Observation of Hybrid Guarded Command Programs

Domitilla Del Vecchio[1] Eric Klavins[2]

Division of Engineering and Applied Science
California Institute of Technology
{ddomitilla,klavins}@cds.caltech.edu

Abstract

We consider the problem of estimating the internal state (hidden variables) of a class of hybrid guarded-command programs. Such programs model dynamical systems that have both continuous and discrete states. For these systems we supply a definition of *weak observability* and for the case where a given system is weakly observable we construct an observer that takes advantage of the special guarded-command structure of the program. We then focus on a particular example, the "RoboFlag Drill" wherein two teams of robots compete in a simplified capture-the-flag-like game. For this system, the state is large enough and complex enough that the simple observer is not practical. Thus we propose an efficient observer that takes advantage of the particular structure of the RoboFlag Drill.

1 Introduction

In this paper we examine the problem of observing the values of hidden variables in a class of *guarded command programs*. Such programs, which consist of a set of guard-rule pairs, are typically used in program verification to formally model algorithms. In this paper we extend this use to modeling a kind of hybrid system wherein there is an interplay between discrete and continuous variables. In particular, we are concerned with decentralized multi-robot systems, such as are found in robot soccer, where continuous variables represent physical quantities such as position and velocity, and discrete variables represent the state of the internal logical system or communications protocol used by the robots to coordinate their actions. The observation problem of interest is then to observe the internal discrete part of the system given the evolution of the continuous part.

The main contributions of this paper are to define, in Section 2, the observation problem for transition systems as represented by guarded command programs and to construct observers for such systems. After defining the requirements of an observer for a guarded command program in general, we examine in Section 3 a particular class of guarded command programs. For this class we propose a candidate observer and show that if the system is observable then the candidate indeed satisfies the requirements of an observer.

In Section 5 we introduce a multi-robot task similar to the game "capture the flag" and specify it in the form of the particular class of guarded command programs defined in Section 3. We show that, given the evolution of the positions of the robots, the discrete state of the program, representing an assignment of defending robots to attacking robots, is observable. We then note in Section 6.3 that, due to the large state space of the system, the observer defined in Section 3 is not practicable. Therefore we define a more efficient, but possibly slower, observer. We end the paper with simulations of the capture the flag system that demonstrate the convergence of the two observers on various examples.

Related Work: Hybrid systems characterized by the interaction of continuous variables, governed by differential or difference equations, and by discrete variables, described by finite state machines, if-then-else rules or propositional and temporal logic have been examined by many researchers [3, 8, 4]. Guarded commands programs are introduced in [7] and are used to model capture the flag like systems in [9]. We have adopted this way of modeling hybrid systems because it is particularly suitable for describing a distributed system that may be parameterized by the number of agents in it. The guarded command formalism allows us to implicitly represent large state spaces that would have to be explicitly represented in other formalisms. Observability of hybrid systems has been examined in [2] for the MDL modeling framework, in [11] with piecewise discrete time linear systems, and in [1] where piecewise-linear continuous-time systems are studied. In the discrete event literature the observability problem for finite automata is examined in, for example, [5] where an observer similar to the one described in this paper is proposed. The systems we explore in this paper have continuous variables, however, and it is not obvious that such observers can be used in our case. Observability of programs is also related to information flow security [10] where the problem of ensuring that hidden variables *can not* be observed from observable variables.

[1]ONR grant N00014-10-1-0890 under the MURI program

[2]Supported in part by the DARPA SEC program under grant number F33615-98-C-3613 and by AFOSR grant number F49620-01-1-0361.

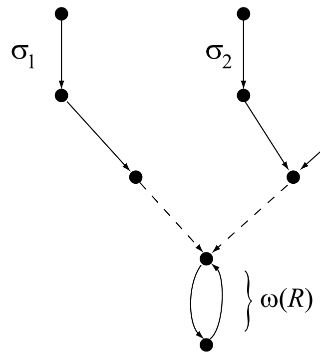


Fig. 1: Trajectories $\sigma_2(t)$ and $\sigma_3(t)$ are weakly equivalent trajectories according to Definition 2.4 while $\sigma_3(t)$ is not weakly equivalent to either $\sigma_1(t)$ or $\sigma_2(t)$.

2 Definitions

2.1 State Transition Systems

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 $U = \bigcup_{v \in V} type(v)$. The set of all states is denoted S. For a subset W of V, we denote by $S|_W$ the set of states

$$S|_W = \{s|_W \ : \ s \in S\}.$$

A **transition relation** on S is a relation $R \subseteq S \times S$. The set S is called the **domain** of R and is denoted $dom(R)$. 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

$$x' < y \vee y' = z \tag{1}$$

then $sRs' \Leftrightarrow s'(x) < s(y) \vee s'(y) = s(z)$.

Given a transition relation R, an **execution** of R is a sequence $\sigma = \{s_t\}_{t \in \mathbb{N}}$ such that $s_t R s_{t+1}$ for all $t \in \mathbb{N}$. 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 $v(t)$ the value $\sigma(t)(v)$. The **trajectory** of $v \in V$ with respect to σ is the sequence $\{\sigma(t)(v)\}_{t \in \mathbb{N}}$.

