

Discrete Dynamic Feedback for a Class of Hybrid Systems on a Lattice

Domitilla Del Vecchio

Abstract—We address the problem of designing a dynamic output feedback control for a hybrid system in order to satisfy system specifications, once the continuous variables are measured. In absence of a structure on the discrete variable space, the design of such a controller requires a number of computations at least proportional to the size of the discrete variable set and to the size of the control set. In this paper, we propose to exploit a partial order structure on the set of discrete variables and inputs in order to compute the control as a function of two discrete variable values that are updated at each step. This algorithm is applied to a multi-robot game involving two teams competing against each other in a “capture the flag”-like game.

I. I

Controller design problems under language specification have been extensively studied for discrete systems in the computer science literature (see [15] for an overview). A control perspective in the context of discrete event systems was given by [11]. The approach has been extended to specific classes of hybrid systems such as timed automata ([1] for example) and rectangular automata ([17]). For these classes of hybrid systems, implementation results using tools such as [9] showed that in practice the synthesis procedure is limited to control problems with small numbers of control modes. Large part of the work on safety verification for general classes of hybrid systems has been concerned with the computation of reachable sets (see for example [16], and the references therein). As noted also by [16], the problems solved with these methods are of low dimension. Also, these works are concerned with state feedback, that is, the state is available for measurement. An output map is considered in the literature of viability theory for hybrid systems (see for example, [2], [7], [13]). However, static output feedback is usually performed.

In this paper, the continuous variables are available for measurement, thus the control and state estimation problems concern only the discrete variables. This scenario has practical interest in multi-robot systems in which the continuous variables represent the position and the velocity of a robot, while the discrete variables regulate the internal communication and coordination protocol. In [11], the control problem of discrete event systems under language specifications is considered. The proposed control algorithms with full observation have polynomial complexity in the number of states. In the case of partial observations, the control problem becomes NP complete at worse. In a practical system, the

number of states can be exponential in the number of constituent processes, and therefore these control methodologies are prohibitive. Caines and Wang [8], consider the problem of steering the state of a partially observed automata to a final desired state. A dynamic programming methodology is proposed, which leads to a complexity of the control computation that is polynomial in the size of the state set, of the input set, and of the output set. Modular synthesis and special structures on the process are suggested in order to reduce computation. For example, [12] proposes algorithms for the synthesis problems of safe control policies in decentralized control of discrete event systems.

In this work, we aim at exploiting some special structure of the system in order to decrease the amounts of computation needed for state estimation and control. In particular, we propose a methodology based on partial order structures for computing dynamic feedback control that satisfies system specifications. This methodology uses the state estimator on a lattice already developed in [6] for computing the inputs that satisfy the system specifications. This work addresses the case in which the number of discrete variables and inputs is so large that enumeration methods for state estimation and exhaustive searches for control design are prohibitive. By using a partial order on the set of inputs and on the set of discrete variables, the state estimation algorithm updates two variables at each step. These are then used to compute the lower and upper bounds (in a specified lattice) of the set of inputs that satisfy the system specifications. This can be achieved under suitable order preserving assumptions of the system dynamics. A multi-robot example is proposed, which shows how to apply the proposed methodology in an attack-defense game. In particular, an attacker team runs the proposed estimation and control algorithm to design the next move based on the observation of the other team behavior. The scope of the attackers is to win the game. This scope is encoded in a system specification, which is then guaranteed by the dynamic control algorithm.

This paper is organized as follows. In section II, we introduce deterministic transition systems. In section III, we formulate the control problem of a system on a partial order. In section IV, we propose a solution to the problem. In section V, we show a multi-robot example. A small appendix revises some partial order theory.

II. D T S

In this section, we define the model of the systems that we are going to use. This model is referred to as deterministic transition system and it is introduced in the following definition.

Definition 2.1: A *Deterministic transition system* is a tuple $\Sigma = (\mathcal{S}, \mathcal{I}, \mathcal{Y}, F, g)$ in which \mathcal{S} is a set of states, \mathcal{Y} is a set of outputs, \mathcal{I} is a set of inputs, $F : \mathcal{S} \times \mathcal{I} \rightarrow \mathcal{S}$ is a transition function, and $g : \mathcal{S} \rightarrow \mathcal{Y}$ is an output function. An *execution* of Σ is any sequence $\sigma = \{s(k)\}_{k \in \mathbb{N}}$ such that $s(0) \in \mathcal{S}$ and $s(k+1) = F(s(k), u(k))$ for $u(k) \in \mathcal{I}$ for any $k \in \mathbb{N}$. The set of all executions of Σ is denoted $\mathcal{E}(\Sigma)$. The output sequence $g(\sigma)$ is also denoted $\{y(k)\}_{k \in \mathbb{N}}$ with $y(k) = g(s(k))$. Given a system execution σ , we will also use the notation $s(k) = \sigma(k)(s)$ to denote the value of the state at step k along such an execution.

We now give some definitions about controllability properties of system Σ given an output measurements and a specification. Let $P : \mathcal{S} \rightarrow \{\text{T}, \text{F}\}$ be a predicate on the set of states that can be either true (T) or false (F). Assume that we would like to design a control input that guarantees that such a predicate is true at any time step along the system execution. Since we have only output information to design such an input, we give next two definitions about output controllability with respect to the given predicate. Let $S = \{s \in \mathcal{S} \mid P(s) = \text{T}\}$ denote the *true set* and let $Y = \{s \in \mathcal{S} \mid g(s) = y, y \in \mathcal{Y}\}$ denote the *output set*, with $Y(k) = \{s \in \mathcal{S} \mid g(s) = y(k), y(k) \in \mathcal{Y}\}$, in which $\{y(k)\}_{k \in \mathbb{N}}$ is an output sequence of Σ . For any $X \subseteq \mathcal{S}$, we use the notation $I_X(S) := \{u \in \mathcal{I} \mid F(X, u) \subseteq S\}$ to represent the set of inputs that map a set X inside S through F . The next definition proposes a concept of dynamic output feedback analogous to the one proposed by [14]. Let $\mathcal{P}(\mathcal{S})$ denote the set of all subsets of \mathcal{S} , that is, $\mathcal{P}(\mathcal{S}) = \{A \mid A \subseteq \mathcal{S}\}$.

Definition 2.2: The system Σ is said to be *controllable by dynamic output feedback* with respect to true set S and initial set X_0 if there exist functions $H_1 : \mathcal{P}(\mathcal{S}) \times \mathcal{Y} \rightarrow \mathcal{P}(\mathcal{I})$ and $H_2 : \mathcal{P}(\mathcal{S}) \times \mathcal{Y} \rightarrow \mathcal{P}(\mathcal{S})$ such that for any execution $\sigma \in \mathcal{E}(\Sigma)$ and any k if

$$\begin{aligned} X(k+1) &= H_2(X(k), y(k)) \\ u(k) &\in H_1(X(k), y(k)) \end{aligned}$$

with $X(0) = X_0$ and $y(k) = g(\sigma(k)(s))$, then

- (i) $s(k) \in X(k)$ with $s(k) = \sigma(k)(s)$;
- (ii) $X(k) \subseteq S$.

This definition implies that there is a system $\Sigma_f = (\mathcal{P}(\mathcal{S}), \mathcal{Y}, \mathcal{P}(\mathcal{I}), H_2, H_1)$ (in which the subscript f stands for feedback) that takes as input the output of Σ , $y \in \mathcal{Y}$, has internal state $X \in \mathcal{P}(\mathcal{S})$ a subset of \mathcal{S} , and it has a set of inputs to Σ as its output. The set H_1 is the set of inputs that map the set of states X inside the desired true set S . Among these inputs, $u(k)$ is chosen to be applied to the system Σ .

The next proposition gives a sufficient condition for Σ to be controllable by dynamic output feedback.

Proposition 2.1: Let $I_{Y \cap S}(S)$ be nonempty for any $y \in \mathcal{Y}$ and let $s(0) \in S$. Then, system Σ is controllable by dynamic output feedback with respect to true set S and initial set $X_0 = S$.

Proof: We prove this by choosing specific functions H_1 and H_2 . Let $X_0 = S$, and define $H_1 : \mathcal{P}(\mathcal{S}) \times \mathcal{Y} \rightarrow \mathcal{P}(\mathcal{I})$ and $H_2 : \mathcal{P}(\mathcal{S}) \times \mathcal{Y} \rightarrow \mathcal{P}(\mathcal{S})$ as $H_2(X(k), y(k)) = F(X(k) \cap$

$Y(k), u(k))$, $u(k) \in H_1(X(k), y(k))$, and $H_1(X(k), y(k)) = \{u \in \mathcal{I} \mid F(X(k) \cap Y(k), u) \subseteq S\}$. We need to show that the set $H_1(X(k), y(k))$ is not empty for all k . Then, we need to show properties (i) and (ii) of Definition 2.2.

