Supervisory control of differentially flat systems based on abstraction

Alessandro Colombo and Domitilla Del Vecchio

Abstract— **The limiting factor in most implementations of safety enforcing controllers is the model's complexity, and a common work-around includes the abstraction of the physical model, based on differential equations, to a finite symbolic model. We exploit the specific structure of a class of systems (the differentially flat systems) to perform the abstraction. The objective is to construct a supervisor enforcing a set of safety rules, while imposing as little constraints as possible on the system's functionality. An example – a collision avoidance algorithm for a fleet of vehicles converging to an intersection – is presented. Our approach improves on previous results by providing a deterministic symbolic model irrespective of the stability properties of a system, and by addressing explicitly the problem of safety enforcing.**

I. INTRODUCTION

The problem of controlling multi-agent systems with safety specifications is often addressed in the framework of supervisory control of discrete event systems [1], [2]. The advantage of casting the control problem in this framework lies in the relative simplicity of formally verifying discrete event systems, with respect to dealing with geometric constraints on sets of differential equations. Additionally, the discrete event structure couples naturally with the digital systems that implement the controller. One of the greatest challenges in this approach resides in finding an efficient way to map the physical, continuous-time system onto a discrete event system, without losing too much structure along the way. Typical solutions involve defining a space and time discretization of the continuous-time model, restricting space and input sets so that the discretization is finite, and then proving an equivalence relation between the discretized system and a suitable discrete event system (as in [3], [4], [5], [6], [7]). The discrete event equivalent is then called a *finite-state abstraction* of the continuous-time system. In the absence of incremental stability properties, the above cited works yield nondeterministic discrete event systems. Moreover, none of these approaches addresses explicitly the issue of safety enforcement. An exception is found in [8], where abstraction techniques for safety enforcement are addressed in the context of reachability analysis.

In this paper, we exploit the specific structure of an important class of systems –the differentially flat systems [9], [10], [12], [11]– to reduce the dynamics of a general model to that of the trivial system $\dot{\chi} = u$. We construct a finite-state abstraction of the trivial system, and use it as the backbone for the design of a supervisor. Safety is addressed explicitly and ensured by construction, irrespective of the chosen discretization step, so that a coarse discretization can be used without compromising the specifications.

This work was supported by the NSF Award # CNS 0930081

Alessandro Colombo and Domitilla Del Vecchio are with the Department of Mechanical Engineering, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, USA. Email: acolombo@mit.edu, ddv@mit.edu

We design the least restrictive supervisor for the trivial system, and use it to construct the supervisor for the general model. We provide a measure of the maximum distance of the trajectories allowed by the supervisor of the general system and the trajectories of the trivial system. We also show that by refining the discretization, the supervisor allows any trajectory that does not intersect the bad set. Moreover, we provide an upper bound on the distance of the allowed trajectories of the general model from the bad set. This distance can be made arbitrarily small by refining the discretization.

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. A superscript between parentheses (e.g., $x^{(i)}$) indicates an i-th order time-derivative. Finally, $\mathrm{Conv}(\cdot)$ indicates the convex hull of a set.

The paper is organised as follows: in the next section, we formalize the problem. In Section III we propose a solution, and in Section IV we provide details on its numerical implementation. Then, in Section V we apply our algorithm to the control of a set of vehicles at an intersection, and we draw conclusions in Section VI.

II. PROBLEM STATEMENT

We analyse a system of the form

$$\dot{x} = f(x,a), \quad y = h(x), \tag{1}$$

with $x(t) \in X \subset \mathbb{R}^m$, $a(t) \in A \subset \mathbb{R}^n$, $y(t) \in Y \subset \mathbb{R}^n$. Functions f and h are C^k for sufficiently large k. The vector $x(t)$ is the state of the system, $a(t)$ is the input, and $y(t)$ is the output. We assume that (1) is differentially flat [9], [10], [11], with y as flat output. This means that it satisfies the following assumption:

Assumption 2.1: Function h has rank n, and there exist two functions $\Gamma : (\mathbb{R}^n)^{q+1} \mapsto \mathbb{R}^m$ and $\Theta : (\mathbb{R}^n)^{q+2} \mapsto \mathbb{R}^n$ of rank m and n, respectively, in their domains, such that the integral curves of (1) identically satisfy the equations

$$x = \Gamma(y, \dot{y}, ..., y^{(q)}), \quad a = \Theta(y, \dot{y}, ..., y^{(q+1)}). \tag{2}$$

We also require that $n(q+1) = m$, which together with the rank condition implies that the function Γ is invertible.

Call $\mathcal{A}$, $\mathcal{X}$, and $\mathcal{Y}$ the sets of all possible functions of time a, x, and y, respectively. We assume that $\mathcal{A}$ is the space of C^q functions of t. We consider the set $B \subset Y$, called the *bad set*, and assume that it is the union of a finite number of convex polytopes. This is the setting, for example, of mechanical systems in the form $\dot{x}^1 = x^2$, $\dot{x}^2 = f^1(x^1,x^2)+f^2(x^1,x^2)a$, where the flat output is $y = x^1$, and the bad set is a set of collision positions.

Let us introduce the *flow* $\phi_t(x(0),a) \in \mathbb{R}^m$ of system (1), such that $\phi_0(x(0),a) = (x(0))$ and x with $x(t) = \phi_t(x(0),a)$ for all t satisfies equation (1). Let $\phi_{[t_1,t_2]}(x(0),a) := \bigcup_{t\in[t_1,t_2]} \phi_t(x(0),a)$. Our objective is to design a supervisor [1], [2] for system (1) that prevents trajectories from entering the bad set. This requirement can be formally expressed using the concept of *ϵ-safe trajectory*:

Definition 2.1: A trajectory $\phi_{[0,T]}(x(0),a)$ with $T \in \mathbb{R}_+$ is ϵ-safe provided $\inf_{t\in[0,T]} \inf_{b\in B} \|h(\phi_t(x(0),a))-b\| > \epsilon$.

