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

Eric Dallal, Alessandro Colombo, Domitilla Del Vecchio and Stéphane Lafortune

***Abstract*—We consider the problem of collision avoidance at vehicular intersections for a set of controlled and uncontrolled vehicles that are linked by wireless communication. Each vehicle is modeled by a first order system. We use a disturbance to account for bounded model uncertainty. We construct a discrete event system abstraction and formulate the problem in the context of supervisory control for discrete event systems with uncontrollable events. This allows us to mitigate computational limitations related to the presence of continuous dynamics and infinite state spaces. For solving the resulting supervisory control problem at the discrete event level, we develop an algorithm that exploits the structure of the transition map to compute the supremal controllable sublanguage more efficiently than standard algorithms. We present implementation results on an intersection with several vehicles.**

I. INTRODUCTION

Vehicle collisions cause, on average, 4156 injuries and 84 deaths per day in the United States [1]. About a quarter of all reported light vehicle fatalities are due to side impacts, suggesting collisions at traffic intersections and merges [2]. A side impact avoidance system at traffic intersections must deal in real time with multiple vehicles, uncontrolled vehicles, and model uncertainty.

In this paper, the collision avoidance problem is formulated in the framework of the supervisory control theory of discrete event systems [3]. The computational complexity is tamed by reducing the continuous dynamics of a multi-vehicle system to a finite representation, namely, a discrete event system (DES). This approach is commonly known as *abstraction*. Then, the control map is synthesized based on the set of allowed transitions of the DES. Abstraction-based control schemes were proposed in [4], [5], [6], [7] for incrementally stable systems, and extended in [8] to general systems, using non-deterministic DES. In [9], abstraction techniques for safety enforcement were discussed in the context of reachability analysis. In [10], the problem of robot control is considered by applying an abstraction based on triangularization and designing a low-level (continuous) control to satisfy path specifications expressed in linear temporal logic (LTL). In [11], the dynamic properties of common mechanical systems (including models of vehicle dynamics) were exploited to obtain a safety-enforcing supervisory control based on a *deterministic* abstraction, irrespective of the stability properties of the dynamics. An alternative solution, based on an equivalence relation between the collision avoidance problem and a scheduling problem, is discussed in [12]. These results are, however, limited to the case of perfectly known models, and do not address the presence of uncontrolled agents. A different approach to the collision avoidance problem is proposed in [13], based on a centralized scheduling of the intersection crossing times of all vehicles.

We directly exploit the simple structure of first order vehicle dynamics with model uncertainty to construct a finite deterministic DES abstraction. In contrast to [11], our results deal with model uncertainty and handle the presence of uncontrolled vehicles. In particular, we introduce a deterministic DES that *simulates* the original continuous system, and such that the continuous system *alternatingly simulates* the DES (see, e.g., [5] for the definitions of similarity and alternating similarity). The actions of uncontrolled vehicles are modeled naturally as uncontrollable events. Modeling uncertainty is handled by adding suitable uncontrollable transitions to the DES model. We prove that safety at the continuous level is implied by a notion of safety at the discrete event level. We then pose the desired collision avoidance problem in the framework of the theory of supervisory control of DES [3]. The problem has three requirements: (R1) *safety*, i.e., vehicular collisions must be avoided; (R2) *non-blockingness*, i.e., vehicles should not deadlock and must reach their final destinations, which in this case means they must completely cross the intersection; and (R3) *maximal permissiveness*, i.e., the supervisory control actions should leave as much autonomy as possible to the individual vehicles. In the presence of uncontrollable events, the solution is obtained by computing the *supremal controllable sublanguage* [14] of the specification language with respect to the system language and the set of uncontrollable events.

The resulting supervisory control problem to solve at the discrete event level is the well-known "Basic Supervisory Control Problem - Nonblocking case", or BSCP-NB [15]. Its solution provably satisfies requirements (R1)-(R3). Instead of using standard techniques for computing the supremal controllable sublanguage, the key step in solving BSCP-NB, we develop new algorithmic techniques that are customized to the specific application under consideration and thereby achieve greater computational efficiency. Our approach has conceptual similarities with the "variable lookahead" technique of [16] but differs in two respects. First, we do not do limited lookahead but we perform a *depth-first search* over the entire state space. This is required as eventually, we want to find the optimal control action from every (safe) state in the state space. Second, in our approach, we deal with the uncontrollability resulting from unmodeled dynamics and uncontrolled vehicles; specifically, we exploit *structural properties* of the transition structure of

Research supported in part by NSF grant CNS-0930081.

E. Dallal and S. Lafortune are with the EECS Dept. at the University of Michigan, MI, USA [edallal,stephane]@umich.edu

A. Colombo is with the DEIB at Politecnico di Milano, Italy alessandro.colombo@polimi.it

D. Del Vecchio is with the ME Dept. at MIT, MA, USA ddv@mit.edu

the underlying automaton obtained from the abstraction step. We present implementation results that illustrate the computational efficiency of our approach. Finally, we note that the approach that we follow has the advantage of leaving complete freedom to the driver over the timing of the intersection crossing, thus reducing the control action to the minimum required to enforce safety; this follows from the maximal-permissiveness of the supremal controllable sublanguage.

The simulation-alternating simulation structure that we exploit to construct our abstraction is analogous to the one obtained in [8]. Our results, however, apply to systems with model uncertainty. Furthermore, our focus here is on determining the largest subset of transitions of the DES that preserves requirements (R1) and (R2), something that is not addressed in [8].

The paper is organized as follows. In Section II, we describe the continuous model and define the problems we will solve. In Section III, we construct the DES abstraction. Section IV presents our customized algorithm for computing the supervisory controller at the discrete event level. Performance results from an implementation of this algorithm are presented in Section V, while Section VI concludes the paper. Due to space limitations, several proofs are not presented here; they are available from the authors.