We define transition relations over subsets W of V, as in $R \subseteq S|_W \times S|_W$, to enforce the notion that R does not have information about variables in $V - W$. We may then extend R to all of V as follows.

Definition 2.1 *The* extension *of R to S is defined by*

$$ex_S(R) = \{(s, s') \in S \times S \ : \ s|_{dom(R)} \ R \ s'|_{dom(R)}\}.$$

2.2 Observability

We now define two notions of observability for transition systems (both in Definition 2.5). The first is the standard notion: The system is observable if any two execution sequences can be distinguished by their outputs. The second is a weaker definition motivated by the fact that in the systems in which we are interested (such as that described in Section 5), uniqueness is not guaranteed. That is two different states may transition to the same state. Thus, we use the notion of "weakly observable": The system is observable as long as any two executions that do not collapse onto the same state before stabilizing can be distinguished from their outputs. The following definitions state these ideas formally.

Definition 2.2 *Given a transition relation R on S and an output function $h : S \to U$, two executions $\sigma_1, \sigma_2 \in \mathcal{E}(R)$ are* distinguishable *if there exists a time t such that $h \circ \sigma_1(t) \neq h \circ \sigma_2(t)$.*

Definition 2.3 *Let R be a transition relation on S, the set $A \subset S$ is the* ω-limit set *of R, denoted by $\omega(R)$, if the following hold:*

(i) if $s \in A$ and $s \ R \ s'$, then $s' \in A$;

(ii) for each $\sigma \in \mathcal{E}(R)$, there exists a time t_σ such that $\sigma(t_\sigma) \in A$ for all $t \geq t_\sigma$.

Definition 2.4 *Given a transition relation R, two executions $\sigma_1, \sigma_2 \in \mathcal{E}(R)$ are* weakly equivalent*, denoted $\sigma_1 \sim \sigma_2$, if there exists a time t^* such that $\sigma_1(t^*) \notin \omega(R)$ and $\sigma_1(t) = \sigma_2(t)$ for all $t \geq t^*$.*

Examples of weakly equivalent and inequivalent trajectories are illustrated in Figure 1.

Definition 2.5 *(Observability and Weak Observability) The transition relation R is said to be* observable *with respect to the output function $h : S \to U$ if any two executions $\sigma_1, \sigma_2 \in \mathcal{E}(R)$ are distinguishable. The system is* weakly observable *if whenever $\sigma_1 \nsim \sigma_2$ then σ_1 and σ_2 are distinguishable.*

In this paper we suppose that the programs we wish to observe are defined on $V = \mathcal{H} \cup \mathcal{O}$, where $\mathcal{H} \cap \mathcal{O} = \emptyset$, and $\mathcal{H}$ and $\mathcal{O}$ consist of *hidden* and *observable* variables respectively.[1] In this case h is essentially a projection. In the sequel we construct an observer $\hat{R}$ for the couple (R, h) that is defined over a variable set W such that $\mathcal{O} \subseteq W$ and $\mathcal{H} \cap W = \emptyset$. We will denote by α the vector of all variables in $\mathcal{H}$ and by $\hat{\alpha}$ a variable symbol in $W - \mathcal{O}$ that $\hat{R}$ uses to estimate the value of α.

Problem 2.1 *(Observer) Let $V = \mathcal{H} \cup \mathcal{O}$ and W be such that $\mathcal{O} \subseteq W$ and $\mathcal{H} \cap W = \emptyset$. Suppose that α is the vector*

[1] If the system under observation has output function h that is not simply a projection of the state, we could define an auxiliary variable y, set $\mathcal{O} = \{y\}$ and require that $y = h \circ \sigma(t)$ for all t and all $\sigma \in \mathcal{E}(R)$.

of all variables in $\mathcal{H}$ and suppose that $\hat{\alpha} \in W - \mathcal{O}$. Given a transition relation $R : S|_V \times S|_V$, the transition relation $\hat{R} : S|_W \times S|_W$ is an observer *for R if the following hold for all $\sigma \in \mathcal{E}(ex(R) \cap ex(\hat{R}))$:*

(i) there exists a time t^* such that $\hat{\alpha}(t) = \alpha(t)$ for all $t \geq t^*$;

(ii) there exists a metric d on $type(\hat{\alpha})$ such that for each ε there exists a δ such that for all t

$$d(\hat{\alpha}(0), \alpha(0)) < \delta \;\Rightarrow\; d(\hat{\alpha}(t), \alpha(t)) < \varepsilon.$$

2.3 Guarded Command Programs

One way to specify transition relations is with *guarded command programs*, which we now define. A **guard** is a predicate on states and a **rule** (or *command*) is a relation on states. A **guarded command** is then a pair $g : r$ where g is a guard and r is a rule. As in (1), we denote guarded commands using primed and unprimed variable symbols. For example,

$$x > 0 \;:\; x' = x + y \tag{2}$$

denotes the guarded command relating two states s_1 and s_2 by

$$s_1(x) > 0 \;:\; s_2(x) = s_1(x) + s_1(y).$$

If s is a state and $g : r$ is a guarded command, we say that $g : r$ is **applicable** in state s if $g(s)$ is true.

