Observers for a Class of Hybrid Systems on a Lattice

Domitilla Del Vecchio and Richard M. Murray

Control and Dynamical Systems
California Institute of Technology
1200 E California Boulevard, Mail Stop 107-81
Pasadena, CA 91125
{ddomitilla, murray}@cds.caltech.edu

Abstract. In this paper we consider the problem of estimating discrete variables in a class of hybrid systems where we assume that the continuous variables are available for measurement. Using lattice and order theory we develop a framework for constructing an observer on an enlarged space of variables with lattice structure, which updates only two variables at each step. We apply our ideas to a multi-robot system example, the RoboFlag Drill.

1 Introduction

In the last decade hybrid systems models have become very popular in the control community. Some of the systems under study have been changing from systems governed by continuous differential equations, to systems characterized by very large numbers of discrete and continuous variables whose evolution is determined by both continuous dynamics and logics. Examples include internet systems, continuous plants controlled by digital controllers, and multi-agent systems. The interplay of continuous dynamics and decision protocols renders these system interesting and complicated enough that new mathematical tools are needed for the sake of analysis and control. Issues like controllability and observability arise naturally when trying to analyze the properties of these systems for control.

The problem of estimating and tracking the values of non-measurable variables in hybrid systems with computational effort comparable to the one needed for simulating the hybrid system itself is a challenging one. Bemporad et al. [1] show that observability properties are hard to check for hybrid systems and an observer is proposed that requires large amounts of computation. Caines [2] shows that the complexity of the observer often arises from the need to compute maps on large sets of values, corresponding to the set of all possible internal states compatible with the observed output sequence. These same difficulties are encountered in [8], where the proposed observer fails to be applicable for large problem sizes.

In this paper we attack the problem of estimation and tracking of non-measurable variables in hybrid systems by considering a simplified scenario where among all the system's variables the continuous variables are available for measurement. We thus focus on the problem of estimating the discrete variables of the system. The discrete and continuous variables values are usually heavily coupled through logical operations and continuous dynamics, and the question of estimating the values of the discrete variables is not a trivial one. The continuous variables may represent physical quantities such as positions and velocities, while discrete variables may represent the state of the internal logical system that is used for control and coordination such as in the case of decentralized multi-robot systems.

Our point of view is that some of the complexity issues such as those encountered in [8] or [2] can be avoided by finding a good way of representing the sets of interest and by finding a good way of computing maps on them. As a naive example consider the set $\mathcal{S}$ of all natural numbers between one and one thousand. This set can be represented as $\mathcal{S} = [1, 1000]$, without the need of listing all its elements. Suppose we want to know what set $\mathcal{S}$ is mapped to by a map ϕ. We can either compute such a map on all the elements of $\mathcal{S}$, or we can compute it on the least and greatest elements of $\mathcal{S}$. In this case if the map ϕ has some properties, the set $\phi(\mathcal{S})$ can be deduced by $\phi(1)$ and $\phi(1000)$ without too much additional computation. This simplification is possible thanks to the order structure naturally associated to $\mathbb{N}$ and thanks to the structure of the map ϕ.

In this paper we formalize these ideas using lattice theory. In particular given a system Σ defined on its space of variables, we extend it to a larger space of variables that has lattice structure so as to obtain the extended system $\tilde{\Sigma}$. Under certain properties verified by the extension $\tilde{\Sigma}$, an observer for system Σ can be constructed, which updates at each step only two variables. Namely it updates the least and greatest element of the set of all values of discrete variables compatible with the output sequence and with the dynamics of Σ. The structure of the obtained observer resembles the structure of the Luenberger observer as it is obtained by "copying" the dynamics of the system Σ and by correcting it according to the measured output values. We present initial results that introduce the mathematical framework and point to next steps.

The contents of this paper is as follows. In Section 2 we review some basic definitions and results on lattice theory, and some basic terminology of transition systems. The main result is given in Section 3 where we provide an explicit construction for the observer which updates least and maximum elements on a proper lattice structure. In Section 4 we introduce a multi-robot system, the RoboFlag Drill, and in Section 5 we show how to apply our ideas to this example. We then conclude the paper with some simulation results on the RoboFlag Drill system in Section 6.

2 Basic Concepts

In this section we give first some background on lattice theory as it can be found in [3]. Then we recall basic definitions on transition systems (see [7] for

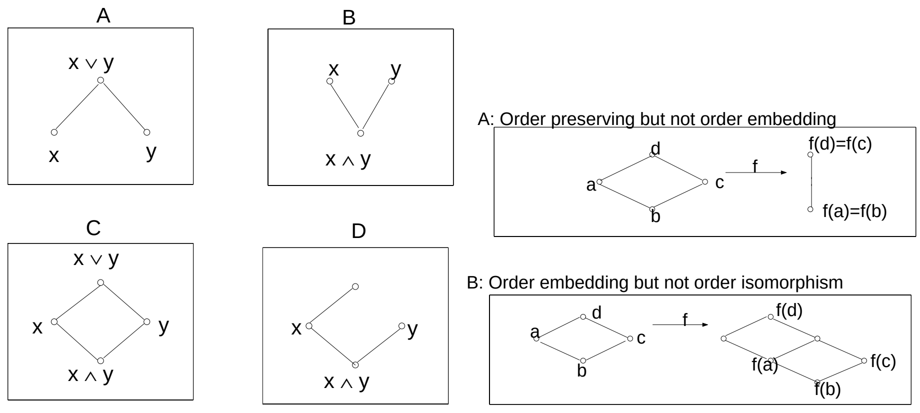
A
x ∨ y
x
y
B
x
y
x ∧ y
C
x ∨ y
x
y
x ∧ y
D
x
y
x ∧ y
A: Order preserving but not order embedding
d
a
c
b
f
f(d)=f(c)
f(a)=f(b)
B: Order embedding but not order isomorphism
d
a
c
b
f
f(d)
f(a)
f(c)
f(b)

Fig. 1. (Left) In diagram A and B x and y are not related, but they have a join and a meet. In diagram C we show a complete lattice, and in diagram D we show an ordered set that is not a lattice, since the elements x and y have a meet, but not a join. (Right) In diagram A we show a map that is order preserving but not order embedding. In diagram B we show an order embedding map that is not order isomorphism: any two elements maintain the same order relation, but in between c and d there is nothing, while in between $f(c)$ and $f(d)$ some other elements appears (i.e. it is not onto).

more details). Finally we recall some basic observability definitions as they can be found in many references (see [8] for example).

