Immediate Observability of Discrete Event Systems with Application to User-Interface Design

Meeko Oishi, Inseok Hwang, and Claire Tomlin
Hybrid Systems Lab, Stanford University, Stanford, CA
moishi,ishwang,tomlin@stanford.edu

***Abstract*— A human interacting with a hybrid system is often presented, through information displays, with a simplified representation of the underlying system. This *interface* should not overwhelm the human with unnecessary information, and thus usually contains only a subset of information about the true system model, yet, if properly designed, represents an abstraction of the true system which the human is able to use to safely interact with the system [1]. For cases in which the human interacts with all or part of the system from a remote location, and communication has a high cost, the need for a simple abstraction which reduces the amount of information that must be transmitted is of the utmost importance. The user should be able to immediately determine the actual state of the system, based on the information displayed through the interface. In this paper, we derive conditions for *immediate observability* in which the current state of the system can be unambiguously reconstructed from the output associated with the current state and the last or next event. Then, we show how to construct a discrete event system output function which makes a system immediately observable, and apply this to a reduced state machine which represents an interface.**

Keywords: discrete observability, discrete event systems, interface design, remote systems

I. INTRODUCTION

Human-automation systems are automated systems with which a human interacts. These types of systems are pervasive, occurring in consumer products (alarm clocks, VCRs, cellular phones), transportation systems (automobiles, commercial aircraft), scientific research platforms (unmanned ocean- and aerial-vehicles), military systems (fleets of single-man aircraft), and in other realms. Many of the systems users interact with have *hybrid* behavior, in which the underlying system's continuous-state dynamics switch according to discrete rules governed by operating conditions or mode-logic, for example. The user, when interacting with such complex systems, is often presented with a simplified representation of the underlying system, which we call the *interface*.

This representation can be based upon a discrete abstraction of the system: discrete event systems easily capture the way in which a user interacts with a hybrid system. For example, one abstraction might be based on requirements of safety [2], [3], while another abstraction could be based on the desired trajectory, as in autopilot mode-logic. In aircraft autopilots, a simple discrete indication, a word, can represent an infinite number of physical aircraft trajectories. A pilot recognizes, upon seeing the word "FLARE" on the cockpit mode control panel, that the aircraft is following a path to touchdown smoothly on the runway. The user interacts with the system in discrete ways, as well: pilots push buttons, adjust levers, and flip toggles – all discrete events under the pilot's control.

We presume to begin with such an abstraction to represent the underlying hybrid system. The problem of determining an acceptable interface for a system involves finding a suitably reduced discrete event system which accurately represents the information relevant for the user. We are motivated by the work of Heymann and Degani, who have determined a method to synthesize interfaces using state reduction techniques [1]. Here, we study the relationship between observability and interface design. For an interface to be useful, the user must be able to reconstruct, on the basis of a given output, information about the original system.

We define a type of observability necessary for user-interfaces: *immediate observability*, or the ability to determine the current state of the system based only on the current output and events associated with the current state. Discrete systems which model user-interfaces of safety-critical systems must be immediately observable in order to be "good" interfaces: that is, to accurately represent the underlying system to the user, so that the user will not be misled or confused. We formulate conditions for immediate observability, as well as an output synthesis method to synthesize state outputs which fulfill these conditions and which are minimal in cardinality. In remote and distributed systems, in which communication is prohibitively expensive, this method could be used to synthesize outputs for interfaces. The result is an output which minimizes necessary communications, yet provides all information relevant to the user.

There has been a wealth of research on discrete observability, mostly with application to the synthesis of discrete supervisors. In this paper, we examine discrete observability in the context of reconstruction of state information from a given output. Discrete observability has been defined in many ways. Ramadge [4] derives conditions for current state observability under which the current state can be uniquely determined from a sequence of past events and state observations. Özveren et al. [5] consider a discrete event system whose output is a subset of an event set, and derive a condition to determine the current state uniquely at times separated by bounded numbers of transitions. Caines et al. [6] propose a dynamical logic observer for a partially observed discrete event system and develop observer synthesis methods using a current state observer tree. The state observer in [5] may not always determine the current state and that proposed in [6] needs a number of state

transitions to determine the current state, and even then can determine only the state which belongs to a subset of states. Based on [4], Lin et al. [7] develop necessary and sufficient conditions for the existence of a supervisor which enables and disables controllable events based on recorded occurrences of observable events, so that the closed-loop system meets some specifications.