II. MODEL AND PROBLEM FORMULATION

Notation: In the text, the symbol $\|\cdot\|$ denotes the infinity norm of a vector, a subscripted index (e.g., x_i) indicates an element of a vector, and a superscripted index (e.g., x^i) indicates a vector out of a set of vectors. The symbols $\lfloor t \rfloor$ and $\lceil t \rceil$ denote the greatest integer less than or equal to t and the smallest integer greater than or equal to t, respectively.

Consider a set $\mathcal{N} = \{1, \ldots, n\}$ of vehicles, where $n = |\mathcal{N}|$, moving along p roads, $p \leq n$, that intersect at a unique point. The vehicles are modeled as single integrators and their collective dynamics are described by the system

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

where $x \in X \subset \mathbb{R}^n$ is the state, with X compact, $v \in V \subset \mathbb{R}^n$ is the control input, and $d \in D \subset \mathbb{R}^n$ is a disturbance input representing unmodeled dynamics. Assume that $v \in V$ is a vector with elements in the finite set $\{\mu a, \mu(a+1), \ldots, \mu b\}$, with $a, b \in \mathbb{N}$ and $\mu \in \mathbb{R}_+$, and that $d \in D = [d_{min}, d_{max}]^n$, with the vector $[0, \ldots, 0] \in [d_{min}, d_{max}]^n$. We refer to $a\mu$ and $b\mu$ as v_{min} and v_{max}, respectively. We allow the possibility that a subset of the vehicles cannot be controlled. To represent this, we partition the vector v into two subvectors, $v_c \in V_c$ and $v_{uc} \in V_{uc}$, where v_c represents the control inputs of the controlled vehicles, whereas v_{uc} represents the control inputs of the uncontrollable vehicles, such that $v = (v_c, v_{uc})$ and $V = V_c \times V_{uc}$. Assume also that $v_{min} + d_{min} \geq \mu$, so that μ constitutes a lower bound on the velocity of the vehicles. Finally, assume that the input v is kept constant over time intervals $[k\tau, (k+1)\tau]$, and discretize the above system in time, with step τ, 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 $u \in U$ and $\delta \in \Delta$. As for the set V, we write $U = U_c \times U_{uc}$, where U_c is the set of available actions for the controllable vehicles and U_{uc} is the set of actions of the uncontrollable vehicles. We use the notation $u = (u_c, u_{uc})$ to denote the actions of the controllable and uncontrollable vehicles for any vector $u \in U$.

Define a grid with hypercubic cells of side $\tau\mu$ covering X, and consider a regular lattice $\tilde{Q}$ of step $\tau\mu$ such that an element q of the lattice lies in the centre of each hypercubic cell. Since both $q \in \tilde{Q}$ and $x \in X$ are elements of $\mathbb{R}^n$, the infinity norm defines a distance for any pair (q, x). Take the function

$$\ell(x) := \min_{q \in Q} \{q : \|x - q\| \leq \tau\mu/2\}, \tag{3}$$

where $\min$ is taken in the lexicographical order. Assume that the lattice $\tilde{Q}$ is such that, for all $q \in \tilde{Q}$, there exists an $x \in X$ such that $\ell(x) = q$. Let W be a set of symbols with cardinality $(\lceil d_{max}/\mu \rceil - \lfloor d_{min}/\mu \rfloor + 1)^n$. Now, for each pair (q, u), define the set of reachable states

$$Q_{q,u} := \{q' : \exists x \in X, \delta \in \Delta : \ell(x) = q, \ell(x + u + \delta) = q'\},$$

and given a subset $W_{q,u}$ of W with the cardinality of $Q_{q,u}$, define a bijective assignment

$$W_{q,u} \overset{q,u}{\Leftrightarrow} Q_{q,u}.$$

Finally, define the DES

$$\tilde{G} := (\tilde{Q}, U \times W, \psi, Q_0) \tag{4}$$

with state set $\tilde{Q}$, events set $U \times W$, and transition function $\psi(q, u, w)$ defined as follows:

$$\psi(q, u, w) = q' \text{ iff } w \in W_{q,u}, \text{ and } w \overset{q,u}{\Leftrightarrow} q'. \tag{5}$$

Hereafter, we write $q \notin \tilde{Q}$ whenever, for all x such that $\ell(x) = q$, all vehicles have crossed the intersection in state x. The set Q_0 is the initial state set, assumed hereafter to be any subset of $\tilde{Q}$. In the construction above, a transition is described by a pair of events, u and w. Events u of the DES are in one to one relation with the possible control inputs of the continuous system, whereas events w account for the difference between the expected reached state and the state that is effectively reached, due to the effect of the disturbance input. We denote by $\mathcal{L}(\tilde{G}, q)$ the *language* of G starting from state q, that is, the set of all possible strings of events $u^1 w^1 u^2 w^2 ...$ that can occur starting from initial state $q \in Q_0$. The symbol s is used to denote a generic string. Also, given a state $q \in \tilde{Q}$, we denote by (q, u, w) a transition of (4) from state q with event u, w, and by (q, s) an execution of (4) starting from initial state q, with events string $s = u^1 w^1 u^2 w^2 ....$ We write $\psi(q, s)$ to denote the last state reached by an execution. Since each vector u is itself composed of a subvector of controllable inputs u_c and a subvector of uncontrollable inputs u_{uc}, a string of events s can also be a sequence of the form $s = u_c^1 u_{uc}^1 w^1 u_c^2 u_{uc}^2 w^2 \ldots$.

Let us now endow system (2) with the observation map $H_x(x) := \ell(x)$, and $\tilde{G}$ with the observation map $H_q(q) := q$. With this observation map, we can define the notions of

simulation and alternating simulation, whose definitions have been adapted from [5] for our context.