Definition 2.6 *A* guarded command program *consists of a set Σ of guarded commands.*

A guarded command program defines a transition relation (giving the operational semantics of the program) wherein all commands are *executed* in parallel to give a new state.

Definition 2.7 *Given a guarded command program Σ over variables V, the* transition relation corresponding to Σ *is the relation $R_\Sigma \subseteq S|_V \times S|_V$ where $s \; R_\Sigma \; s'$ if and only if*

$$\forall g : r \in \Sigma \;.\; g(s) \Rightarrow s \; r \; s'.$$

Furthermore, if a variable $v \in V$ does not occur primed in any command applicable in s, then $s(v) = s'(v)$.

Note that the composition $\Sigma_1 \cup \Sigma_2$ of two guarded command programs Σ_1 and Σ_2 has defining relation $R_{\Sigma_1} \cap R_{\Sigma_2}$. The observer problem for guarded command programs is: Given Σ construct $\hat{\Sigma}$ so that $R_{\hat{\Sigma}}$ is an observer for R_Σ.

3 Problem Statement

We now restrict our attention to a certain class of guarded commands programs. Let $\mathcal{O} = \{z_1, ..., z_{N_\mathcal{O}}\}$ and $\mathcal{H} = \{\alpha_1, ..., \alpha_{N_\mathcal{H}}\}$ and put $V = \mathcal{O} \cup \mathcal{H}$. We suppose that each z_i has a associated with it K_i commands of the form

$$P_{i,j}(z, \alpha) : z_i' = f_{i,j}(z), \quad j \in \{1, ..., K_i\} \tag{3}$$

and each α_k has associated with it M_k commands of the form:

$$Q_{k,l}(z, \alpha) : \alpha_k' = g_{k,l}(\alpha) \quad l \in \{1, ..., M_k\}, \tag{4}$$

where $f_{i,j}(\cdot)$ and $g_{k,l}(\cdot)$ are functions. We use Σ to denote the set of all the commands for the hidden and observable variables described in (3) and (4). We suppose that $type(z_i) = \mathbb{R}$ and denote the vector $(z_1, ..., z_{N_\mathcal{O}})$ by z and the vector $(\alpha_1, ..., \alpha_{N_\mathcal{H}})$ by α. We leave $U \triangleq type(\alpha)$ unspecified for now and suppose that it represents the discrete part of the system.[2] Thus, Σ defines a relation $R|_V$ with domain $(V \to \mathbb{R}^{N_\mathcal{O}} \times U)$. We require that the following for any state s:

(A1) For each i there is exactly one $j \in \{1, ..., K_i\}$ such that $P_{i,j}(z, \alpha)$ is true, and for each k there is exactly one $l \in \{1, ..., M_k\}$ such that $Q_{k,l}(z, \alpha)$ is true.

Assumption (A1) implies that there cannot be two different update rules for z (or α) acting simultaneously. (A1) also implies that at any time there is at least one update rule holding at that time.

The other assumption we have made (implied by structure (3) and (4)) is that Σ is deterministic (i.e. that R_Σ is a function). We intend to relax this somewhat strong assumption in future work.

4 Observer Construction

We now turn our attention to the question of when an observer exists for Σ. We first propose a candidate observer $\hat{\Sigma}$ for (3)-(4). We then show property 2.1(ii), by choosing a particular d defined on U. Further if Σ is weakly observable we can also show property 2.1(i) – that is, that $\hat{\Sigma}$ is an observer for Σ.

We use the variable symbol $\hat{\alpha}$ to represent an estimate of α, with $type(\hat{\alpha}) = 2^U$. The intention is that $\hat{\alpha}$ will denote the *set* of all possible values of α at any given time in an execution of $\Sigma \cup \hat{\Sigma}$. Initially $\hat{\alpha} = U$. We

[2] For simplicity we assume that $type(\alpha_i) = type(\alpha_j)$ although this certainly does not need to be the case.

define $\hat{\Sigma}$ to be the program containing the single clause:

$$true : B' = \bigcap_{i}^{N_\mathcal{O}} \bigcup_{j}^{K_i} \{\alpha \,:\, z_i' = f_{i,j}(z) \wedge P_{i,j}(z,\alpha)\} \cap \hat{\alpha}$$

$$\wedge\ \hat{\alpha}' = \bigcup_{\alpha \in B'} \left\{ \beta : \forall k. \beta_k \in \bigcup_{l=1}^{M_k} g_{k,l}(\{\alpha\} \cap \{\gamma : Q_{k,l}(z,\gamma)\}) \right\} \tag{5}$$

where B is an auxiliary variable used for clarity. The assignment to B' collects the set of all values of α that agree with the observation $z_i' = f_{i,j}(z,\alpha)$ and that are currently candidates (they are also in $\hat{\alpha}$). The assignment to $\hat{\alpha}'$ maps this forward using the functions $g_{k,l}$ on each component. An example illustrates the process.

