take this set and "deflate it" by $\mu\tau/2$, to capture the effect of the discretization. This yields the equations used in the capture set version of the NatVic subroutine in Alg. 2. Similarly, we can obtain the set of states $q$ for which there exists some $x \in \ell^{-1}(q)$ satisfying Eqs. (84)-(87) by taking this set and "inflating it" by $\mu\tau/2$. This yields the equations used in the capture set version of the ContLoss subroutine in Alg. 2.

In the former case (NatVic), the equations become

$$q_i < \alpha_{r_{i,2}} - \mu\tau/2 \tag{88}$$

$$q_j < \alpha_{r_{j,2}} - \mu\tau/2 \tag{89}$$

$$\overline{v}_i^c(q_j + \alpha_{r_{j,1}} - \mu\tau/2) - \underline{v}_j^c(q_i - \alpha_{r_{i,2}} + \mu\tau/2) > 0 \tag{90}$$

$$\overline{v}_j^c(q_i + \alpha_{r_{i,1}} - \mu\tau/2) - \underline{v}_i^c(q_j - \alpha_{r_{j,2}} + \mu\tau/2) > 0 \tag{91}$$

The latter case (ContLoss) results in one of two possibilities, depending on whether the set of Eqs. (84)-(87) is open or closed. The set will be open if $\frac{\overline{v}_j^c}{\underline{v}_i^c} \leq \frac{\underline{v}_j^c}{\overline{v}_i^c}$ and closed if $\frac{\overline{v}_j^c}{\underline{v}_i^c} > \frac{\underline{v}_j^c}{\overline{v}_i^c}$. If the set is open, the equations become:

$$q_i < \alpha_{r_{i,2}} + \mu\tau/2 \tag{92}$$

$$q_j < \alpha_{r_{j,2}} + \mu\tau/2 \tag{93}$$

$$\overline{v}_i^c(q_j + \alpha_{r_{j,1}} + \mu\tau/2) - \underline{v}_j^c(q_i - \alpha_{r_{i,2}} - \mu\tau/2) > 0 \tag{94}$$

$$\overline{v}_j^c(q_i + \alpha_{r_{i,1}} + \mu\tau/2) - \underline{v}_i^c(q_j - \alpha_{r_{j,2}} - \mu\tau/2) > 0 \tag{95}$$

If the set is closed, then two more equations must be added in general (see Fig. 9)

$$q_i > \frac{\overline{v}_i^c\overline{v}_j^c\alpha_{r_{i,1}} + \overline{v}_i^c\underline{v}_i^c\alpha_{r_{j,2}} + \overline{v}_i^c\underline{v}_i^c\alpha_{r_{j,1}} + \underline{v}_i^c\underline{v}_j^c\alpha_{r_{i,2}}}{\overline{v}_i^c\overline{v}_j^c - \underline{v}_i^c\underline{v}_j^c} - \mu\tau/2 \tag{96}$$

$$q_j > \frac{\overline{v}_i^c\overline{v}_j^c\alpha_{r_{j,1}} + \overline{v}_j^c\underline{v}_j^c\alpha_{r_{i,2}} + \overline{v}_j^c\underline{v}_j^c\alpha_{r_{i,1}} + \underline{v}_i^c\underline{v}_j^c\alpha_{r_{j,2}}}{\overline{v}_i^c\overline{v}_j^c - \underline{v}_i^c\underline{v}_j^c} - \mu\tau/2 \tag{97}$$

If $d_{min}$ and $d_{max}$ are integer multiples of $\mu$, then it can be shown these last two equations become unnecessary. We first prove a lemma.