Consider a compact set of initial conditions $X_0 \subset X$. Given a grid with hypercubic cells of side η on the set Y, consider a compact set $\bar{Y} \subset Y$ composed of a finite number of cells, such that for all $x(0) \in X_0$, $h(x(0)) \in \bar{Y}$. For all $x(0) \in X_0$ and $a \in \mathcal{A}$, call $T(x(0),a)$ the infimum of the set $\{T \in \mathbb{R}_+ : h(\phi_T(x(0),a)) \notin \bar{Y}\}$, and call $a_{(k\tau,(k+1)\tau]}$ the signal a restricted to the interval $(k\tau,(k+1)\tau]$.

Problem 2.1 (Supervisor design): Determine a map $\sigma : X \mapsto 2^{\mathcal{A}}$, called a *supervisor*, depending on the parameter $0 < \tau < \infty$, with the two following properties:

(P.1) For all $x(0) \in X_0$ and for all a such that $a_{(k\tau,(k+1)\tau]} \in \sigma(x(k\tau))$, $\phi_{[0,T(x(0),a)]}(x(0),a)$ is 0-safe.

(P.2) For all $x(0) \in X_0$, if there exist $0 < \delta < \infty$, $-\infty < \gamma_l < \gamma_u < \infty$, and $a \in \mathcal{A}$ such that $\phi_{[0,T(x(0),a)]}(x(0),a)$ is δ-safe and $\gamma_l \le \dot{h}_i(\phi_t(x(0),a)) \le \gamma_u$ for all $t \in [0,T(x(0),a)]$ and for all i, then there exists a $\tau^*(\delta) > 0$ such that, for all $\tau < \tau^*(\delta)$, there exists an $a^* \in \mathcal{A}$ such that $a^*_{(k\tau,(k+1)\tau]} \in \sigma(x(k\tau))$ and $\|h(\phi_t(x(0),a^*)) - h(\phi_t(x(0),a))\| < \delta$ for all $t \in [0,\min(T(x(0),a),T(x(0),a^*))]$.

Property (P.1) requires that all trajectories allowed by the supervisor are 0-safe. Property (P.2) requires that, if there exists a 0-safe trajectory originating at $x(0)$ for input a, whose output has bounded derivative, then for τ sufficiently small the output is approximated arbitrarily well by the output of a 0-safe trajectory allowed by the supervisor.

III. DESIGN OF THE SUPERVISOR

Here, we design a supervisor to solve Problem 2.1. In Section III-A, we introduce the trivial system, with dynamics in the output set Y, and design a supervisor for this system. Then, in Section III-B, we use this result to construct a supervisor for (1).

A. SUPERVISOR OF THE TRIVIAL SYSTEM

Consider the system

$$\dot{\chi} = u, \tag{3}$$

where $\chi : \mathbb{R} \mapsto Y$ is the state trajectory and $u : \mathbb{R} \mapsto U \subset \mathbb{R}^n$ is the input. To set the basis for the construction of a finite-state abstraction, we restrict the input set of (3) to a finite set, and we construct a finite lattice over the state set $\bar{Y}$. Define U as a set of vectors with elements in $u_{adm} := \{k\mu : k \in \mathbb{Z}\}$, where μ is a fixed positive constant, and assume that u is constant over intervals $(k\tau,(k+1)\tau]$. Call $\mathcal{U}$ the set of all such signals u. Consider a regular lattice $G \subset \bar{Y}$ of step η over $\bar{Y}$, such that an element of the lattice lies in the centre of each hypercubic cell composing $\bar{Y}$. Let g denote an element of G. Since both $g \in G$ and $\chi(t) \in \bar{Y}$ are elements of $\mathbb{R}^n$, the infinity norm defines a distance for any pair $(g,\chi(t))$. Clearly, the lexicographical order is a total order on the elements of G, so that any subset of G has a unique minimum. Then, define the map $\ell : Y \mapsto G$ as

$$\ell(\chi(t)) := \min_{g\in G}\{g : \|\chi(t) - g\| \le \eta/2\}. \tag{4}$$

In analogy to the previous section, call $\phi_t(\chi(0),u)$ and $\phi_{[t_1,t_2]}(\chi(0),u)$, $t_1 < t_2$, respectively, the flow and trajectories of (3). The concept of ϵ-safety in Definition 2.1 is extended to trajectories of (3) simply by substituting the flow $\phi_t(\chi(0),u)$ to the function $h(\phi_t(x(0),a))$ in Definition 2.1. For all $\chi(0) \in \bar{Y}$ and $u \in \mathcal{U}$, call $T(\chi(0),u)$ the infimum of the set $\{T \in \mathbb{R}_+ : \phi_T(\chi(0),u) \notin \bar{Y}\}$.

Definition 3.1: A supervisor $\sigma_C : \bar{Y} \mapsto 2^U$ of (3) is the *least restrictive* with respect to ϵ if for all $\chi(0) \in \bar{Y}$, for all $u \in \mathcal{U}$ such that $u(t) \notin \sigma_C(\chi(k\tau))$ for $k = \sup\{k : k\tau < t\}$, there exists a trajectory $\phi_{[0,T(\tilde{\chi}(0),u)]}(\tilde{\chi}(0),u)$ with $\ell(\tilde{\chi}(0)) = \ell(\chi(0))$ that is not ϵ-safe.

