Proceedings of the
46th IEEE Conference on Decision and Control
New Orleans, LA, USA, Dec. 12-14, 2007

Observer-based Control for Block-triangular Hybrid Automata

Domitilla Del Vecchio
University of Michigan, USA

Abstract— **The safety control problem for the class of block-triangular order preserving hybrid automata with imperfect state information is addressed. A dynamic feedback law is proposed, which exploits the order preserving properties of the dynamics to construct state estimation and control algorithms that have linear complexity in the number of variables. The proposed algorithms are applied to a collision avoidance problem arising in the context of intelligent transportation.**

I. I

The problem addressed in this paper is the dynamic control (state estimator plus control) of the parallel composition of a class of hybrid automata (triangular order preserving hybrid automata) under safety specifications. Motivating applications both for the model and for the problem considered include multi-agent hybrid systems such as intelligent transportation systems and railway control systems. In these systems, each agent (a vehicle) can be modeled as a hybrid automaton, in which the continuous state dynamics has triangular structure and models the physical motion of the agent. The discrete state can model a control mode in which the agent can be (turning, accelerating, run-out, etc.) or it can model input and state constraints. The entire system is given as the parallel composition of the component systems modeling the agents. In particular, one problem for which automated solutions are sought [1], [2] is the collision prediction and avoidance at traffic intersections and at railway mergings. In these systems, the state (speed and position, for example) is known to the controller only within some uncertainty bounds. This uncertainty is due to measurement errors or to missing measurements as it happens for example with the position measurement obtained by the Global Positioning System (GPS).

The control problem under safety specifications assuming perfect state information has been addressed by several researchers (see [3]–[5], for example). There is a large body of literature about safety control design and the list here provided is not exhaustive. In these works, the safety control problem has been addressed by computing the set of states that lead to an unsafe configuration independently of an input choice (called the backward reachable set [3] or the uncontrollable predecessor [6] of an unsafe set). Then, a feedback is computed that guarantees that the state never enters such a set. As it appears from these works, a bottleneck in solving this problem is complexity. For classes of hybrid automata for which the continuous dynamics reachable set can be computed, computational constraints usually limit the system to four or five continuous variables and to two or three discrete states. Furthermore, the proposed algorithms are not guaranteed to terminate [3], [4]. To reduce the computational load, approximate algorithms have been proposed to compute an over-approximation of the backward reachable set of the unsafe set [7]–[9]. The safety control problem with imperfect or partial state information has been scarcely addressed in the literature. Pioneering work in this direction can be found in [10] and in [11], for example. In [10], a controller that relies on a state estimator is proposed for finite state systems. The results are then extended to control a class of rectangular hybrid automata with imperfect state information. The proposed algorithm has exponential complexity in the size of the system. In [11], a partial order approach is proposed for the design of computationally efficient state estimation and control algorithms. In such work, only discrete dynamic feedback is considered. Also, no algorithm is provided to compute the set of initial states in the state space from which a state estimation-based control strategy would work.

In this paper, we propose a solution for the dynamic control of block triangular order preserving hybrid automata under imperfect state information. The proposed approach relies on an approximate computation of the uncontrollable predecessor as performed in [7]–[9]. However, a control law that relies on a state estimator is constructed to handle imperfect state information. The state estimation and control algorithms are constructed exploiting the partial order structure associated with the system dynamics. By virtue of this structure, our algorithms have linear complexity with the number of variables.

The contents of this paper are as follows. In Section II, we provide some preliminary notions on partial orders, transition systems, and we define the escape set. In Section III, we introduce the block triangular order preserving hybrid automaton model. In Section IV, we introduce the state estimator while the dynamic controller is constructed in Section V. Finally, Section VI proposes two application examples arising in the context of intelligent transportation.

II. P , ,

A partial order [12] is a set P with a partial order relation "$\leq$", and we denote it by the pair $(P, \leq)$. For all $x, w \in P$, the $\sup\{x, w\}$, denoted $x \vee w$, is the smallest element that is larger than both x and w. The $\inf\{x, w\}$, denoted $x \wedge w$, is the largest element that is smaller than both x and w. If $S \subseteq P$, $\bigvee S := \sup S$ and $\bigwedge S := \inf S$. If $x \wedge w \in X$ and $x \vee w \in X$ for all $x, w \in X$, then $(X, \leq)$ is a *lattice*. Any interval sublattice of $(P, \leq)$ is given by $[L, U] = \{w \in P \mid L \leq$

This work was in part supported by the Crosby Award at University of Michigan and by the NSF CAREER award number CNS-0642719.

1-4244-1498-9/07/$25.00 ©2007 IEEE.

$w \leq U\}$ for $L, U \in P$. That is, this special sublattice can be represented by only two elements. Let $(P, \leq)$ and $(Q, \leq)$ be partially ordered sets. A map $f : P \to Q$ is (i) an *order preserving map* if $x \leq w \implies f(x) \leq f(w)$; (ii) an *order isomorphism* if $x \leq w \iff f(x) \leq f(w)$ and it maps P *onto* Q; (iii) *order continuous* if $f(\bigvee S) = \bigvee f(S)$ and $f(\bigwedge S) = \bigwedge f(S)$ for $S \subseteq P$. An order isomorphism is always order continuous. A particular partial order that we will consider in the sequel, is the power set of a set S, that is, the set of all subsets of S, denoted 2^S, ordered according to inclusion relation. This partial order will be denoted by $(2^S, \subseteq)$.

We introduce the escape set for the general modeling formalism of transition systems (see [6], for example) as the notion of escape set is independent on whether the system has continuous or discrete variables. A *transition system with output* is a tuple $\Sigma = (S, \mathcal{I}, \mathcal{Y}, \tau, \gamma)$, in which S is a (possibly infinite) set of states, $\mathcal{I}$ is a set of inputs, $\mathcal{Y}$ is a set of outputs, $\tau : S \times \mathcal{I} \to S$ is a transition map, and $\gamma : \mathcal{Y} \to 2^S$ is the output map. We denote a state by $s \in S$ and an input by $u \in \mathcal{I}$. If a state $s \in \gamma(y)$, we say that s is compatible with measurement y. Also, we will say that an output measurement y is compatible with state s if $s \in \gamma(y)$. An execution of Σ is an infinite sequence $\{s^k\}_{k\in\mathbb{N}}$ such that $s^{k+1} = \tau(s^k, u^k)$ for $u^k \in \mathcal{I}$. An input sequence is denoted by $\{u^k\}_{k\in\mathbb{N}}$ and an output sequence by $\{y^k\}_{k\in\mathbb{N}}$, in which y^k is such that $s^k \in \gamma(y^k)$.

For transition system Σ, we define the operator $\hat{\tau} : 2^S \times \mathcal{I} \times \mathcal{Y} \to 2^S$ as follows. Let $\hat{s} \subseteq S$, then $\hat{\tau}(\hat{s}, u, y) := \tau(\hat{s}, u) \cap \gamma(y)$. Given an output sequence of Σ, $\{y^k\}_{k\in\mathbb{N}}$, corresponding to the execution $\{s^k\}_{k\in\mathbb{N}}$, the set of all states at step k that are compatible with such output sequence up to step k and with the system transition map is given by

$$\begin{aligned} \hat{s}^{k+1} &= \hat{\tau}(\hat{s}^k, u^k, y^{k+1}) \\ \hat{s}^0 &= \gamma(y^0). \end{aligned} \tag{1}$$

One can verify that $s^k \in \hat{s}^k$ for all k. We refer to equations (1) as a *state estimator* for the transition system Σ. A *state estimator-based control strategy* is one in which the control law depends on $\hat{s}$, that is, $u = g(\hat{s})$ for $\hat{s} \subseteq S$. We define the notation $\hat{\tau}^{n+1}(\hat{s}, \{u^k\}_{k<n+1}, \{y^k\}_{k\leq n+1}) := \hat{\tau}(\hat{\tau}^n(\hat{s}, \{u^k\}_{k<n}, \{y^k\}_{k\leq n}), u^n, y^{n+1})\hat{\tau}^0(\hat{s}, \{u^k\}_{k<0}, \{y^k\}_{k\leq 0}) = \gamma(y^0)$.