Definition II.1. *We say that $\tilde{G}$ simulates (2) if*

(i) *for all $x \in X$ there exists a $q \in \tilde{Q}$ such that $q = \ell(x)$,*
(ii) *for all $x \in X$ and $q \in \tilde{Q}$ such that $q = \ell(x)$, $H_x(x) = H_q(q)$,*
(iii) *for all $x \in X$ and $q \in \tilde{Q}$ such that $q = \ell(x)$, if there exists $(u^1, \delta) \in U \times \Delta$ such that $x + u^1 + \delta = x'$, then there exists $(u^2, w) \in U \times W$ such that $\psi(q, u^2, w) = \ell(x')$.*

Definition II.2. *We say that (2) alternatingly simulates $\tilde{G}$ if*

(i) *for all $q \in \tilde{Q}$ there exists an $x \in X$ such that $q = \ell(x)$,*
(ii) *for all $x \in X$ and $q \in \tilde{Q}$ such that $q = \ell(x)$, $H_x(x) = H_q(q)$,*
(iii) *for all $x \in X$ and $q \in \tilde{Q}$ such that $q = \ell(x)$, for all $u_c^1 \in U_c$ there exists $u_c^2 \in U_c$ such that for all $u_{uc}^2 \in U_{uc}$, $\delta \in \Delta$ and $q' = \ell(x + u^2 + \delta)$ there exists $u_{uc}^1 \in U_{uc}$ and $w \in W_{q,u^1}$ satisfying $\psi(q, u^1, w) = q'$, where $u^1 = (u_c^1, u_{uc}^1)$ and $u^2 = (u_c^2, u_{uc}^2)$.*

Ignoring the fact the state spaces of (2) and $\tilde{G}$ are different, the intuition behind alternating similarity is that, for any control decision u_c^1 of system $\tilde{G}$, there exists a control decision u_c^2 of system (2) such that the set of reachable states in the latter case is contained in the set of reachable states in the former case. This is useful for a safety objective where the goal is to avoid a set of states. In such a case, the safety of u_c^1 in system $\tilde{G}$ implies the safety of u_c^2 in system (2).

Given the above construction and the definitions of similarity and alternating similarity, we can prove the following.

Lemma II.1. *$\tilde{G}$ simulates (2) and (2) alternatingly simulates $\tilde{G}$.*

Proof: ($\tilde{G}$ simulates (2)): In Definition II.1, (i) is true because the grid defining the lattice $\tilde{Q}$ covers X; (ii) follows from the definition of the observation maps; (iii) is a consequence of the definition of transitions in (5) by taking $u^1 = u^2$.

((2) alternatingly simulates $\tilde{G}$): In Definition II.2, (i) is true by assumption; (ii) follows from the definition of the observation maps; (iii) follows by taking $u^1 = u^2 = u$ and $w = q' - q - u$, which is guaranteed to exist by virtue of the transition structure of equation (5). ∎

Define a set Π_k for each road, and say that vehicle $i \in \Pi_k$ if vehicle i drives along road k. Describe the length of the portion of each road that belongs to the intersection as an interval $[\alpha_k, \beta_k] \subset \mathbb{R}$, and define a safety distance $\gamma \in \mathbb{R}_+$, common to all vehicles. We say that two vehicles $i \in \Pi_k$, $j \in \Pi_l$ with $k \neq l$ undergo a collision whenever $x_i \in [\alpha_k, \beta_k]$ and $x_j \in [\alpha_l, \beta_l]$ simultaneously. Similarly, we say that two vehicles $i, j \in \Pi_k$ undergo a collision whenever $|x_i - x_j| < \gamma$. The subset of X of all collision points is called the *bad set* B. A trajectory $x(t)$ of (1) is *ϵ-safe* provided

$$\inf_{t \geq 0, b \in B} \|x(t) - b\| \geq \epsilon.$$

Let X/ℓ denote the quotient set of X by the equivalence classes induced by ℓ. We aim to design a supervisor $\sigma : X/\ell \to 2^{V_c}$ for (1) that enforces 0-safety, where $V_c = U_c/\tau$. More precisely, we aim to solve the following problem.

Problem II.1. *Given X/ℓ, define a supervisor that associates 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 $x(t)$ is 0-safe in the same time interval (0-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 $x((k+1)\tau) \in X$, then $\sigma(x((k+1)\tau)) \neq \emptyset$ (non-blockingness)*
- *if $\tilde{\sigma} \neq \sigma$ and $\tilde{\sigma}$ satisfies the two properties above, then $\tilde{\sigma}(x(k\tau)) \subseteq \sigma(x(k\tau))$ for all $k \geq 0$ (maximal permissiveness).*

By reducing (2) to a finite abstraction, we can reduce Problem II.1 to the problem of selecting a particular set T of executions of $\tilde{G}$. Given such a set T, construct a supervisor map for each time interval $[k\tau, (k+1)\tau]$ as follows

$$\sigma_T(x(k\tau)) := \left\{ \begin{array}{l} v_c \in V_c : \exists (\ell(x(k\tau)), u_c^1 u_{uc}^1 w^1 u_c^2 u_{uc}^2 w^2 ...) \in T \\ \text{with } u_c^1 = \tau v_c \end{array} \right\}. \quad (6)$$

Intuitively, we would like T to be selected as the set of executions that should be allowed by the system in order to preclude only unsafe behavior.

Definition II.3. *A set T of executions is suffix-closed if $(q^1, u^1 w^1 s) \in T$ implies that $(\psi(q^1, u^1, w^1), s) \in T$.*

Definition II.4. *A set T of executions is non-escaping if $(q^1, u_c^1 u_{uc}^1 w^1 s) \in T$ implies that, for all $u_{uc}^{1\prime} \in U_{uc}$ and $w^{1\prime} \in W_{q^1,u^1}$, $\exists s'$ such that $(q^1, u_c^1 u_{uc}^{1\prime} w^{1\prime} s') \in T$.*

The non-escaping property implies that the presence or absence of an execution from T should depend only on the controllable events u_c. Otherwise, the supervisor σ_T could potentially allow executions that are not in the set T, due to uncontrollable events u_{uc} or w. This property is the analogue of the controllability property of DES [15].