Our first step consists in solving the following problem:

Problem 3.1: Given the set $\bar{Y}$, design the least restrictive supervisor σ_C with respect to ϵ such that for all $u \in \mathcal{U}$ with $u(t) \in \sigma_C(\chi(k\tau))$ for $k = \sup\{k : k\tau < t\}$, $\phi_{[0,T]}(\chi(0),u)$ is ϵ-safe.

We construct a finite-state abstraction of (3) by considering its time discretization, called Σ_{DT}, and then constructing an equivalent finite discrete event system, called Σ_{DE}.

Definition 3.2: Given $\tau > 0$, the *time-τ discretization* of (3) is a discrete event system and is denoted $\Sigma_{DT} = \{Y, U, f\}$, where Y is the state space, U is the input space, and $f : Y \times U \mapsto Y$ is the transition function. We call the state $z \in Y$ and the input $w \in U$, to distinguish them from the state and input functions of (3). Given any $z \in Y$ and $w \in U$, and given $u \in \mathcal{U}$ such that $u(t) = w$ for all $t \in (0,\tau]$, and $\chi(0) = z$, then $f(z,w) := \phi_\tau(\chi(0),u)$. A transition of Σ_{DT} is a state/input pair and is denoted $(z,w)_{DT}$.

We can thus say that a transition $(z,w)_{DT}$ of Σ_{DT} *corresponds* to a trajectory $\phi_{[0,\tau]}(\chi(0),u)$ of (3) if $\chi(0) = z$ and $u(t) = w$ for all $t \in (0,\tau]$.

Definition 3.3: A finite *execution* of length m is a sequence of transitions $\{(z^0,w^0),(z^1,w^1),...\}$ such that $z^{i+1} = \psi(z^i,w^i)$. To identify an execution of length m, we introduce the shorthand notation $(z^0,w^0,...,w^{m-1})_{DE}$. An execution of arbitrary (finite or infinite) length is denoted $(z^0,w^0,...)_{DE}$. We also introduce the symbol $\psi(\xi,w^0,...,w^{m-1})$ for the terminal state reached by an execution of length m and $\psi(\xi,w^0,...)$ to indicate the terminal state reached by a finite execution of arbitrary length.

System Σ_{DT} has an infinite number of states. We use the concept of *bisimulation* (see definition in [2]) to prove the an equivalence of Σ_{DT} and a suitably constructed, finite discrete event system Σ_{DE}. Consider the equivalence relation $\simeq$ defined by $z^1 \simeq z^2$ if $\ell(z^1) = \ell(z^2)$, with ℓ defined in (4), and the partition L of Y that it induces. Use G as the state set of Σ_{DE} and U as its event set, and define its transition function by

$$\psi(g^1, w) = g^2 \text{ iff } \exists\, z \in Y : g^1 = \ell(z),\ g^2 = \ell(f(z, w)).$$

Furthermore, set

$$\eta = \tau\mu. \tag{5}$$

Lemma 3.1 (Finite-state abstraction): Σ_{DT} introduced in Definition 3.2 is bisimilar to $\Sigma_{DE} = \{G, U, \psi\}$ with respect to the relation $g \simeq z$ if $g = \ell(z)$.

Notice that our choice of partition ensures that Σ_{DE} is deterministic, since the dynamics of Σ_{DT} maps elements of L onto elements of L. We can now define a supervisor for (3) using the finite-state abstraction Σ_{DE}. First we extend the concept of ϵ-safety to transitions of Σ_{DT} and Σ_{DE}.

Definition 3.4: Given a real number $\epsilon > 0$.

- Take a transition $(z, w)_{DT}$ of Σ_{DT} and the corresponding trajectory $\phi_{[0,\tau]}(\chi(0), u)$ of (3), with $\chi(0) = z$ and $u(t) = w$ for all $t \in [0, \tau]$. The transition $(z, w)_{DT}$ is ϵ-safe if and only if $\phi_{[0,\tau]}(\chi(0), u)$ is ϵ-safe.
- Take $\Sigma_{DE} = \{G, U, \psi\}$ bisimilar to $\Sigma_{DT} = \{Y, U, f\}$. Consider a single transition $(g, w)_{DE}$ of Σ_{DE}. The transition $(g, w)_{DE}$ is ϵ-safe provided $(z, w)_{DT}$ with $z = g$ is an $(\epsilon + \eta/2)$-safe transition of Σ_{DT}

The definition of ϵ-safety of a transition is extended to ϵ-safety of executions of Σ_{DE} as follows.

Definition 3.5: The execution $(g, w^0, ...)_{DE}$ is ϵ-safe if and only if all the transitions that compose it are ϵ-safe.

The above definitions imply the following

Lemma 3.2: ($(g, w)_{DE}$ is ϵ-safe) implies that for all $\chi(0) \in \bar{Y}$ such that $\ell(\chi(0)) = g$, $\phi_{[0,\tau]}(\chi(0), u)$ with $u(t) = w$ for $t \in (0, \tau]$ is ϵ-safe.

We now define the supervisor σ_C, using the concept of *forward-maximal* executions, defined as follows.

Definition 3.6: An execution of finite length $(g^0, w^0, ..., w^{m-1})_{DE}$ is forward-maximal if $\psi(g^0, w^0, .., w^{m-n}) \in G$ for all $n \in \{2, ..., m\}$, and $\psi(g^0, w^0, .., w^{m-1}) \notin G$. An execution $(g^0, w^0, ...)$ of infinite length is always forward-maximal.