We proceed by induction argument on the step k . By assumption, $X(0) = S$ (i.e., $X(0) \subseteq S$) and $s(0) \in X(0)$. We show (base step) that $H_1(X(0), y(0)) \neq \emptyset$. Then we show (induction step) that if $X(k-1) \subseteq S$, we also have that $H_1(X(k-1), y(k-1)) \neq \emptyset$, $X(k) \subseteq S$ with $u(k-1) \in H_1(X(k-1), y(k-1))$, and $s(k) \in X(k)$.

(Base step) By assumption, $X(0) = S$, thus $H_1(X(0), y(0)) = \{u \in \mathcal{I} \mid F(X(0) \cap Y(0), u) \subseteq S\}$ is nonempty because $I_{Y \cap S}(S)$ is nonempty for any $y \in Y$ and $y(0) \in \mathcal{Y}$.

(Induction step) Assume $X(k-1) \subseteq S$, then $H_1(X(k-1), y(k-1)) = \{u \in \mathcal{I} \mid F(X(k-1) \cap Y(k-1), u) \subseteq S\} \neq \emptyset$ because $\{u \in \mathcal{I} \mid F(X(k-1) \cap Y(k-1), u) \subseteq S\} \supseteq \{u \in \mathcal{I} \mid F(S \cap Y(k-1), u) \subseteq S\}$ and $I_{Y(k-1) \cap S}$ is nonempty by assumption. Thus, if $u(k-1) \in H_1(X(k-1), y(k-1))$ we have by construction that $X(k) \subseteq S$. Also, since $s(k-1) \in X(k-1)$ and $s(k-1) \in Y(k-1)$, we have that $s(k) \in X(k)$. ■

In the next section, we specialize the structure of system Σ to explicitly model the evolution of continuous and discrete variables.

III. P S

Given a deterministic transition system $\Sigma = (\mathcal{S}, \mathcal{I}, \mathcal{Y}, F, g)$, we specialize it to the case $\mathcal{S} = \mathcal{U} \times \mathcal{Z}$, in which \mathcal{U} is a finite set of discrete variables denoted $\alpha \in \mathcal{U}$ and \mathcal{Z} is a set of continuous variables denoted $z \in \mathcal{Z}$. The transition function will also have two components, i.e., $F = (f, h)$, in which $f : \mathcal{U} \times \mathcal{Z} \times \mathcal{I} \rightarrow \mathcal{U}$ and $h : \mathcal{U} \times \mathcal{Z} \rightarrow \mathcal{Z}$. The set of outputs is defined as $\mathcal{Y} = \mathcal{Z} \times \mathcal{Z}$ and the output function is $g : \mathcal{U} \times \mathcal{Z} \rightarrow \mathcal{Y}$. For the remainder of this paper, we denote by $\Sigma = (\mathcal{U} \times \mathcal{Z}, \mathcal{I}, \mathcal{Y}, (f, h), g)$ the system represented by the following difference equations

$$\begin{aligned} \alpha(k+1) &= f(\alpha(k), z(k), u(k)) \\ z(k+1) &= h(\alpha(k), z(k)) \end{aligned} \quad (1)$$

$$(y_1(k), y_2(k)) = (z(k), h(\alpha(k), z(k))). \quad (2)$$

Any execution of the system Σ is of the form $\sigma = \{\alpha(k), z(k)\}_{k \in \mathbb{N}}$ and the output sequence is given by $\{y(k)\}_{k \in \mathbb{N}} = \{y_1(k), y_2(k)\}_{k \in \mathbb{N}}$. Given any execution σ of the system, we will denote the values of z and α at step k in such an execution by $\sigma(k)(z)$ and $\sigma(k)(\alpha)$, respectively. At each step k , the output y has two components basically corresponding to the value of $z(k)$ and of $z(k+1)$.

Since the z variables are measured, we consider the problem of satisfying a predicate P on the discrete variables only. Thus, $P : \mathcal{U} \rightarrow \{\text{T}, \text{F}\}$. Let the true set $S \subset \mathcal{U}$ be defined as $S = \{\alpha \in \mathcal{U} \mid P(\alpha) = \text{T}\}$, then we consider the problem of determining an input to the system that satisfies the specification. In the present case, the output set is given by $Y = \{\alpha \in \mathcal{U} \mid y_2 = h(\alpha, y_1), y_1, y_2 \in \mathcal{Z}\}$.

Assume that we would like to compute the set of inputs that maintain an output set Y in S . In principle, if \mathcal{U} and

\mathcal{I} are finite and discrete, for any $\alpha \in Y$ we can compute $f(\alpha, z, u)$ for any $u \in \mathcal{I}$ and check whether it is contained in S . Assuming the size of Y of the order of the size of \mathcal{U} , this requires a number of computations of the order of $|\mathcal{I}||\mathcal{U}||S|$. If we also assume that the size of S is of the order of the size of \mathcal{U} , we then require a number of computations of the order of $|\mathcal{I}||\mathcal{U}|^2$. If \mathcal{U} is given by the product of a number of sets because, for example, Σ is a multi-agent system, then the number of computations required for solving our control problem is exponential in the number of agents as also noted by Ramadge and Wonham [11].

We thus propose an alternative procedure that exploits a partial order structure on the set of discrete states as well as on the set of inputs. The idea can be explained in the following simple example. Assume $\alpha \in \mathbb{N}$, $Y = [2, 11]$, $S = [1, 10]$, $u \in \mathbb{Z}$, and that $f(\alpha, z, u) = f(\alpha, u) = \alpha + u$. For computing the set of inputs in \mathbb{Z} such that $f([2, 11], u) \subset [1, 10]$, it is not necessary to compute f for all pairs (α, u) and check whether $f(\alpha, u) \in S$. In fact, it is enough to compute the set of $u \in \mathbb{Z}$ such that $f(2, u) \geq 1$ and the set of $u \in \mathbb{Z}$ such that $f(11, u) \leq 10$, and then intersect the two found sets, which turn out to be intervals in \mathbb{Z} : $[-1, \infty)$ and $(-\infty, -1]$, respectively. This simply gives the answer $u = \{-1\}$. For this example, it is enough to compare the image through f of the lower and upper bounds of Y with the lower and upper bounds of S respectively. From this comparison, one can compute the lower and upper bounds of the set of allowed inputs. This is due to the fact that the spaces \mathcal{U} and \mathcal{I} are equipped with a (total in this case) order while the function f preserves such orders. This argument will be made more formal and more general in this paper by using some basic partial order theory. We next state the problem of determining a dynamic output feedback on a partial order.

Problem 1: (Dynamic Output Feedback on a Lattice) Given system $\Sigma = (\mathcal{U} \times \mathcal{Z}, \mathcal{I}, \mathcal{Y}, (f, h), g)$ with $\alpha(0) \in X_0 \subseteq S$, find a deterministic transition system $\Sigma_f = (\chi \times \chi, \mathcal{Y}, \tilde{\mathcal{I}} \times \tilde{\mathcal{I}}, (H_{21}, H_{22}), (H_{11}, H_{12}))$ with $H_{21} : \chi \times \chi \times \mathcal{Y} \rightarrow \chi$, $H_{22} : \chi \times \chi \times \mathcal{Y} \rightarrow \chi$, $H_{11} : \chi \times \chi \times \mathcal{Y} \rightarrow \tilde{\mathcal{I}}$, $H_{12} : \chi \times \chi \times \mathcal{Y} \rightarrow \tilde{\mathcal{I}}$, (χ, \leq) and $(\tilde{\mathcal{I}}, \leq)$ lattices, with $\mathcal{U} \subseteq \chi$ and $\mathcal{I} \subseteq \tilde{\mathcal{I}}$, such that if $u(k) \in [H_{11}(L(k), U(k), y(k)), H_{12}(L(k), U(k), y(k))] \cap \mathcal{I}$, and $L(k), U(k) \in \chi$ are updated by

$$\begin{aligned} L(k) &= H_{21}(L(k-1), U(k-1), y(k-1), y(k)) \\ U(k) &= H_{22}(L(k-1), U(k-1), y(k-1), y(k)), \end{aligned}$$

with $\{y(k)\}_{k \geq 0} = g(\sigma)$ we have

- (i) $\sigma(k)(\alpha) \in [L(k), U(k)] \cap \mathcal{U}$;
- (ii) $[L(k), U(k)] \cap \mathcal{U} \subseteq S$.

The variables $L(k)$ and $U(k)$ represent the lower and upper bound of the set of possible discrete states compatible with the output sequence and with the system dynamics. The functions H_{11} and H_{12} compute the lower and upper bounds of the set of possible inputs that map the interval $[L(k), U(k)] \cap \mathcal{U}$ inside the true set S . In the next section, we propose a solution for the dynamic output feedback problem.