We introduce the following definitions of safety for $\tilde{G}$.

Definition II.5. *Given functions $\epsilon_1 : [0, \tau] \to \mathbb{R}^n$ and $\epsilon_2 : [0, \tau] \to \mathbb{R}^n$, a transition (q, u, w) such that $\psi(q, u, w) = q'$ is (ϵ_1, ϵ_2)-safe if $\nexists b \in B : \epsilon_1(t) - \boldsymbol{\mu}\tau/2 < b - [q + t(q' - q)/\tau] \leq \epsilon_2(t) + \boldsymbol{\mu}\tau/2$, where $\boldsymbol{\mu} = \{\mu, \ldots, \mu\} \in \mathbb{R}^n$ and the inequalities are taken component-wise. An execution (q, s) is (ϵ_1, ϵ_2)-safe if all the transitions that compose it are (ϵ_1, ϵ_2)-safe.*

The above definition captures the idea that the interpolated trajectory from q to q' (given by $q + t(q' - q)/\tau$) must be a certain (potentially time-dependent) distance away from any point in the bad set. Let $\underline{\epsilon}$ and $\bar{\epsilon}$ be defined as follows:

$$\underline{\epsilon}_{q,u,w}(t) = \begin{cases} [\underline{v}_u - (q'-q)/\tau]\, t, & t \le \underline{t}_{q,u,w} \\ [(q'-q)/\tau - \overline{v}_u]\,(\tau - t), & t \ge \underline{t}_{q,u,w} \end{cases} \quad (7)$$

$$\underline{t}_{q,u,w} = \frac{\tau \overline{v}_u - q' + q}{\overline{v}_u - \underline{v}_u} \quad (8)$$

$$\overline{\epsilon}_{q,u,w}(t) = \begin{cases} [\overline{v}_u - (q'-q)/\tau]\, t, & t \le \overline{t}_{q,u,w} \\ [(q'-q)/\tau - \underline{v}_u]\,(\tau - t), & t \ge \overline{t}_{q,u,w} \end{cases} \quad (9)$$

$$\overline{t}_{q,u,w} = \frac{q' - q - \tau \underline{v}_u}{\overline{v}_u - \underline{v}_u} \quad (10)$$

$$\underline{v}_u = u/\tau + \boldsymbol{d}_{min} \quad (11)$$

$$\overline{v}_u = u/\tau + \boldsymbol{d}_{max} \quad (12)$$

where $\boldsymbol{d}_{min} = (d_{min}, \ldots, d_{min}) \in \mathbb{R}^n$ and similarly for $\boldsymbol{d}_{max}$.

We have shown that for these definitions of $\underline{\epsilon}$ and $\overline{\epsilon}$ (which depend on q, u, and w), $(\underline{\epsilon}, \overline{\epsilon})$-safety of a transition implies 0-safety of the corresponding trajectory $x(k\tau + t)$ for $t \in [0, \tau]$ when $\ell(x(k\tau)) = q$, DE control decision u_c is issued at time $k\tau$, the uncontrolled vehicles take action u_{uc}, and the disturbance is w.

Lemma II.2. *If T is a set of suffix-closed and $(\underline{\epsilon}, \overline{\epsilon})$-safe executions, and is non-escaping, then σ_T in (6) enforces 0-safety.*

Then, let us define forward-maximal executions as follows.

Definition II.6. *An execution $(q, s) = (q, u_c^1, u_{uc}^1, w^1, .., u_c^m, u_{uc}^m, w^m)$ is forward-maximal if $\psi(q, u_c^1, u_{uc}^1, w^1, .., u_c^n, u_{uc}^n, w^n) \in \tilde{Q}$ for all $n < m$, and $\psi(q, u_c^1, u_{uc}^1, w^1, .., u_c^m, u_{uc}^m, w^m) \notin \tilde{Q}$.*

This definition ensures that we allow only executions that eventually reach some goal (e.g., crossing the intersection). By this definition and the fact that (2) alternatingly simulates $\tilde{G}$, we can prove the next result.

Lemma II.3. *If T is a set of forward-maximal executions, suffix-closed and non-escaping, then σ_T in (6) is non-blocking.*

From the above results, we can conclude that

Theorem II.4. *If T is selected as the largest set of $(\underline{\epsilon}, \overline{\epsilon})$-safe, suffix-closed, non-escaping and forward-maximal executions of (4), then the supervisor (6) solves Problem II.1.*

Proof: 0-safety and nonblockingness follow from Lemmas II.2 and II.3 and maximal permissiveness is ensured by taking T as the largest set satisfying all required conditions. ■

In the following sections, we present an algorithm to construct T in order to enforce safety in the case of unmodeled dynamics and in the presence of uncontrolled vehicles. Specifically, we propose an algorithm to construct sets of executions T to solve the three following problems:

Problem II.2. *Determine the largest set T of $(\underline{\epsilon}, \overline{\epsilon})$-safe suffix-closed, non-escaping and forward-maximal executions when $d_{min} = d_{max} = 0$ and vehicles $k+1, ..., n$ are uncontrolled.*

Problem II.3. *Determine the largest set T of $(\underline{\epsilon}, \overline{\epsilon})$-safe suffix-closed, non-escaping and forward-maximal executions when $d_{min}, d_{max} \neq 0$ and all vehicles are controlled.*

Problem II.4. *Determine the largest set T of $(\underline{\epsilon}, \overline{\epsilon})$-safe suffix-closed, non-escaping and forward-maximal executions when $d_{min}, d_{max} \neq 0$ and vehicles $k+1, ..., n$ are uncontrolled.*

III. DISCRETE EVENT SYSTEM PROBLEM FORMULATION

By fixing the initial state of the automaton to be $q \in Q_0$, the resulting language, denoted by $\mathcal{L}(\tilde{G}, q)$, is in a one-to-one relation with the set of all possible executions (q, s), that have q as initial condition[1]. In order to simulate multiple possible initial states, we introduce a "dummy" state with no physical meaning, denoted by q^0, and for each $q \in Q_0$, we create a transition with label e_q from q^0 to q. We denote by E_Q the set of all such events e_q: $E_Q = \{e_q : q \in Q_0\}$. Mathematically, we define the transition function ψ acting on state q^0 by $\psi(q^0, e_q) = q$ for any $e_q \in E_Q$.