2.1 Lattice theory

Given a set χ with an order relation "$\leq$", 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 $S \subseteq \chi$, $\bigwedge S = \inf\ S$;

where by $\sup\{x, w\}$ we denote the smallest element in χ that is bigger than both x and w, and we denote by $\inf\{x, w\}$ the biggest element in χ that is smaller than both x and w. In analogous way we denote by $\sup\ S$ the smallest element in χ that is bigger than all the elements in S, and we denote by $\inf\ S$ the biggest element in χ that is smaller than all the elements in S.

Let χ be a non-empty ordered set. If $x \wedge w$ and $x \vee w$ exist for any $x, w \in \chi$, then χ is a *lattice*. If $\bigvee S$ and $\bigwedge S$ exist for all $S \subseteq \chi$, then χ is a *complete lattice*. Notice that any finite lattice is complete. In Figure 1 (left) we report diagrams showing ordered sets. From the diagram it is easy to tell when one element is less than another: $x < w$ if and only if there is a sequence of connected line segments moving upward from x to w. Let χ be an ordered set. Then χ is a *chain* if for all $x, w \in \chi$, either $x \leq y$ or $y \leq x$, that is any two elements are comparable. At the opposite extreme of a chain is an antichain. The ordered set χ is an *antichain* if $x \leq y$ if and only if $x = y$. Let χ be a lattice and let $\emptyset \neq S \subseteq \chi$ be a subset of χ. Then S is a *sublattice* of χ if $a, b \in S$ implies that $a \vee b \in S$ and $a \wedge b \in S$.

Definition 1. Let P and Q be ordered sets. A map $f : P \to Q$ is said to be

(i) *order preserving* if $x \leq w \implies f(x) \leq f(w)$;
(ii) *order embedding* if $x \leq w \iff f(x) \leq f(w)$;
(iii) *order isomorphic* if it is order embedding and it maps P *onto* Q.

Definition 2. If P and Q are lattices, then a map $f : P \to Q$ is said to be an *homomorphism* if f is *join-preserving* and *meet-preserving*, that is for all $x, w \in P$ we have that $f(x \vee w) = f(x) \vee f(w)$ and $f(x \wedge w) = f(x) \wedge f(w)$. A bijective homomorphism is a *(lattice) isomorphism.*

Every order isomorphic map faithfully mirrors the structure of P onto Q. In Figure 1 (right) we show some examples.

Lemma 1. *Let P and Q be ordered sets and $f : P \to Q$ be an order isomorphism. Then f preserves all joins and meets, that is for any $S \subseteq P$ whenever $\bigvee S$ ($\bigwedge S$) exists in P then $\bigvee f(S)$ ($\bigwedge f(S)$) exists in Q and*

$$f(\bigvee S) = \bigvee f(S), \quad \text{and} \quad f(\bigwedge S) = \bigwedge f(S)\ .$$

2.2 State Transition Systems

For completeness, we review the basic definitions used in transition systems as described more completely in other work [7]. Consider a set of variable symbols V with types *type*(v) for each $v \in V$. A *state* s is a function from V into U where $U = \bigcup_{v \in V} \mathit{type}(v)$. The set of all states is denoted S. For a subset W of V, we denote by $s|_W$ the restriction of s to W, so that we have that $S|_W = \{s|_W \ : \ s \in S\}$. A *transition relation* on S is a relation $R \subseteq S \times S$. If sRs' and $v \in V$, we will write v to refer to $s(v)$ and v' to refer to $s'(v)$. For example, if we denote R by $x' < y \vee y' = z$ then $sRs' \iff s'(x) < s(y) \vee s'(y) = s(z)$. Note that sRs' is equivalent to writing $(s, s') \in R$.

Given a transition relation R, an *execution* of R is a sequence $\sigma = \{s_k\}_{k \in \mathbb{N}}$ such that $s_k R s_{k+1}$ for all $k \in \mathbb{N}$. The set of all executions of R is denoted $\mathcal{E}(R)$. If $\sigma \in \mathcal{E}(R)$ is fixed and $v \in V$ we denote by $v(k)$ the value $\sigma(k)(v)$. The *trajectory* of $v \in V$ with respect to σ is the sequence $\{\sigma(k)(v)\}_{k \in \mathbb{N}}$.

We now recall the notion of observability for transition systems as it can be found in [8].

Definition 3. Given a transition relation R on S and an output map $g : S \to U$, for some U, two executions $\sigma_1, \sigma_2 \in \mathcal{E}(R)$ are *distinguishable* if there exists a k such that $g(\sigma_1(k)) \neq g(\sigma_2(k))$.

Definition 4. (Observability) The transition relation R is said to be *observable* with respect to the output function $g : S \to U$ if any two executions $\sigma_1, \sigma_2 \in \mathcal{E}(R)$ are distinguishable.

We will consider state transition systems with both discrete and continuous variables. V_C is the set of continuous variables that we denote with z with $type(z) = \mathbb{R}^N$ for all $z \in V_C$, and V_D is the set of discrete variables that we denote with α with $type(\alpha) = \mathcal{U}$ for all $\alpha \in V_D$. In this paper we assume that $V = V_C \cup V_D$, with $V_C \cap V_D = \emptyset$. Then a state $s \in S$ can be also reviewed as the pair $(s|_{V_C}, s|_{V_D})$. We leave $\mathcal{U}$ unspecified for the moment. We also assume that the set of continuous variables coincide with the set of measurable variables, that is $g : S \to S|_{V_C}$.

Let $(s, s') \in R$, that is $((s|_{V_C}, s_{V_D}), (s'|_{V_C}, s'|_{V_D})) \in R$. We now define two other relations R_C and R_D from R. The first relates s with any other state of the form $(s'|_{V_C}, *)$ and the second relates s with any other state of the form $(*, s'|_{V_D})$. Clearly $R_C \cap R_D = R$. Moreover we restrict to the case in which R is deterministic, so that R_C and R_D become functions, which we denote respectively by $h : \mathcal{U} \times \mathbb{R}^N \to \mathbb{R}^N$ and $f : \mathcal{U} \times \mathbb{R}^N \to \mathcal{U}$. In what follows we will denote a transition system Σ by the couple (f, h) assuming that V_C is the set of measurable variables.