Definition 1: Let $B \subseteq S$ be a bad set of states. The *escape set* $\mathcal{E}$ for system $\Sigma = (S, \mathcal{I}, \mathcal{Y}, \tau, \gamma)$ is defined as

$$\mathcal{E} = \{X \in 2^S \mid \forall\ \{u^k\}_{k\in\mathbb{N}},\ \exists\ N \text{ and } \{y^k\}_{k\leq N}, \text{ such that } \hat{\tau}^N(X, \{u^k\}_{k<N}, \{y^k\}_{k\leq N}) \cap B \neq \emptyset\}.$$

$\mathcal{E}$ is the set of all subsets of S such that if the state estimator is initialized with one of such subsets of S, then there will be an output sequence for which the state estimate at a later time will intersect the bad set no matter what input sequence is applied to the system. This definition is analogous to the one in [10]. To maintain safety, we must thus guarantee that the state estimator (1) will never have as a state one of the sets in $\mathcal{E}$. For a system Σ with only discrete states, set $\mathcal{E}$ can be computed in a finite number of steps [10], but the algorithm computation scales exponentially with the number of states. For a system Σ with also continuous states, the computation of $\mathcal{E}$ is impractical because the sets are infinite. Therefore, instead of considering $\mathcal{E} \in 2^S$, we consider another set. That is, we consider the smallest set $E \subseteq S$ containing B, if it exists, with the property that if the state estimate does not intersect it, there is an input sequence that will guarantee that the state estimate will never intersect E at a later time. This is formally stated as follows.

Problem 1: (State estimator-based safety control problem) Determine the smallest set $E \subseteq S$ with $B \subseteq E$, if it exists, and a dynamic feedback law $u^k = g(\hat{s}^k)$, with $\hat{s}^k = \hat{\tau}(\hat{s}^{k-1}, u^{k-1}, y^k)$, with $\hat{s}^0 = \gamma(y^0)$ and with $\{y^k\}_{k\in\mathbb{N}}$ output sequence of Σ such that if $\hat{s}^0 \cap E = \emptyset$ then $\hat{s}^k \cap E = \emptyset$ for all k.

Since $s^k \in \hat{s}^k$ for all k, the dynamic control law given in Problem 1 guarantees that the state of the system never enters the bad set B. In general, the set E could be the entire space S even if $\mathcal{E}$ is not equal to the entire 2^S. For the class of systems that we consider here, this will not be the case. In particular, we will show that we can have an overapproximation of E, called $\overline{E}$, to be the same as the one obtained for the full information case, which was shown to be a tight overapproximation of the escape set for the full information case [13].

III. B

We start by defining the discrete time hybrid automaton in a way analogous to the continuous time counterpart [3].

Definition 2: A *discrete time hybrid automaton* is a tuple $H = (Q, X, \mathcal{I}, \iota, \mathcal{Y}, f, Dom, R, \gamma)$, in which $Q = \{q_1, ..., q_m\}$ is a set of discrete states (or modes); $X = \mathbb{R}^p$ is the set of continuous states; $\mathcal{I} = \mathcal{I}_D \times \mathcal{I}_C$, is the set of discrete and continuous inputs, respectively; $\iota : Q \to 2^{\mathcal{I}}$ is a function that attaches to each discrete state the set of enabled inputs; $\mathcal{Y}$ is a set of outputs; $f : Q \times X \times \mathcal{I}_C \to X$ is the continuous state update function; $Dom : Q \to 2^X$ is a map that for each mode establishes the domain in X in which such mode holds; $R : Q \times X \times \mathcal{I}_D \to Q$ is the discrete state update map, which for any current discrete state, continuous state, and input determines the new discrete state; $\gamma : \mathcal{Y} \to 2^X$ is the output map.

We denote by $q \in Q$ the mode, by $x \in X$ the continuous state, by $u \in \mathcal{I}_C$ the continuous input, and by $\sigma \in \mathcal{I}_D$ the discrete input. We assume that the reset function is static, that is, it does not contain memory of previous discrete states. Thus, we have that $q = R(x, \sigma)$. We make an explicit distinction between two types of modes: the modes q such that $Dom(q) = \mathbb{R}^p$ and the modes q such that $Dom(q) \neq \mathbb{R}^p$. In particular, we assume that a transition to a mode with $Dom(q) = \mathbb{R}^p$ can happen *only* by a suitable choice of discrete input $\sigma \in \mathcal{I}_D$, while a transition to a mode with $Dom(q) \neq \mathbb{R}^p$ can happen only autonomously and thus cannot be controlled. This is formalized by the following structure of R:

$$R(x, \sigma) := \begin{cases} R(\sigma) & \text{if } \sigma \neq \emptyset \\ R(x) & \text{if } \sigma = \emptyset, \end{cases} \tag{2}$$

in which we define $R(x) := q$ if $x \in Dom(q)$. One can verify that this update is *deterministic* if $Dom(q_1) \cap Dom(q_2) = \emptyset$

whenever $Dom(q_1) \neq \mathbb{R}^p$ and $Dom(q_2) \neq \mathbb{R}^p$. Also, we assume that for any mode with $Dom(q) = \mathbb{R}^p$, there exists a discrete input $\sigma \in \mathcal{I}_D$ such that $q = R(\sigma)$. The *non-blocking* condition can be guaranteed if $\bigcup_{\{q \mid Dom(q) \neq \mathbb{R}^p\}} Dom(q) = \mathbb{R}^p$. In the sequel, we use the notation $\overline{Q} := \{q \in Q \mid Dom(q) \neq \mathbb{R}^p\}$. Hybrid automaton H corresponds to the transition system $\Sigma_H = (S, \mathcal{I}, \mathcal{Y}, \tau, \gamma)$, in which $S = X, \mathcal{I} = \mathcal{I}_D \times \mathcal{I}_C$. An input $\overline{u} \in \mathcal{I}$ is a pair $\overline{u} = (u, \sigma)$, in which $u \in \mathcal{I}_C$ and $\sigma \in \mathcal{I}_D$. The transition map is given by $\tau(x, \overline{u}) := f(R(x, \sigma), x, u)$ with $q = R(x, \sigma)$ and $u \in \iota(q)$. Finally, $\gamma : \mathcal{Y} \to 2^X$. Thus, the definition of escape set for H is the same as the definition of escape set for transition system Σ_H as given in Definition 1.

Definition 3: Let $(\mathbb{R}^n, \leq)$ be the partial order established according to component-wise ordering. A *triangular order preserving hybrid automaton* is a hybrid automaton $H = (Q, X, \mathcal{I}, \iota, \mathcal{Y}, f, Dom, R, \gamma)$, in which

(i) The update map $f(q, x, u)$ for every $q \in Q$ and $x = (x_1, ..., x_n) \in \mathbb{R}^n$ has the following triangular structure $f(q, x, u) = (f_1(x_1, ..., x_n), ..., f_i(x_i, ..., x_n), ..., f_n(x_n, q, u))$, in which $f_i : \mathbb{R}^{n-(i-1)} \to \mathbb{R}$ for $i \in \{1, ..., n-1\}$, $f_n : \mathbb{R} \times Q \times \mathcal{I}_C \to \mathbb{R}$ with $\mathcal{I}_C = \mathbb{R}$, and $Dom(q) \subseteq \mathbb{R}^n$.

(ii) We assume that the set of discrete states with $Dom(q) = \mathbb{R}^n$ is a lattice with minimum α and with maximum β, that is, $\{q \in Q \mid Dom(q) = \mathbb{R}^n\} = [\alpha, \beta]$. For all $q \in Q$, we assume that $\iota(q)$ is an interval in $\mathbb{R}$, that is, $\iota(q) = [u_L(q), u_U(q)]$. Also, the functions $u_L(\cdot)$ and $u_U(\cdot)$ are order preserving in q with $Dom(q) = \mathbb{R}^n$.