Consider the following set of all ϵ-safe forward-maximal executions:

$$\begin{aligned} S := \{&(g, w^1, ...)_{DE} : g \in G, (g, w^1, ...)_{DE} \text{ is } \epsilon\text{-safe}, \\ &\text{and } (g, w^1, ...)_{DE} \text{ is forward-maximal}\}. \end{aligned} \tag{6}$$

The supervisor σ_C is defined as

$$\sigma_C(z) := \{w : \exists\, (\ell(z), w, ...)_{DE} \in S\}. \tag{7}$$

Theorem 3.3: Problem 3.1 is solved by σ_C.

Proof: By definition of σ_C and S, the trajectory $\phi_{[0,\tau]}(\chi(0), u)$ with $u(t) \in \sigma_C(\chi(0))$ for all $t \in (0, \tau]$ corresponds to a transition $(z^0, w^0)_{DT}$ with $z^0 = \chi(0)$ and $w^0 = u(\tau)$ which, by the bisimilarity of Σ_{DT} and Σ_{DE}, corresponds to the first transition of an ϵ-safe execution $(\ell(z^0), w^0, ...)_{DE} \in S$. Hence, by Lemma 3.2 the trajectory is ϵ-safe, and the state $\ell(\phi_\tau(\chi(0), u))$ belongs to an execution in S. Since all executions in S are forward-maximal, any sequence of such trajectories is ϵ-safe as long as it remains in $\bar{Y}$. Thus the supervisor (7) solves Problem 3.1.

We are left to show that the supervisor is the least-restrictive. We proceed by contradiction. Consider an input signal $u \in \mathcal{U}$ such that $u(t) \notin \sigma_C(\chi(k\tau))$, $k = \sup\{k : k\tau < t\}$, for some $t \in [0, T(\chi(0), u)]$. Assume that all trajectories $\phi_{[0,T(\chi(0),u)]}(\tilde{\chi}(0), u)$, with $\ell(\tilde{\chi}(0)) = \ell(\chi(0))$, are ϵ-safe. Applying Lemma 3.2 to each segment of length τ of each trajectory we conclude that the execution $(\ell(z), w^0, w^1, ...)_{DE}$, where $w^0 = u(t)$ in the interval $(0, \tau]$, $w^1 = u(t)$ in the interval $(\tau, 2\tau]$, etc., is ϵ-safe and forward-maximal, which contradicts the hypothesis that $u(t) \notin \sigma_C(\chi(k\tau))$. ∎

B. SOLUTION OF PROBLEM 2.1

To solve Problem 2.1 we need to find a map from the inputs u of system (3), to the inputs a of system (1). For the supervisor to enforce safety, this map must ensure that outputs $y = h(\phi_{[0,t]}(x(0), a))$ of (1) follow closely the trajectories $\phi_{[0,t]}(\chi(0), u)$ of system (3). Here, we exploit the fact that y is a flat output for system (1), hence by virtue of (2) we can design a path in y, and map it onto signals a and x satisfying (1) and (2).

Having set $\chi(0) = h(x(0))$, for $t \in (k\tau, (k+1)\tau]$ we construct $y(t) = c_0 + c_1(t - k\tau) + ... + c_{2q+1}(t - k\tau)^{2q+1}$ taking the coefficients $c_0, ..., c_{2q+1}$ so that y is C^q everywhere and passes through the points $\chi(k\tau) = \phi_{k\tau}(\chi(0), u)$, $k \in \mathbb{N}$, at $t = k\tau$. Since the supervisor is causal, we must determine the coefficients using only the value $x(k\tau) = \phi_{k\tau}(x(0), a)$ with $k = \sup\{k : k\tau < t\}$. For a given $u(t) = \tilde{u} \in \sigma_C(x(k\tau))$, the coefficients $c_0, ..., c_q$ are obtained by using the first equation of (2), by imposing that

$$\lim_{t \to k\tau} (y(t), \dot{y}(t), ..., y^{(q)}(t)) = \Gamma^{-1}(x(k\tau)). \tag{8}$$

The coefficients $c_{q+1}, ..., c_{2q+1}$ are obtained by imposing the following $q + 1$ conditions at the boundary $t = (k + 1)\tau$:

$$\begin{array}{l} y((k+1)\tau) = h(x(k\tau)) + \tau\tilde{u} \\ \dot{y}((k+1)\tau) = \tilde{u} \\ y^{(2)}((k+1)\tau) = ... = y^{(q)}((k+1)\tau) = 0. \end{array} \tag{9}$$

Notice that, since $\chi(k\tau) = h(x(k\tau))$ and $u(t) = \tilde{u}$, the polynomial above indeed passes through the point $\chi((k + 1)\tau) = h(x(k\tau)) + \tau\tilde{u}$ at time $(k + 1)\tau$. Moreover, (8) properly ensures that $y \in C^q$. Finally, since the polynomials composing y have coefficients and domain in a compact set, the quantity

$$\epsilon_a = \sup_{u \in \mathcal{U}, k \in \mathbb{N}, t \in [k\tau, (k+1)\tau]} \|y(t) - \phi_t(\chi(0), u)\|, \tag{10}$$

with $y(0) = \chi(0)$, which is the maximum distance of $y(t)$ from $\chi(t)$, is finite.

Let us define

$$\sigma(x(k\tau)) := \Theta(y(t)), \dot{y}(t), ..., y^{(q+1)}(t)), \tag{11}$$

for t spanning the set $(k\tau,(k+1)\tau]$. Here, $y(t)$ on the right hand side is given by (9), and $\tilde{u}$ is set by σ_C.

Theorem 3.4 (Main result): Setting $\epsilon > \epsilon_a$ and constructing σ_C to enforce ϵ-safety, with u_{adm} such that $[\gamma_l, \gamma_u] \subset \mathrm{Conv}(u_{adm})$, the supervisor (11) solves Problem 2.1.