3 Observer Construction

In this section we show that if we can extend the space $\mathcal{U}$ to a space χ with lattice structure, and if we can extend the maps f and g to the whole χ such that f is order isomorphic on suitable subsets of χ, then in the case in which Σ is observable we can construct a system that at each step updates only two variables. These variables are the join and the meet of the set of all possible α's values compatible with the output sequence; moreover the set that they define converges asymptotically to a set whose intersection with $\mathcal{U}$ is the current value of α. This is stated formally in the following theorem

Theorem 1. *Consider the system $\Sigma = (f, h)$ with $h : \mathcal{U} \times \mathbb{R}^N \to \mathbb{R}^N$ and $f : \mathcal{U} \times \mathbb{R}^N \to \mathcal{U}$. Let $z \in \mathbb{R}^N$ denote the continuous variables and $\alpha \in \mathcal{U}$ the discrete variables. Assume that variables z are measurable, that is $y = z$. Assume that*

(i) There exist a lattice χ such that $\mathcal{U} \subseteq \chi$;

(ii) The map $h : \mathcal{U} \times \mathbb{R}^N \to \mathbb{R}^N$ can be extended to the whole χ as $\tilde{h} : \chi \times \mathbb{R}^N \to \mathbb{R}^N$, such that $\tilde{h}|_{\mathcal{U} \times \mathbb{R}^N} = h$ and

$$A_y(k) := \{x \in \chi : y(k+1)\} = \tilde{h}(y(k), x) = [\bigwedge A_y(k), \bigvee A_y(k)],$$

which means that $A_y \subseteq \chi$ is a lattice and is equal to $\{x : x \geq \bigwedge A_y \wedge x \leq \bigvee A_y\}$;

(iii) The map $f : \mathcal{U} \times \mathbb{R}^N \to \mathcal{U}$ can be extended to the whole χ as $\tilde{f} : \chi \times \mathbb{R}^N \to \chi$, such that $\tilde{f}|_{\mathcal{U} \times \mathbb{R}^N} = f$ and $\tilde{f} : A_y \to [\tilde{f}(\bigwedge A_y), \tilde{f}(\bigvee A_y)]$ is an order isomorphism;

(iv) System Σ is observable.

Then the following system

$$L(k) = \tilde{f}(L(k-1)) \vee (\bigwedge A_y(k)) \ , \tag{1}$$

$$U(k) = \tilde{f}(U(k-1)) \wedge (\bigvee A_y(k)) \ , \tag{2}$$

with $L(0) = \bigvee A_y(0)$ *and* $U(0) = \bigwedge A_y(0)$*, is such that*

(a) $\alpha \in [L(k), U(k)] \cap \mathcal{U}$ *for all* k *(correctness);*
(b) $|[L(k+1), U(k+1)]| \leq |[L(k), U(k)]|$ *(non-increasing error);*
(c) $|[L(k), U(k)] \cap \mathcal{U} - \alpha| \to 0$ *as* $k \to \infty$ *(convergence),*

where $\lfloor S \rfloor$ *denotes the cardinality of the set* S*. Moreover, if the extended system* $\tilde{\Sigma} = (\tilde{f}, \tilde{h})$ *defined on* $\chi \times \mathbb{R}^N$ *with output* z *is also observable, then properties (a)–(c) become:*

(a') $\alpha \in [L(k), U(k)]$*;*
(b') $|[L(k+1), U(k+1)]| \leq |[L(k), U(k)]|$*;*
(c') $L(k) \to \alpha(k)$ *and* $U(k) \to \alpha(k)$ *as* $k \to \infty$*.*

Proof. The proof proceeds in two steps. In the first step we show that Assumptions (i)–(iii) imply that

$$\text{for all } w \ \in [L(k+1), U(k+1)], \text{ there exists } x \in [L(k), U(k)] : w = \tilde{f}(x) \tag{3}$$

$$\text{for all } x \in [L(k), U(k)], \ x \in A_y(k). \tag{4}$$

In the second step we show that properties (3) and (4) together with Assumption (iv) imply property (c), (a) and (b).

Step 1. Property (4) can be proved directly using the definition of $L(k)$ and $U(k)$ given in expressions (1) and (2). In fact if $x \in [L(k), U(k)]$ then $x \leq U(k)$ and by (2) we have $x \leq \bigvee A_y(k)$. Also $x \geq L(k)$, which by (1) implies $x \geq \bigwedge A_y(k)$.

To prove (3) we first show that $\tilde{f}$ is an order isomorphism on each sub-lattice of $A_y(k)$, then we notice that $[L(k), U(k)]$ is a sublattice of $A_y(k)$, and therefore property (3) is a direct consequence of the definition of order isomorphic maps. Let $\bar{A}_y(k) = [\bar{l}_y, \bar{u}_y]$ for some $\bar{l}_y \in A_y(k)$ and $\bar{u}_y \in A_y(k)$. Clearly $\bar{A}_y(k) \subseteq A_y(k)$, and $\bar{A}_y(k)$ is a lattice. Therefore $\tilde{f} : \bar{A}_y(k) \to \tilde{f}(\bar{A}_y(k))$ is a lattice isomorphism since $\tilde{f}$ is an isomorphism on $A_y(k)$. We now show that $\tilde{f}(\bar{A}_y(k)) = [f(\bar{l}_y), f(\bar{u}_y)]$. To prove this equality we need to show that each element of the first set is contained in the second set, and *viceversa.* For any $z \in \tilde{f}(\bar{A}_y(k))$ we have $z \in [\bigwedge \tilde{f}(\bar{A}_y(k)), \bigvee \tilde{f}(\bar{A}_y(k))]$. Since $\tilde{f} : \bar{A}_y(k) \to \tilde{f}(\bar{A}_y(k))$ is a lattice isomorphism, by Lemma 1 $\tilde{f}(\bigwedge \bar{A}_y(k)) = \bigwedge \tilde{f}(\bar{A}_y(k))$ and $\tilde{f}(\bigvee \bar{A}_y(k)) = \bigvee \tilde{f}(\bar{A}_y(k))$. Then, since $\bar{l}_y = \bigwedge \tilde{f}(\bar{A}_y(k))$ and $\bar{u}_y = \bigvee \tilde{f}(\bar{A}_y(k))$, we have $z \in [\tilde{f}(\bar{l}_y), f(\bar{u}_y)]$. We now show that for any $z \in [\bigwedge \tilde{f}(\bar{A}_y(k)), \bigvee \tilde{f}(\bar{A}_y(k))]$ we also have $z \in \tilde{f}(\bar{A}_y(k))$. Since $\bigvee \bar{A}_y(k) \in A_y(k)$, and $\bigwedge \bar{A}_y(k) \in A_y(k)$, we have that $\tilde{f}(\bigvee \bar{A}_y(k)) \in \tilde{f}(A_y(k))$, and $\tilde{f}(\bigwedge \bar{A}_y(k)) \in \tilde{f}(A_y(k))$. This in turn implies that $\tilde{f}(\bigwedge A_y(k)) \leq$