**Lemma 3** *If $d_{min}$ and $d_{max}$ are integer multiples of $\mu$, $\frac{\overline{v}_j^c}{\underline{v}_i^c} > \frac{\underline{v}_j^c}{\overline{v}_i^c}$, and $q \in \tilde{Q}$ satisfies Eqs. (94) and (95) then, for any $u_c \in U_c$, there exists $q' \in \mathbf{Post}_{u_c}(q)$ that also satisfies Eqs. (94) and (95).*

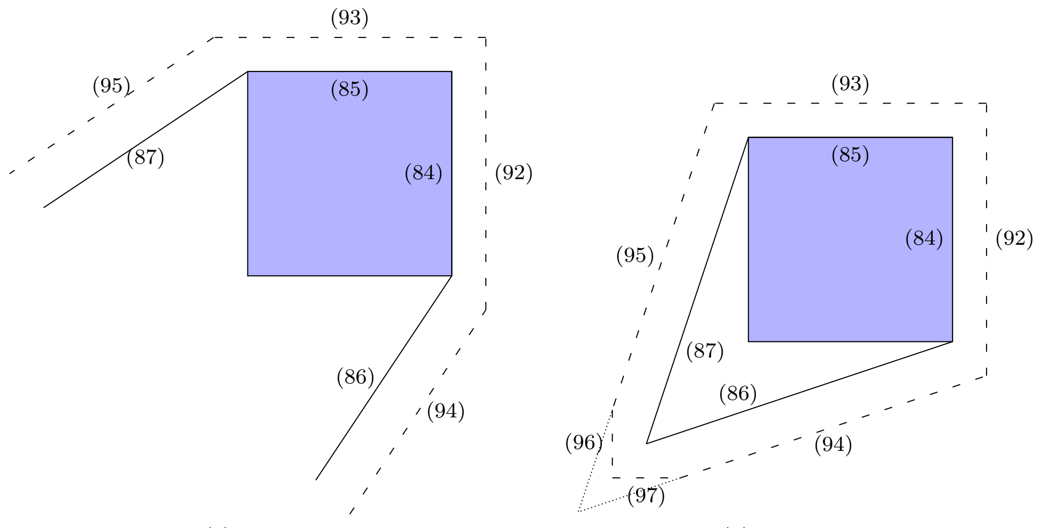
*Proof.* First note from Eqs. (82) and (83) that, if either vehicle is uncontrolled, then $\frac{\overline{v}_j^c}{\underline{v}_i^c} \leq 1$ and $\frac{\underline{v}_j^c}{\overline{v}_i^c} \geq 1$, violating $\frac{\overline{v}_j^c}{\underline{v}_i^c} > \frac{\underline{v}_j^c}{\overline{v}_i^c}$. It follows that both vehicles are controlled, and that $\overline{v}_i^c = \overline{v}_j^c > \underline{v}_i^c = \underline{v}_j^c$. We prove the following claim:
<u>Claim</u>: For any $u_c \in U_c$, there exists some $d_i \in [d_{min}, d_{max}]$ such that $u_{c,i}/\tau + d_i \in [\underline{v}_i^c, \overline{v}_i^c]$ and $u_{c,i}/\tau + d_i$ is an integer multiple of $\mu$.
It suffices to prove that, for any $u_c \in U_c$, $[\underline{v}_i^c - u_{c,i}/\tau, \overline{v}_i^c - u_{c,i}/\tau] \cap [d_{min}, d_{max}]$ contains some integral multiple of $\mu$, since we may then take such a value as $d_i$. Clearly, $u_{c,i}/\tau \in [v_{min}, v_{max}]$, from which it follows that $\underline{v}_i^c - u_{c,i}/\tau = v_{min} + d_{max} - u_{c,i}/\tau \leq d_{max}$ and that $\overline{v}_i^c - u_{c,i}/\tau = v_{max} + d_{min} - u_{c,i}/\tau \geq d_{min}$. Thus, $[\underline{v}_i^c - u_{c,i}/\tau, \overline{v}_i^c - u_{c,i}/\tau] \cap [d_{min}, d_{max}]$ is non-empty. Since it is non-empty, there must be at least one of $d_{min}$ and $\underline{v}_i^c - u_{c,i}/\tau$ in the intersection of the two sets. Since both $d_{min}$ and $\underline{v}_i^c - u_{c,i}/\tau$ are multiples of $\mu$, the claim is proven.
Constructing $d_i$ and $d_j$ as in the claim, we obtain

$$\frac{\overline{v}_j^c}{\underline{v}_i^c} \geq \frac{u_{c,j}/\tau + d_j}{u_{c,i}/\tau + d_i} \geq \frac{\underline{v}_j^c}{\overline{v}_i^c}.$$