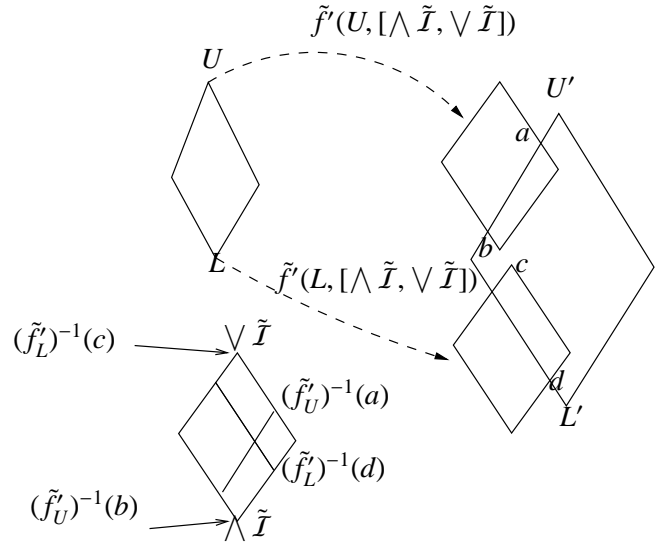


Fig. 1. This figure explains how u_L and u_U of Lemma 4.1 are computed. The dependence on z has been omitted. Each polygon represents an interval sublattice in the partial order. u_L is the supremum (\vee) of $(\tilde{f}'_U)^{-1}(b)$ and $(\tilde{f}'_L)^{-1}(d)$, while u_U is the infimum (\wedge) of $(\tilde{f}'_L)^{-1}(c)$ and $(\tilde{f}'_U)^{-1}(a)$.

IV. P S

Here, we give the main result for the computation of a dynamic output feedback control. We first give some definitions.

Definition 4.1: A system Σ is said to satisfy the dynamic controllability condition with respect to $S \subset \mathcal{U}$ if $\{u \in \mathcal{I} \mid f(S \cap Y, z, u) \subseteq S\}$ is not empty for any $y \in \mathcal{Y}$ and $z \in \mathcal{Z}$.

Definition 4.2: Let $\Sigma = (\mathcal{U} \times \mathcal{Z}, \mathcal{I}, \mathcal{Y}, (f, h), g)$, and let (χ, \leq) and $(\tilde{\mathcal{I}}, \leq)$ be lattices with $\mathcal{U} \subseteq \chi$ and $\mathcal{I} \subseteq \tilde{\mathcal{I}}$. An extension of Σ on (χ, \leq) and $(\tilde{\mathcal{I}}, \leq)$ is any system $\tilde{\Sigma} = (\chi \times \mathcal{Z}, \tilde{\mathcal{I}}, \mathcal{Y}, (\tilde{f}, \tilde{h}), \tilde{g})$ such that $\tilde{f}|_{\mathcal{U} \times \mathcal{Z} \times \mathcal{I}} = f$, $\tilde{h}|_{\mathcal{U} \times \mathcal{Z}} = h$, and $\tilde{g}|_{\mathcal{U} \times \mathcal{Z}} = g$.

Let $\tilde{f}' : \chi \times \tilde{\mathcal{I}} \rightarrow \chi$ be a map with (χ, \leq) and $(\tilde{\mathcal{I}}, \leq)$ lattices. Let $\tilde{f}'(x, [\wedge \tilde{\mathcal{I}}, \vee \tilde{\mathcal{I}}]) \rightarrow [\tilde{f}'(x, \wedge \tilde{\mathcal{I}}), \tilde{f}'(x, \vee \tilde{\mathcal{I}})]$ be a bijection for any $x \in \chi$. For any $w \in [\tilde{f}'(x, \wedge \tilde{\mathcal{I}}), \tilde{f}'(x, \vee \tilde{\mathcal{I}})]$ we denote by $(\tilde{f}'_x)^{-1}(w)$ the input $\tilde{u} \in [\wedge \tilde{\mathcal{I}}, \vee \tilde{\mathcal{I}}]$ such that $\tilde{f}'(x, \tilde{u}) = w$. Then, we have the following lemma.

Lemma 4.1: Let $(\tilde{\mathcal{I}}, \leq)$ and (χ, \leq) be a lattices. Let $L, U \in \chi$ with $L \leq U$ and $L', U' \in \chi$ with $L' \leq U'$. Assume that

- (i) the function $\tilde{f}' : \chi \times \tilde{\mathcal{I}} \rightarrow \chi$ is order preserving on $[L, U]$ in its first argument;
- (ii) $\tilde{f}' : (x, [\wedge \tilde{\mathcal{I}}, \vee \tilde{\mathcal{I}}]) \rightarrow [\tilde{f}'(x, \wedge \tilde{\mathcal{I}}), \tilde{f}'(x, \vee \tilde{\mathcal{I}})]$ is an order isomorphism for any $x \in [L, U]$;
- (iii) the set $\{\tilde{u} \in \tilde{\mathcal{I}} \mid \tilde{f}'([L, U], \tilde{u}) \subseteq [L', U']\}$ is not empty.

Then, $\{\tilde{u} \in \tilde{\mathcal{I}} \mid \tilde{f}'([L, U], \tilde{u}) \subseteq [L', U']\} = [u_L, u_U]$, with

$$\begin{aligned} u_L &= (\tilde{f}'_L)^{-1}(\tilde{f}'(L, \wedge \tilde{\mathcal{I}}) \vee L') \vee (\tilde{f}'_U)^{-1}(\tilde{f}'(U, \wedge \tilde{\mathcal{I}}) \vee L') \\ u_U &= (\tilde{f}'_L)^{-1}(\tilde{f}'(L, \vee \tilde{\mathcal{I}}) \wedge U') \wedge (\tilde{f}'_U)^{-1}(\tilde{f}'(U, \vee \tilde{\mathcal{I}}) \wedge U'). \end{aligned}$$

Proof: We have that $\{\tilde{u} \in \tilde{\mathcal{I}} \mid [\tilde{f}'([L, U], \tilde{u}) \subseteq [L', U']\} = \{\tilde{u} \in \tilde{\mathcal{I}} \mid \tilde{f}'(L, \tilde{u}) \in [L', U'] \text{ and } \tilde{f}'(U, \tilde{u}) \in [L', U']\}$ by the order preserving property of \tilde{f}' in its first argument (assumption (i)). As a consequence, we have that $\{\tilde{u} \in$

$\tilde{I} \mid \tilde{f}'(L, \tilde{u}) \in [L', U']$ and $\tilde{f}'(U, \tilde{u}) \in [L', U'] = \{\tilde{u} \in \tilde{I} \mid \tilde{f}'(L, \tilde{u}) \in [L', U']\} \cap \{\tilde{u} \in \tilde{I} \mid \tilde{f}'(U, \tilde{u}) \in [L', U']\}$. By the fact that \tilde{f}' is an order isomorphism in its second argument by assumption (ii) and by the fact that $\tilde{f}'(L, \wedge \tilde{I}) \vee L' \in [\tilde{f}'(L, \wedge \tilde{I}), \tilde{f}'(L, \vee \tilde{I})]$ because of assumption (iii), we have that $\{\tilde{u} \in \tilde{I} \mid \tilde{f}'(L, \tilde{u}) \in [L', U']\} = [(\tilde{f}'_L)^{-1}(\tilde{f}'(L, \wedge \tilde{I}) \vee L'), (\tilde{f}'_L)^{-1}(\tilde{f}'(L, \vee \tilde{I}) \wedge U')]$, and similarly $\{\tilde{u} \in \tilde{I} \mid \tilde{f}'(U, \tilde{u}) \in [L', U']\} = [(\tilde{f}'_U)^{-1}(\tilde{f}'(U, \wedge \tilde{I}) \vee L'), (\tilde{f}'_U)^{-1}(\tilde{f}'(U, \vee \tilde{I}) \wedge U')]$. Intersecting these two sets, we obtain the interval whose lower and upper bounds are given by u_L and u_U , respectively. ■

The meaning of this lemma is as follows. Since \tilde{f}' is an order preserving map on the interval $[L, U]$, in order to map such interval into another interval $[L', U']$ by means of a good choice of inputs, it is enough to do as follows. It is enough to (1) compute the set of inputs that map L into $[L', U']$, (2) compute the set of inputs that map U into $[L', U']$, (3) intersect these sets. If \tilde{f}' is an order isomorphism in its second argument, the sets computed in (1) and in (2) are intervals. These are intervals because the inverse images of $[a, b]$ and of $[d, c]$ through $(\tilde{f}'_L)^{-1}$ and $(\tilde{f}'_U)^{-1}$ are intervals. This is due to the fact that the functions $(\tilde{f}'_L)^{-1} : [\tilde{f}'(L, \wedge \tilde{I}), \tilde{f}'(L, \vee \tilde{I})] \rightarrow [\wedge \tilde{I}, \vee \tilde{I}]$ and $(\tilde{f}'_U)^{-1} : [\tilde{f}'(U, \wedge \tilde{I}), \tilde{f}'(U, \vee \tilde{I})] \rightarrow [\wedge \tilde{I}, \vee \tilde{I}]$ are order isomorphisms. This is shown in Figure 1.