Example 4.1 *Let $N_\mathcal{O} = N_\mathcal{H} = 1$, type$(\alpha) = \{-2,-1,1,2\}$, and $z \in \mathbb{R}$. Instantiate (3)-(4) by:*

$$\begin{aligned} z < \alpha &\ :\ z' = z + 0.1 \\ z > \alpha &\ :\ z' = z - 0.1 \\ z = \alpha &\ :\ z' = z \\ |\alpha - z| \le 0.5 &\ :\ \alpha' = -\alpha \\ |\alpha - z| > 0.5 &\ :\ \alpha' = \alpha \end{aligned}$$

where z observable variable and α needs to be estimated. Here $K_1 = 3$ and $M_1 = 2$. Suppose that initially $z = 0$, and $\alpha = 2$. The first eight steps of the resulting execution of $\Sigma \cup \hat{\Sigma}$ are shown in the following table:

z	α	$\hat{\alpha}$
0.0	2	$\{-2,-1,1,2\}$
0.1	2	$\{1,2\}$
0.2	2	$\{1,2\}$
0.3	2	$\{1,2\}$
0.4	2	$\{1,2\}$
0.5	2	$\{1,2\}$
0.6	2	$\{-1,2\}$
0.7	2	$\{2\}$
0.8	2	$\{2\}$

From the first step ($z := 0.1 \to z := 0.2$) the observer determines that α must be positive because the first z clause was used. The estimate then remains the same until z changes from 0.6 to 0.7.

We now show that $\hat{\Sigma}$ is indeed an observer for Σ.

Theorem 4.1 *Given Σ defined in (3)-(4), the program $\hat{\Sigma}$ defined in equation (5) satisfies the following properties:*

(1) For all t, $\alpha(t) \in \hat{\alpha}(t)$ (correctness);

(2) If Σ is weakly observable, then 2.1(i) holds for $\hat{\Sigma}$ (convergence);

(3) Property 2.1(ii) also holds for $\hat{\Sigma}$ (small error).

Therefore, $\hat{\Sigma}$ is an observer for Σ.

Proof:

(1) Fix a particular execution. We prove (1) by induction on t. By assumption, $\alpha(0) \in \hat{\alpha}(0) = U$. For the inductive step, suppose that $\alpha(t-1) \in \hat{\alpha}(t-1)$. It suffices to determine how the set-valued map in (5) taking $\hat{\alpha}$ to $\hat{\alpha}'$ operates on the singleton $\{\alpha(t-1)\}$. First note that if $\hat{\alpha} = \{\alpha(t-1)\}$ then $B' = \{\alpha(t-1)\}$ as well. In this case, for each k there is at least one l for which the argument for $g_{k,l}$ equal to $\{\alpha(t-1)\}$ and for which $g_{k,l}(\alpha(t-1)) = \alpha_k(t)$.

(2) By (5) for any given $\beta' \in \hat{\alpha}'$ there exists an $\beta \in B'$ such that

$$\beta_k' \in \bigcup_{l=1}^{M_k} g_{k,l}(\{\beta\} \cap \{\gamma : Q_{k,l}(z,\gamma)\})$$

for each k. Also $\beta \in B'$ implies that $\beta \in \hat{\alpha}$ and that for every i there is a j such that $z_i' = f_{i,j}(z) \wedge P_{i,j}(z,\beta)$. This in turn implies that the sequence $\{(z(t),\beta(t))\}_{t\in\mathbb{N}}$ corresponds to an execution σ of Σ with $\sigma(t)(\beta) = \beta(t)$ and $\sigma(t)(z) = z(t)$ for all t. Also, $\beta(t) \in \hat{\alpha}(t)$ for all t. Therefore, for any $\beta', \gamma' \in \hat{\alpha}'$ there exist sequences $\{(z(t),\beta(t))\}_{t\in\mathbb{N}}$ and $\{(z(t),\gamma(t))\}_{t\in\mathbb{N}}$ corresponding to executions of σ_1 and σ_2 of Σ, where $\sigma_1(t)(\beta) = \beta(t)$, $\sigma_2(t)(\gamma) = \gamma(t)$, $\sigma_1(t)(z) = \sigma_2(t)(z) = z(t)$ for all t. Since $h \circ \sigma_1(t) = h \circ \sigma_2(t) = z(t)$, $\sigma_1 \sim \sigma_2$ and so there exists a time t such that $\sigma^1(t) = \sigma^2(t)$ implying that $\beta(t) = \gamma(t)$. Thus, the two sequences $\{\beta(t)\}_{t\in\mathbb{N}}$ and $\{\gamma(t)\}_{t\in\mathbb{N}}$ collapse onto the same value. This will occur for the sequences corresponding to any two elements in $\hat{\alpha}$, thus we conclude that $\hat{\alpha}$ converges to a singleton.