To represent the states reached by (4) when a transition *leaves* $\tilde{Q}$, we define the set of *marked* (in standard DES terminology) states Q_m

$$Q_m := \left\{ q' : q' \notin \tilde{Q} \text{ and } \exists q \in \tilde{Q}, (u, w) \in U \times W, \psi(q, u, w) = q' \right\}$$

The set Q_m satisfies $\tilde{Q} \cap Q_m = \emptyset$ and allows us to translate Definition II.6 into a *non-blocking* condition on the DES. Intuitively, a forward-maximal execution is an execution that extends forward in time as much as possible in $\tilde{Q}$. By definition of Q_m it is clear that $\psi(q, s) \in Q_m$ whenever $\psi(q, s)$ is forward-maximal. As previously stated, we write $U = U_c \times U_{uc}$, where U_c consists of the discrete-event (DE) controlled input and U_{uc} consists of the DE uncontrolled input. Events in the set U_c correspond to controllable events and events in the sets U_{uc} or W correspond to *uncontrollable* events. Since each event must take us to a new state, we define sets of intermediate states Q_{I1} and Q_{I2} along with intermediate transition functions $\psi_1 : \tilde{Q} \times U_c \to Q_{I1}$, $\psi_2 : Q_{I1} \times U_{uc} \to Q_{I2}$, and $\psi_3 : Q_{I2} \times W \to \tilde{Q} \cup Q_m$. These are defined only by $\psi(q, u_c, u_{uc}, w) = \psi(q, (u_c, u_{uc}), w) = \psi_3(\psi_2(\psi_1(q, u_c), u_{uc}), w)$. With these additions, we redefine G as the complete DES

$$G := (Q, U_c \times U_{uc} \times W \cup E_Q, \psi, q^0, Q_m) \quad (13)$$

where $Q = \{q^0\} \cup \tilde{Q} \cup Q_m \cup Q_{I1} \cup Q_{I2}$. Note that G has a single initial state, the dummy state q^0. Note also that the set of events E_Q are considered controllable. Finally, note that Q_{I1} and Q_{I2} are disjoint from $\tilde{Q}$. This is done in order to enforce the strict alternation of controllable and uncontrollable events. Mathematically, the language $\mathcal{L}(G) \subseteq E_Q(U_c U_{uc} W)^*$.

The first step in computing the desired set T is finding the set $\hat{T}$ of all executions (not necessarily forward-maximal) that are $(\underline{\epsilon}, \overline{\epsilon})$-safe executions of system (13). From $\hat{T}$, we obtain the language L_a, called the safety specification, by mapping

[1] We use standard DES notations, as in [3], [15].

executions to strings (an execution (q, s) maps to the string $e_q s$ and vice-versa). Note that $L_a \subseteq \mathcal{L}(G)$ and is prefix-closed. Next, we define the set L_{am} of marked strings as the subset of L_a corresponding to forward-maximal executions. From the way we have defined Q_m, we see that $L_{am} = L_a \cap \mathcal{L}_m(G)$. As seen above, dealing with problems II.2, II.3 and II.4 results in the introduction of uncontrollable events. This means that it is possible for there to exist some pair of strings s^1 and s^2 such that $s^1 \in L_a$ and $s^2 = s^1 e \notin L_a$, for some uncontrollable event $e \in U_{uc}$ or $e \in W$. This is called a *control conflict* since we wish to allow s^1 but allowing it makes it impossible to prevent s^2, which we do not wish to allow. The solution to this problem is to "shrink" the language to the largest possible subset that has no control conflicts. This problem always has a unique solution, which is called the supremal controllable sublanguage and is denoted by the operation $\uparrow C$ on languages [15]. The safety specification in our problem is completely expressible in terms of safe states and/or safe transitions over G at the discrete event level. Therefore, L_a and L_{am} are generated and marked, respectively, by a subautomaton of G. In this case, without loss of generality, we can define the domain of the DE supervisor to be the state set of G, Q, instead of the domain $\mathcal{L}(G)$ used in the development of the results in supervisory control theory. Since we also require the solution to be non-blocking, we have to solve at the discrete event level the basic supervisory control problem in the non-blocking case (BSCP-NB), as defined in [15]:

Problem III.1. *Given system G with event set E, uncontrollable event set $E_{uc} \subseteq E$, and admissible marked language $L_{am} \subseteq \mathcal{L}_m(G)$, find a non-blocking DE-supervisor $S : Q \to 2^U$ (that chooses which events to allow in each state) such that:*

1) $\mathcal{L}_m(S/G) \subseteq L_{am}$
2) $\mathcal{L}_m(S/G)$ *is "the largest it can be", that is, for any other non-blocking DE-supervisor S_{other} such that $\mathcal{L}_m(S_{other}/G) \subseteq L_{am}$, $\mathcal{L}_m(S_{other}/G) \subseteq \mathcal{L}_m(S/G)$.*