Let $\tilde{\Sigma} = (\chi \times \mathcal{Z}, \mathcal{I}, \mathcal{Y}, (\tilde{f}, \tilde{h}), \tilde{g})$ be an extension of Σ on (χ, \leq) . We will assume that the output set of the extended system is an interval, that is, $\tilde{Y} = \{x \in \chi \mid y_2 = \tilde{h}(x, y_1), (y_1, y_2) \in \mathcal{Y}\} = [\wedge \tilde{Y}, \vee \tilde{Y}]$. For an output sequence $\{y(k)\}_{k \geq 0}$, we will denote the output set by $\tilde{Y}(k)$ and its lower and upper bounds by $\wedge Y(k) = L_y(k)$ and $\vee Y(k) = U_y(k)$. We also assume that $\tilde{S} \subset \chi$ is a set such that $\tilde{S} \cap \mathcal{U} = S$ and that $\tilde{\Sigma}$ satisfies the dynamic output controllability condition with respect to \tilde{S} . This implies by the definition of \tilde{f} that Σ satisfies the dynamic output controllability condition with respect to S .

Definition 4.3: Let $\tilde{\Sigma} = (\chi \times \mathcal{Z}, \mathcal{I}, \mathcal{Y}, (\tilde{f}, \tilde{h}), \tilde{g})$ be an extension of Σ on (χ, \leq) . Let (\tilde{I}, \leq) be a lattice with $\mathcal{I} \subseteq \tilde{I}$. Let $[L, U]$ be any interval in $\tilde{S} \cap \tilde{Y}$. If there are a function $\tilde{f}' : \chi \times \tilde{I} \rightarrow \chi$ such that $\tilde{f}' : (x, [\wedge \tilde{I}, \vee \tilde{I}]) \rightarrow [\tilde{f}'(x, \wedge \tilde{I}), \tilde{f}'(x, \vee \tilde{I})]$ is an order isomorphism for any $x \in \chi$ and an order preserving map in the first argument, constants $L^*, U^* \in [L, U]$, and $L', U' \in \tilde{S}$ such that $\{u \in \mathcal{I} \mid \tilde{f}([L, U], z, u) \subseteq \tilde{S}\} \supseteq \mathcal{I} \cap \{\tilde{u} \in \tilde{I} \mid \tilde{f}'([L^*, U^*], \tilde{u}) \subseteq [L', U']\}$, with the righthand set not empty, then the extension $\tilde{\Sigma}$ of Σ is said to admit an order compatible input extension on (\tilde{I}, \leq) with respect to \tilde{S} . The tuple $(\tilde{f}', L^*, U^*, L', U')$ is named an order compatible tuple for $[L, U]$.

The reason for introducing this definition is that while the computation of the set $\{u \in \mathcal{I} \mid \tilde{f}([L, U], z, u) \subseteq \tilde{S}\}$ might be hard, the computation of the set $\{\tilde{u} \in \tilde{I} \mid \tilde{f}'([L^*, U^*], \tilde{u}) \subseteq [L', U']\}$ is simpler by virtue of Lemma 4.1.

Definition 4.4: The system extension $\tilde{\Sigma}$ is said to be *output interval compatible* with respect to \tilde{S} if for any $[L, U] \subset \chi$, we have that $\tilde{f}([L, U], z, u) \subseteq \tilde{S}$ implies $[\wedge(\tilde{f}([L, U], z, u) \cap \tilde{Y}), \vee(\tilde{f}([L, U], z, u) \cap \tilde{Y})] \subseteq \tilde{S}$ for any $y \in \mathcal{Y}$ and $z \in \mathcal{Z}$.

In this paper, we are not making any order preserving assumption on the function f . Note that it is always possible to break a function into order preserving pieces on an interval. Let $f : \chi \rightarrow \chi$ with (χ, \leq) a lattice. Let $[L, U] \subseteq \chi$ be an interval. The function \tilde{f} is said to be a piecewise order isomorphism on $[L, U]$ if there are a finite number of points $L^j, U^j \in [L, U]$ such that $[L, U] = [L^1, U^1] \cup \dots \cup [L^M, U^M]$ with $[L^i, U^i] \cap [L^j, U^j] = \emptyset$ for $i \neq j$ and such that $f : [L^j, U^j] \rightarrow [f(L^j), f(U^j)]$ is an order isomorphism for any j . The points L^j are said lower knot points and the point U^j are said upper knot points of f in $[L, U]$. On a finite discrete set, a map can always be broken into order isomorphisms: in the limit, each interval will contain only one element. On an infinite dense set, the notion of piece-wise order isomorphism will have to be replaced by piece-wise monotone function.

The next theorem provides a solution to Problem 1.

Theorem 4.1: Let $\tilde{\Sigma}$ be an extension of Σ on (χ, \leq) . Let $\alpha(0)$ be such that $\tilde{Y}(0) \subseteq \tilde{S}$. Assume that $\tilde{\Sigma}$ admits an order compatible input extension on (\tilde{I}, \leq) with respect to \tilde{S} . Assume that $\tilde{\Sigma}$ is output interval compatible with respect to \tilde{S} . Then,

$$\begin{aligned} L(k+1) &= \bigwedge_{\tilde{U}^j \leq \tilde{U}^i} \tilde{L}^j, \tilde{L}^j = \tilde{f}(L^j(k), z(k), u(k)) \vee L_y(k+1) \\ U(k+1) &= \bigvee_{L^j \leq U^i} \tilde{U}^j, \tilde{U}^j = \tilde{f}(U^j(k), z(k), u(k)) \wedge U_y(k+1), \end{aligned}$$

with $L^j(k), U^j(k)$ knot points of \tilde{f} on $[L(k), U(k)]$, $L(0) = L_y(0)$, $U(0) = U_y(0)$, and $u(k) \in [u_L(k), u_U(k)] \cap \mathcal{I}$ with $u_L(k) = (\tilde{f}'_{L^*(k)})^{-1}(\tilde{f}'(L^*(k), \wedge \tilde{I}) \vee L'(k)) \vee (\tilde{f}'_{U^*(k)})^{-1}(\tilde{f}'(U^*(k), \wedge \tilde{I}) \vee L'(k))$ and $u_U(k) = (\tilde{f}'_{L^*(k)})^{-1}(\tilde{f}'(L^*(k), \vee \tilde{I}) \wedge U'(k)) \wedge (\tilde{f}'_{U^*(k)})^{-1}(\tilde{f}'(U^*(k), \vee \tilde{I}) \wedge U'(k))$, with $(\tilde{f}', L^*(k), U^*(k), L'(k), U'(k))$ an order compatible tuple for $[L(k), U(k)]$, solve Problem 1.

Proof: (Sketch). For any k , if $[L(k), U(k)] \subseteq \tilde{S} \cap \tilde{Y}(k)$, we have $\{u \in \mathcal{I} \mid \tilde{f}([L(k), U(k)], u) \subseteq \tilde{S}\} \supseteq \mathcal{I} \cap \{\tilde{u} \in \tilde{I} \mid \tilde{f}'([L^*(k), U^*(k)], \tilde{u}) \subseteq [L'(k), U'(k)]\}$, in which the righthand side is not empty for $(\tilde{f}', L^*(k), U^*(k), L'(k), U'(k))$ an order compatible tuple for $[L(k), U(k)]$. As a consequence, we can apply Lemma 4.1 to obtain $u_L(k)$ and $u_U(k)$. We need to show that $[L(k), U(k)] \subseteq \tilde{S} \cap \tilde{Y}(k)$ for any k . We proceed by induction argument on k . We omit the dependence of \tilde{f} on z for simplicity. By assumption, we have that $[L(0), U(0)] = \tilde{Y}(0) \subset \tilde{S}$. As a consequence, $u(0) \in [u_L(0), u_U(0)] \cap \mathcal{I}$ such that $\tilde{f}([L(0), U(0)], u(0)) \subseteq \tilde{S}$. Let us then assume that $[L(k), U(k)] \subseteq \tilde{Y}(k) \cap \tilde{S}$ and show that also $[L(k+1), U(k+1)] \subseteq \tilde{Y}(k+1) \cap \tilde{S}$. Since $[L(k), U(k)] \subseteq \tilde{Y}(k) \cap \tilde{S}$, $\tilde{f}([L(k), U(k)], u) \subseteq \tilde{S}$ for $u \in [u_L(k), u_U(k)] \cap \mathcal{I}$. Let $L^i(k), U^i(k)$ be knot points of \tilde{f} on $[L(k), U(k)]$. Then, we have that $\tilde{f}([L(k), U(k)], u) = [\tilde{f}(L^1(k), u), \tilde{f}(U^1(k), u)] \cup \dots \cup [\tilde{f}(L^M(k), u), \tilde{f}(U^M(k), u)]$ and thus $\tilde{f}([L(k), U(k)], u) \cap \tilde{Y}(k+1) = [\tilde{f}(L^1(k), u) \vee L_y(k+1), \tilde{f}(U^1(k), u) \wedge U_y(k+1)] \cup \dots \cup [\tilde{f}(L^M(k), u) \vee L_y(k+1), \tilde{f}(U^M(k), u) \wedge U_y(k+1)]$. It follows that $\wedge(\tilde{f}([L(k), U(k)], u) \cap \tilde{Y}(k+1)) = \wedge_{i \in I} \tilde{f}(L^i(k), u) \vee L_y(k+1) = L(k+1)$ and $\vee(\tilde{f}([L(k), U(k)], u) \cap \tilde{Y}(k+1)) = \vee_{i \in I} \tilde{f}(U^i(k), u) \wedge U_y(k+1) = U(k+1)$ with

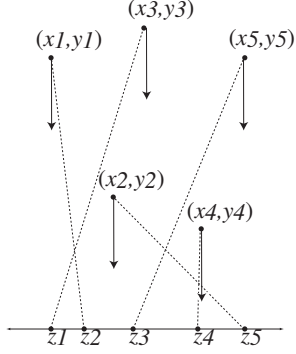


Fig. 2. Example with five robots per team.