We start with background for the problem as well as some basic definitions, then define immediate observability and derive conditions for a system which is immediately observable. We show how to use these conditions to synthesize a state output of minimal cardinality for a reduced system which represents an interface. We illustrate the output synthesis through an example: minimal, observable interface design for an individual aircraft in a group of formation-flying aircraft.

II. Problem Formulation

While the conditions we derive for immediate observability are quite general, we typically only wish to apply the output synthesis method on systems which are already reduced. A reduced system produces the same behavior (output sequences) as the original system, but with fewer discrete states. (If the system we wish to observe were not reduced, the output would create distinctions which may not be necessary for the user.) While methods for state reduction were developed in the late 1950s ([8], [9], [10], [11]), obtaining a minimal model, in which the number of states of a reduced model is minimal, is more complicated. A method to determine minimal models was developed in the 1960s [12], and much research has been done since then on methods to compute the minimal model ([13], [14], [15], [16], [17]). Techniques for state reduction or state minimization are not limited to deterministic systems ([15], [18]) and this is also an active topic of research. However, interfaces by nature must be deterministic: not only must the user know the current state of the system, but the user must also be able to predict, uniquely, where the system will transition to next. Nondeterminism is responsible for such phenomena as mode confusion and automaton surprises [19].

Interface design involves selecting relevant information about the underlying system to display to the user: too much information can overwhelm the user, and too little information can confuse and mislead the user [19]. There have been many recent efforts to use formal methods [20], [21], [22], [23], as well as a hybrid system verification method [3], [2] to verify that an interface accurately represents its underlying system. In [1], the authors use state reduction techniques to synthesize an interface as a reduced model of a more complex system. The reduction process occurs by effectively combining states for which all input sequences produce the same output sequences: examining the system's behavior from the output, the two states appear to function in exactly the same way. For interfaces, combining states is a way to eliminate information unnecessary to the user, while maintaining necessary distinctions allows the user to distinguish between potentially important states.

In this paper, we make use of two discrete event system models: an automaton G and another automaton G_{sys}. While we first consider G to be a generic, nondeterministic automaton, when considering application of immediate observability to user-interfaces, we presume that G is a deterministic and reduced automaton which represents a more complicated, but also deterministic, automaton G_{sys}. Additionally, the notation $|\cdot|$ indicates the cardinality of a set.

Let the nondeterministic discrete event system $G = (Q, \Sigma, \delta, Q_0)$ consist of a finite state set Q, a finite event set $\Sigma = \Sigma_o \cup \Sigma_{uo}$, which is composed of the set of observable events Σ_o and the set of unobservable events Σ_{uo}, the one-to-many state transition function $\delta : Q \times \Sigma \to 2^Q$, and the set of the possible initial states $Q_0 \subset Q$. We define the finite set Ψ as the set of observable events combined with the ϵ event which represents all unobservable events, i.e. $\Psi = \Sigma_o \cup \{\epsilon\}$. In addition, the finite output set Y, defined by the many-to-one output map $h : Q \to Y$, is assumed to be available. Thus, we can infer information about the state from the sequence of events belonging to Ψ and the sequence of outputs.

III. Immediate Observability and Immediately Observable Output Synthesis

Using the system $G = (Q, \Sigma, \delta, Q_0)$ as defined in Section II, we first enunciate the conditions for *co-observability* from [4], which we call *eventual observability* here. We then derive conditions for immediate state observability of a generic system G and provide an illustrative example. Using this result, we describe a method to synthesize an output map h such that (G, h) is immediately observable.