From the supervisor construction it follows trivially that σ satisfies P.1 of Problem 2.1. To prove that it also satisfies P.2, we first need to prove two lemmas.

Lemma 3.5: Consider $-\infty < \gamma_l < \gamma_u < \infty$, $a \in \mathcal{A}$, and $x(0) \in X_0$ such that $\gamma_l < \dot{h}_i(\phi_t(x(0),a)) < \gamma_u$ for all $t \in [0, T(x(0),a)]$ and for all $i \in \{1,...,n\}$. Let $\chi(0) = h(x(0))$ and assume that $[\gamma_l,\gamma_u] \subset \mathrm{Conv}(u_{adm})$. Then for $\tau \to 0$ there exists a $u^\tau \in \mathcal{U}$ such that $\|\phi_t(\chi(0),u^\tau) - h(\phi_t(x(0),a))\| \le \tau(\gamma_u - \gamma_l)/2$.

Proof: [sketch] In the case $\gamma_u = -\gamma_l = \gamma > 0$, the lemma is proved by induction on k. Call $x(t) = \phi_t(x(0),a)$ and $\chi(t) = \phi_t(\chi(0),u)$. The base case is true by assumption, as $\chi(0) = h(x(0))$, and the induction step requires to prove that if $\|\chi(k\tau) - h(x(k\tau))\| < \gamma\tau$, then $\|\chi(t) - h(x(t))\| < \gamma\tau$ in the interval $(k\tau,(k+1)\tau]$. This is done taking, for each given τ, each component u_i^τ of u^τ in the interval $(k\tau,(k+1)\tau]$ with the following rule:

(i) If $\chi_i(k\tau) \le h_i(x((k+1)\tau))$ then $u_i^\tau(t) = \gamma$
(ii) If $\chi_i(k\tau) > h_i(x((k+1)\tau))$ then $u_i^\tau(t) = -\gamma$.

The case of generic γ_l and γ_u follows by defining $\gamma = (\gamma_u - \gamma_l)/2$, and subtracting $t(\gamma_u+\gamma_l)/2$ from both $\chi(t)$ and $h(x(t))$. ■

Lemma 3.6: Consider y defined by (8) and (9), interpolating χ at the points $\chi(k\tau)$, $k \in \mathbb{N}$. Then, $\|y - \chi\| \le C\tau$ for some $C > 0$.

Proof: The component l of y is

$$y_l(t) = \sum_{i=0}^{2q+1} c_i (t-k\tau)^i,\ t \in (k\tau,(k+1)\tau] \qquad (12)$$

with $c_i \in \mathbb{R}$. From (8) we see that $c_0,...,c_q$ are functions of $x(0)$ when $k=0$. In the following intervals instead, (9) together with the C^q-continuity ensured by (8) imply that $c_0 = h_l(x(k\tau))$, $c_1 = \dot{h}_l(x(k\tau))$, and $c_2,...,c_q = 0$. Thus, these coefficients are always independent of τ. The coefficients $c_{q+1},...,c_{2q+1}$ are obtained by solving the linear system (9), which has form $M(\tau)C = V(\tau)$ where M is a matrix function of τ, C is the vector of coefficients $c_{q+1},...,c_{2q+1}$, and $V(\tau)$ is a vector function of τ and of $c_0,...,c_q$, $h(x(k\tau))$, and $\tilde{u}$. Now consider $\tau \to 0$. Exploiting the form of M it is easy to prove that the element i,j of M^{-1} is $m_{i,j} = O(\tau^{-q-1+j-i})$, while for the elements j of V it holds $v_1 = O(\tau)$, $v_j = O(1)$ for $j > 1$. Thus, $c_{q+i} = \sum_{j=1}^{q+1} m_{i,j} v_j = O(\tau^{-q-i+1})$ for $i \in \{1,...,q+1\}$. Writing $c_i(t-k\tau)^i = \tilde{c}_i\lambda^i$ with $\lambda = (t-k\tau)/\tau \in [0,1]$ and $\tilde{c}_i = c_i\tau^i$, (12) becomes $y_l = \sum_{i=0}^{2q+1}\tilde{c}_i\lambda^i$ where $\tilde{c}_0 = c_0 = h(x(k\tau))$, and $\tilde{c}_i = O(\tau)$ for all $i \in \{1,...,2q+1\}$. Now, since $\chi_l(t) = h(x(k\tau)) + (t-k\tau)\tilde{u}_l$ in the interval $(k\tau,(k+1)\tau]$, $\|y_l(t) - \chi_l(t)\| = O(\tau)$. This reasoning applies for all components of y and χ, and in particular applies to $\sup_{u\in\mathcal{U},t>0}\|y(t)-\chi(t)\|$, which is finite as we have established in (10). Thus, $\|y-\chi\| \le C\tau$ for some $C > 0$. ■

We can now prove that the supervisor σ satisfies Property (P.2).