(iii) We assume that f_i is order preserving in all of its arguments, that is, if $(x_i^a, ..., x_n^a) \leq (x_i^b, ..., x_n^b)$ then $f_i(x_i^a, ..., x_n^a) \leq f_i(x_i^b, ..., x_n^b)$ for $i < n$, and $f_n(x_n^a, q, u) \leq f_n(x_n^b, q, u)$. Also, $f_n : Q|_{\{q \in Q \mid Dom(q) = \mathbb{R}^n\}} \times \mathbb{R} \times \mathcal{I}_C \to \mathbb{R}$ is order preserving in all of its arguments. Additionally, f_i is one-one and onto in x_i, that is, fixed $x_{i+1}, .., x_n, q, u$, for any x_i' there is one and only one x_i such that $f_i(x_i, ..., x_n) = x_i'$ if $i < n$ or $f_i(x_i, q, u) = x_i'$ if $i = n$. We denote the first one by $f_i^{-1}(x_i', x_{i+1}, ..., x_n)$ and the second one by $f_i^{-1}(x_i', q, u)$.

(iv) The maps f_i are non-decreasing: $f_i(x_i, ..., x_n) \geq x_i$, for $i < n$ and $f_n(x_n, q, u_U(q)) > x_n$ for all q.

(v) For all $y \in \mathcal{Y}$, the set $\gamma(y) \subseteq \mathbb{R}^n$ is an interval in $(\mathbb{R}^n, \leq)$, that is, $\gamma(y) = [\bigwedge \gamma(y), \bigvee \gamma(y)]$.

Item (v) of the above definition implies that there is a bounded measurement uncertainty for each continuous variable. The parallel composition of a number of triangular order preserving hybrid automata generates a block-triangular order preserving hybrid automaton. This is made more precise by defining the parallel composition of hybrid automata in a way similar to [14].

Definition 4: Let $H_1 = (Q_1, X_1, \mathcal{I}_1, \iota_1, \mathcal{Y}_1, f_1, Dom_1, R_1, \gamma_1)$ and $H_2 = (Q_2, X_2, \mathcal{I}_2, \iota_2, \mathcal{Y}_2, f_2, Dom_2, R_2, \gamma_2)$ be two hybrid automata. The parallel composition, denoted $H = H_1 \| H_2$, is given by $H = (Q, X, \mathcal{I}, \iota, \mathcal{Y}, f, Dom, R, \gamma)$, in which $Q = Q_1 \times Q_2$, $X = X_1 \times X_2$, $\mathcal{I} = \mathcal{I}_C \times \mathcal{I}_D$ with $\mathcal{I}_C = \mathcal{I}_{C,1} \times \mathcal{I}_{C,2}$ and $\mathcal{I}_D = \mathcal{I}_{D,1} \times \mathcal{I}_{D,2}$; $\iota : Q \to \mathcal{I}_C$ is given by $\iota = (\iota_1, \iota_2)$; $\mathcal{Y} = \mathcal{Y}_1 \times \mathcal{Y}_2$; $f : Q \times X \times \mathcal{I}_C \to X$ is given by $f = (f_1, f_2)$; $Dom(q) = Dom_1(q_1) \times Dom_2(q_2)$; $R(x, \sigma) = (R_1(x_1, \sigma_1), R_2(x_2, \sigma_2))$; $\gamma = (\gamma_1, \gamma_2)$.

Definition 5: A *block triangular order preserving hybrid automaton* is the parallel composition of N triangular order preserving hybrid automata $H_1, ..., H_N$.

Let $x_i = (x_{1,i}, ..., x_{n,i}) \in \mathbb{R}^n$, $q_i \in Q_i$, $u_i \in \iota(q_i)$, $\sigma_i \in \mathcal{I}_{D,i}$ represent the continuous state, the discrete state, the continuous input, and the discrete input of the triangular hybrid automaton H_i, respectively. Then, in each mode $q = (q_1, ..., q_N)$ of the hybrid automaton $H = H_1 \| ... \| H_N$, the continuous state update map has the following form

$$\begin{aligned} x'_{j,i} &= f_{j,i}(x_{j,i}, ..., x_{n,i}),\ j < n\ i \in \{1, ..., N\} \\ x'_{n,i} &= f_{n,i}(x_{n,i}, q_i, u_i),\ i \in \{1, ..., N\}, \end{aligned} \tag{3}$$

in which primed variables denote updated variables. In the sequel, we will use the notation $f_i(q_i, x_i, u_i) = (f_{1,i}(x_{1,i}, ..., x_{n,i}), ..., f_{n,i}(q_i, x_{n,i}, u_i))$. For this system, we model the safety requirement by requesting that the state x never enter the bad set

$$\begin{aligned} B &= \{(x_{1,1}, ..., x_{n,1}, ..., x_{1,N}, ..., x_{n,N}) \mid (x_{1,1}, ..., x_{1,N}) \in \overline{B}\}, \\ \overline{B} &= [L_1, U_1] \times ... \times [L_N, U_N],\ \text{with } L_i, U_i \in \mathbb{R}. \end{aligned} \tag{4}$$

In the sequel, we denote $L = (L_1, ..., L_N)$ and $U = (U_1, ..., U_N)$. This choice of the safety requirement to involve only the variables $(x_{1,1}, ..., x_{1,N})$ is motivated by the applications that we are targeting (see Section VI).

To the block triangular order preserving hybrid automaton, the following transition system corresponds. Let $H = H_1 \| ... \| H_N$ be the block triangular order preserving hybrid automaton with $H_i = (Q_i, X_i, \mathcal{I}_i, \iota_i, \mathcal{Y}_i, f_i, Dom_i, R_i, \gamma_i)$. Let $\Sigma_{H_i} = (X_i, \mathcal{I}_i, \mathcal{Y}_i, \tau_i, \gamma_i)$ be the transition system associated with H_i and $\Sigma_H = (X, \mathcal{I}, \mathcal{Y}, \tau, \gamma)$ be the one associated with H. We thus have that $\tau(x, \overline{u}) = (\tau_1(x_1, \overline{u}_1), ..., \tau_N(x_N, \overline{u}_N))$, in which $\tau_i(x_i, \overline{u}_i) = f_i(R_i(x_i, \sigma_i), x_i, u_i)$ with $x_i \in X_i$ and $\overline{u}_i = (\sigma_i, u_i) \in \mathcal{I}_i$. As a consequence, the escape set E for $H = H_1 \| ... \| H_N$ is the same as the escape set defined in Definition 1 for transition system Σ_H.

IV. S

Consider hybrid automaton H and let $\hat{x} \subseteq X$. A state estimator for H of the type of the one in equation (1) can take, for example, the form

$$\hat{x}' = f(\hat{q}, \hat{x}, u) \cap \gamma(y'),\ \text{with } \hat{q} = R(\hat{x}, \sigma), \tag{5}$$

$$\text{and } R(\hat{x}, \sigma) = \begin{cases} R(\sigma) & \text{if } \sigma \neq \emptyset \\ R(\hat{x}) & \text{if } \sigma = \emptyset, \end{cases}$$

in which $R(\hat{x}) = \{q \in \overline{Q} \mid \exists\ x \in \hat{x},\ \text{with},\ x \in Dom(q)\}$ and $\hat{x}^0 = \gamma(y^0)$. One can verify that $x^k \in \hat{x}^k$ for all k. This type of estimator is impractical for implementation because the sets $\hat{x}$ are in general infinite sets. However, since H is the parallel composition of order preserving (triangular) hybrid automata H_i, we have that the update maps f_i are order preserving and that $\gamma(y_i) = [\wedge\gamma(y_i), \vee\gamma(y_i)]$. As a consequence, one can keep track of the lower and upper bounds of $\hat{x}$ as follows. Let $\vee\hat{x} = (\vee\hat{x}_1, ..., \vee\hat{x}_N)$ and $\wedge\hat{x} = (\wedge\hat{x}_1, ..., \wedge\hat{x}_N)$, denote the