Definition 1: A system is *eventually observable* if the following two properties from Proposition 6.2 of [4] hold:

1) For each pair $(q, \sigma) \in Q \times \Sigma$, if $q_1, q_2 \in \delta(q, \sigma)$ with $q_1 \neq q_2$, then $h(q_1) \neq h(q_2)$.
2) No two distinct states q_1, q_2 have a common event sequence which can generate a common output sequence.

Remark 1: The first condition (called *trackability* in [4]) allows for the determination of the next-state, given the next output and the current event. The second condition allows for unique determination of the initial state.

Remark 2: In an eventually observable system, given a sequence of events and a sequence of outputs, the initial state and the current state can be determined. While these sequences are finite, there are no further constraints on the length of these sequences, hence the notion of "eventual" observability.

A. Condition for immediate state observability

Definition 2: A discrete event system is *immediately observable* if the current state can be determined uniquely from the current state output and either the last or next event.

First define the following sets:

$$\begin{array}{rcl} Q_y & := & \{q \in Q \mid \exists y \in Y,\, y = h(q)\} \\ I^f_\sigma & := & \{q' \in Q \mid \forall q \in Q, \exists \sigma \in \Psi,\, q' = \delta(q,\sigma)\} \\ I^b_\sigma & := & \{q \in Q \mid \forall q' \in Q, \exists \sigma \in \Psi,\, q' = \delta(q,\sigma)\} \end{array} \quad (1)$$

where Q_y is the set of all states whose output is $y \in Y$, I^f_σ is the set of all states reachable through an event $\sigma \in \Psi$ from any $q \in Q$, and I^b_σ is the set of all states which can reach any state $q' \in Q$ through an event $\sigma \in \Psi$. Using this notation, we state conditions for immediate observability.

Proposition 1: The system $G = (Q, \Sigma, \delta, Q_0)$ is immediately observable if and only if the following conditions hold ($\forall \sigma,\, \sigma' \in \Psi,\, \forall y \in Y$):

1) (For initial state: $y_0 = h(q_0)$) for all $q_0 \in Q_0$
   a) $h^{-1}(y_0)$ exists, (only Q_{y_0} available) $\bigvee$
   b) $|Q_{y_0} \cap I^b_\sigma| = 1$ (Q_{y_0} and the next event available).
2) (For any state: $y = h(q)$ for all $q \in Q$ and for $i \in \mathbb{N}^+$
   a) $h(q(i-1)) \neq h(q(i))$ if $q(i) \in \delta(q(i-1), \epsilon)$, and
   b) $|Q_y \cap I^f_\sigma| = 1$ (Q_y and last event available), $\bigvee$
   c) $|Q_y \cap I^b_\sigma| = 1$ (Q_y and next event available), $\bigvee$
   d) $|I^f_\sigma \cap Q_y \cap I^b_{\sigma'}| = 1$ (Q_y, last, and next events available).

Proof: **(if)** (i) For the initial states, if Condition (1a) is true, the initial state can be uniquely determined from the state output. If Condition (1b) is true, there is a unique initial state with the output y_0 and from which the event $\sigma \in \Psi$ occurs. Thus, the initial state can be determined uniquely. (ii) Condition (2a) states that the unobservable events can be detected and thus validates the assumption that both the current state output and the last (or the next) event are available to determine the current state. If Condition (2b) is true, there is only one state which has the current state output y and is reachable through the last event $\sigma \in \Psi$. Thus, the current state can be uniquely reconstructed. Similarly, Conditions (2c) and (2d) can be proved.