$I = \{i \mid \tilde{f}(L^i(k), u) \vee L_y(k+1) \leq \tilde{f}(U^i(k), u) \wedge U_y(k+1)$ and $u \in [u_L(k), u_U(k)] \cap I\}$. By the output interval compatibility assumption, $[\wedge(\tilde{f}([L(k), U(k)], u) \cap \tilde{Y}(k+1))]$, $\vee(\tilde{f}([L(k), U(k)], u) \cap \tilde{Y}(k+1)) \subseteq \tilde{S}$, thus we also have that $[L(k+1), U(k+1)] \subseteq \tilde{S}$. Also, $\alpha(k) \in [L(k), U(k)] \cap \mathcal{U}$ by construction, which shows (i) of Problem 1. Since $[L(k), U(k)] \subseteq \tilde{S}$ and $\tilde{S} \cap \mathcal{U} = S$, we also have that $[L(k), U(k)] \cap \mathcal{U} \subseteq S$, which shows (ii) of Problem 1. ■

This theorem considers the case in which \tilde{f} can be broken into order isomorphisms on an interval in the output lattice. The counter part is that we need to compute more than two joins and meets in order to compute $L(k)$ and $U(k)$, and thus the amounts of computation increase with respect to the case in which \tilde{f} is an order isomorphism on the entire output interval (as assumed in [6]). In the next section, we propose a multi-robot game to show how to apply Theorem 4.1.

V. E : A M - S

We consider a task that represents a defensive maneuver for a robotic “capture the flag” game [3]. We consider a simplified version of the game called “RoboFlag Drill” already considered in [6]. In particular, we show how to use the estimator on a lattice constructed therein in order to design a control algorithm for the attackers, which causes the defenders to be always “sufficiently” far from stabilization. This implies that the defenders loose the game. We describe the system in the next section.

A. System description

Some number of robots with positions $(z_i, 0) \in \mathbb{R}^2$, which we refer to as blue robots, must defend their zone $\{(x, y) \in \mathbb{R}^2 \mid y \leq 0\}$ from an equal number of incoming robots, which we refer to as red robots. The positions of the red robots are $(x_i, y_i) \in \mathbb{R}^2$. An example with five robots is illustrated in Figure 2. The red robots move toward the blue defensive zone. The blue robots are assigned each to a red robot and they coordinate to intercept the red robots. In previous work, such as in [10], it was assumed that the red robots had no “intelligence” and were just coming down straight. In this work, we allow the red robots to move horizontally on the basis of their observations (i.e., the positions of the blue

robots) in order to prevent the blue robots to become close to intercept them.

Let N represent the number of robots in each team. The robots start with an arbitrary (bijective) assignment $\alpha : \{1, \dots, N\} \rightarrow \{1, \dots, N\}$, where α_i is the red robot that blue robot i is required to intercept. At each step, each blue robot communicates with its neighbors and decides to either switch assignments with its left or right neighbor or keep its assignment. In the case in which the red robots are just coming down straight, it is possible to show that the α assignment reaches the equilibrium value $(1, \dots, N)$ ([10]). In this work, we consider the problem of estimating the current assignment α given the motions of the blue robots, which is then used by the red robots to determine a strategy of attack. We thus assume that the red robots can also move horizontally by exchanging position with a close red robot. The RoboFlag Drill system can be specified by the following rules

$$y_i(k+1) = y_i(k) - \delta \quad \text{if } y_i(k) \geq \delta \quad (3)$$

$$z_i(k+1) = z_i(k) + \delta \quad \text{if } z_i(k) < x_{\alpha_i(k)} \quad (4)$$

$$z_i(k+1) = z_i(k) - \delta \quad \text{if } z_i(k) > x_{\alpha_i(k)} \quad (5)$$

$$(\alpha_i(k+1), \alpha_{i+1}(k+1)) = (\alpha_{i+1}(k), \alpha_i(k)) \text{ if } x_{\alpha_i(k)} \geq z_{i+1}(k) \wedge x_{\alpha_{i+1}(k)} \leq z_{i+1}(k), \quad (6)$$

$$(x_i(k+1), x_j(k+1)) = (x_j(k), x_i(k)) \text{ if } \lambda_j(k) = \lambda_i(k) + 1 \text{ and } \text{swap}_{i,j}(k) = 1, \quad (7)$$

$$(\lambda_i(k+1), \lambda_j(k+1)) = (\lambda_j(k), \lambda_i(k)) \text{ if } \lambda_j(k) = \lambda_i(k) + 1 \text{ and } \text{swap}_{i,j}(k) = 1. \quad (8)$$

Equation (6) establishes that two blue robots trade their assignments if the current assignments cause them to go toward each other. Condition (7) allows two adjacent red robots to swap their location. Here, λ_i is the location of red robot i . Locations are $\{1, \dots, N\}$ and denote the order along the x direction in which the red robots are displaced. We start the game with $\lambda = (1, \dots, N)$, $z_i \leq z_{i+1}$, and $x_i < z_i < x_{i+1}$. The variable $\text{swap}_{i,j}$ is the control input to the system represented by equations (4–6). This input must be designed in order to satisfy system specifications.

For this purpose, we define the system $\Sigma = (\mathcal{U} \times \mathcal{Z}, \mathcal{I}, \mathcal{Y}, (f, h), g)$ that models the dynamics of the blue robots with inputs $\text{swap}_{i,j}$ (4–6) as follows. Let $\bar{\alpha}_i = \lambda_{\alpha_i}$ represent the assignment of blue robot i to one of the locations. Let $\bar{x}_i = x_i(0)$ represent the x coordinate of location i . We let the variable $u_i = 1$ if and only if the red robot at location i swaps with the red robot at location $i+1$. Then, set $\mathcal{U} = \text{perm}(N)$, with $\bar{\alpha} \in \mathcal{U}$, $\mathcal{Z} = \mathbb{R}^N$, $\mathcal{I} = \{u \in \{-1, 0, 1\}^N \mid u_i = 1 \Leftrightarrow u_{i+1} = -1, \text{ and } u_N \neq 1 \text{ and } u_1 \neq -1\}$, $\mathcal{Y} = \mathbb{R}^N \times \mathbb{R}^N$. The functions are defined as follows: $f(\bar{\alpha}, z, u) = G(F(\bar{\alpha}, z), u)$, in which $F(\bar{\alpha}, z)$ is represented by relations (6) with $x_i(k) = x_i(0) = \bar{x}_i$ and α_i replaced by $\bar{\alpha}_i$ and $G(\beta, u) = \beta'$, with $u_j = 1 \Rightarrow$ (if $\beta_i = j \Rightarrow \beta'_i = j+1$) and (if $\beta_i = j+1 \Rightarrow \beta'_i = j$). Note that we also will use the following notation $G(\beta, u) = (G_1(\beta_1, u), \dots, G_N(\beta_N, u))$, in which $G_i(\beta_i, u) = \beta'_i$, with $u_j = 1 \Rightarrow$ (if $\beta_i = j \Rightarrow \beta'_i = j+1$) and (if $\beta_i = j+1 \Rightarrow \beta'_i = j$).

We assume for simplicity that $z_i \leq z_{i+1}$ and $\bar{x}_i < z_i < \bar{x}_{i+1}$ for all k . In the sequel, with abuse of notation we will remove the bar and denote $\bar{\alpha}$ and \bar{x} by α and x , respectively. The function $h(\alpha, z)$ is represented by relations (4-5).

Let the entropy of the blue robots be defined by $E(\alpha) = \frac{1}{2} \sum_{i=1}^N |\alpha_i - i|$. In absence of any input to the system (i.e. $u(k) = 0$ for any k), this entropy converges to zero. We define the control specification $P : \mathcal{U} \rightarrow \{\text{T}, \text{F}\}$ to be $P(\alpha) = \text{T}$ if $E(\alpha) \geq 2$ and $P(\alpha) = \text{F}$ otherwise. In words, we want to solve the following problem: Given measurements z determine a dynamic control input $u(k)$ such that $E(\alpha(k)) \geq 2$ for all $k \in \mathbb{N}$.

Having entropy greater or equal than two implies that two or more swappings of assignments among the blue robots are needed before the assignment reaches the value $(1, \dots, N)$. If this is verified at all time, one can show that there will be four red robots that will pass over the blue robots defensive zone. Note that asking the entropy to be always even larger implies that in the end there will be more than four red robots that will be able to pass over the defensive zone. We next compute the set $S = \{\alpha \mid E(\alpha) \geq 2\}$.

Proposition 5.1: The set $S = \{\alpha \mid E(\alpha) \geq 2\} = \{\alpha \mid \exists i, j, \text{ with } j > i + 1 \text{ such that } \alpha_i \neq i \text{ and } \alpha_j \neq j\}$.