upper and lower bounds of $\hat{x}$, respectively. Then, we have that $\vee\hat{x}_i = (\vee\hat{x}_{1,i}, ..., \vee\hat{x}_{n,i}) \in \mathbb{R}^n$ and $\wedge\hat{x}_i = (\wedge\hat{x}_{1,i}, ..., \wedge\hat{x}_{n,i}) \in \mathbb{R}^n$ are the lower and the upper bounds of $\hat{x}_i$, respectively, in which $\hat{x}_i \subseteq \mathbb{R}^n$ is the state estimate of the component automaton H_i. Then, the bounds of $\hat{x}_i$ for all i are updated according to the following equations

$$\text{if } \sigma_i \neq \emptyset \begin{cases} \wedge\hat{x}_i' = f_i(\wedge\hat{x}_i, R(\sigma_i), u_i) \vee \wedge\gamma(y_i') \\ \vee\hat{x}_i' = f_i(\vee\hat{x}_i, R(\sigma_i), u_i) \wedge \vee\gamma(y_i') \end{cases} \tag{6}$$

$$\text{if } \sigma_i = \emptyset \begin{cases} \wedge\hat{x}_i' = \bigwedge_{q_i \in \hat{\overline{Q}}_i} f_i(\wedge\hat{x}_i, q_i, u_i) \vee \wedge\gamma(y_i') \\ \vee\hat{x}_i' = \bigvee_{q_i \in \hat{\overline{Q}}_i} f_i(\vee\hat{x}_i, q_i, u_i) \wedge \vee\gamma(y_i'), \end{cases} \tag{7}$$

in which y_i' is the output observation of H_i, $\hat{\overline{Q}}_i$ is the set of possible modes that are compatible with the interval of states $[\wedge\hat{x}_i, \vee\hat{x}_i]$, that is,

$$\hat{\overline{Q}}_i = \{q_i \in \overline{Q}_i \mid \exists\ x_i \in [\textstyle\bigwedge\hat{x}_i, \bigvee\hat{x}_i], \text{ such that } x_i \in Dom_i(q_i)\}, \tag{8}$$

$$\textstyle\bigwedge\hat{x}_i^0 = \bigwedge\gamma(y_i^0) \qquad \bigvee\hat{x}_i^0 = \bigvee\gamma(y_i^0). \tag{9}$$

Proposition 1: Let $\{u^k\}_{k\in\mathbb{N}}$ be an input sequence for the block triangular order preserving hybrid automaton $H = H_1\|...\|H_N$, and let $\{x^k\}_{k\in\mathbb{N}}$ and $\{y^k\}_{k\in\mathbb{N}}$ be the corresponding execution and output sequence. Let $\{\wedge\hat{x}^k\}_{k\in\mathbb{N}}$ and $\{\vee\hat{x}^k\}_{k\in\mathbb{N}}$ with $\vee\hat{x} = (\vee\hat{x}_1, ..., \vee\hat{x}_N)$ and $\wedge\hat{x} = (\wedge\hat{x}_1, ..., \wedge\hat{x}_N)$ be generated by equations (6-9). Then, $x^k \in [\wedge\hat{x}^k, \vee\hat{x}^k]$ for all k.

The proof of this proposition is a consequence of the order preserving property of f_i and of the interval structure of $\gamma(y_i)$. For more details on these types of estimators and for convergence conditions, the reader is referred to [15].

V. C

To solve Problem 1, we compute set $\overline{E}$ as follows. Denote $F_i(x_{2,i}, ..., x_{n,i}, q_i, u_i) := f_{2,i}(x_{2,i}, ..., x_{n,i}), ..., f_{n,i}(x_{n_i}, q_i, u_i)$ and $\overline{x}_i = (x_{2,i}, ..., x_{n,i})$. Let $F_i^k(\overline{x}_i, q_i, u_i) := F(F^{k-1}(\overline{x}_i, q_i, u_i), q_i, u_i)$ and define $\overline{x} = (\overline{x}_1, ..., \overline{x}_N)$. In particular, we have that $\overline{E} = \{x \mid (x_{1,1}, ..., x_{1,N}) \in \overline{E}^*(\overline{x})\}$, in which $\overline{E}^*(\overline{x})$ is given by the following algorithm.

Algorithm 1.

$\overline{E}^*(\overline{x}) = \bigcup_{k\geq 0}[\overline{L}^k(\overline{x}), \overline{U}^k(\overline{x})]$, $\overline{L}^0(\overline{x}) = L$, $\overline{U}^0(\overline{x}) = U$, $\overline{L}^k = (\overline{L}_1^k(\overline{x}_1), ..., \overline{L}_N^k(\overline{x}_N))$, $\overline{U}^k = (\overline{U}_1^k(\overline{x}_1), ..., \overline{U}_N^k(\overline{x}_N))$ with

$$\begin{aligned} \overline{L}_i^1(\overline{x}_i) &= f_{1,i}^{-1}(\overline{L}_i^0, \overline{x}_i) \\ \overline{U}_i^1(\overline{x}_i) &= f_{1,i}^{-1}(\overline{U}_i^0, \overline{x}_i), \end{aligned}$$

while for $k > 1$, we have

$$L_i^k(\overline{x}_i) = L_i^{k,a}(\overline{x}_i) \vee L_i^{k,b}(\overline{x}_i) \tag{10}$$

$$L_i^{k,a}(\overline{x}_i) = \bigwedge_{q_i \in \overline{Q}_i} f_{1,i}^{-1}(\overline{L}_i^{k-1}(F_i(\overline{x}_i, q_i, u_L(q_i))), \overline{x}_i) \tag{11}$$

$$L_i^{k,b}(\overline{x}_i) = f_{1,i}^{-1}(\overline{L}_i^{k-1}(F_i(\overline{x}_i, \alpha_i, u_L(\alpha_i))), \overline{x}_i) \tag{12}$$

$$U_i^k(\overline{x}_i) = U_i^{k,a}(\overline{x}_i) \wedge U_i^{k,b}(\overline{x}_i) \tag{13}$$

$$U_i^{k,a}(\overline{x}_i) = \bigvee_{q_i \in \overline{Q}_i} f_{1,i}^{-1}(\overline{U}_i^{k-1}(F_i(\overline{x}_i, q_i, u_U(q_i))), \overline{x}_i) \tag{14}$$

$$U_i^{k,b}(\overline{x}_i) = f_{1,i}^{-1}(\overline{U}_i^{k-1}(F_i(\overline{x}_i, \beta_i, u_U(\beta_i))), \overline{x}_i) \tag{15}$$

with for $k < k^*$ (removing the dependence on $\overline{x}_i$ for shortness of notation)

$$\overline{L}_i^k = \inf(L_i^k, \overline{L}_i^{k-1}) \tag{16}$$

$$\overline{U}_i^k = \begin{cases} \sup(U_i^k, \overline{L}_i^{k-1}), & \text{if } \exists\ j \text{ such that } U_j^k > \overline{L}_j^{k-1} \\ U_i^k, & \text{if } U_j^k \leq \overline{L}_j^{k-1}\ \forall\ j, \end{cases} \tag{17}$$

with k^* the smallest k such that

$$U_i^k \leq \overline{L}_i^{k-1}\ \forall\ i \text{ and } \exists\ j \text{ such that } \overline{U}_j^{k+1} < \overline{L}_j^{k+1}.$$

For $k \geq k^*$, we instead define

$$\overline{U}_i^{k+1} = \sup(U_i^{k+1}, L_i^{k+1}) \tag{18}$$

$$\overline{L}_i^{k+1} = \inf(U_i^{k+1}, L_i^{k+1}). \tag{19}$$

□

If k^* is finite, it means that the iteration of Algorithm 1 terminates. For termination conditions, the reader is referred to [13]. In order to solve Problem 1, one needs to check whether $\overline{E} \cap \hat{x} = \emptyset$ and in such a case compute a controller that guarantees that $\overline{E} \cap \hat{x}' = \emptyset$. We then proceed by giving two results. The first result exploits the structure of the set $\overline{E}$ and of the set $\hat{x}$ to provide a simple check for determining whether $\overline{E} \cap \hat{x} = \emptyset$. The second result provides a controller such that if $\overline{E} \cap \hat{x} = \emptyset$ then $\overline{E} \cap \hat{x}' = \emptyset$. First, let us give the following intermediate result that states that whenever $x \notin \overline{E}$ (and thus $(x_{1,1}, ..., x_{1,N}) \notin \overline{E}^*(\overline{x}) \subseteq \mathbb{R}^N$) there is a two-dimensional projection of $\overline{E}^*(\overline{x}) \subseteq \mathbb{R}^N$ and of $(x_{1,1}, ..., x_{1,N})$ along coordinate axis (i, j) in $\mathbb{R}^N$, such that $(x_{1,i}, x_{1,j})$, is not contained in $\bigcup_{k=0}^{k^*}[\overline{L}_i^k(\overline{x}_i), \overline{U}_i^k(\overline{x}_i)] \times [\overline{L}_j^k(\overline{x}_j), \overline{U}_j^k(\overline{x}_j)]$.