In the above, $\mathcal{L}(S/G)$ is defined to be the set of strings $s \in \mathcal{L}(G)$ that are allowed by the DE-supervisor S and $\mathcal{L}_m(S/G)$ is the subset of those strings that are marked. A unique solution exists when L_{am} satisfies $L_{am} = \overline{L_{am}} \cap \mathcal{L}_m(G)$, the so-called $\mathcal{L}_m(G)$-closure condition [15]. This is always the case when $L_{am} = L_a \cap \mathcal{L}_m(G)$, as it is here. The unique solution in this case is $\mathcal{L}_m(S/G) = L_{am}^{\uparrow C}$ and $\mathcal{L}(S/G) = \overline{L_{am}^{\uparrow C}}$. Finally, the set T consist of those executions that correspond to strings in $L_{am}^{\uparrow C}$. The $(\underline{\epsilon}, \overline{\epsilon})$-safety of T can be ensured by properly defining safety of the transitions of G and its suffix-closure is guaranteed by the structure of languages. Non-escapingness of T is ensured by the fact that $L_{am}^{\uparrow C}$ will be free of control conflicts and forward-maximality is ensured by the non-blocking property of $L_{am}^{\uparrow C}$. Finally, T is the largest set satisfying the above properties because $L_{am}^{\uparrow C}$ is the *supremal* controllable sublanguage of L_{am}. The supervisor σ can be computed from S as follows:

$$\sigma(x(k\tau)) = \{u_c/\tau : u_c \in S(\ell(x(k\tau)))\} \tag{14}$$

IV. CONTROL OF THE DISCRETE EVENT SYSTEM

In principle, the solution to BSCP-NB can be computed using the standard iterative algorithm for computing $\overline{L_{am}^{\uparrow C}}$, or the linear-time version that applies to livelock-free systems (DES G is livelock-free if every cycle in G contains a marked state), such as the one in [16]. We chose to develop our own customized algorithm because the special structure of our problem allows us to do even better in the presence of a disturbance or of uncontrollable vehicles. Specifically, it can be shown that, to verify the safety of DE control decision u_c from DE state q, we do not need to check safety of every transition of the form $(q, \psi(q, u_c, u_{uc}, w))$. Instead, we can use a single test with a running time of $O(n^2)$ by computing the set of positions which the vehicles could be in at time $t \in [k\tau, (k+1)\tau]$ when DE control decision u_c is made from DE state q at time $k\tau$. In this section, we provide theorems that establish the basis for our algorithmic solutions.

Definition IV.1. *The set of reachable states from state q given DE control decision u_c is defined by $R(q, u_c) := \{\psi(q, u_c, u_{uc}, w) : u_{uc} \in U_{uc} \wedge w \in W\}$.*

Definition IV.2. *Given $t \in [0, \tau]$, let the set $A_{q,u,w}(t) \subseteq X$ denote the set of points x_d such that it is possible for $x(k\tau + t) = x_d$ when $\ell(x(k\tau)) = q$, DE control decision u_c is issued at time $k\tau$, the uncontrolled vehicles take action u_{uc}, and the disturbance is w.*

From the above definition, we can see that a transition from q to q' is 0-safe in the continuous domain if $A_{q,u,w}(t)$ does not cross the bad set for $t \in [0, \tau]$.

Definition IV.3. *Given $t \in [0, \tau]$, let the set $A_{q,u_c}(t) \subseteq X$ denote the set of points x_d such that it is possible for $x(k\tau + t) = x_d$ when $\ell(x(k\tau)) = q$ and DE control decision u_c is issued at time $k\tau$.*

Clearly, $A_{q,u_c}(t) = (q - \boldsymbol{\mu}\tau/2 + \underline{v}_{u_c} t, q + \boldsymbol{\mu}\tau/2 + \overline{v}_{u_c} t]$, where $\underline{v}_{u_c}$ and $\overline{v}_{u_c}$ are as given in equations (11) and (12) for the controlled vehicles and are equal to $v_{min} + d_{min}$ and $v_{max} + d_{max}$, respectively, for the uncontrolled vehicles. Thus, $A_{q,u_c}(t)$ is the analogue of $A_{q,u,w}(t)$ when we do not fix u_{uc} and w. By the above two definitions, it must be that $A_{q,u_c}(t) = \bigcup_{u_{uc} \in U_{uc}} \bigcup_{w \in W} A_{q,u,w}(t)$.

The set $A_{q,u,w}(t)$ has a non-trivial shape and it is not simple to check whether it crosses the bad set. Also of importance is the fact that $A_{q,u,w}(t)$ does indeed have a dependence on u so that the safety of an individual transition is not uniquely determined by its start and end state. Instead, we can determine if DE control decision u_c is safe from DE state q by checking if the set $A_{q,u_c}(t)$ crosses the bad set for some $t \in [0, \tau]$, which can be done in $O(n^2)$ time. The idea is to verify intersection for each pair of vehicles, since the bad set reduces to a rectangle in this case. Thus, rather than determining the safety of each of the $|U_{uc}||W|$ transitions that could result from DE control decision $u_c \in U_c$, it suffices to make a single safety test for each control decision. Hence we make $|U_c|$ safety tests instead of $|U_c||U_{uc}||W|$ safety tests.

Theorem IV.1 (Basis for Depth-First Search)**.** *Let S be the minimally restrictive non-blocking DE-supervisor. Then S must satisfy the following, for all $q \in Q$, $q \neq q^0$:*

$$u_c \in S(q) \Leftrightarrow \begin{array}{c} \textit{SafeDECon}(G, q, u_c) \textit{ and} \\ \forall q' \in R(q, u_c), \quad [q' \in Q_m \vee S(q') \neq \emptyset], \end{array} \tag{15}$$

where SafeDECon(G, q, u_c) is true if and only if the set $A_{q,u_c}(t)$ does not cross the bad set for any $t \in [0, \tau]$.

The above theorem suggests that computing the set $S(q)$ for some $q \in Q$ can be done by checking, for each $u_c \in U_c$, whether taking DE control decision u_c from DE state q could result in the bad set being crossed during the following interval of time τ and, for each $q' \in R(q, u_c)$, whether q' is either terminal or non-deadlocked. Since checking whether q' is non-deadlocked requires determining whether $S(q') = \emptyset$, this can be accomplished through recursion. This forms the basis for the use of depth-first seach to compute the DE-supervisor S.