Proof: First, we show that if there are i, j with $j > i + 1$ such that $\alpha_i \neq i$ and $\alpha_j \neq j$ then $E(\alpha) \geq 2$. Since $\alpha_i \neq i$ and $\alpha_j \neq j$, we have that $|\alpha_i - i| + |\alpha_j - j| \geq 2$. Then, there are two cases. The first case is $\alpha_i = j$ and $\alpha_j = i$. The second case is $\alpha_i \neq j$. In the first case, we have that $|\alpha_i - i| + |\alpha_j - j| = 2|j - i|$, and since $j > i + 1$, we have that $|j - i| \geq 2$. In the second case, we consider (a) $\alpha_i \neq j$ and $\alpha_j = i$ and (b) $\alpha_i \neq j$ and $\alpha_j \neq i$. In case (a), there is $k \neq i, j$ such that $\alpha_k = j$. Thus, $|\alpha_j - j| \geq 2$ and $|\alpha_k - j| + |\alpha_i - i| \geq 2$. Therefore $|\alpha_j - j| + |\alpha_k - j| + |\alpha_i - i| \geq 4$. In case (b), there are l, k with $l \neq \{i, j\}$ and $k \neq \{i, j\}$ such that $\alpha_k = j$ and $\alpha_l = i$. Thus, we have that $|\alpha_i - i| + |\alpha_j - j| + |\alpha_k - k| + |\alpha_l - l| \geq 4$.

Now, we show that of $E(\alpha) \geq 2$, there are i, j with $j > i + 1$ such that $\alpha_i \neq i$ and $\alpha_j \neq j$. By assumption, we have that $\sum_{i=1}^N |\alpha_i - i| \geq 4$. This immediately implies that there are i, j such that $\alpha_i \neq i$ and $\alpha_j \neq j$. We show that $j > i + 1$. Assume instead that $j = i + 1$. In this case, there must be $k \neq \{i, j\}$ such that $\alpha_k \neq k$ because otherwise we would have $\alpha_i = j$ and $\alpha_j = i$ and therefore $\sum_{i=1}^N |\alpha_i - i| = 2$. Thus, we have two possibilities $k < i$ or $k > j$. If $k < i$, we have $\alpha_k \neq k$ and $\alpha_j \neq j$ with $j > k + 1$. If $k > j$, we have $\alpha_k \neq k$ and $\alpha_i \neq i$ with $k > i + 1$. This concludes the proof. ■

In the next section, we show how to use the state estimation and control computation on a partial order for the solution of the described control problem.

B. Dynamic control on a partial order

In this section, we introduce the partial order (χ, \leq) and the set \tilde{S} extension of S in χ . Let (χ, \leq) be the partial order with $\chi = \mathbb{N}^N$ with order established componentwise. For any set $X \subseteq \mathbb{N}^N$, we denote $[X]_j$ the projection along j of such set. Obviously, if X is an interval in \mathbb{N}^N , the set $[X]_j$ will be an interval in \mathbb{N} . Then, we have the following characterization of the set \tilde{S} .

Proposition 5.2: A set $\tilde{S} \subseteq \chi$ with $S = \tilde{S} \cap \mathcal{U}$ is given by $\tilde{S} = \cup_i \tilde{S}_i$, in which \tilde{S}_i for each i are intervals of four types:

- (a) there are $l < j$ such that $[\tilde{S}_i]_l = [l + 1, N]$ and $[\tilde{S}_i]_j = [j + 1, N]$;
- (b) there are $l < j$ such that $[\tilde{S}_i]_l = [1, l - 1]$ and $[\tilde{S}_i]_j = [j + 1, N]$;
- (c) there are $l < j$ with $j > l + 1$ such that $[\tilde{S}_i]_l = [l + 1, N]$ and $[\tilde{S}_i]_j = [1, j - 1]$;
- (d) there are $l < j$ such that $[\tilde{S}_i]_l = [1, l - 1]$ and $[\tilde{S}_i]_j = [1, j - 1]$.

Proof: We show that if \tilde{S} satisfies (a)–(d) then $\tilde{S} = S \cap \mathcal{U}$. To show this, we show that for any $\alpha \in S$ also $\alpha \in \tilde{S}$ and for any $\alpha \in \tilde{S}$ we have that also $\alpha \in S$.

If $\alpha \in \tilde{S}$ then, we have one of the four cases above. In case (a) $\alpha_l \neq l$ and $\alpha_j \neq j$. If $j > l + 1$ then $\alpha \in S$, otherwise there must exist k such that $\alpha_k \neq k$ smaller than l or greater than j . In case (b) also $\alpha_l \neq l$ and $\alpha_j \neq j$ and since $\alpha_l \neq j$, there must be $\alpha_k \neq k$. Case (c) is trivial as $\alpha_j \neq j$ and $\alpha_l \neq l$ with $j > l + 1$. In case (d), there must be $k < l$ such that $\alpha_k = l$. As a consequence, we have shown that if $\alpha \in \tilde{S}$, also $\alpha \in S$.

If $\alpha \in S$, we have that $\alpha_l \neq l$ and $\alpha_j \neq j$ for $j > l + 1$. This implies that $\alpha_l \in [l + 1, N]$ or $\alpha_l \in [1, l - 1]$ and the same for α_j with the constraint that if $\alpha_l \in [l + 1, N]$ and $\alpha_j \in [1, j - 1]$, then $j > l + 1$. ■

Next, we show that the system admits a dynamic output feedback and we apply Theorem 4.1 to construct it. The proofs of the following propositions are not included in this paper for space limitations and they can be found in [5]. Let us define the extension $\tilde{F} : \chi \times \mathcal{Z} \rightarrow \chi$ as F with now $\alpha \in \mathbb{N}^N$. Clearly, $\tilde{F}|_{\mathcal{U} \times \mathcal{Z}} = F$. Also, we define $\tilde{h} : \chi \times \mathcal{Z} \rightarrow \mathcal{Z}$ as h with $\alpha \in \mathbb{N}^N$. Also, we have that $\tilde{h}|_{\mathcal{U} \times \mathcal{Z}} = h$. Let us denote $\tilde{Y} = \{x \in \chi \mid z' = \tilde{h}(x, z)\}$. One can easily check that this set is an interval for any pair z, z' . Also, the function \tilde{F} is an order isomorphism on the set \tilde{Y} (more details can be found in [6]). The function $\tilde{G} : \chi \times \mathcal{I} \rightarrow \chi$ is defined as G in which the first argument now belongs to \mathbb{N}^N . Then $\tilde{f} = \tilde{G} \circ \tilde{F}$, in which one can check that $\tilde{f}|_{\mathcal{U} \times \mathcal{Z} \times \mathcal{I}} = f$. We thus have the extended system $\tilde{\Sigma} = (\chi \times \mathcal{Z}, \mathcal{Y}, \mathcal{I}, (\tilde{f}, \tilde{h}), \tilde{g})$. One can verify that $\tilde{Y} \cap \tilde{S}$ is an interval in χ for any y and that if P is an interval, also $\tilde{F}(P, z)$ is an interval. Then, we have the following proposition that establishes that the extended system $\tilde{\Sigma} = (\chi \times \mathcal{Z}, \mathcal{I}, \mathcal{Y}, (\tilde{f}, \tilde{h}), \tilde{g})$ satisfies the dynamic controllability condition with respect to \tilde{S} if $N > 4$.

Proposition 5.3: Let $P = \tilde{Y} \cap \tilde{S}$. If $N > 4$ there is $u \in \mathcal{I}$ such that $\tilde{f}(P, z, u) \subseteq \tilde{S}$.

The set \tilde{S} is given by the union of a large number of intervals and at each step there may be several choices of sets contained in \tilde{S} in which it is possible to keep the system state. Thus, at each step we determine a particular subset $\tilde{X} \subset \tilde{S}$ in which we want (and for which it is possible) to keep the system state. On the basis of such a set, we then show how to find an order compatible tuple for an interval $P \subseteq \chi$ once a lattice $(\tilde{\mathcal{I}}, \leq)$ has been established. The following algorithm, computes \tilde{X} componentwise.

Algorithm 5.1: Let $P \subseteq \tilde{Y} \cap \tilde{S}$ be an interval, and let $P' = \tilde{F}(P, z)$.

Initialize $\tilde{X}_i = [1, N]$, $flag_i = 0$ for all i

For $i = 1 : N$

If $\min(P'_i) = i$ and $flag_{i-1} = 0 \implies \tilde{X}_i = [1, i-1] \cup [i+1, N]$ and $flag_i = 1$

End

For $i = 2 : N$

If $\max(P'_{i-1}) = i-1$ and $flag_{i-1} = 0 \implies \tilde{X}_{i-1} = [1, i-2] \cup [i, N]$ and $flag_i = 1$

End

For $i = 1 : N$

If $\min(P'_i) \geq i+1$ and $flag_{i+1} = 0 \implies \tilde{X}_i = [i+1, N]$
If $\max(P'_i) \leq i-1$ and $flag_i = 0 \implies \tilde{X}_i = [1, i-1]$

End

In the next proposition, we show that the set $\tilde{X} = \tilde{X}_1 \times \dots \times \tilde{X}_N$ is contained in the set \tilde{S} and that there exist an input $u \in \mathcal{I}$ that maps the interval $P \subseteq \tilde{S} \cap \tilde{Y}$ into \tilde{X} . Since \tilde{X} depends on the set P of Algorithm 5.1, we will use the notation $\tilde{X}(P)$. Then, we show how to exploit $\tilde{X}(P)$ in order to compute an order compatible tuple for P .