Proposition 2: If $(x_{1,1}, ..., x_{1,N}) \notin \overline{E}^*(\overline{x})$ with $\overline{E}^*(\overline{x})$ as given in Algorithm 1, then there is a pair of coordinates (i, j) such that $(x_{1,i}, x_{1,j}) \notin [\overline{L}_i^k(\overline{x}_i), \overline{U}_i^k(\overline{x}_i)] \times [\overline{L}_j^k(\overline{x}_j), \overline{U}_j^k(\overline{x}_j)]$ for all k.

For the proof, the reader is referred to [13].

Proposition 3: We have that $[\wedge\hat{x}, \vee\hat{x}] \cap \overline{E} = \emptyset$ if and only if

$$[\textstyle\bigwedge\hat{x}_{1,1}, \bigvee\hat{x}_{1,1}] \times ... \times [\bigwedge\hat{x}_{1,N}, \bigvee\hat{x}_{1,N}] \cap \bigcup_{k\geq 0}[\overline{L}^k(\vee\hat{\overline{x}}), \overline{U}^k(\wedge\hat{\overline{x}})] = \emptyset.$$

Proof: ($\Leftarrow$) We show that if $[\wedge\hat{x}_{1,1}, \vee\hat{x}_{1,1}] \times ... \times [\wedge\hat{x}_{1,N}, \vee\hat{x}_{1,N}] \cap \bigcup_{k\geq 0}[\overline{L}^k(\vee\hat{\overline{x}}), \overline{U}^k(\wedge\hat{\overline{x}})] = \emptyset$, then $[\wedge\hat{x}, \vee\hat{x}] \cap \overline{E} = \emptyset$. This derives directly from the fact that $\overline{E} = \{x \mid (x_{1,1}, ..., x_{1,N}) \in \overline{E}^*(\overline{x})\}$, in which $\overline{E}^*(\overline{x})$ is given by Algorithm 1, and by the fact that the functions $\overline{L}_i^k(\overline{x}_i)$ and $\overline{U}_i^k(\overline{x}_i)$ are order reversing functions of their arguments.

($\Rightarrow$) We show that if $[\wedge\hat{x}, \vee\hat{x}] \cap \overline{E} = \emptyset$, then also $[\wedge\hat{x}_{1,1}, \vee\hat{x}_{1,1}] \times ... \times [\wedge\hat{x}_{1,N}, \vee\hat{x}_{1,N}] \cap \bigcup_{k\geq 0}[\overline{L}^k(\vee\hat{\overline{x}}), \overline{U}^k(\wedge\hat{\overline{x}})] = \emptyset$. If $[\wedge\hat{x}, \vee\hat{x}] \cap \overline{E} = \emptyset$ we have that all $x \in [\wedge\hat{x}, \vee\hat{x}]$ are not inside $\overline{E}$. If $x \notin \overline{E}$, it means that $(x_{1,1}, ..., x_{1,N}) \notin \bigcup_{k\geq 0}[\overline{L}^k(\overline{x}), \overline{U}^k(\overline{x})]$. By virtue of Proposition 2, we have that there is a pair of coordinates (i, j) such that $(x_{1,i}, x_{1,j}) \notin [\overline{L}_i^k(\overline{x}_i), \overline{U}_i^k(\overline{x}_i)] \times [\overline{L}_j^k(\overline{x}_j), \overline{U}_j^k(\overline{x}_j)]$ for all k. This means that for all k either $x_{1,i} > \overline{U}_i^k(\overline{x}_i)$ or $x_{1,i} < \overline{L}_i^k(\overline{x}_i)$ or either $x_{1,j} > \overline{U}_j^k(\overline{x}_j)$ or $x_{1,j} < \overline{L}_j^k(\overline{x}_j)$. Assume that for a given k $x_{1,i} > \overline{U}_i^k(\overline{x}_i)$

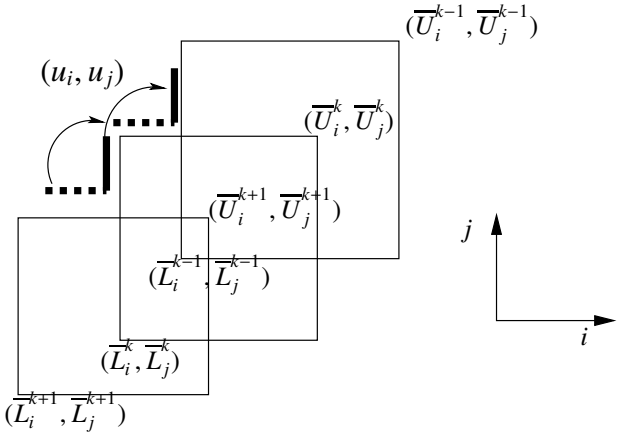

Fig. 1. Conceptual picture explaining how the controller exploits the order preserving property of the update map to maintain sets outside $\overline{E}$. The picture shows a slice of $\overline{E}$ for a specific value of $\overline{x}_i, \overline{x}_j$ in the $x_{1,i}, x_{1,j}$ plane. The points on the lower dashed line are such that $x_{1,j} > \overline{U}_j^{k+1}(\overline{x}_j)$. Thus, there is an input u_j that will map such points above $\overline{U}_j^k(\overline{x}_j')$, that is, $x_{1,j}' > \overline{U}_j^k(\overline{x}_j')$. The points on the lower solid line are such that $x_{1,i} < \overline{L}_i^k(\overline{x}_i)$. Thus, there is an input u_i that will map such points on the left of $\overline{L}_i^{k-1}(\overline{x}_i')$, that is, $x_{1,i}' < \overline{L}_i^{k-1}(\overline{x}_i')$.

and $x_{1,j} < \overline{L}_j^k(\overline{x}_j)$. Since this must be true for all $x_{1,i} \in [\wedge\hat{x}_{1,i}, \vee\hat{x}_{1,i}]$ and all $x_{1,j} \in [\wedge\hat{x}_{1,j}, \vee\hat{x}_{1,j}]$, then in particular we must have that $\wedge\hat{x}_{1,i} > \overline{U}_i^k(\hat{\wedge\overline{x}}_i)$ and $\vee\hat{x}_{1,j} < \overline{L}_j^k(\hat{\vee\overline{x}}_j)$. Since the same reasoning is true for all k, we have that $[\wedge\hat{x}_{1,1}, \vee\hat{x}_{1,1}] \times ... \times [\wedge\hat{x}_{1,N}, \vee\hat{x}_{1,N}] \cap [\overline{L}^k(\hat{\vee\overline{x}}), \overline{U}^k(\hat{\wedge\overline{x}})] = \emptyset$ for all k. This gives us the desired result. ■