(only if) [proof by contradiction] Since Condition (1) is obvious, we consider Condition (2) only. (i) Suppose Condition (2a) is not true. Since the event is unobservable, only the current state output is available to determine the current state. However, more than one state has the same output. Therefore, the current state cannot be uniquely determined. This is a contradiction to the assumption that the current state is immediately observable. (ii) Suppose Condition (2b) is not true. Then, there is more than one state which has the current output y and is reachable through the last event $\sigma \in \Psi$. Therefore, the current state cannot be uniquely determined. This is a contradiction. Similarly, Conditions (2c) and (2d) can be proved. ■

To test for immediate observability, choose the appropriate condition of Proposition 1 to test, depending on available information. For the remainder of the paper, we consider only the case in which the current state output and the last occurring event are available. For this discrete event system

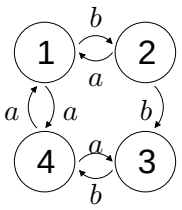

Fig. 1. Nondeterministic discrete event system G

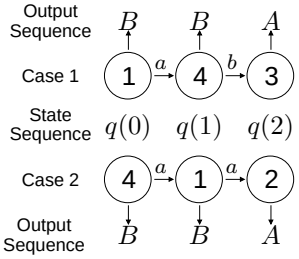

Fig. 2. Output and event sequences show G is not immediately observable.

to be immediately observable, Conditions (1a), (2a), and (2b) must be true for all combinations of $\sigma \in \Psi$ and $y \in Y$ dictated by the transition function δ. Without Condition (2a) we cannot determine the current state reached through the ϵ event whose output is the same as the previous output.

The complexity of the tests for immediate observability depends on the number of outputs and events. For completely specified automata (in which δ is defined over all combinations of $(q, \sigma) \in Q \times \Sigma$) with $n_Y = |Y|$ outputs and $n_\Psi = |\Psi|$ events at most $n_Y n_\Psi$ intersections must be evaluated. Incompletely specified automata (in which δ is defined for some combinations of $(q, \sigma) \in Q \times \Sigma$) will require fewer intersections, depending on the relation δ.

Example: To illustrate Proposition 1, we consider a nondeterministic discrete event system $G = (Q, \Sigma, \delta, Q_0)$ where $Q = \{1, 2, 3, 4\}$, $\Sigma = \{a, b\}$, $Q_0 = \{1, 4\}$, and the state transition function δ is defined as in Figure 1. We assume events $\Psi = \{a, b\}$ outputs $Y = \{A, B\}$, and the output map h is defined by: $h(1) = h(4) = B$ and $h(2) = h(3) = A$. Thus, we have two observed sequences: the output sequences and the event sequences. Consider two state sequences shown in Figure 2. Start from the initial state $q(0)$: the only available information is the output B. Since both cases have the same output, we cannot identify the initial state. Let the state move to the next state $q(1)$ through an observable event a. New information available is the event a and the current output B. Since both cases generate the same observed sequences, we still cannot distinguish the current state or the initial state uniquely. Now, let the state again move to the next state $q(2)$. New information distinguishes the two cases: Although both cases show state output A, in case 1 event b is observed, and in case 2 event a is observed. Using the observability condition of Definition 1, we can uniquely reconstruct the initial conditions for both cases and thus determine all the states up to $q(2)$ uniquely for both cases. This system satisfies the observability condition of Definition 1, yet it is not immediately observable. From Condition (2b) of Proposition 1, with $Q_A = \{2, 3\}$, $Q_B = \{1, 4\}$, $I^f_a = \{1, 2, 4\}$, and $I^f_b = \{1, 3\}$, we obtain $Q_A \bigcap I^f_a = \{2\}$, $Q_A \bigcap I^f_b = \{3\}$, $Q_B \bigcap I^f_a = \{1, 4\}$, and $Q_B \bigcap I^f_b = \{1\}$. Since $Q_B \bigcap I^f_a$ is not a singleton, the system is not immediately observable and the states 1 and 4 are indistinguishable. ■

B. Relationship to Eventual Observability

Proposition 2: If a system G is immediately observable, then it is also eventually observable.

Proof: In an immediately observable system, only one event (the last occurring event) and one output (the current output) are necessary to determine the current state. The initial state can be uniquely determined from the initial state output. Thus an immediately observable system fulfills the two conditions of Definition 1, and so is eventually observable. ∎

We repeat the following from [8], [9], [10], [11]:

Definition 3: Two modes $q_i, q_j \in Q_{\text{sys}}$ are *compatible* if 1) $h_{\text{sys}}(q_i) = h_{\text{sys}}(q_j)$, and 2) for all events $\sigma \in \Psi_{\text{sys}}$ possible from both modes, the two resulting modes after the event σ is applied also have the same output: $h_{\text{sys}}(\delta_{\text{sys}}(q_i, \sigma)) = h_{\text{sys}}(\delta_{\text{sys}}(q_j, \sigma))$.