Proposition 5.4: Let $\tilde{X}(P)$ be computed by Algorithm 5.1 for $P \subseteq \tilde{Y} \cap \tilde{S}$ and let $P' = \tilde{F}(P, z)$. Then $\tilde{X}(P) \subseteq \tilde{S}$ and $\{u \in \mathcal{I} \mid \tilde{G}(P', u) \subseteq \tilde{X}(P)\}$ is not empty.

Thus, $\{u \in \mathcal{I} \mid \tilde{f}(P, z, u) \subseteq \tilde{X}(P)\}$ is nonempty for any interval $P \subseteq \tilde{Y} \cap \tilde{S}$ and it is contained in the set $\{u \in \mathcal{I} \mid \tilde{f}(P, z, u) \subseteq \tilde{S}\}$. The next proposition shows that system $\tilde{\Sigma}$ admits an order compatible input extension on $(\tilde{\mathcal{I}}, \leq)$, where $\tilde{\mathcal{I}} = \{-1, 0, 1\}^N$.

Proposition 5.5: Let $P \subseteq \tilde{Y} \cap \tilde{S}$ be an interval, let $P' = \tilde{F}(P, z)$, and let $\tilde{X}(P)$ as computed by Algorithm 5.1. Then, $\{u \in \mathcal{I} \mid \tilde{G}(P', u) \subseteq \tilde{X}(P)\} = \mathcal{I} \cap \{\tilde{u} \in \tilde{\mathcal{I}} \mid \tilde{f}'([L^*, U^*], u) \subseteq [L', U']\}$ with $\tilde{\mathcal{I}} = \{-1, 0, 1\}^N$, in which

- (i) $L_i^* = U_i^* = i$ for all i such that $\tilde{X}_i = [1, i-1] \cup [i+1, N]$ or $\tilde{X}_i = [1, N]$, $L_i^* = U_i^* = i-1$ if $\tilde{X}_i = [1, i-1]$, and $L_i^* = U_i^* = i+1$ if $\tilde{X}_i = [i+1, N]$;
- (ii) $\tilde{f}'([L^*, U^*], u) = (\tilde{f}'_1([L_1^*, U_1^*], u_{k_1}), \dots, \tilde{f}'_N([L_N^*, U_N^*], u_{k_N}))$ with $u_{k_i} = u_i$ if $\tilde{X}_i = [1, i-1] \cup [i+1, N]$ or if $\tilde{X}_i = [1, N]$, $u_{k_i} = u_{i-1}$ if $\tilde{X}_i = [1, i-1]$, and $u_{k_i} = u_{i+1}$ if $\tilde{X}_i = [i+1, N]$;
- (iii) $\tilde{f}'_i(x_1, x_2) = x_1 + x_2$ for $x_1 \in [1, N]$ and $x_2 \in \{-1, 0, 1\}$;
- (iv) $[L'_i, U'_i] = [1, i-1]$ if $P'_i = [i, N]$ and $\tilde{X}_i = [1, i-1] \cup [i+1, N]$, or if $\tilde{X}_i = [1, i-1]$; $[L'_i, U'_i] = [i+1, N]$ if $P'_i = [1, i]$ and $\tilde{X}_i = [1, i-1] \cup [i+1, N]$, or if $\tilde{X}_i = [i+1, N]$; $[L'_i, U'_i] = [1, N]$ if $\tilde{X}_i = [1, N]$.

Clearly, \tilde{f}' is an order isomorphism in the second argument for any $x \in [L^*, U^*]$ and order preserving in the first argument. This along with $\{u \in \mathcal{I} \mid \tilde{f}(P, z, u) \subseteq \tilde{X}(P)\} \subseteq \{u \in \mathcal{I} \mid \tilde{f}(P, z, u) \subseteq \tilde{S}\}$ with $P \subseteq \tilde{Y} \cap \tilde{S}$ implies that system $\tilde{\Sigma}$ admits an order compatible input extension.

For mapping any interval $[L, U] \subset \chi$ through the function \tilde{G} with an input $u \in \mathcal{I}$ by computing \tilde{G} on the minimal number of elements in $[L, U]$, we break the set $[L, U]$ in intervals on which \tilde{G} is an order isomorphism for that particular input. This can always be done. The next proposition provides the intervals on which \tilde{G} is an order isomorphism.

Proposition 5.6: For any $u \in \mathcal{I}$ and any $[L, U] \in \chi$, let $[L, U] = [L_1, U_1] \times \dots \times [L_N, U_N]$ with $[L_i, U_i] = [L_i^1, U_i^1] \cup$

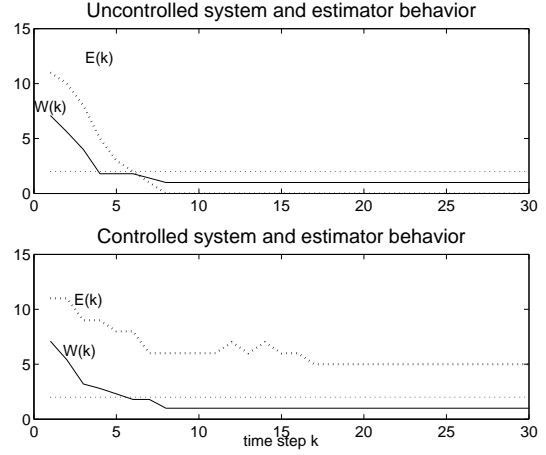


Fig. 3. Convergence plots of the estimator and of the entropy for the case in which $N = 10$ and $\alpha(0) = (2, 5, 6, 1, 3, 10, 7, 8, 4, 9)$. The upper plots show the behaviors when the system is not controlled. The lower plots show the behavior of the system when the control input is applied. The horizontal dotted line corresponds to an entropy value equal to two.

$\dots \cup [L_i^{M_i}, U_i^{M_i}]$ and $[L_i^j, U_i^j] \cap [L_i^k, U_i^k] = \emptyset$. Assume that the sets $[L_i^j, U_i^j]$ for any j satisfy the following conditions

- (i) if $u_j = 1$ then $j \in [L_i, U_i] \implies L_i^p = U_i^p = j$ for some p , $j+1 \in [L_i, U_i] \implies L_i^p = U_i^p = j+1$ for some p ;
- (ii) if $u_j = -1$ then $j \in [L_i, U_i] \implies L_i^p = U_i^p = j$ for some p , $j-1 \in [L_i, U_i] \implies L_i^p = U_i^p = j-1$ for some p .

Then, $\tilde{G}([L^J, U^J], u) \implies [\tilde{G}([L^J, U^J], u), \tilde{G}([L^J, U^J], u)]$ is an order isomorphism for any $L^J \leq U^J$ with $L^J = (L_1^{j_1}, \dots, L_N^{j_N})$ and $U^J = (U_1^{j_1}, \dots, U_N^{j_N})$ for $j_i \in \{1, \dots, M_i\}$.

The proof of this proposition is apparent once one notices that $u_j = 1$ leads to a swapping of j with $j+1$ in each coordinate set $[L_i, U_i]$. The next proposition shows that system $\tilde{\Sigma}$ is output interval compatible with respect to \tilde{S} .

Proposition 5.7: Let $\tilde{Y} = [L_y, U_y]$ and $[L, U] \subseteq \chi$. Assume $\tilde{G}([L, U], u) \subseteq \tilde{X} \subset \tilde{S}$ and let $[L, U] = [L_1, U_1] \times \dots \times [L_N, U_N]$ with $[L_i, U_i] = [L_i^1, U_i^1] \cup \dots \cup [L_i^{M_i}, U_i^{M_i}]$ and the sets $[L_i^j, U_i^j]$ as in Proposition 5.6. Then, we have that $\tilde{G}_i([L_i, U_i], u) \cap [L_{y,i}, U_{y,i}] \subseteq \tilde{X}_i$ for all i .

By virtue of Proposition 5.4 and of Proposition 5.5, system $\tilde{\Sigma}$ admits an order compatible input extension with respect to \tilde{S} on $(\tilde{\mathcal{I}}, \leq)$. Finally, by virtue of Proposition 5.7 system $\tilde{\Sigma}$ is also output interval compatible with respect to \tilde{S} . Thus, all the properties required for the application of Theorem 4.1 are satisfied. The next section shows simulation results obtained by constructing the lower-upper bound estimator of Theorem 4.1, which is used to compute an allowed input to the system.