A. The algorithm

Algorithm 1 shows pseudo-code for the algorithm which computes the DE-supervisor S defined in Section III and hence solves each of the four problems of Section II. The subroutine "Terminal(G, q)" checks if all the vehicles have crossed the finish line (if $q \in Q_m$). The variable "Done(q)" is true if and only if the algorithm has already been called with position q. The variable Safe(q) is true if q was determined to be safe, which means that $S(q) \neq \emptyset$. The variable Safe(q) is valid only if Done(q) is true. Finally, the subroutine SafeDECon(G, q, u_c)" checks if the set $A_{q,u_c}(t)$ does not cross the bad set for any $t \in [0, \tau]$.

Although the pseudo-code for the solution to all problems is the same, each of the four problems will result in different ways of computing the set $R(q, u_c)$ and therefore each problem will have small differences in implementation of the algorithm. The call to DoDFS(G, q) computes the set $S(q)$ and returns true if and only if $q \in Q_m \vee S(q) \neq \emptyset$. It is thus readily observed that lines 9-20 of the algorithm implement equation (15) and hence correctly computes S. The running time is dependent on the size of the set $R(q, u_c)$ and is given by the expression $O(|\tilde{Q}||U_c|n^2 + |\tilde{Q}||U_c||R(q, u_c)|)$. The values for each of the four cases are given in Table I. In the last case, k is the number of controlled vehicles, and $\phi = \lceil d_{max}/\mu \rceil - \lfloor d_{min}/\mu \rfloor + 1$, so that $|W| = \phi^n$.

Case	Running Time
no uncontrollable vehicles no disturbance	$O(\|\tilde{Q}\|\|U\|n^2)$
uncontrollable vehicles no disturbance	$O(\|\tilde{Q}\|\|U_c\|n^2 + \|\tilde{Q}\|\|U\|)$
no uncontrollable vehicles disturbance	$O(\|\tilde{Q}\|\|U_c\|n^2 + \|\tilde{Q}\|\|U_c\|\|W\|)$
uncontrollable vehicles disturbance	$O(\|\tilde{Q}\|\|U_c\|n^2)$ $+O(\|\tilde{Q}\|\|U_c\|\phi^k(b-a+\phi)^{n-k})$

TABLE I
RUNNING TIMES IN EACH OF THE FOUR PROBLEM SCENARIOS

The standard algorithm for computing the supremal controllable sublanguage works by first constructing an automaton H representing the legal language specification and taking the product $G \times H$, then performing an iterative procedure until convergence, which runs in time quadratic in the size of $G \times H$. Let X_G and X_H be the state spaces of G and H, respectively, and E be the set of events. Recalling that verifying safety of a transition takes time $O(n^2)$, the asymptotic running time will therefore be $O(|X_G||E|n^2)$ to construct H (step 1), and $O(|X_G|^2|X_H|^2|E|)$ for the iterative process (step 2). Algorithm 1 achieves better asymptotic complexity in three ways. First, as previously mentioned, we can use a linear time algorithm since our system is livelock-free. Second, because our legal language specification is represented by a sub-automaton of G at the outset (rather than merely being a sublanguage of $\mathcal{L}(G)$), the product automaton $G \times H$ is isomorphic to H. The running time of step 2 is therefore $O(|X_H||E|)$. In our context, that gives the expression $O(|\tilde{Q}||U_c||U_{uc}||W|)$. For step 1, the running time would be $O(|\tilde{Q}||U_c||U_{uc}||W|n^2)$ in our context. Third, by making only a single test of safety for each control decision, we reduce the running time of step 1 to $O(|\tilde{Q}||U_c|n^2)$.

Algorithm 1 The Algorithm

```
1:  procedure DoDFS(G, q)
2:      if Terminal(G, q) then
3:          return true
4:      else if Done(q) then
5:          return Safe(q)
6:      end if
7:      Safe(q) ← false
8:      S(q) ← U_c
9:      for all u_c ∈ U_c do
10:         if not SafeDECon(G, q, u_c) then
11:             S(q) ← S(q) \ {u_c}
12:             continue
13:         end if
14:         for all q' ∈ R(q, u_c) do
15:             if not DoDFS(G, q') then
16:                 S(q) ← S(q) \ {u_c}
17:                 break
18:             end if
19:         end for
20:     end for
21:     if S(q) ≠ ∅ then
22:         Safe(q) ← true
23:     end if
24:     Done(q) ← true
25:     return Safe(q)
26: end procedure
```

V. PERFORMANCE OF THE ALGORITHM

We have implemented the algorithm presented above as an interactive Java application. Sample trajectories are shown in the three panels of Figure 1. In all cases, we took $\mu = \tau = 1$. The intersections are at the same position on each road and are shown in gray. The first panel corresponds to the model with $U_c = U$, $d_{min} = d_{max} = 0$, $n = 6$ and $V = \{1, 2\}^n$.

The number of states and DE control decisions were $|\tilde{Q}| \approx 4.80 \times 10^8$ and $|U| = 64$, and it took 15.3 seconds to compute the solution. The second panel corresponds to the model with $U_c = U$, $d_{min} = -1$, $d_{max} = 1$, $n = 5$ and $V = \{2, 3, 4\}^n$. The number of states and DE control decisions were $|\tilde{Q}| \approx 6.12 \times 10^5$ and $|U| = 243$, and it took 0.81 seconds to compute the solution. The third panel corresponds to the model with one uncontrolled vehicle (dotted and blue), $d_{min} = d_{max} = 0$, $n = 5$ and $V = \{1, 2\}^n$. The number of states and DE control decisions were $|\tilde{Q}| \approx 6.83 \times 10^4$ and $|U| = 32$, and it took 0.034 seconds to compute the solution. All simulations were run on a 1.6GHz laptop computer and used under 300MB of memory.