Definition 4: A set of modes S is compatible if and only if all possible pairs of modes in S are compatible.

Proposition 3: Given a deterministic automaton G_{sys} and an output ψ, the minimal (or simply reduced), deterministic automaton G is eventually observable with respect to ψ.

Proof: By definition, compatible states are those which have the property that no sequence of events will distinguish between them. The states of the reduced automaton G are the largest sets of compatible states which produce the same output sequence. The states of G_{sys} which have not been lumped together (distinct states of G) can be distinguished after a finite number of steps, discounting loops. ∎

C. Immediately Observable Output Synthesis

Interfaces can be formed through state reduction [1]. However, there is often freedom in forming a reduced automaton G from the underlying system G_{sys}. A reduced or minimized model G can be formed by first identifying its modes Q, through state reduction. The set of events in G is $\Sigma = \Psi_{\text{sys}}$, and the deterministic transition function δ is defined according to the event-successors of each transition $\sigma \in \Sigma$ possible for each compatible. Recall that event-successors are defined by:

Definition 5: An *event-successor* of a compatible $C_i = \{q_i^{(1)}, ...q_i^{(n_i)}\}$ for a given event σ is a compatible C_j which contains $\{\delta(q_i^{(1)}, \sigma), ..., \delta(q_i^{(n_i)}, \sigma)\}$.

If more than one event-successor is possible for a given event σ, only one event-successor should be selected. In anticipation of the upcoming output synthesis, the event-successor can be chosen to minimize constraints on that output. For example, if we know $C_1 \xrightarrow{\sigma} C_2$ and C_2 is one of multiple event-successors for σ from another compatible C_3, we should choose $C_3 \xrightarrow{\sigma} C_2$.

We now want to determine how to choose the output function h such that (G, h) is immediately observable. There are many possible choices; one option is the identity map. With $y = q$, (G, h) is immediately observable. For situations in which the interface is physically connected to the underlying system, this would likely be the best solution, since the user has an intuitive sense of the various interface modes. However, in distributed systems, information about the state of the system is dependent on outside information. Given communication bandwidth limitations, in which minimizing information passed to each of the distributed units is of the utmost importance, an output h can be used to determine the minimum of information which must be transferred in order to be able to reconstruct the interface mode.

The conditions on a system G and its output h for immediate observability result in a minimal set of restrictions on h. We can then use these restrictions to determine the output of minimal cardinality which fulfills these conditions, and guarantees immediate observability of the interface.

Conditions (2a) and (2b) of Proposition 1 for immediate observability result in the following constraints on the output:

$$\forall p, q \text{ such that } p \xrightarrow{\varepsilon} q, \qquad h(q) \neq h(p) \tag{2}$$

$$\forall p, q \in I_\sigma^f \subseteq Q, \qquad h(p) \neq h(q) \tag{3}$$

From Condition (2a), we know the state outputs before and after an unobservable event must be distinct. From Condition (2b), we know that the current state must be uniquely and immediately determined from the last occurring event and the current output. For a given set of states I_σ^f, and an output y, there must uniquely exist one state in I_σ^f with output y. (If more than one state maps to the same output y, $|I_\sigma^f \cap Q_y| > 1$ so the system cannot be immediately observable.) Therefore none of the states in I_σ^f can map to the same output: all states in I_σ^f must have different outputs.

To make use of immediate observability constraints (3) and (2), we define the following, based on [8], [9], [10], [11]:

Definition 6: A pair of modes $(q_i, q_j) \in Q \times Q$ is *allowable* if $h(q_i) = h(q_j)$.

Definition 7: A set of modes S is an *allowable set* if each mode pair in S is an allowable pair.