As a consequence of Proposition 3, to compute an input that maintains the intersection $[\wedge\hat{x}^k, \vee\hat{x}^k] \cap \overline{E}$ empty at all time it is then sufficient (and necessary) to compute a control law such that whenever $[\wedge\hat{x}_{1,1}, \vee\hat{x}_{1,1}] \times ... \times [\wedge\hat{x}_{1,N}, \vee\hat{x}_{1,N}] \cap \bigcup_{k\geq 0}[\overline{L}^k(\hat{\vee\overline{x}}), \overline{U}^k(\hat{\wedge\overline{x}})] = \emptyset$ we have that also $[\wedge\hat{x}_{1,1}', \vee\hat{x}_{1,1}'] \times ... \times [\wedge\hat{x}_{1,N}', \vee\hat{x}_{1,N}'] \cap \bigcup_{k\geq 0}[\overline{L}^k(\hat{\vee\overline{x}'}), \overline{U}^k(\hat{\wedge\overline{x}'})] = \emptyset$. This control law can be easily computed by exploiting the order preserving properties of the map f_i. We first make the following assumption

Assumption 1: We assume that $\bigcap_{q_i\in\overline{Q}_i}[u_L(q_i), u_U(q_i)] \neq \emptyset$. Also let $\bigvee_{q_i\in\overline{Q}_i} F_i(\overline{x}_i, q_i, u_L(q_i)) = F_i(\overline{x}_i, q_{i,M}, u_L(q_{i,M}))$ in which $u_L(q_{i,M}) = \bigvee_{q_i\in\overline{Q}_i} u_L(q_i)$. Similarly, let $\bigwedge_{q_i\in\overline{Q}_i} F_i(\overline{x}_i, q_i, u_U(q_i)) = F_i(\overline{x}_i, q_{i,m}, u_U(q_{i,m}))$ in which $u_U(q_{i,m}) = \bigwedge_{q_i\in\overline{Q}_i} u_U(q_i)$.

Proposition 4: Let $\overline{L}_i^k(\overline{x}_i)$ and $\overline{U}_i^k(\overline{x}_i)$ be as in Algorithm 1 and let Assumption 1 hold. Let $\vee\hat{x}_i, \wedge\hat{x}_i \in \mathbb{R}^n$ and let $\vee\hat{x}_i', \wedge\hat{x}_i' \in \mathbb{R}^n$ be the updated values according to equations (6) and (7). If $\vee\hat{x}_{1,i} < \overline{L}_i^k(\hat{\vee\overline{x}}_i)$, $(\wedge\hat{x}_{1,i} > \overline{U}_i^k(\hat{\wedge\overline{x}}_i))$ then there exists a continuous/discrete control law (σ_i, u_i) such that $\vee\hat{x}_{1,i}' < \overline{L}_i^k(\hat{\vee\overline{x}'_i})$, $(\wedge\hat{x}_{1,i}' > \overline{U}_i^k(\hat{\wedge\overline{x}'_i}))$. In particular, such control laws are as follows:

$$\text{if } \bigvee\hat{x}_{1,i} < \overline{L}_i^k(\hat{\bigvee\overline{x}}_i), \text{ then}$$
$$\begin{cases} R_i(\sigma_i) = \alpha_i,\ u_i = u_L(\alpha_i) & \text{if } L_i^{k,a}(\hat{\vee\overline{x}}_i) < L_i^{k,b}(\hat{\vee\overline{x}}_i) \\ u_i = \bigvee_{q_i\in\hat{\overline{Q}}_i} u_L(q_i) & \text{if } L_i^{k,a}(\hat{\vee\overline{x}}_i) \geq L_i^{k,b}(\hat{\vee\overline{x}}_i) \end{cases} \quad (20)$$

$$\text{if } \bigwedge\hat{x}_{1,i} > \overline{U}_i^k(\hat{\bigwedge\overline{x}}_i),$$
$$\begin{cases} R_i(\sigma_i) = \beta_i,\ u_i = u_U(\beta_i) & \text{if } U_i^{k,a}(\hat{\wedge\overline{x}}_i) > U_i^{k,b}(\hat{\wedge\overline{x}}_i) \\ u_i = \bigwedge_{q_i\in\hat{\overline{Q}}_i} u_U(q_i) & \text{if } U_i^{k,a}(\hat{\wedge\overline{x}}_i) \leq U_i^{k,b}(\hat{\wedge\overline{x}}_i) \end{cases} \quad (21)$$

Proof: (sketch) We show that if $\wedge\hat{x}_{1,i} > \overline{U}_i^k(\hat{\wedge\overline{x}}_i)$ then there exists a continuous/discrete control law (σ_i, u_i) such that $\wedge\hat{x}_{1,i}' > \overline{U}_i^k(\hat{\wedge\overline{x}'_i})$ (the other case can be shown in a similar way). There are two cases: (1) $k < k^*$; (2) $k \geq k^*$.

Case (1): $k < k^*$. In such a case, the update law is the one in equation (17). If $\wedge\hat{x}_{1,i} > \overline{U}_i^k(\hat{\wedge\overline{x}}_i)$ and $U_i^{k,a} < U_i^{k,b}$, we will have that $\wedge\hat{x}_{1,i} > U_i^{k,a}(\hat{\wedge\overline{x}}_i)$. Applying $f_{1,i}$ both sides and taking into account that $f_{1,i}$ preserves the ordering, we obtain that $f_{1,i}(\wedge\hat{x}_i) > f_{1,i}(U_i^{k,a}(\hat{\wedge\overline{x}}_i), \hat{\wedge\overline{x}}_i)$. By expression (14), we have that $f_{1,i}(U_i^{k,a}(\hat{\wedge\overline{x}}_i), \hat{\wedge\overline{x}}_i) = \bigvee_{q_i\in\overline{Q}_i} \overline{U}_i^{k-1}(F_i(\hat{\wedge\overline{x}}_i, q_i, u_U(q_i))$. By the fact that $U_i(\cdot)$ are order reversing functions of their arguments, we have that $\bigvee_{q_i\in\overline{Q}_i} \overline{U}_i^{k-1}(F_i(\hat{\wedge\overline{x}}_i, q_i, u_U(q_i)) = \overline{U}_i^{k-1}(\bigwedge_{q_i\in\overline{Q}_i} F_i(\hat{\wedge\overline{x}}_i, q_i, u_U(q_i)))$. As a consequence, if we choose $u_i = \bigwedge_{q_i\in\hat{\overline{Q}}_i} u_U(q_i)$, we obtain that $\wedge\hat{x}_{1,i}' > \overline{U}_i^{k-1}(\hat{\wedge\overline{x}'_i})$ by virtue of Assumption 1. If instead $U_i^{k,a} \geq \overline{U}_i^{k,b}$, we will have that $\hat{x}_{1,i} > U_i^{k,b}(\hat{\wedge\overline{x}}_i)$. Applying $f_{1,i}$ both sides and taking into account that $f_{1,i}$ preserves the ordering, we obtain that $f_{1,i}(\wedge\hat{x}_i) > f_{1,i}(U_i^{k,b}(\hat{\wedge\overline{x}}_i), \hat{\wedge\overline{x}}_i)$. By expression (14), we have that $f_{1,i}(U_i^{k,b}(\hat{\wedge\overline{x}}_i), \hat{\wedge\overline{x}}_i) = \overline{U}_i^{k-1}(F_i(\hat{\wedge\overline{x}}_i, \beta_i, u_U(\beta_i))$. Therefore, choosing $R_i(\sigma_i) = \beta_i$ and $u_i = u_U(\beta_i)$, we obtain that $\wedge\hat{x}_{1,i}' > \overline{U}_i^{k-1}(\hat{\wedge\overline{x}'_i})$.

Case (2): $k \geq k^*$. In such a case, the update law is the one in equation (18). Since $\overline{U}_i^k = \sup(U_i^k, L_i^k)$ then $\wedge\hat{x}_{1,i} > \overline{U}_i^k(\hat{\wedge\overline{x}}_i)$ implies that also $\wedge\hat{x}_{1,i} > U_i^k(\hat{\wedge\overline{x}}_i)$ and the proof proceeds as in case (1). ■

The idea of this proposition is visualized in Figure 1. All points that have the *jth* coordinate larger than $\overline{U}_j^{k+1}$ can be mapped to points with *jth* coordinate larger than $\overline{U}_j^k$ by suitable choice of input u_j (or σ_j). Similarly, all points with *ith* coordinate smaller than $\overline{L}_i^k$ can be mapped to points with *ith* coordinate smaller than $\overline{L}_i^{k-1}$ by suitable choice of input u_i (or σ_i).