$\tilde{f}(\bigwedge \bar{A}_y(k)) \leq z \leq \tilde{f}(\bigvee \bar{A}_y(k)) \leq \tilde{f}(\bigvee A_y(k))$, so that $z \in f(A_y(k))$. Since $z \in f(A_y(k))$, there exist $x \in A_y(k)$ such that $z = \tilde{f}(x)$. Since $\tilde{f} : A_y \to f(A_y(k))$ is order embedding we have that $\tilde{f}(\bigwedge \bar{A}_y(k)) \leq z = \tilde{f}(x) \leq \tilde{f}(\bigvee \bar{A}_y(k))$ implies $\bigwedge \bar{A}_y(k) \leq x \leq \bigvee \bar{A}_y(k)$, which in turn implies that $x \in \bar{A}_y(k)$, and therefore $z \in \tilde{f}(\bar{A}_y(k))$.

Step 2. Let us prove (a) first (correctness). We show this by induction on the step k. Initially $\alpha \in [L(0), U(0)] = [l_y, u_y]$. For the induction step assume that $\alpha(k) \in [L(k), U(k)]$, let us show that $\alpha(k+1) \in [L(k+1), U(k+1)]$. This can be shown by using the fact that $\tilde{f}$ is order preserving. In fact $L(k) \leq \alpha(k) \leq U(k)$ implies $\tilde{f}(L(k)) \leq \tilde{f}(\alpha(k)) \leq \tilde{f}(U(k))$. Also $\alpha(k+1) = \tilde{f}(\alpha(k)) \in [l_y(k+1), u_y(k+1)]$ therefore $\alpha(k+1) \leq (\tilde{f}(U(k)) \wedge u_y(k+1)) = U(k+1)$, and $\alpha(k+1) \geq (\tilde{f}(L(k)) \vee l_y(k+1)) = L(k+1)$.

To prove (b) we can directly use property (3). In fact by (3) we have that for each $w \in [L(k+1), U(k+1)]$ there is a $x \in [L(k), U(k)]$ such that $w = \tilde{f}(x)$. This in turn implies that $|[L(k+1), U(k+1)]| \leq |[L(k), U(k)]|$.

To prove (c) notice that by properties (3), (4), and the fact that $\mathcal{U}$ is invariant with respect to $\tilde{f}$, we have that for each $x' \in [L(k+1), U(k+1)] \cap \mathcal{U}$ there is $x \in [L(k), U(k)] \cap \mathcal{U}$, such that $x' = f(x)$, and $x \in A_y(k)$. This in turn implies that the sequence $\{x(k), y(k)\}_{k \in \mathbb{N}}$ corresponds to an execution σ of system Σ, that is $x(k) = \sigma(k)(x)$. Therefore for any $x, w \in [L(k), U(k)] \cap \mathcal{U}$, there are sequences $\{x(k), y(k)\}_{k \in \mathbb{N}}$ and $\{w(k), y(k)\}_{k \in \mathbb{N}}$ corresponding to executions σ_1 and σ_2 of Σ, where $x(k) = \sigma_1(k)(x)$, $w(k) = \sigma_1(k)(w)$, and $\sigma_1(k)(y) = \sigma_2(k)(y) = y(k)$ for all k. Since the system is observable, the two executions must coincide, that is $x(k) = w(k)$. Therefore there exist a k_0 such that for all $k \geq k_0$ we have that $|[L(k), U(k)] \cap \mathcal{U}| = 1$. This together with (a) proves (c).

We then refer to the system in equations (1) and (2) as an observer for Σ.

4 An Example: The RoboFlag Drill

In this section we consider a simplified version of the RoboFlag Drill system described in [5] that is similar to "capture the flag", only for robots. We do not propose to devise a strategy that addresses the full complexity of the game. Instead we examine the following very simple *drill* or exercise. Some number of blue robots with positions $(z_i, 0) \in \mathbb{R}^2$ must defend their zone $\{(x, y) \in \mathbb{R}^2 \mid y \leq 0\}$ from an equal number of incoming red robots. The positions of the red robots are $(x_i, y_i) \in \mathbb{R}^2$. An example for 5 robots is illustrated in Figure 2. The red robots move straight toward the blue defensive zone. The blue robots are assigned each to a red robot and they coordinate to intercept the red robots. Let N represent the number of robots in each team. The robots start with a random (bijective) assignment $\alpha : \{1, ..., N\}$ into $\{1, ..., N\}$. 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. We consider the problem

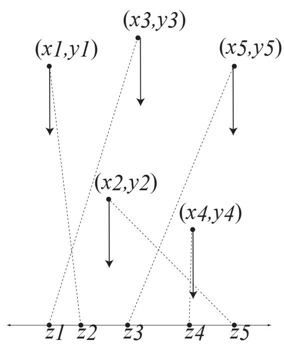

Fig. 2. Example of the RoboFloag Drill for 5 robots. Here $\alpha = \{3, 1, 5, 4, 2\}$ and the dashed lines indicates these assignments. The attackers are moving down along y direction as indicated by the arrows.

of estimating the current assignment α given the motions of the blue robots – which might be of interest to, for example, the red robots in that they may use such information to determine a better strategy of attack. We do not consider the problem of how they would change their strategy in this paper.

The system can be described with a guarded command program. Such programs are constituted by a set of clauses. Each clause is of the form *guard* : *rule*, where *guard* is the guard, and *rule* is the rule. When a guard becomes true the corresponding rule is executed (for more details see [4]). The description here is similar to the one in [5]). The red robot dynamics Σ_{Red} are described by the N clauses

$$y_i - \delta > 0 : y_i' = y_i - \delta$$

for $i \in \{1, ..., N\}$. These state simply that the red robots move a distance δ toward the defensive zone at each step. The blue robot dynamics Σ_{Blue} are described by the $3N$ clauses

$$z_i < x_{\alpha_i} : z_i' = z_i + \delta \ , \quad z_i > x_{\alpha_i} : z_i' = z_i - \delta, \quad z_i = x_{\alpha_i} : z_i' = z_i \tag{5}$$

for $i \in \{1, ..., N\}$. For the blue robots we assume that initially $z_i \in [z_{min}, z_{max}]$ and $z_i < z_{i+1}$ and that $x_i \leq z_i \leq x_{i+1}$ for all time. We define $x = (x_1, ..., x_N)$, $z = (z_1, ..., z_N)$, $\alpha = (\alpha_1, ..., \alpha_N)$. The assignment protocol dynamics Σ_{Assign} is defined by

$$x_{\alpha_i} \geq z_{i+1} \wedge x_{\alpha_{i+1}} \leq z_{i+1} : (\alpha_i', \alpha_{i+1}') = (\alpha_{i+1}, \alpha_i) \ , \tag{6}$$

which is a modification of the protocol presented in [5], since two adjacent robots switch assignments only if they are moving one against the other. The complete RoboFalg specification is then given by $\Sigma_{RF} := \Sigma_{Red} \cup \Sigma_{Blue} \cup \Sigma_{Assign}$. In particular the clauses in (5), representing Σ_{Blue}, model the function h that updates the continuous variables, and the clauses in (6), representing Σ_{Assign}, model the function f that updates the discrete variables.