It follows that we can take $w \in W$ such that $w_i = d_i\tau$ and $w_j = d_j\tau$, obtaining $q'$ with $q'_i = q_i + u_{c,i} + w_i$, $q'_j = q_j + u_{c,j} + w_j$ such that $q' \in \mathbf{Post}_{u_c}(q)$ satisfies Eqs. (94) and (95). □

**Corollary 1** *If $d_{min}$ and $d_{max}$ are integer multiples of $\mu$ then, given two vehicles $i$ and $j$ on different roads, there does not exist any safe and non-deadlocking supervisor $\sigma : \tilde{Q} \to 2^{U_c}$ with $\sigma(q) \neq \emptyset$, for any $q \in \tilde{Q}$ satisfying Eqs. (92)-(95) only (i.e., without satisfying Eqs. (96) and (97)), even when $\frac{\overline{v}_j^c}{\underline{v}_i^c} > \frac{\underline{v}_j^c}{\overline{v}_i^c}$.*

*Proof.* We have already shown that the result holds if $\frac{\overline{v}_j^c}{\underline{v}_i^c} \leq \frac{\underline{v}_j^c}{\overline{v}_i^c}$, or $\frac{\overline{v}_j^c}{\underline{v}_i^c} > \frac{\underline{v}_j^c}{\overline{v}_i^c}$ and $q$ satisfies Eqs. (92)-(97). It remains to be shown that the result also holds if $d_{min}$ and $d_{max}$ are integer multiples of $\mu$, $\frac{\overline{v}_j^c}{\underline{v}_i^c} > \frac{\underline{v}_j^c}{\overline{v}_i^c}$, and $q$ satisfies Eqs. (92)-(95), but not Eqs. (96) and (97). Consider any $u_c \in U_c$. By Lemma 3, there exists $q' \in \mathbf{Post}_{u_c}(q)$ that also satisfies Eqs. (94) and (95). There are now three cases to consider:
<u>Case 1</u>: $q'$ satisfies Eqs. (92)-(97).
We have shown in this case there exists no safe and non-deadlocking supervisor from $q'$.
<u>Case 2</u>: $q'$ satisfies Eqs. (92)-(95), but not both of Eqs. (96) and (97).
Because $d_{min} + v_{min} > 0$, Lemma 3 can be applied repeatedly, until a $q'$ is obtained which satisfies Eqs. (96) and (97).
<u>Case 3</u>: $q'$ does not satisfy both of Eqs. (92) and (93).
In this case, the line segment from $q$ to $q'$ either crosses the bad set, or comes



(a) Open Case (b) Closed Case

**Fig. 9** The capture sets of Eqs. (84)-(97) in the open (left) and closed (right) cases. The blue square denotes the bad set. The set of Eqs. (84)-(87) is depicted with solid lines, and its inflation by $\mu\tau/2$ is depicted in dashed lines. Right: If $d_{min}$ and $d_{max}$ are integer multiples of $\mu$, then Eqs. (96) and (97) are unnecessary, which is shown by the dotted lines.

within a distance of $\mu\tau/2$ of it (see Fig. 9). In the latter case, we can find some pair $x \in \ell^{-1}(q)$ and $x' \in \ell^{-1}(q')$ such that the line segment from $x$ to $x'$ crosses the bad set. $\square$

Figure 9 depicts the set described by Eqs. (84)-(87) of Prop. 8, the inflated set of Eqs. (92)-(97), and the special case of Cor. 1. The simulations of Sec. 9 satisfied the property that $d_{\min}$ and $d_{\max}$ were integer multiples of $\mu$, and hence the code used Eqs. (92)-(95) only.