(3) Define $d : 2^U \times 2^U \to \mathbb{R}$ by

$$d(A,B) \triangleq |A - B| + |B - A|. \tag{6}$$

It is straightforward to show that d is a distance function. Note that by (1), $\alpha(t) \in \hat{\alpha}(t)$ and thus

$$d(\hat{\alpha}(t), \alpha(t)) = |\hat{\alpha}(t) - \alpha(t)| = |\hat{\alpha}(t)| - 1.$$

Thus, to show 2.1(ii), we need only demonstrate that $|\hat{\alpha}|$ is non-increasing. Since B' is of the form $X \cap \hat{\alpha}$, clearly $|B'| \le |\hat{\alpha}|$. Now, by (A1), we have that for each α and each k there is exactly one l such that $Q_{k,l}(z(t),\alpha)$ is true. This implies that in (5) $|\hat{\alpha}'| \le |B'|$. ∎

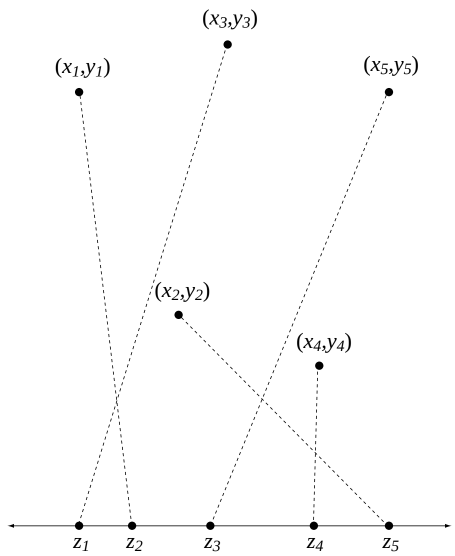


Fig. 2: An example state of the RoboFloag Drill for 5 robots. Here $\alpha = \{3, 1, 5, 4, 2\}$.

5 An Example: The RoboFlag Drill

In this section we consider a game called *RoboFlag* that is similar to "capture the flag", only for robots [6]. Two teams of robots, say *red* and *blue*, each have a defensive zone that they must protect (it contains the team's flag). If a red robot enters the blue team's defensive zone without being tagged by a blue robot, it captures the blue flag and earns a point. If a red robot is tagged by a blue robot in the vicinity of the blue defensive zone, it is disabled. 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 $\{(x, y) \mid y \leq 0\}$ 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. They start with a random (bijective) assignment $\alpha : \{1, ..., N\} \rightarrow \{1, ..., N\}$. 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 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. However, we do not consider the problem of how they would change their strategy in this paper.

The system can be described with guarded commands as follows (the description here is similar to that in [9]). The red robot dynamics Σ_{Red} are described by the N clauses

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

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

$$\begin{aligned} z_i < x_{\alpha_i} \quad &: \quad z_i' = z_i + \delta \\ z_i > x_{\alpha_i} \quad &: \quad z_i' = z_i - \delta \\ z_i = x_{\alpha_i} \quad &: \quad z_i' = z_i \end{aligned}$$

for $i \in \{1, ..., N\}$. To define the assignment protocol, it is useful to define, for $i \in \{2, ..., N-1\}$, the predicates

$$\begin{aligned} down_i \quad &\triangleq \quad up_{i-1} \\ up_i \quad &\triangleq \quad \neg down_i \wedge x_{\alpha_i} > x_{\alpha_{i+1}} \end{aligned}$$

and for robots 1 and N the predicates

$$\begin{aligned} down_1 \quad &\triangleq \quad false & \qquad down_N \quad &\triangleq \quad up_{N-1} \\ up_1 \quad &\triangleq \quad x_{\alpha_1} > x_{\alpha_2} & \qquad up_N \quad &\triangleq \quad false\ . \end{aligned}$$

The assignment protocol dynamics Σ_{Assign} are then given by the $3N$ clauses

$$\begin{aligned} down_i \quad &: \quad \alpha_i' = \alpha_{i-1} \\ up_i \quad &: \quad \alpha_i' = \alpha_{i+1} \\ \neg(down_i \vee up_i) \quad &: \quad \alpha_i' = \alpha_i . \end{aligned}$$

for $i \in \{1, ..., N\}$. Note that we have defined Σ_{Assign} so that α remains a permutation of $\{1, ..., N\}$ at every step. The complete RoboFlag specification is then given by

$$\Sigma_{RF} \triangleq \Sigma_{Red} \cup \Sigma_{Blue} \cup \Sigma_{Assign}.$$

For the blue robots we assume that initially $z_i \in [z_{min}, z_{max}]$ and $z_i < z_{i+1}$. For the red robots, we assume it is always the case that $x_i \in (z_{i-1}, z_i)$ and $y_i > 0$. We will denote with $x = (x_1, ..., x_N)$, $y = (y_1, ..., y_N)$, $z = (z_1, ..., z_N)$, $\alpha = (\alpha_1, ..., \alpha_N)$.