Definition 8: A set of modes S is a *maximum allowable set* if it is not contained in any other allowable set.

Definition 9: A set of allowables T *covers* Q if and only if it contains all elements of Q.

Although we define the map h such that it is many-to-one, and defined for all $q \in Q$, we do not invoke closure conditions as they are invoked in the construction of a reduced automaton from compatibles.

We can determine the maximum allowable sets for a given set of constraints through a merger table [11] for

$Q = \{q_1, q_2, \cdots, q_n\}$.

q_2	*				
q_3	*	*			
$\vdots$	$\vdots$		$\ddots$		
q_{n-1}	*	*	$\cdots$	*	
q_n	*	*	$\cdots$	*	*
	q_1	q_2	$\cdots$	q_{n-2}	q_{n-1}

(4)

We first enumerate all possible pairs (4), then mark pairs which are not allowable by "×", as determined by each I_σ^f. (For I_σ^f with n_σ elements, there are $n_\sigma(n_\sigma - 1)/2$ constraints.) As opposed to the procedure used in [11], we do not need to iterate (4) through the transition function δ: we are constructing an output, rather than an automaton. (The user will not interact directly with the minimized system. Rather, this output can be decoded so that the reduced model (the interface) will be displayed to the user).

We determine the maximum allowables by examining the resultant table according to the same procedure as in [11] to determine maximum compatibles. From the maximum allowables, we now can create a list of all possible allowable sets, which will also be the set of all possible output mappings. For each maximum allowable A_i, enumerate all subsets which have not been enumerated through examining the subsets of another maximum allowable A_j (so that each subset is counted only once). This results in a total of n_A allowables, each of which is a potential output. We use integer programming to determine which outputs (allowable sets) must be selected to minimize the cardinality of the output as well as guarantee immediate observability. For each mode $q_k \in Q$, one allowable set which contains q_k must be selected in the minimal output map. (Each mode must have exactly one output associated with it: the output Y must disjointly cover Q.) This results in a constraint in n_k variables, where n_k is the number of allowables which contain q_k.

$$x_k^{(1)} + ... + x_k^{(n_k)} = 1 \tag{5}$$

The binary variable $x_k^{(j)}$ represents the jth allowable which contains q_k: $x_k^{(j)} = 1$ if the allowable it represents is selected, and $x_k^{(j)} = 0$ otherwise. There will be a total of n such constraints, one for each mode of the interface. This differs from the process of obtaining prime compatibles in that we do not need to satisfy any closure conditions.

To solve the problem of minimizing an interface we form a set-partitioning problem, in which a set of states Q must be partitioned subject to constraints (5) necessary to enforce immediate observability. These problems are, in general, NP complete. However, for this particular problem,

$$\begin{aligned} \min \quad & \mathbf{1}^T x \\ \text{subject to} \quad & Ax = \mathbf{1} \end{aligned} \tag{6}$$

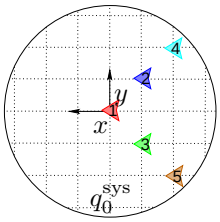

Fig. 3. Five aircraft flying in formation q_0^{sys}. Each aircraft knows its position relative to a the lead aircraft (Aircraft 1) and communicates with Aircraft 1, which issues control commands.

with $A \in \{0,1\}^{n \times n_A}$ and $x \in \{0,1\}^{n_A}$ the vector of all possible allowables $x_k^{(j)}$, can be solved in polynomially bounded time [24]. For a fixed n_A, the set-partitioning can be accomplished in $\mathcal{O}({n_A}^2 \log n_A)$ steps [24], [25].

Unlike linear programs, there is no certificate of optimality; however all possible solutions, given a feasible set of restrictions $Ax = \mathbf{1}$, can be enumerated. This computation would likely be done off-line.

Proposition 4: Given a reduced model G and a set of m equality constraints $Ax = \mathbf{1}$ which enforce immediate observability, the solution x^* to an integer program (6) determines the output map $h : Q \to Y$ of minimal cardinality which results in immediate observability for (G, h).