C. Simulation results

We consider the system starting with a random (unknown) assignment $\alpha(0) \in S$ such that $\tilde{Y}(0) \subset \tilde{S}$, that is, the initial entropy of the assignment is larger or equal than two. Given $P \subseteq \chi$, which at the first step is equal to $\tilde{Y}(0)$ and in general it is equal to $[L(k), U(k)]$, we compute by means of Algorithm 5.1 the set $\tilde{X}(P)$ in which we want to map the state at the

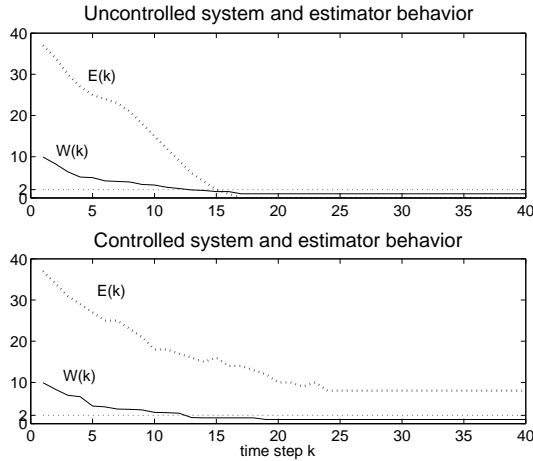


Fig. 4. Convergence plots of the estimator and of the entropy for the case in which $N = 15$ and $\alpha(0) = (4, 8, 9, 2, 13, 15, 6, 5, 12, 10, 1, 14, 3, 7, 11)$. The upper plots show the behaviors when the system is not controlled. The lower plots show the behavior of the system when the control input is applied. The horizontal dotted line corresponds to an entropy value equal to two.

next step. Once $\tilde{X}(P)$ is computed, we use Proposition 5.5 to compute an order compatible tuple. Then, we use Lemma 4.1 to compute $u_L(k)$ and $u_U(k)$, that is, the lower and the upper bounds of the set of possible inputs. Among these inputs, we choose the one that has the largest number of zero components (smallest control action). We thus apply this input to the system. Then, we update $[L(k), U(k)]$ by exploiting Propositions 5.6 and 5.7 for using the update laws of Theorem 4.1.

In Figure 3 and in Figure 4, we report convergence plots of the estimator and of the entropy for different numbers N of robots. In particular, the convergence plot of the estimator is given by $W(k) = 1/N \sum_{i=1}^N |m_i(k)|$, in which $m_i(k)$ is the coordinate set $[L_i(k), U_i(k)]$ minus all the singletons that occur at other coordinates. The behavior of the entropy $E(k) = 1/2 \sum_{i=1}^N |\alpha_i(k) - i|$ is also depicted. Note that the controlled system has always $E(k) \geq 2$. Also, note that the estimator with no input applied to the system not always converges before the state $\alpha(k)$ has reached $E(k) = 2$. Thus, a strategy that runs the estimator first, and only when it has converged the controller is applied to the system does not guarantee the specification. Even when the control input is applied to the system, the estimate sets converge to the current value $\alpha(k)$. In this work however, no statement has been made about the convergence properties of the estimator when a control input is applied to the system as we were only interested in satisfying the specifications.

The computation requirement for the implementation of the dynamic controller is proportional to N , that is, to the number of variables that need to be controlled. If we had not used any structure, we would have incurred in a number of computations at each step at least of the order of $(N!)^2$ as the size of the output set and of the set S are both of the order of $N!$. Note also that this simplification is not due to the fact that the dynamics decouples as it is heavily coupled

between the robots.

VI. C F W

In this work, we have shown how to exploit a partial order structure on the set of inputs and discrete states in order to design a dynamic controller that keeps the system state in a desired set. The proposed algorithm updates two variables at each step: the lower and upper bound of the set of all possible discrete variables compatible with the measured output sequence. These bounds are then used to compute a lower and an upper bound of a set of inputs that keep the system in the desired set at the next step. We showed how to apply the proposed algorithm to a multi-robot system involving two teams competing against each other in order to design a dynamic winning controller for one of the teams.

The computation requirement for implementing the algorithm depends on the partial order chosen. Conditions under which the assumptions needed for the proposed controller construction are verified need to be investigated. We conjecture that, as in the case of state estimation (see [6]), also for the dynamic control problem if the system is controllable by dynamic output feedback it is always possible to find suitable partial orders and system extensions for which the assumptions are verified. The proposed method does not rely on enumeration and exhaustive search, and therefore it is in principle applicable to state estimation and control of continuous states. This will be investigated in future work. Finally, we would like to apply this approach to more complex multi-agent game scenarios in which the agents can move in a plane as opposed to a line with more complex dynamics.

R

- [1] E. Asarin, O. Maler, and A. Pnueli. Symbolic controller design for discrete and timed systems. *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science, volume 999, P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, Eds. Springer Verlag, pages 1–20, 1995.
- [2] J. Aubin, J. Lygeros, M. Quincampoix, S. Sastry, and N. Seube. Impulse differential inclusions: A viability approach to hybrid systems. *IEEE Transactions on Automatic Control*, 47(1):2–20, 2002.
- [3] R. D’Andrea, R. M. Murray, J. A. Adams, A. T. Hayes, M. Campbell, and A. Chaudry. The RoboFlag Game. In *American Control Conference*, pages 661–666, 2003.
- [4] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 2002.
- [5] D. DelVecchio. Discrete output feedback for a class of hybrid systems on a lattice. In [url:http://www.eecs.umi.ch/~ddv/Report1.pdf](http://www.eecs.umi.ch/~ddv/Report1.pdf).
- [6] D. DelVecchio, R. M. Murray, and E. Klavins. Discrete state estimators for systems on a lattice. *Automatica*, 42(2):271–285, 2006.
- [7] A. Deshpande and P. Varaiya. Viable control of hybrid systems. In *Hybrid Systems II*, Lecture Notes in Computer Science, volume 999, P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, Eds. Springer Verlag, pages 128–147, 1995.
- [8] P. E. Caines and S. Wang. Classical and logic based regulator design and its complexity for partially observed automata. In *Conf. on Decision and Control*, pages 132–137, 1989.
- [9] T. A. Henzinger, P. H. Ho, and H. Wong-Toi. A user guide to HYTECH. In *TACAS 95: Tools and Algorithms for the construction and analysis of systems*, Lecture Notes in Computer Science, vol. 1019, E. Brinksma, W. Cleaveland, K. Larsen, T. Margaria, and B. Steffen, Eds. Springer-Verlag, pages 41–71, 1995.
- [10] E. Klavins and R. M. Murray. Distributed algorithms for cooperative control. *Pervasive Computing*, 3:56–65, 2004.

- [11] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, 1989.
- [12] K. Rohloff and S. Lafortune. On the synthesis of safe control policies in decentralized control of discrete event systems. *IEEE Transactions on Automatic Control*, 48(6):1064–1068, 2003.
- [13] P. Saint-Pierre. Approximation of viability kernels and capture basins for hybrid systems. In *European Control Conference*, pages 2776–2783, 2001.
- [14] E. D. Sontag. *Mathematical Control Theory*. Springer, 1998.
- [15] W. Thomas. On the synthesis of strategies in infinite games. In *Proceedings of the STACS 95*, E. W. Mayr and C. Puech, Eds. Springer Verlag, pages 1–13, 1995.
- [16] C. J. Tomlin, I. Mitchell, A. M. Bayen, and M. Oishi. Computational techniques for the verification of hybrid systems. *Proceedings of the IEEE*, 91(7):986–1001, 2003.
- [17] H. Wong-Toi. The synthesis of controllers for linear hybrid automata. In *Conf. on Decision and Control*, pages 4607–4613, 1997.

A. Appendix

A partial order is a set χ with a partial order relation “ \leq ”, and we denote it by the pair (χ, \leq) . For any $x, w \in \chi$, the $\sup\{x, w\}$ is the smallest element that is larger than both x and w . In a similar way, the $\inf\{x, w\}$ is the largest element that is smaller than both x and w . We define the *join* “ \vee ” and the *meet* “ \wedge ” of two elements x and w in χ as (1) $x \vee w = \sup\{x, w\}$ and $x \wedge w = \inf\{x, w\}$; (2) if $S \subseteq \chi$, $\bigvee S = \sup S$ and $\bigwedge S = \inf S$. If $x \wedge w \in \chi$ and $x \vee w \in \chi$ for any $x, w \in \chi$, then (χ, \leq) is a *lattice*. Let (χ, \leq) be a lattice and let $S \subseteq \chi$ be a non-empty subset of χ . Then, (S, \leq) is a *sublattice* of χ if $a, b \in S$ implies that $a \vee b \in S$ and $a \wedge b \in S$. Any interval sublattice of (χ, \leq) is given by $[L, U] = \{w \in \chi \mid L \leq w \leq U\}$ for $L, U \in \chi$. That is, this special sublattice can be represented by only two elements. Let (P, \leq) and (Q, \leq) be partially ordered sets. A map $f : P \rightarrow Q$ is (i) an *order preserving map* if $x \leq w \implies f(x) \leq f(w)$; (ii) an *order embedding* if $x \leq w \iff f(x) \leq f(w)$; (iii) an *order isomorphism* if it is order embedding and it maps P onto Q . For more details, the reader is referred to [4].