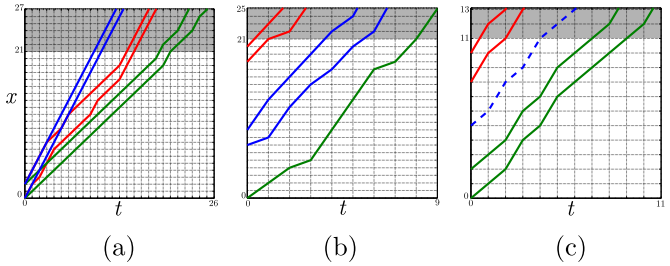

Fig. 1. Sample trajectories for various parameters and model types. (a): Zero disturbance, no uncontrolled vehicles. (b): No uncontrolled vehicles, disturbance with parameters $d_{min} = -1$, $d_{max} = 1$. (c): Zero disturbance, one uncontrolled vehicle. Trajectories of the same color represent vehicles on the same road. Dashed trajectories represent uncontrolled vehicles. The intersection is represented by a gray area.

VI. CONCLUSION

We have considered the problem of collision avoidance in vehicular networks as a supervisory control problem for a discrete event abstraction of the underlying continuous dynamics modeled by a first order system with model uncertainty. We have demonstrated that our abstraction methodology guarantees that the supervisor designed at the discrete event level, when lifted to the continuous level, satisfies the desired safety property, i.e., collisions are avoided. By considering the effect of unmodeled dynamics and uncontrolled vehicles as uncontrollable events at the discrete event level, we have been able to leverage the concepts and techniques of the theory of supervisory control of DES, in particular the nonblockingness and maximal permissiveness properties of the supremal controllable sublanguage of the marked version of the safe language. Moreover, we have exploited the structural properties of the transition structure of the discrete event model obtained by abstraction and developed a new algorithm for computing the supremal controllable sublanguage customized to this particular application, which achieves greater efficiency than the standard one. This work constitutes a new application area of DES theory, beyond those in manufacturing and software for instance (see, e.g., [17], [18], [19], [20]), with the distinctive feature that the DES model is obtained by abstraction from a continuous one, not by direct modeling of the discrete transition structure. Current issues of interest include refinement of our methodology to handle continuous models with second order dynamics, imperfect state information, acceleration constraints, and further algorithmic improvements.

REFERENCES

[1] *U. S. DOT National Highway Traffic Administration (NHTSA). Traffic safety facts. 2009.*

[2] (2011) Vehicle safety and fuel economy rulemaking and research priority plan 2011-2013. [Online]. Available: http://www.nhtsa.gov/staticfiles/rulemaking/pdf/2011-2013 Vehicle Safety-Fuel Economy Rulemaking-Research Priority Plan.pdf

[3] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control and Optimization*, vol. 25, no. 1, pp. 206–230, Jan. 1987.

[4] P. Tabuada, "An approximate simulation approach to symbolic control," *IEEE Trans. Autom. Control*, vol. 53, pp. 1406–1418, 2008.

[5] ——, *Verification and control of hybrid systems*. Springer-Verlag, 2009.

[6] A. Girard, G. Pola, and P. Tabuada, "Approximately bisimilar symbolic models for incrementally stable switched systems," *IEEE Trans. Autom. Control*, vol. 55, pp. 116–126, 2010.

[7] M. Broucke, M. D. Di Benedetto, S. Di Gennaro, and A. Sangiovani-Vincentelli, "Efficient solution of optimal control problems using hybrid systems," *SIAM J. Contr. Opt.*, vol. 43, pp. 1923–1952, 2005.

[8] M. Zamani, G. Pola, M. Mazo Jr., and P. Tabuada, "Symbolic models for nonlinear control systems without stability assumptions," *IEEE Trans. Autom. Control*, vol. 57, pp. 1804–1809, 2012.

[9] R. Alur, T. Dang, and F. Ivancic, "Predicate abstraction for reachability analysis of hybrid systems," *ACM Trans. on Embedded Computing Systems*, vol. 5, pp. 152–199, 2006.

[10] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. Pappas, "Symbolic planning and control of robot motion [grand challenges of robotics]," *Robotics Automation Magazine, IEEE*, vol. 14, no. 1, pp. 61 –70, march 2007.

[11] A. Colombo and D. Del Vecchio, "Supervisory control of differentially flat systems based on abstraction," in *50th IEEE Conference on Decision and Control*, 2011.

[12] ——, "Efficient algorithms for collision avoidance at intersections," in *Hybrid Systems: Computation and Control*, 2012.

[13] H. Kowshik, D. Caveney, and P. R. Kumar, "Provable systemwide safety in intelligent intersections," *IEEE Trans. Veh. Technol.*, vol. 60, pp. 804–818, 2011.

[14] W. M. Wonham and P. J. Ramadge, "On the supremal controllable sublanguage of a given language," *SIAM J. Control and Optimization*, vol. 25, no. 3, pp. 637–659, May 1987.

[15] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Springer-Verlag, 2008.

[16] N. Ben Hadj-Alouane, S. Lafortune, and F. Lin, "Variable lookahead supervisory control with state information," *IEEE Trans. Automatic Control*, vol. 39, no. 12, pp. 2398–2410, Dec. 1994.

[17] B. Brandin, "The real-time supervisory control of an experimental manufacturing cell," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 1, pp. 1 –14, feb 1996.

[18] T. Kelly, Y. Wang, S. Lafortune, and S. Mahlke, "Eliminating concurrency bugs with control engineering," *Computer*, vol. 42, no. 12, pp. 52–60, 2009.

[19] R. Leduc, M. Lawford, and P. Dai, "Hierarchical interface-based supervisory control of a flexible manufacturing system," *Control Systems Technology, IEEE Transactions on*, vol. 14, no. 4, pp. 654 – 668, july 2006.

[20] T. Moor, K. Schmidt, and S. Perk, "Applied supervisory control for a flexible manufacturing system," *Proceedings of the 10th International Workshop on Discrete Event Systems - WODES'10*, pp. 263–268, Sep. 2010.