It should be apparent that Σ_{RF} has the form described in Equations (3) and (4). Furthermore, it is straightforward to show that assumption (A1) holds. In the sequel we will be concerned only with the system $\Sigma_{Blue} \cup \Sigma_{Assign}$. This is because the evolution of $\Sigma_{Blue} \cup \Sigma_{Assign}$ depends only on the initial values of x and y and not on the evolution of Σ_{Red}. Therefore, we may treat x and y as parameters of $\Sigma_{Blue} \cup \Sigma_{Assign}$ and put $\mathcal{O} = \{z_1, ..., z_N\}$ and $\mathcal{H} = \{\alpha_1, ..., \alpha_N\}$ corresponding to the problem definition 2.1.

It can be shown that α stabilizes to the assignment $\alpha^* = (1, ..., N)$ by showing that (1) the number of "conflicts" (pairs (i, j) such that $i < j$ but $x_{\alpha_i} > x_{\alpha_j}$) decreases at each step that changes an assignment and (2) once $down_i$ and up_i are both false for all i, they remain false forever after. Once α stabilizes, the values of z_i converge to the interval $(x_{\alpha_i} - \delta, x_{\alpha_i} + \delta)$. For a given execution $\sigma \in \mathcal{E}(\Sigma_{Blue} \cup \Sigma_{Assign})$ we denote the time that α stabilizes by t^α_σ and the time that the whole system stabilizes by t_σ. Note that $t^\alpha_\sigma \leq t_\sigma$. The observation problem of interest is then

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.

6 Observability of the RoboFlag Drill

To solve the RoboFlag Drill observation problem, we first determine whether $\Sigma_{Blue} \cup \Sigma_{Assign}$ is weakly observable. In particular, we want to know if inequivalent executions of $\Sigma_{Blue} \cup \Sigma_{Assign}$ lead to different sequences for z.

6.1 Observability

Lemma 6.1 *The program $\Sigma_{Blue} \cup \Sigma_{Assign}$ is weakly observable.*

Proof: (*Sketch*) Suppose $x_i \in (z_{i-1}, z_i)$ is invariant (i.e. δ is small). For given initial values of x and y consider any two executions $\sigma_1 \not\approx \sigma_2$ of $\Sigma_{Blue} \cup \Sigma_{Assign}$ (that might arise from different initial values of α and z). Put $t^\alpha = max\{t^\alpha_{\sigma_1}, t^\alpha_{\sigma_2}\}$ and $t^* = max\{t_{\sigma_1}, t_{\sigma_2}\}$. There are two cases:

1. $t^\alpha < t^*$: Then $\sigma_1(t^\alpha)(\alpha) = \sigma_2(t^\alpha)(\alpha)$. Since by assumption $\sigma_1 \not\approx \sigma_2$, it must be that $\sigma_1(t^\alpha)(z) \neq \sigma_2(t^\alpha)(z)$.
2. $t^\alpha = t^*$: Then $\sigma_1(t^\alpha - 1)(\alpha) \neq \sigma_2(t^\alpha - 1)(\alpha)$. If $\sigma_1(t^\alpha)(z) \neq \sigma_2(t^\alpha)(z)$ then we have the desired result. Thus, suppose $\sigma_1(t^\alpha)(z) = \sigma_2(t^\alpha)(z) \triangleq z^*$. Then it can be shown that for some i,

   $$\sigma(t^\alpha - 1)(z_i) = z^*_i - \delta \text{ but}$$

   $$\sigma(t^\alpha - 1)(z_i) = z^*_i + \delta.$$

   This is because at time $t^\alpha - 1$ the values of α under the two executions differ.

■

6.2 RoboFlag Observer

We now examine the observer $\hat{\Sigma}$ as defined by (5) with respect to $\Sigma_{Blue} \cup \Sigma_{Assign}$. We have

$$\begin{array}{ll} P_{i,1}(z,\alpha) \Leftrightarrow z_i < x_{\alpha_i}, & f_{i,1}(z) = z_i + \delta, \\ P_{i,2}(z,\alpha) \Leftrightarrow z_i > x_{\alpha_i}, & f_{i,2}(z) = z_i - \delta, \\ P_{i,3}(z,\alpha) \Leftrightarrow z_i = x_{\alpha_i}, & f_{i,3}(z) = z_i \end{array}$$

and

$$\begin{array}{ll} Q_{k,1}(z,\alpha) \Leftrightarrow down_k, & g_{k,1}(\alpha) = \alpha_{k-1}, \\ Q_{k,2}(z,\alpha) \Leftrightarrow up_k, & g_{k,2}(\alpha) = \alpha_{k+1}, \\ Q_{k,3}(z,\alpha) \Leftrightarrow \neg(down_i \vee up_i), & g_{k,1}(\alpha) = \alpha_k. \end{array}$$

Note that $P_{i,j}$ only depends on z_i and α_i and so we may also write $P_{i,j}(z_i, \alpha_i)$ and similarly for $f_{i,j}$. Since the system is weakly observable, the properties listed in Theorem 4.1 hold. We may also determine the rate of convergence of $\hat{\Sigma}$. We have