RoboFlag Drill Observation Problem: Given initial values for x and y and the values of z corresponding to an execution of $\Sigma_{Blue} \cup \Sigma_{Assign}$, determine the value of α during that execution.

5 Observer Construction for The RoboFlag Drill System

In this section we first show that the RoboFalg Drill is observable, and then we show how Theorem 1 can be applied to construct the observer in equations (1- 2).

5.1 Observability

Lemma 2. *The system Σ represented by the guarded command program (5) and (6) with measurable variables z is observable.*

Proof. Since we are interested in the observability of the α trajectories, for proving observability we show that any two executions of $\Sigma_{Blue} \cup \Sigma_{Assign}$, σ_1 and σ_2, with $\{\alpha_1(k)\}_{k\in\mathbb{N}} \neq \{\alpha_2(k)\}_{k\in\mathbb{N}}$ have different output sequences. For output we consider the vector of directions of motion of the z_i . Since the measurable variables are the z_i's, also their direction of motion is measurable; let $g(\sigma(k)) = (g_1(\sigma(k)), ..., g_N(\sigma(k)))$ denote the vector of directions at step k for the execution σ. From equations (5) it is clear that the direction of motion depends only on α and not on z, therefore $g(\sigma) = g(\alpha)$. Note that every α trajectory reaches the equilibrium value $[1, ..., N]$, and therefore there is a step k at which $f(\alpha_1(k)) = f(\alpha_2(k))$ and $\alpha_1(k) \neq \alpha_2(k)$. Therefore it is enough to prove that for any $\alpha_1 \neq \alpha_2$ we have $g(\alpha_1) = g(\alpha_2) \implies f(\alpha_1) \neq f(\alpha_2)$. $g(\alpha_1) = g(\alpha_2)$ by (5) implies that (1) $z_{i,1} < x_{\alpha_{i,1}} \iff z_{i,2} < x_{\alpha_{i,2}}$ and (2) $z_{i,1} \geq x_{\alpha_{i,1}} \Leftrightarrow z_{i,2} \geq x_{\alpha_{i,2}}$. This implies that $x_{\alpha_{i,1}} \geq z_{i+1,1} \wedge x_{\alpha_{i+1,1}} \leq z_{i+1,1} \Leftrightarrow x_{\alpha_{i,2}} \geq z_{i+1,2} \wedge x_{\alpha_{i+1,2}} \leq z_{i+1,2}$. By letting $\alpha' = f(\alpha)$, we thus have that $(\alpha'_{i,1}, \alpha'_{i+1,1}) = (\alpha_{i+1,1}, \alpha_{i,1}) \Leftrightarrow (\alpha'_{i,2}, \alpha'_{i+1,2}) = (\alpha_{i+1,2}, \alpha_{i,2})$. Hence if there exists an i such that $\alpha_{i,1} \neq \alpha_{i,2}$, then there exists a j such that $\alpha'_{j,1} \neq \alpha'_{j,2}$, and therefore $f(\alpha_1) \neq f(\alpha_2)$.

5.2 Lattice Structure

We now construct a lattice χ and extensions of f and h such that assumptions (i)–(iii) of Theorem 1 are verified.

We choose as χ the set of vectors in $\mathbb{N}^N$ with coordinates $x_i \in [1, .., N]$, that is

$$\chi = \{x \in \mathbb{N}^N : x_i \in [1, ..., N]\} .$$

The partial order that we choose on such a set is given by

$$\forall x, w \in \chi,\ x \leq w \text{ if } x_i \leq w_i\ \forall i . \tag{7}$$

We define join and meet in the following way:

$$\forall\, x, w \ \in \chi,\ v = x \vee w \text{ if } v_i = \max\{x_i, w_i\}$$
$$\forall\, x, w \ \in \chi,\ v = x \wedge w \text{ if } v_i = \min\{x_i, w_i\}\ .$$

In this way we have for example $\bigvee \chi = [N, ..., N]$ and $\bigwedge \chi = [1, ..., 1]$, and the set χ with the order defined in (7) is clearly a lattice. The set $\mathcal{U}$ is the set of all permutations of N elements and it is a subset of χ. All the elements in $\mathcal{U}$ form an anti-chain of the lattice, that is any two elements in $\mathcal{U}$ in χ are not related by the order defined in (7). In the sequel we will denote by w the variables with type χ not specifying if the type is $\mathcal{U}$, and we will always denote by α the variables with type $\mathcal{U}$.

The function h can be naturally extended in the following way

$$z_i < x_{w_i} : z_i' = z_i + \delta\ ,\quad z_i > x_{w_i} : z_i' = z_i - \delta,\quad z_i = x_{w_i} : z_i' = z_i \tag{8}$$

for $w \in \chi$. Then the clauses (8) model the function $\tilde{h}$. In analogous way f is extended as

$$x_{w_i} \geq z_{i+1} \wedge x_{w_{i+1}} \leq z_{i+1}\ :\ (w_i', w_{i+1}') = (w_{i+1}, w_i)\ , \tag{9}$$

for $w \in \chi$. Then the clauses (9) model the function $\tilde{f}$. As a consequence we have two new functions $\tilde{h} : \mathbb{R}^N \times \chi \to \mathbb{R}^N$ and $\tilde{f} : \mathbb{R}^N \times \chi \to \chi$, such that $\tilde{f}|_{\mathbb{R}^N \times \mathcal{U}} = f$ and $\tilde{h}|_{\mathbb{R}^N \times \mathcal{U}} = h$.