Algorithm 2.

(i) Evaluate all $\overline{L}_i^k(\hat{\vee\overline{x}}_i)$ and $\overline{U}_i^k(\hat{\wedge\overline{x}}_i)$ for all i and all k until $\wedge\hat{\overline{x}}_{1,i} > \overline{U}_i^k(\hat{\wedge\overline{x}}_i)$ for all i;

(ii) If there is a k and a pair of coordinates (i, j) such that $\wedge\hat{x}_{1,i} > \overline{U}_i^{k+1}(\hat{\wedge\overline{x}}_i)$ and $\vee\hat{x}_{1,j} < \overline{L}_j^k(\hat{\vee\overline{x}}_j)$ then set (σ_i, u_i) as in equation (21) with $k+1$ in place of k, and set (σ_j, u_j) as in equation (20);

Theorem 1: Algorithm 2 solves the state estimator-based control problem (Problem 1). Furthermore, Algorithm 2 terminates.

The fact that Algorithm 2 solves Problem 1 is a straightforward consequence of Proposition 2, Proposition 3, and Proposition 4. Algorithm 2 terminates if and only if (i) of Algorithm 2 terminates. Termination of step (i) is guaranteed by the fact that the sequence $\{\overline{U}_i^k(\hat{\wedge\overline{x}}_i)\}_{k\in\mathbb{N}}$ is strictly decreasing by virtue of the property (iv) of the model in Definition 3. The fact that $E \subseteq \overline{E}$ is implied by the above theorem.

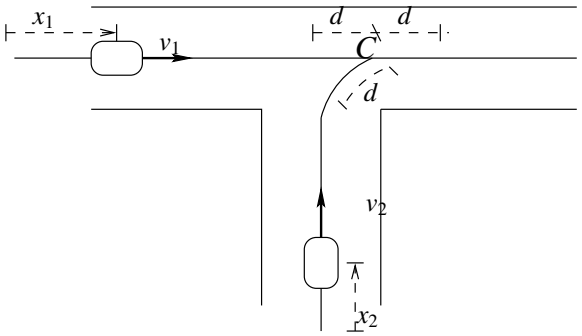

Fig. 2. Vehicles at a traffic intersection. The bad set is the set of all vehicle 1/vehicle 2 configurations in which the vehicles are both closer than some distance d from the intersection C of their paths. This corresponds to a rectangle in the x_1, x_2 plane.

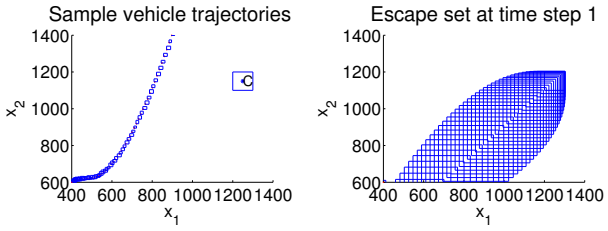

Fig. 3. Vehicles at a traffic intersection. On the left, we show a sample trajectory in the x_1, x_2 plane. The dots represent the position of the vehicles and the rectangle surrounding them is given by the state estimator (equations (7)). On the right, we show a slice of the set $\overline{E}$ at the initial time for the initial values of v_1, v_2.

Remark. The entire control strategy is established as follows. Based on the current values of $\wedge\hat{x}$ and of $\vee\hat{x}$, we predict the values of $\wedge\hat{x}$ and of $\vee\hat{x}$ at the next step by using equations (6) and (7) with the previous input and without intersecting with $\gamma_i(y_i')$. Let these predicted values be $\wedge\hat{x}_{pred}$ and $\vee\hat{x}_{pred}$. Thus, the intervals $[\overline{L}^k(\vee\hat{\hat{x}}_{pred}), \overline{U}^k(\wedge\hat{\hat{x}}_{pred})]$ are computed so that we can check whether $[\wedge\hat{x}_{pred}, \vee\hat{x}_{pred}] \cap \overline{E} = \emptyset$ (using Proposition 3). If this intersection is empty, then the input is set to its previous value. If this intersection is not empty, we use Algorithm 2 to compute the new input.

VI. A

Example 1: Vehicles at a traffic intersection. Let us consider two vehicles converging to a traffic intersection (Figure 2). The vehicle's physical motion can be modeled by considering its longitudinal dynamics along its geometric path (determined by the geometry of the lanes) following a similar modeling framework as performed in [3]. Let then x_1 and x_2 denote the position of the two vehicles along their path with respect to some fixed reference point. Let v_1 and v_2 be the velocities of the two cars along their lanes. We assume that each car dynamics along its lane can be modeled as a second order system, which in discrete time becomes:

$$x_i' = x_i + v_i(\Delta T),\ v_i' = v_i + u_i(\Delta T), i \in \{1, 2\}, \quad (22)$$

in which ΔT is the time interval. The controller u_i can directly affect the acceleration by acting on the throttle pedal or on the brake. The system also has the following constraints. When a vehicle is inside the intersection, it cannot stop as it has to free the intersection as soon as possible, while it can stop before entering the intersection. In addition, a vehicle cannot move backwards in its lane. These constraints can be modeled by requiring, for a suitable x_i^A that for $x_i \le x_i^A$ then $v_i \ge 0$, while for $x_i > x_i^A$ we must have $v_i \ge v_m$ with $v_m > 0$. Let $u_m < 0 < u_M$. Each vehicle can be described by a hybrid automaton with two modes: $q_i = q_{1,i}$ if ($x_i \le x_i^A$ and $v_i \le 0$) or ($x_i > x_i^A$ and $v_i \le v_m$); $q_i = q_{2,i}$ if ($x_i \le x_i^A$ and $v_i > 0$) or ($x_i > x_i^A$ and $v_i > v_m$). In each one of these modes, the update map f is given by equations (22), in which $\iota(q_{1,i}) = [0, u_M]$, $\iota(q_{2,i}) = [u_m, u_M]$. Since $\mathcal{I}_D = \emptyset$, the hybrid automaton admits only autonomous mode transitions.

Example 2: Trains at a railway merging. Consider two trains in the proximity of a railway merging. Assuming a second order dynamics along their rail, each train can be modeled again as in equations (22). However, now the input sets will be different from the previous example. In digital control mode [16], the input u_i can take four values corresponding to a "hard-brake" mode, a "run-out" mode, a "constant-speed" mode, and an "acceleration" mode. Let these 4 values be denoted by respectively $\alpha, \gamma, \delta, \beta$ so that $\alpha < \gamma < \delta < \beta$. Each vehicle dynamics can thus be modeled by a hybrid automaton with four modes such that $q_i = q_{1,i}$ iff $u_i = \alpha$, $q_i = q_{2,i}$ iff $u_i = \gamma$, $q_i = q_{3,i}$ iff $u_3 = \delta$, and $q_i = q_{4,i}$ iff $u_i = \beta$. There are not autonomous switches in this system, so that for each train $R(x_i, v_i, \sigma_i) = R(\sigma_i)$ where σ_i is the discrete input.

For both examples, the measurement model is given by $\mathcal{Y}_i = \mathbb{R}$ and $\gamma(y_i) = [y_i - \Delta, y_i + \Delta]$, in which Δ can be an uncertainty of several meters. One can verify that the above models for each vehicle are triangular order preserving hybrid automata. The safety requirement is that the two vehicles never are in a ball of radius d around the conflict point C at the same time. This is encoded by a bad set $B = \{(x_1, v_1, x_2, v_2) \mid (x_1, x_2) \in \overline{B}\}$, in which $\overline{B} = [L_1, U_1] \times [L_2, U_2]$ for suitable $L_1, U_1, L_2, U_2 \in \mathbb{R}$. Algorithm 1 and Algorithm 2 were implemented for both examples and the results are shown in Figure 3 and in Figure 4.