Proposition 6.1 *The observer $\hat{\Sigma}$ applied to $\Sigma_{Blue} \cup \Sigma_{Assign}$ converges in at most $t^\alpha_\sigma + 1$ steps in any execution σ of $\Sigma_{Blue} \cup \Sigma_{Assign} \cup \hat{\Sigma}$.*

Proof: (*Sketch*) Recall that we have assumed that $z_i \in (x_i, x_{i+1})$ is invariant. Since $\alpha(t^\alpha_\sigma) = \alpha^* = (1, ..., N)$, from (5) we have that $z_i(t^\alpha_\sigma) > x_{\alpha_i}$ for all i. This implies that $B' = \alpha^*$. ■

6.3 A More Efficient Observer

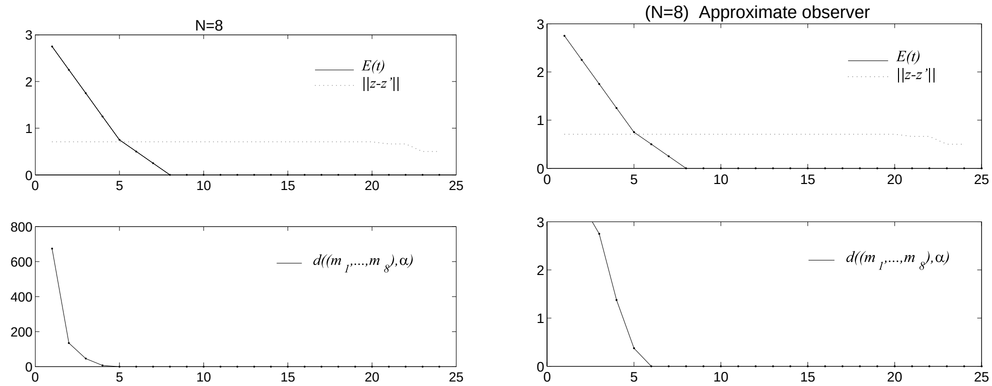
Note that the number of possible assignments $|U|$ is $N!$. Therefore, without some efficient scheme for representing $\hat{\alpha}$, the space and computational requirements for computing the clause in (5) is prohibitively high. In this section we propose an approximate observer for α (in the sense that it may over-approximate the observer defined previously) in the following manner. For each i we keep a set $m_i \subseteq \{1, ..., N\}$ of possible assignments to the *ith* blue robot. Initially, $m_i = \{1, ..., N\}$.

First, for $A, B \subseteq \{1, ..., N\}$, define

$$A \leq B \;\Leftrightarrow\; \forall i \in A\, \forall j \in B \,.\, x_i \leq x_j$$

and define $A^{\leq} = \{j \;:\; A \leq \{j\}\}$. Also define $A^{\geq} = \{j \;:\; A \geq \{j\}\}$. We use $A \not\leq B$ to mean $\neg A \leq B$ and similarly for $\not\geq$. We now describe in Algorithm 1 a procedure for mapping forward the sets m_1,...,m_N at each step. Although we write the procedure as a loop to better show its structure, it could equally be written (with some effort) as a rule in a guarded command program.

In each iteration i of the for-loop in Algorithm 1 we compute four sets: AP, DP, AN and DN (for "Add Previous", "Delete Previous", "Add Next" and "Delete Next"). AP and AN consist of elements from m_{i-1} and m_{i+1} respectively that should be added to m_i because there is a possibility that α_i could take on these values. Similarly, DP and DN consist of elements from m_i that should be deleted from m_i. Note that DP is the set of elements in m_i that *must* be exchange with some element in m_{i-1} under the assumption that nothing in m_{i-1} has exchanged with an element of m_{i-2}. The boolean variable *flag* is used to denote the truth of this assumption.



(a) (b)

Fig. 3: The performance of $\hat{\Sigma}$ (a) and $\tilde{\Sigma}$ (b) for the RoboFlag Drill. Here, N=8.

Algorithm 1 Approximate Observer

$m_1' = (m_1 - m_2^{\leq}) \cup (m_1^{\not\leq} \cap m_2)$
flag $=$ *true*
for $i = 2$ to N **do**
  $AP = m_i^{\not\geq} \cap m_{i-1}$
  if flag **then**
    $DP = m_{i-1}^{\geq} \cap m_i$
  else
    $DP = \emptyset$
  end if
  $AN = (m_i - DP)^{\not\leq} \cap m_{i+1}$
  $DN = m_{i+1}^{\leq} \cap m_{i-1}^{\leq} \cap m_i$
  $m_i' = (m_i - DP - DN) \cup AP \cup AN$
  flag $= (AP \cup DP = \emptyset)$
end for

Call the function computed by the above procedure $\hat{g}$ so that $m_i' = \hat{g}_i(m)$. We may then represent the approximate observer $\tilde{\Sigma}$ by the single clause

$$true : b_i' = m_i \cap \bigcup_{j=1}^{3} \{\alpha_i \; : \; z_i' = f_{i,j}(z_i) \wedge P_{i,j}(z_i, \alpha_i)\}$$

$$\wedge (c_1', ..., c_N') = Refine(b_1', ..., b_N')$$

$$\wedge m_i' = \hat{g}_i(c_1', ..., c_N') \qquad (7)$$

where $Refine(b_1, ..., b_N)$ takes the assignment sets $b_1, ..., b_N$ and produces assignment sets $c_1, ..., c_N$ with the following property: If $c_i = \{k\}$ then $k \notin c_i$ for any $j \neq i$. This is helps increase the rate of convergence of $\tilde{\Sigma}$ by decreasing the size of the sets m_i at each step.