Proof: A sketch of the proof follows: The solution x^* specifies which allowables should be chosen to produce an output Y which is of minimal cardinality $n_Y^* = \mathbf{1}^T x^*$ and which satisfies conditions of immediate observability for the reduced system G with respect to the output map h. This solution is not necessarily unique. ■

IV. Example: Coordinated Aircraft Maneuvers

We consider a distributed system with five formation-flying aircraft. While this is an inherently hybrid system due to the nonlinear aircraft dynamics, we assume that the aircraft are completely controllable, and therefore only consider discrete relative position changes based on a known sequence of desired formations for all five aircraft. While we assume for simplicity that each aircraft in the formation communicates only with the lead aircraft, other communication structures might also be considered, such as one in which each aircraft communicates only with its immediate neighbors [26].

Figure 3 shows the five aircraft in their initial configuration (mode q_0^{sys}), flying nominally at a constant velocity. Each aircraft knows its position relative to the lead aircraft (Aircraft 1), and all five aircraft travel nominally at the same speed. Each aircraft receives its instructions from the lead aircraft, and does not have access to information regarding the other aircraft in formation. These instructions may be events (i.e. 'Change formation') or outputs (i.e. 'Maneuver A'). Only some of these events and outputs may affect a given aircraft in the formation (for example, perhaps 'Maneuver A' involves only the leading aircraft) – so an individual aircraft

may not need to know about every single event and output that occurs in order to maintain its proper location in the formation. Each aircraft only needs to know sequences of events and outputs which will affect its position relative to the lead aircraft.

The five aircraft maneuver according to the automaton shown in Figure 4. Transitions between formations occur based on simple, physically constrained trajectories: for example, for aircraft to directly invert their "V" formation in mode q_0^{sys} to that in q_2^{sys} (when event α_2 occurs), Aircraft 2 and 3 must move forward in the x direction two grid spaces, while Aircraft 4 and 5 move four grid spaces. (Note that aircraft maintain the same orientation in the y direction for the entire automaton: 4-2-1-3-5.) In Figure 4, all aircraft following Aircraft 1 are placed according to their position relative to the lead aircraft along an (x, y) coordinate system (shown by a grid). However, some formations involve separation of the five aircraft into either two or three groups (see modes q_4^{sys} and q_6^{sys}). Aircraft not following the lead aircraft are not drawn on top of a grid: these aircraft maintain their relative spacing to other aircraft in their group. For example, in mode q_4^{sys}, while Aircraft 2 and 4 maintain their positions relative to Aircraft 1, Aircraft 3 and 5 are separate from the group and are following another mission in which Aircraft 3 leads Aircraft 5.

For the remainder of the problem, we will focus on observability from the point of view of Aircraft 2. Examining Figure 4, Aircraft 2 must distinguish between four different formations: these are exemplified by formations in modes $q_0^{\text{sys}}, q_2^{\text{sys}}, q_6^{\text{sys}}$, and q_9^{sys}. Formations in $q_0^{\text{sys}}, q_1^{\text{sys}}, q_5^{\text{sys}}, q_{10}^{\text{sys}}$, and q_{11}^{sys} are essentially equivalent for Aircraft 2, so we map them to the same output, y_0^{sys}. Similarly, $\{q_2^{\text{sys}}, q_3^{\text{sys}}, q_4^{\text{sys}}\} \to y_1^{\text{sys}}$, $\{q_8^{\text{sys}}, q_9^{\text{sys}}\} \to y_2^{\text{sys}}$, and $\{q_6^{\text{sys}}\} \to y_3^{\text{sys}}$. Notice that in q_8^{sys} and q_9^{sys} Aircraft 2 leads Aircraft 4, and in q_6^{sys} Aircraft 4 leads Aircraft 2.