VII. C

In this paper, we have presented a dynamic feedback algorithm for the class of block-triangular order preserving hybrid automata. By virtue of the structure of the system, the hybrid dynamics update map preserves the ordering of the continuous variables (component-wise partial ordering). This allows to construct a state estimator that only keeps track of suitable lower and upper bounds in the partial ordering. Such lower and upper bounds are then used by a safety control law that computes the control actions that guarantee that the state will be mapped outside the escape set. The proposed state estimation and control algorithms have linear complexity in the number of variables. Two application examples from the area of intelligent transportation systems are considered to show that the class of systems considered is general enough to model practically relevant systems. We will investigate in our future work extensions to the case in which the discrete state update map has memory and to the case in which the bad set B involves all the coordinates in the state space.

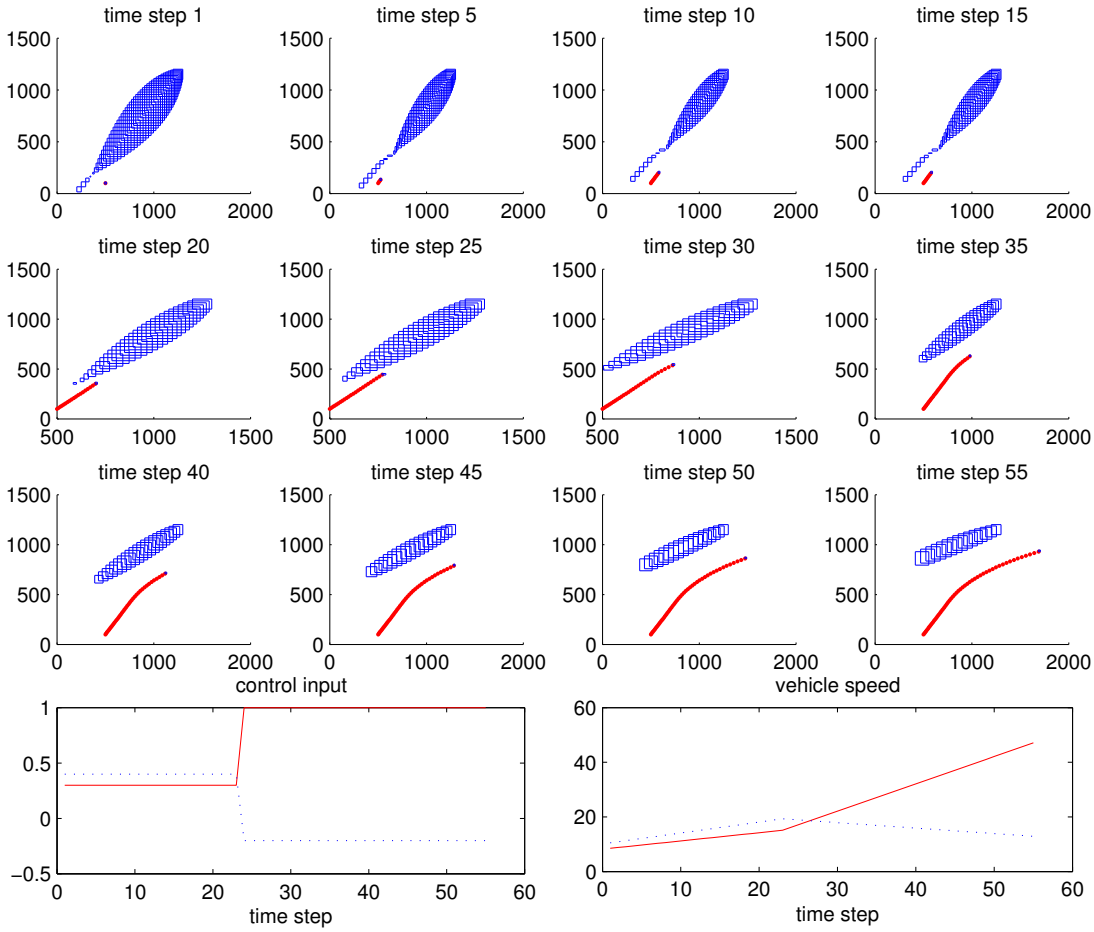

Fig. 4. Trains at a railway merging. This figure shows a run of Algorithm 2 at different time instants in the upper plots. The trajectory of the trains in the x_1, x_2 plane and the sets $\bigcup_{k\geq 0}[\overline{L}^k(\hat{\backslash x}), \overline{U}^k(\hat{\wedge x})]$ are both shown: $\overline{L}_i^k$ and $\overline{U}_i^k$ are computed by Algorithm 1 up to the k established in item (i) of Algorithm 2 are shown in blue. The lower left side plot shows the control commands applied to each train.

R

[1] O. of Safety Analysis, "Accidents/incidents counts," *Federal Railroad Administration, http://safetydata.fra.dot.gov/officeofsafety*, 2005.

[2] K. Laberteaux, L. Caminiti, D. Caveney, and H. Hada, "Pervasive vehicular networks for safety," *IEEE Pervasive Computing, Spotlight*, pp. 60–62, 2006.

[3] C. J. Tomlin, J. Lygeros, and S. Sastry, "A game theoretic approach to controller design for hybrid systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 949–970, 2000.

[4] O. Shakernia, G. J. Pappas, and S. Sastry, "Semi-decidable synthesis for triangular hybrid systems," in *Lecture Notes in Computer Science, volume 2034*, 2001.

[5] E. Asarin, O. Maler, and A.Pnueli, "Symbolic controller synthesis for discrete and timed systems," in *Lecture Notes in Computer Science, volume 999*, 1995, pp. 1–20.

[6] T. A. Henzinger and P. W. Kopke, "Discrete-time control for rectangular hybrid automata," *Theoretical Computer Science*, vol. 221, pp. 369–392, 1999.

[7] I. Mitchell and C. J. Tomlin, "Overapproximating reachable sets by hamilton-jacobi projections," *Journal of Scientific Computation*, vol. 19, no. 1, pp. 323–346, 2003.

[8] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi, "Hytech: A model checker for hybrid systems," *Software Tools for Technology Transfer*, vol. 1, pp. 110–122, 1997.

[9] T. A. Henzinger, B. Horowitz, R. Majumdar, and H. Wong-Toi, "Beyond hytech: Hybrid systems analysis using interval numerical methods," in *Lecture Notes in Computer Science 1790, Springer-Verlag*, 2000, pp. 130–144.

[10] M. D. Wulf, L. Doyen, and J.-F. Raskin, "A lattice theory for solving games of imperfect information," *Hybrid Systems: Computation and Control,* Lecture Notes in Computer Science, vol. 3927, Springer-Verlag, pp. 153–168, 2006.

[11] D. Del Vecchio, "A partial order approach to discrete dynamic feedback in a class of hybrid systems," in *Hybrid Systems: Computation and Control,* Lecture Notes in Computer Science, vol. 4416, A. Bemporad, A. Bicchi, and G. Buttazzo, Eds. Springer-Verlag, 2007, pp. 159–173.

[12] B. A. Davey and H. A. Priesteley, *Introduction to Lattices and Order*. Cambridge University Press, 2002.

[13] D. Del Vecchio, "A partial order approach for low complexity control of block triangular hybrid automata," in *Conf. on Decision and Control*, 2007, p. (To Appear).

[14] T. A. Henzinger, "The theory of hybrid automata," in *Proceedings of the 11th Annual Symposium on Logic in Computer Science*. IEEE press, 1996, pp. 278–292.

[15] D. Del Vecchio, R. M. Murray, and E. Klavins, "Discrete state estimators for systems on a lattice," *Automatica*, vol. 42, no. 2, pp. 271–285, 2006.

[16] J. Pachl, *Railway operation and control*. VTD Rail Publishing, 2002.