5.3 Properties of the extended functions

We analyze in this section the properties of the extensions $\tilde{f}$ and $\tilde{h}$ proposed in the previous section. In particular we show that properties (ii) and (iii) of Theorem 1 hold.

Lemma 3. *Property (ii) of Theorem 1 holds with the lattice structure chosen in Section 5.2.*

Proof. We need to show that

$$A_y(k) = \{w \in \chi : y(k+1) = \tilde{h}(y(k), w) = [\bigwedge A_y(k), \bigvee A_y(k)],$$

where $y = z$. By (8) we have that $\{w \in \chi : z(k+1) = \tilde{h}(z(k), w)\} = \{w | x_{w_i} > z_i,\}$ if $z_i(k+1) = z_i(k) + \delta$, $\{w \in \chi : z(k+1) = \tilde{h}(z(k), w)\} = \{w | x_{w_i} < z_i,\}$ if $z_i(k+1) = z_i(k) - \delta$, and $\{w \in \chi : z(k+1) = \tilde{h}(z(k), w)\} = \{w | x_{w_i} = z_i,\}$ if $z_i(k+1) = z_i(k)$. By assuming $x_i \leq z_i \leq x_{i+1}$ for all time, we have $x_{w_i} > z_i$ if and only if $w_i > i$ and $x_{w_i} < z_i$ if and only if $w_i < i$. Therefore

$$\begin{aligned} A_y(k) = \{w \in \chi\ :\ & [(w_i > i)\ \wedge\ (z_i(k+1) = z_i(k) + \delta)] \\ & \vee [(w_i < i)\ \wedge\ (z_i(k+1) = z_i(k) - \delta)] \\ & \vee [(w_i = i)\ \wedge\ (z_i(k+1) = z_i(k))]\} \end{aligned}$$

Since also $w \in \chi$ we have that $1 \leq w_i \leq N$, and therefore there exist $l_y(k) \in \chi$ and $u_y(k) \in \chi$ such that $A_y(k) = \{w \in \chi\ :\ w \geq l_y(k)\ \wedge\ w \leq u_y(k)\}$, so that (ii) of Theorem 1 holds.

Lemma 4. *Property (iii) of Theorem 1 holds with the lattice structure chosen in Section 5.2.*

Proof. We need to show that $\tilde{f} : A_y(k) \to [\tilde{f}(l_y(k)), \tilde{f}(u_y(k))]$ is an order isomorphism we need to show: a) that it is onto; b) that it is order embedding.

a) To show that it is onto, we show directly that $f(A_y) = [\tilde{f}(l_y), \tilde{f}(u_y)]$. We omit the dependence on k to simplify notation. Our arguments relay on the coordinates structure of the sets A_y and $\tilde{f}(A_y)$. In particular from equations (9) we deduce that $A_y = (A_{y,1}, ..., A_{y,N})$, i.e. the set A_y is a vector of sets whose elements are in $\mathbb{N}$, and in particular $A_{y,i} \in \{[1,i], [i+1,N], [i,i]\}$. Denote by $\tilde{f}(A_y)_i$ the ith coordinate set of $\tilde{f}(A_y)$. By equations (9) we derive that $\tilde{f}(A_y)_i \in \{A_{y,i}, A_{y,i-1}, A_{y,i-1}\}$. We consider the case where $\tilde{f}(A_y)_i = A_{y,i-1}$, the other cases can be treated in analogous way. If $\tilde{f}(A_y)_i = A_{y,i-1}$ then $\tilde{f}(A_y)_{i-1} = A_{y,i}$. Then we have that

$$(i)\ \tilde{f}(A_y)_i = A_{y,i-1} \text{ and } \tilde{f}(A_y)_{i-1} = A_{y,i} \Longrightarrow$$

$$(ii)\ \forall x \in A_{y,i} \text{ we have } x \le i, \text{ and } \forall z \in A_{y,i-1} \text{ we have } z \ge i,$$

which implies (iii) $u_{y,i} \le i,\ l_{y,i} \le i$ and $u_{y,i-1} \ge i,\ l_{y,i-1} \ge i$. This last expression finally implies that

$$(iv)\ \tilde{f}(l_y)_i = l_{y,i-1},\ \tilde{f}(u_y)_i = u_{y,i-1}, \text{ and } \tilde{f}(l_y)_{i-1} = l_{y,i},\ \tilde{f}(u_y)_{i-1} = u_{y,i}\ .$$

Since for any i we have that $A_{y,i} = [\bigwedge A_{y,i}, \bigvee A_{y,i}] = [l_{y,i}, u_{y,i}]$, (i) and (iv) imply that $\tilde{f}(A_y)_i = [\tilde{f}(l_y)_i, \tilde{f}(u_y)_i]$. The same reasoning holds for any $\tilde{f}(A_y)_i \in \{A_{y,i}, A_{y,i-1}, A_{y,i-1}\}$, and for any i, therefore $f(A_y) = [\tilde{f}(l_y), \tilde{f}(u_y)]$.

b) To show that it is order embedding it is enough to note again that $\tilde{f}(A_y)$ is obtained by switching $A_{y,i}$ with $A_{y,i+1}$, $A_{y,i-1}$, or leaving it to $A_{y,i}$. Therefore if $w \le v$ for $w, v \in A_y$ then $f(w) \le f(v)$ since coordinate-wise we will compare the same numbers. By the same reasoning the reverse is also true, that is if $f(w) \le f(v)$ then $w \le v$.

The construction of system in equations (1-2) is straightforward, since we need to "copy" the dynamics reported in (9) and compute a join and a meet. Then write lower and upper bounds L and U coordinate-wise as $U = (U_1, ..., U_N)$ and $L = (L_1, ..., L_N)$ and initialize $L = \bigwedge \chi$ and $U = \bigvee \chi$, so that $L_i = 1$ and $U_i = N$. Then the guarded command program which implements the observer in (1-2) is given by

$$\begin{aligned}
& x_{L_i} \ge z_{i+1}\ \wedge\ x_{L_{i+1}} \le z_{i+1}\ : \\
& \{[(z'_i = z_i + \delta) \Rightarrow (l'_{y,i} = i+1)]\ \vee\ [(z'_i = z_i - \delta) \Rightarrow (l'_{y,i} = 1)]\} \\
& \wedge \{[(z'_{i+1} = z_{i+1} + \delta) \Rightarrow (l'_{y,i+1} = i+2)]\ \vee\ [(z'_{i+1} = z_{i+1} - \delta) \Rightarrow (l'_{y,i+1} = 1)]\} \\
& \wedge (L'_i, L'_{i+1}) = (\max\{L_{i+1}, l'_{y,i}\}, \max\{L_i, l'_{y,i+1}\})
\end{aligned} \tag{10}$$