Proof: (Theorem 3.4) For a given $x(0) \in X_0$, take $\delta > 0$, and assume there exist $-\infty < \gamma_l < \gamma_u < \infty$ and $a \in \mathcal{A}$ such that $\phi_{[0,T(x(0),a)]}(x(0),a)$ is δ-safe and $\gamma_l < \dot{h}_i(\phi_t(x(0),a)) < \gamma_u$ for all $t \in [0,T(x(0),a)]$ and for all i. By Lemma 3.5, there exists a $u \in \mathcal{U}$ such that $\|\phi_t(\chi(0),u) - h(\phi_t(x(0),a))\| < \tau(\gamma_u-\gamma_l)/2$, with $\chi(0) = h(x(0))$. Using Lemma 3.2 we conclude that, if $\phi_{[0,T(x(0),a)]}(x(0),a)$ is δ-safe, then $\phi_{[0,T(\chi(0),u)]}(\chi(0),u)$ is allowed by a supervisor σ_C that ensures ϵ-safety with $\epsilon \le \delta - \tau(\gamma_u-\gamma_l)/2 - \eta$. The quantity $\eta = \tau\mu$ is defined in (5). From Lemma 3.6, we also know that $\epsilon_a \le C\tau$ for some $C > 0$. Hence, a supervisor satisfying (P.2) must have $\epsilon \ge C\tau$. The supervisor exists as long as $C\tau < \delta - \tau(\gamma_u-\gamma_l)/2 - \tau\mu$, that is, as long as $\tau(C + (\gamma_u-\gamma_l)/2 + \mu) \le \delta$, which is always possible for τ sufficiently small. Finally, as $\tau \to 0$, $\phi_{[0,T(\chi(0),u)]}(\chi(0),u)$ converges uniformly to $h(\phi_{[0,T(x(0),a)]}(x(0),a))$ for $t \in [0, \min(T(x(0),a), T(x(0),a^*))]$, and y converges uniformly to $\phi_{[0,T(\chi(0),u)]}(\chi(0),u)$. Setting $a^* = \Theta(y,\dot{y},...)$, then $y(t) = h(\phi_t(x(0),a^*))$ by flatness, so for sufficiently small τ the input a^* ensures that $\|h(\phi_t(x(0),a^*)) - h(\phi_t(x(0),a))\| < \delta$. ■

Additionally, we provide an upper bound on the distance of the output y of the allowed trajectories in $\bar{Y}$ and the bad set:

Theorem 3.7: Place the elements of u_{adm} in increasing order and define δ as the maximum distance between two successive elements. Assume that for a given $x(0) \in X_0$

A.1 $\sigma(x(0)) \neq \emptyset$
A.2 there exists a $u(t) \in \mathcal{U}$ and a correspondng $y(t)$ of the form specified in (9) and (8), with $y(0) = h(x(0))$, such that $y(t) \in B$ for some $t \ge 0$.

Then, for sufficiently large $\bar{Y}$ there exists $a \in \mathcal{A}$ such that $a((k\tau,(k+1)\tau]) \in \sigma(\phi_{k\tau}(x(0),a))$ for all $k \ge 0$, and

$$\min_{t\in[0,T(x(0),a))}\ \min_{b\in B}\|h(\phi_t(x(0),a)) - b\| \le 2\epsilon + \delta\tau.$$

By refining the abstraction, reducing τ and ϵ the supervisor σ allows trajectories that pass arbitrarily close to the bad set.

Proof: [Sketch] Call $\mathcal{S}$ the family of all forward-maximal executions of Σ_{DE} with initial condition $g = \ell(h(x(0))$. One first shows, using assumptions A.1 and A.2, that there exists two executions in $\mathcal{S}$ such that

- the first has transition $s \in \mathbb{N}$ as first non-safe transition,
- the second is ϵ-safe,
- the input sequences of the two executions are identical up to transition s except at one step, where the input value differ by at most δ.

Reasoning on the distance between the trajectories corresponding to the ϵ-safe execution and the ones corresponding to the other one completes the proof. ■

IV. NUMERICAL IMPLEMENTATION

Here, we shall provide an algorithm to implement σ numerically. The main challenge is the implementation of the map σ_C, since its output is readily translated into a set of output signals a through an algebraic mapping. Determining σ_C, in turn, requires to determine the set S.

Let us consider the set $\bar{S}$ of all possible executions of Σ_{DE}: $\bar{S} := \{(g, w, ...)_{DE} : g \in G\}$. The set S defined in (6) is the subset of $\bar{S}$ of all forward-maximal executions that are ϵ-safe. This suggests a way to construct S: given $\bar{S}$, we can first remove all the executions that are not ϵ-safe, obtaining the subset $\hat{S} := \{(g, w, ...)_{DE} : g \in G$ and $(g, w, ...)_{DE}$ is ϵ-safe$\}$, and then remove from $\hat{S}$ all executions that are not forward-maximal. This process can be implemented in a simple way by viewing the three sets $\bar{S}$, $\hat{S}$ and S as the sets of all possible paths in three directed graphs, named $\bar{S}_G$, $\hat{S}_G$ and S_G, respectively. Nodes in the three graphs are states in G, while edges are transitions. The first step –obtaining $\hat{S}_G$ from $\bar{S}_G$– is executed by checking, for each transition $(g, w)_{DE} \in \bar{S}$, if the trajectory $\phi_{[0,\tau]}(\chi(0), u)$ with $\chi(0) = g$ and $u(t) = w$ in $(0, \tau]$ intersects an $(\epsilon+\eta/2)$-neighbourhood of B. A point $\chi(0)$ belonging to a convex polytope composing B verifies an inequality of the form $A\chi(0) < 0$, with A a $p \times n$ matrix, where p is the number of faces of the polytope and n the size of vector $\chi(0)$. The intersection of a trajectory $\phi_{[0,\tau]}(\chi(0), u)$, $u(t) = \tilde{u}$, with any polytope is then detected by checking whether the inequality $A(\chi(0) + t\tilde{u}) < 0$ has solutions for $t \in [0, \tau]$. This conditions must be checked for each polytope, for each possible transition, and for each $g \in G$, thus the complexity of the intersection checking grows linearly with the number of states in G and transitions per state.