For example if $m_1 = \{1, 2, 3\}$, $m_2 = \{2, 3\}$ and $m_3 = \{1, 2\}$ and $x_i = i$, the reader can check that $m_1' = \{1, 2\}$, $m_2' = \{1, 2, 3\}$ and $m_3' = \{1, 2, 3\}$.

It can also be shown that in any execution σ of $\Sigma_{Blue} \cup \Sigma_{Assign}$ each set m_i converges to α_i in at most $t_\sigma^\alpha + 1$ steps, where t_σ^α is time at which α stabilizes. The proof of this fact proceeds similarly to that of 6.1. It is sufficient to notice that at time $t_\sigma^\alpha + 1$, by (7) and *Refine* described above, we have $c_i' = \{i\}$.

6.4 Simulation Results

We implemented the RoboFlag drill in MATLAB 6.0 together with the observer defined in equation (5). We considered eight robots per team, and show the performance of the observer in Figure 3(a). We denote by $d(\hat{\alpha}, \alpha)$ the distance introduced in (6). We also define the *Entropy* of α by

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

In Figure 3(b), we show the performance of the approximate observer on the same execution of the same system. For the approximate observer we plot

$$d((m_1, ..., m_N), \alpha) := \frac{1}{N} \sum_{i=1}^{N} d(m_i, \alpha_i)$$

where $d(m_i, \alpha_i)$ is computed according to (6). Notice that the approximate observer converges more slowly than the full one. Systems where $N > 10$ are computationally too difficult for the full observer but tractable for the approximate observer. We show the performance of the approximate observer in an example with $N = 30$ in Figure 4. In all the simulations the initial assignment was chosen randomly.

7 Conclusions

We have examined the observability problem for a class of hybrid guarded command programs. We proposed a definition of weak observability and proposed an observer that converges when the system is weakly observable. The proposed observer can be computationally expensive as is apparent when it is used with the RoboFlag system. Thus an approximate observer was proposed that has low computational and space requirements.

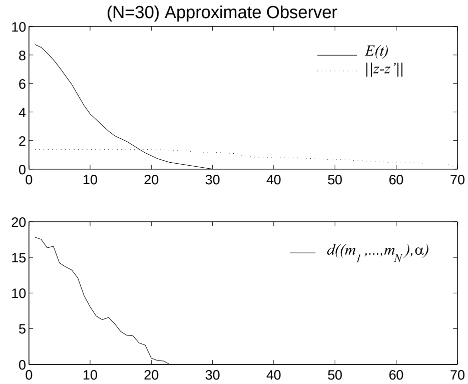


Fig. 4: Approximate observer performance for the RoboFlag Drill where N=30.

We have not provided practical algebraic tests for determining observability of guarded command programs nor have we provided a general construction for a practicable observer for systems with large state spaces. We hope to address these problems in future work. We also hope to explore non-deterministic guarded commands programs that can be used for system specification.

References

[1] A. Balluchi, L. Benvenuti, M. D. Di Benedetto, and A. Sangiovanni-Vincentelli. Design of observers for hybrid systems. *Lecture Notes in Computer Science 2289,C. J. Tomlin and M. R. Greensreet Eds. Springer*, pages 76–89, 2002.

[2] 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.

[3] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics and constraints. *Automatica*, 35:407–427, 1999.

[4] M. S. Branicky, V. S. Borkar, and S. K. Mitter. A unified framework for hybrid control: model and optimal control theory. *IEEE Trans. Automat. Control*, 43:31–45, 1998.

[5] P. E. Caines. Classical and logic-based dynamic observers for finite automata. *IMA J. of Mathematical Control and Information*, pages 45–80, 1991.

[6] R. D'Andrea, R. M. Murray, J. A. Adams, A. T. Hayes, M. Campbell, and A. Chaudry. The RoboFlag Game. In *American Controls Conference*, 2003.

[7] E. W. Dijkstra. Guarded commands, nondeterminacy 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.

[8] R. L. Grossmann, A. Nerode, A. P. Ravn, and H. Rischel. *Hybrid Systems,* Lecture Notes in Computer Science (Vol 736. Springer Verlag, New York, 1993.

[9] E. Klavins. A formal model of a multi-robot control and communication task. In *Conference on Decision and Control*, Hawaii, 2003. Submitted for review.

[10] A Sabelfeld and A. C. Myers. Language-based information-flow security. *IEEE J. on Selected Areas in Communications*, pages 1–15, 2003.

[11] R. Vidal, A. Chiuso, and S. Soatto. Observability and identifiability of jump linear systems. In *Decision and Control Conference*, Las Vegas, 2002.