$$\begin{aligned}
& x_{U_i} \ge z_{i+1}\ \wedge\ x_{U_{i+1}} \le z_{i+1}\ : \\
& \{[(z'_i = z_i + \delta) \Rightarrow (u'_{y,i} = N)]\ \vee\ [(z'_i = z_i - \delta) \Rightarrow (u'_{y,i} = i)]\} \\
& \wedge \{[(z'_{i+1} = z_{i+1} + \delta) \Rightarrow (u'_{y,i+1} = N)]\ \vee\ [(z'_{i+1} = z_{i+1} - \delta) \Rightarrow (u'_{y,i+1} = i+1)]\} \\
& \wedge (U'_i, U'_{i+1}) = (\min\{U_{i+1}, u'_{y,i}\}, \min\{U_i, u'_{y,i+1}\})
\end{aligned} \tag{11}$$

Since we have shown that (i)-(iv) of Theorem 1 are verified, then the sequences $L(k)$ and $U(k)$ have the properties (a)–(c) given by Theorem 1. Note that properties (a')–(c') do not hold since we can prove that the extended system $\tilde{\Sigma} = (\tilde{f}, \tilde{h})$, with measurable variables V_C is not observable.

5.4 Complexity Considerations

The amount of computation required for updating L and U according to (10) and (11) is proportional to the amount of computation required for updating the variables α in system Σ. In fact we have $2N$ clauses, $2N$ variables, and $2N$ computations of "max" and "min" between values in $\mathbb{N}$. Therefore we can roughly say that the complexity of the algorithm that generates the sequences $L(k)$ and $U(k)$ is about twice the complexity of the algorithm that generates the α trajectories. Also note that the clauses in (10) and (11) are obtained by "copying" the clauses in (9) and correcting them by means of the output information, according to how the observer is constructed for dynamical systems (see [6] for details).

By Theorem 1 we have that the function of k $|[L(k), U(k)] \cap \mathcal{U} - \alpha(k)|$ tends to zero and it is non increasing. This function is useful for analysis purposes, but it is not necessary to compute it at any point in the algorithm proposed in equation (10) and (11). However, since the sequence $L(k)$ is not converging to the sequence $U(k)$, once the algorithm has converged, i.e. $|[L(k), U(k)] \cap \mathcal{U}| = 1$, we cannot recover α from the values of $U(k)$ and $L(k)$ directly. Instead of computing directly $[L(k), U(k)] \cap \mathcal{U}$, we carry out a simple algorithm, that in the case of the RoboFlag Drill example takes at most $(N^2 + N)/2$ steps and takes as inputs $L(k)$ and $U(k)$ and gives as output $\alpha(k)$ if the algorithm has converged. This is formally explained in the following paragraph.

Refinement Algorithm. Let $c_i = [L_i, U_i]$. Then the algorithm

$$(m_1, ..., m_N) = \mathit{Refine}(c_1, ..., c_N),$$

which takes assignment sets $c_1, ..., c_N$ and produces assignment sets $m_1, ..., m_N$, is such that If $m_i = \{k\}$ then $k \notin m_j$ for any $j \neq i$.

For such an algorithm we have the properties shown in the following lemmas.

Lemma 5. *When the set $[L(k), U(k)] \cap \mathcal{U}$ has converged to $\alpha(k)$, the refinement algorithm is such that $(m_1(k), ..., m_N(k)) = \alpha(k)$.*

Proof. We notice that when $[L(k), U(k)] \cap \mathcal{U}$ has converged to α, we have that $[L(k), U(k)] \cap \mathcal{U}$ is of the form $\{\alpha(k), \text{elements not in } \mathcal{U}\}$. Denote with c_i the sets $[L_i, U_i]$ before the refinement has occurred, and denote with m_i the refined version of c_i's. Then we show that among the sets $[L_i(k), U_i(k)]$ there is at least one i for which $L_i(k) = U_i(k)$, and therefore we have at least one singleton to take out from the other coordinates. Then the proof proceeds by iteration on N.

To indicate that $\mathcal{U}$ is the set of permutations of N elements, we will write $\mathcal{U}_N$. To show that when $[L(k), U(k)] \cap \mathcal{U}_N$ has converged to $\alpha(k)$ at least for one i $L_i(k) = U_i(k)$ (c_i is a singleton), it is sufficient to notice that if this were not the case we would have more than one possible $\alpha \in \mathcal{U}_N$ in $[L(k), U(k)]$. Without loss in generality assume that such i is equal to N. Then take out that singleton from all the other sets c_j for $j < N$ to obtain new sets m_j whose elements take values in a set of possible $N-1$ natural numbers. Still there is only one $\beta \in \mathcal{U}_{N-1}$ such that $\beta \in (m_1, ..., m_{N-1})$. Then we can apply again the reasoning that for this to be true there must exist at least one singleton among the sets m_j, for $j \in [1, N-1]$. Proceeding iteratively, we get the result.

We can also show that the sum of the cardinalities of the m_i sets is not increasing along the time step k. This is formally shown in the following lemma:

Lemma 6. *Let $c_i(k) = [L_i(k), U_i(k)]$, and denote by $m_i(k)$ the sets obtained with the refinement algorithm. Then*

$$\sum_{i=1}^{N} |m_i(k+1)| \leq \sum_{i=1}^{N} |m_i(k)|$$

Proof. Let us denote with primed variables the variables at step $k+1$ and with unprimed variables the variables at step k. The proof proceeds by showing that for each j there exist a k such that $m'_j \subseteq m_k$. By equations (10) and (11) we deduce that we can have one of the following cases for each i: (a) $c'_i \subseteq c_{i+1} \ \wedge \ c'_{i+1} \subseteq c_i$, (b) $c'_i \subseteq c_i$, (c) $c'_i \subseteq c_{i-1} \ \wedge \ c'_{i-1} \subseteq c_i$. Let us consider case (a), the other cases can be treated in analogous way. Let c_j be a singleton. In the refinement process it is deleted from any other set, so that we have $c_{i+1} = m_{i+1} - c_j$ and $c_i = m_i - c_j$ for all i. Assume that in the first refinement iteration no new singletons are created. We have one of the following situations: $c'_j \subseteq c_{j+1} \ \wedge \ c_{j+1} \subseteq c_j$, $c'_j \subseteq c_j$, $c'_j \subseteq c_{j-1} \ \wedge \ c'_{j-1} \subseteq c_j$. This implies that one of the c'_k is equal to the singleton c_j. The sets m'_i are created removing such singleton for all the other sets, so that we obtain $m'_i + c_j = c'_i \subseteq c_{i+1} = m_{i+1} + c_j$ and $m'_{i+1} + c_j = c'_{i+1} \subseteq c_i = m_i + c_j$. This in turn implies that $m'_i \subseteq m_{i+1}$ and $m'_{i+1} \subseteq m_i$. This holds for all of the cases (a),(b), (c), and for each i. Thus $\sum_{i=1}^{N} |m'_i| \leq \sum_{i=1}^{N} |m_i|$.