The second task –obtaining S_G from $\hat{S}_G$– is accomplished by iteratively removing edges that reach a node with no outgoing edges. The map $\sigma_C(\chi(k\tau))$ is then readily evaluated, finding the node of the graph that corresponds to $\ell(z)$ with $z = \chi(k\tau)$, and returning the values w corresponding to the outgoing edges of node $\ell(z)$. Map σ is found using (11).

Assume that the discretization of X has s elements in each direction. If $X \subset \mathbb{R}^n$ and U has cardinality m, the number of nodes in the graph is s^n, the number of possible edges per node is m^n. Checking which transitions intersect the bad set B thus has complexity $O((sm)^n pn)$ where p, introduced above, is the number of faces of the polytopes composing B. Removing nodes with no outgoing edges has, in the worst case, quadratic complexity in the number of nodes (s^n), so the worst case complexity is $O(s^{2n})$.

V. EXAMPLE

Consider a set of n vehicles travelling in straight lines along m roads that intersect at a common point, as in the example in Fig. 1. A vehicle's position at time t is represented by $x_i^1(t) \in \mathbb{R}, i \in \{1, ..., n\}$. The set of indices $\{1, ..., n\}$ of x^1 can be partitioned into m sets $\Delta^k, k \in \{1, ..., m\}$, where k indicates the road along which vehicle i is travelling. If $i \in \Delta^k$ and $j \in \Delta^k$, then vehicles i and j travel along the same road (e.g., in Fig. 1 indices 1 and 2 are in Δ^1, indices 3 and 4 in Δ^2 and Δ^3, respectively). The set of vehicles obeys the law

$$\dot{x}^1 = x^2, \quad \dot{x}^2 = a - k(x^2)^2, \tag{13}$$

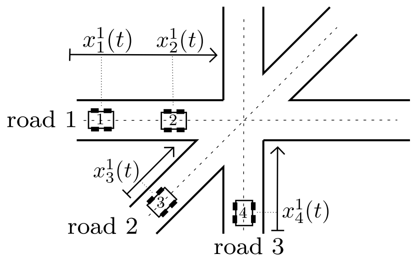

Fig. 1. Four vehicles on three roads. The quantities $x_i^1(t)$ represent the positions of the four vehicles at time t.

where the term $-k(x^2)^2$, with $k > 0$, models air drag, while static friction can be embedded in the input a through a simple change of variables. This is a flat system with flat output $y = x^1$ and

$$\begin{aligned} (x^1, x^2) &= \Gamma(y, \dot{y}, ..., y^{(q)}) = (y, \dot{y}) \\ a &= \Theta(y, \dot{y}, ..., y^{(q+1)}) = y^{(2)} + k\dot{y}^2. \end{aligned} \tag{14}$$

Our objective is to design a control law that prevents vehicle collisions, while ensuring that all vehicles pass the intersection and that their velocities remain nonnegative. States in the bad set satisfy one of two sets of inequalities. The first one is used for two vehicles, $i \in \Delta^k$ and $j \in \Delta^l$, driving along two different roads, where $y_i(t) \in [\alpha_k, \beta_k]$ if vehicle i is in the intersection. Then a collision occurs if

$$\begin{aligned} \exists\, i \in \Delta^k, j \in \Delta^l, k \neq l \text{ such that } \alpha_k < y_i(t) < \beta_k \\ \text{and } \alpha_l < y_j(t) < \beta_l, \alpha_k, \alpha_l, \beta_k, \beta_l \in \mathbb{R}. \end{aligned} \tag{15}$$

The second condition is used for two vehicles driving on the same road. Let $d \in \mathbb{R}$ be the minimum safe distance between vehicles i and j. Then, a collision occurs if

$$\exists i, j \in \Delta^k \text{ such that } -d < y_i(t) - y_j(t) < d. \tag{16}$$

An output $y \in B$ if (15) or (16) is satisfied. The control problem is solved once each vehicle $i \in \Delta^k$ passes a predetermined position p_i, where $p_i \geq \beta_k$. We take $\bar{Y}$ such that each $y_i \in [y_{i,min}, y_{i,max}]$, $y_{i,min}, y_{i,max} \in \mathbb{R}$, making sure that $y_{i,max} > p_i$. The set X_0 is taken so that its image through h is equal to $\bar{Y}$, while the variables v_i are restricted to the interval $[1, 4]$.

We shall assume that x^1 and x^2 are continuous signals, while a can be only piecewise continuous. This implies that y satisfying (14) must be of class C^1. We can thus approximate each trajectory $\phi_{[k\tau,(k+1)\tau]}(\chi(k\tau), u)$, with a cubic polynomial with the following boundary conditions:

$$\begin{aligned} &\lim_{t\to k\tau} y(t) = x^1(k\tau), \quad y((k+1)\tau) = x^1(k\tau) + \tau u(t) \\ &\lim_{t\to k\tau} \dot{y}(t) = x^2(k\tau), \quad \dot{y}((k+1)\tau) = u(t). \end{aligned}$$

Solving the above equations, the signal y is given by

$$\begin{aligned} y(t) = x^1(k\tau) + tx^2(k\tau) - \frac{2t^2}{\tau}(x^2(k\tau) - u(t)) + \\ \tfrac{t^3}{\tau^2}(x^2(k\tau) - u(t)),\ k\tau < t \leq (k+1)\tau. \end{aligned}$$

We set $u_{adm} = \{1, 2, 3, 4\}$, which with the equation above ensures that x^2 is nonnegative, while $\alpha = 1$ and $\beta = 2$ for all vehicles, $d = 1$, $\eta = 1$, $\mu = 1$, and $\tau = 1$. The state set