The same kind of reasoning can be applied if the first refinement iteration of the c_i creates new singletons.

It is easy to show that the refinement algorithm can be executed in at most $(N^2 + N)/2$ steps.

6 Simulation Results

The RoboFlag Drill system represented in equations (5) and (6) has been implemented in MATLAB together with the observer reported in equations (10) and (11). Figure 3 (left) shows the behavior of the quantity

$$V(k) = \log |[L(k), U(k)] \cap \mathcal{U}|$$

and

$$E(k) = \frac{1}{N} \sum_{i=1}^{N} |\alpha_i(k) - i|.$$

$V(k)$ represents the log of the cardinality of the set of all possible assignments at each step. This quantity gives an idea of the convergence rate of the observer. $E(k)$ is a function of α, and it is not increasing along the executions of the system $\Sigma_{Assign} \cup \Sigma_{Red}$. This quantity is showing the rate of convergence of the α assignment to its equilibrium $[1, ..., N]$. In Figure 3 (right) we show the results

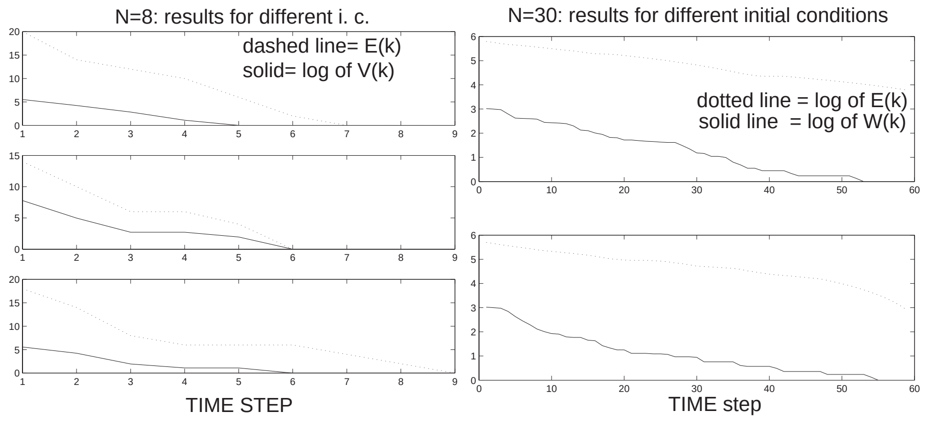

Fig. 3. (Left) Example with N=8: note that the function $V(k)$ is always non-increasing and it is converging to zero. (Right) Example with N=30: note that the function $W(k)$ is always non-increasing and its logarithm is converging to zero.

for $N = 30$ robots per team. In particular we report the log of $E(k)$ and the log of $W(k)$ defined as

$$W(k) = \frac{1}{N} \sum_{i=1}^{N} |m_i(k)|,$$

which by virtue of Lemmas 5 and 6 is non increasing and converging to one, that is the sets $(m_1(k), ..., m_N(k))$ converge to $\alpha(k) = (\alpha_1(k), ..., \alpha_N(k))$. In the same figure we notice that when $W(k)$ converges to one, $E(k)$ has not converged to zero yet. This suggests that the observer is much faster than the dynamics of the system under study. We cannot explain such a good performance formally yet, and the observer speed issue will be addressed in future work.

7 Conclusions

We have proposed a way of constructing an observer for a class of observable hybrid systems where the continuous variables are measured. The observer is constructed on a lattice structure and it updates the least element and the greatest element of the set of all discrete variables values compatible with the observed output sequence. These ideas are applied to a multi-robot example: The RoboFlag Drill. This approach is promising for reducing the computational effort of the observer since it updates a "cheap" representation of a set rather than the set itself.

More work is needed to establish how general the conditions listed in Theorem 1 are, and what is the compromise between generality and complexity. Also more investigation is needed to understand when the extended system is still observable. Computing the intersection $[L(k), U(k)] \cap \mathcal{U}$ is needed once the observer has converged for recovering the α value. The complexity of this computation is smaller than $(N^2 + N)/2$ for the RoboFlag Drill system with N robots, but the general case needs more investigation. A question to be still addressed is concerned with the speed of convergence; the simulation results are encouraging in this regard and a formal analysis needs to be developed. Finally how are the properties listed in Theorem 1 related with observability? All these issues are left to future work.

8 Acknowledgements

This work was supported in part by the ONR grant N00014-10-1-0890 under the MURI program, and the NSF Center for Neuromorphic Systems Engineering. We would like to thank Professor Eric Klavins for his contribution to our knowledge on computer science related subjects such as lattice theory, transition systems, and guarded command programs, and for his insightful comments.

References

1. A. Bemporad, G. Ferrari-Trecate, and M. Morari. Observability and controllability of piecewise affine and hybrid systems. *IEEE Transactions on Automatic Control*, 45:1864–1876, 1999.
2. P. E. Caines. Classical and logic-based dynamic observers for finite automata. *IMA J. of Mathematical Control and Information*, pages 45–80, 1991.
3. B. A. Davey and H. A. Priesteley. *Introduction to Lattices and Order.* Cambridge University Press, 2002.
4. E. W. Dijkstra. Guarded commands, non-determinacy and a calculus for the derivation of programs. In *Proceedings of the international conference on Reliable software*, pages 2 – 2.13, Los Angeles, California, 1975. http://portal.acm.org.
5. E. Klavins. A formal model of a multi-robot control and communication task. In *Conference on Decision and Control*, Hawaii, 2003.
6. David G. Luenberger. An introduction to observers. *IEEE Transactions on Automatic Control*, AC-16:6:596–602, 1971.
7. Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concur- rent Systems: Specication.* Springer-Verlag, 1992.
8. D. Del Vecchio and E. Klavins. Observation of guarded command programs. In *Conference on Decision and Control*, Hawaii, 2003.