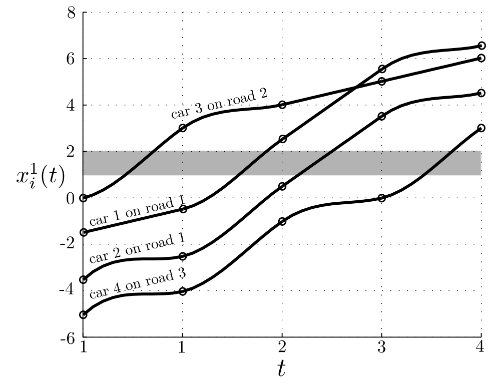

Fig. 2. The components of the trajectory of 4 cars on 3 roads (top), with $\alpha = 1$, $\beta = 2$ (equal for all cars), $d = 1$, $\eta = 1$. The gray area is the crossing $\alpha < x_i^1 < \beta$. Two components corresponding to vehicles on different roads must never be simultaneously in the crossing, and two components corresponding to vehicles on the same road must never be closer than $d = 1$. The initial conditions are: $x_i^1(0) = (0, -1.5, -3.5, -5)$, $x_i^2(0) = (1, 1, 4, 4)$

G of Σ_{DE} is a lattice over the set Y of the positions of all vehicles, and a transition exists from $g^1 \in G$ to $g^2 \in G$ if there is a vector u with $u_i \in u_{adm}$ such that $g^2 = g^1 + u\tau$.

One can easily prove that

$$\begin{aligned}&\sup_{u\in\mathcal{U}} \sup_{t>0} \inf_{s>0} \|y(t) - \phi_s(\chi(0), u)\| < \\ &\sup_{u\in\mathcal{U}} \sup_{t>0} \|y(t) - \phi_t(\chi(0), u)\| \leq \\ &\sup_{u\in\mathcal{U}} \left\|x^2(0) - u(\tau)\right\| \frac{4\tau}{27} < 0.45.\end{aligned}$$

Thus $\epsilon_a < 0.45$. If σ_C ensures 0.45-safety, the supervisor σ ensures 0-safety. We have applied the algorithm detailed above to the case of 4 vehicles driving along 3 roads. The different components of the trajectory of (13) are portrayed in Figure 2. The trajectories lie outside of the bad set as long as any two components corresponding to vehicles on different roads are never in the gray region simultaneously, and any two components corresponding to vehicles on the same road maintain a distance greater than $d = 1$. For a given number of cars the underlying discrete event system can be computed off-line, reducing the online computation to a simple table lookup.

VI. CONCLUSION

We have proposed an algorithm for the supervisory control of differentially flat systems, using model abstraction. By merging the trajectory planning techniques allowed by the flatness property of our system, and the abstraction approach, we have obtained an algorithm that can handle relatively large systems, yet it provides guarantees on the safety of the allowed trajectories. Our approach starts by considering a simple system living in the set of the flat output variables (the trivial system (3)). We construct an abstraction of this system, which by virtue of the simple dynamics can be made deterministic, and design a supervisor based on the abstraction. The control inputs allowed by the supervisor are then mapped back onto the original model using flatness. The allowed trajectories are safe by construction. The algorithm can be applied to differentially flat systems in the form specified in Section II, as long as the bad set is expressed in terms of the output variables. The bad set is an arbitrary but finite union of polytopes.

Unlike previous approaches [7], [3], [4], [5], [6], our technique provides a deterministic abstraction without requirements on the system's stability. This is a definite advantage when the complexity of the abstraction is a bottleneck. Moreover, unlike all these approaches our algorithm provides safety guarantees for the allowed trajectories.

Our approach is similar to a roadmap-based motion planning algorithm [13]. However, rather than a single trajectory, our algorithm provides a set of acceptable trajectories. Since the supervisor σ_C defined in (7) is the least restrictive, this is the largest possible set of trajectories for the trivial system, given the chosen discretization. Moreover, we can provide a minimum distance between the allowed trajectories and the bad set, that can be made arbitrarily small by refining the space and time discretization steps.

With our approach, we have reduced the computational burden of safety control of a continuous-time system. We have succeeded by shifting the problem to a discrete one. The complexity is exponential, but since the abstraction is finite the solution is tractable, whereas known solutions by standard control methods in the continuous domain are prohibitive. A decisive step forward will be the exploitation of the geometric and dynamic properties of a system to reduce the complexity class of the discrete algorithm.

REFERENCES

[1] P. J. Ramdage and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Contr. Opt.*, vol. 25, pp. 206–230, 1987.
[2] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Springer-Verlag, 2008.
[3] P. Tabuada, "An approximate simulation approach to symbolic control," *IEEE Trans. Autom. Control*, vol. 53, pp. 1406–1418, 2008.
[4] ——, *Verification and control of hybrid systems*. Springer-Verlag, 2009.
[5] 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.
[6] M. Zamani, G. Pola, M. Mazo Jr., and P. Tabuada, "Symbolic models for nonlinear control systems without stability assumptions," *arXiv:1002.0822v3*, 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] 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.
[9] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "Flatness and defect of non-linear systems: Introductory theory and examples," *Int. J. Control*, vol. 6, pp. 1327–1361, 1995.
[10] M. van Nieuwstadt, M. Rathinam, and R. M. Murray, "Differential flatness and absolute equivalence of nonlinear control systems," *SIAM J. Contr. Opt.*, vol. 36, pp. 1225–1239, 1998.
[11] J. Lévine, *Analysis and control of nonlinear systems: A flatness-based approach*. Springer, 2009.
[12] P. Tabuada, "Flatness and finite bisimulations in discrete time," in *Sixteenth International Symposium on Mathematical Theory of Networks and Systems*, 2004.
[13] J. C. Latombe, *Robot motion planning*. Kluwer Academic Publishers, 1991.