However, this automaton has more information than Aircraft 2 needs in order to maintain one of its four positions relative to the lead aircraft. We reduce Figure 4 according to the four outputs $y_0^{\text{sys}}, y_1^{\text{sys}}, y_2^{\text{sys}}, y_3^{\text{sys}}$ to find the minimal information Aircraft 2 needs to distinguish between in order to maintain its correct position in the aircraft formation. This allows us to take advantage of situations in which Aircraft 2 may not need to change its relative position in two different formations, for example. Using state reduction techniques, we obtain the minimal model G for Aircraft 2. Its modes Q are formed by the mapping $\{q_8^{\text{sys}}, q_9^{\text{sys}}\} \to q_1$, $\{q_2^{\text{sys}}, q_3^{\text{sys}}, q_4^{\text{sys}}\} \to q_2$, $\{q_1^{\text{sys}}, q_7^{\text{sys}}, q_{11}^{\text{sys}}\} \to q_3$, $\{q_0^{\text{sys}}, q_1^{\text{sys}}, q_5^{\text{sys}}, q_{11}^{\text{sys}}\} \to q_4$, $\{q_0^{\text{sys}}, q_1^{\text{sys}}, q_{10}^{\text{sys}}, q_{11}^{\text{sys}}\} \to q_5$, $\{q_6^{\text{sys}}\} \to q_6$. The aircraft *must* be able to distinguish between the six modes shown in Figure 5. For the pilot of Aircraft 2, this defines the pilot's "interface": this is the information about the underlying five-aircraft system G_{sys} which the pilot must have at his disposal.

Communication costs are often prohibitively high in distributed systems [27]. To reduce those costs, we find the minimal information to broadcast to each aircraft. To do this, we synthesize an output which is minimal in cardinality, and which allows reconstruction of information in the reduced model, Figure 5. The output is constructed as in Section III-C from the conditions for immediate observability of (G, h): We first find the maximum allowables and their subsets, and then solve an integer optimization to determine which set of allowables results in an output of minimal cardinality, subject to covering conditions.

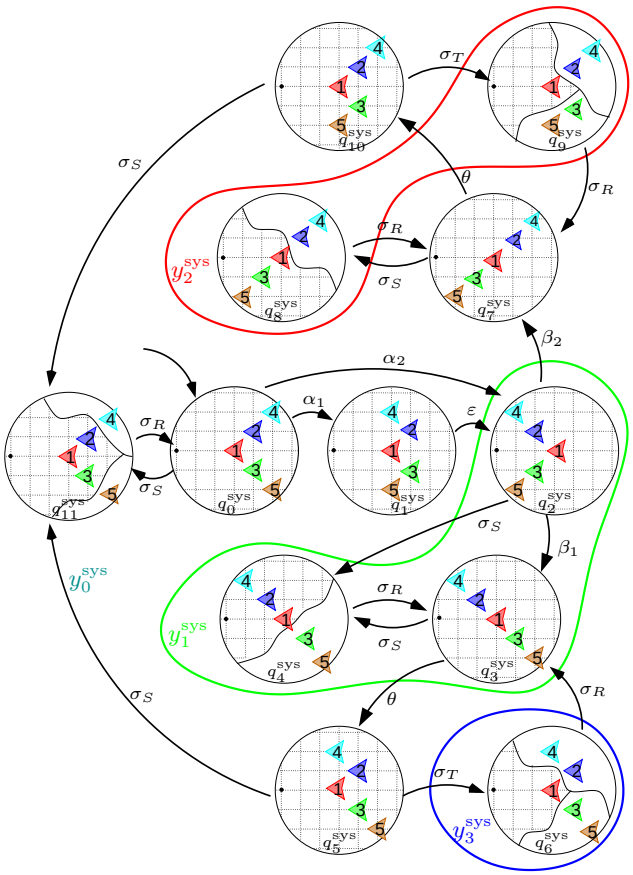

Fig. 4. The position of Aircraft 2 in G_{sys} is indicated by enclosing curves with different shadings: all modes unshaded map to y_0^{sys}, the lightest shading maps to y_1^{sys}, the medium-shading maps to y_2^{sys}, and the darkest shading maps to y_3^{sys}.

Assuming that the last event and the current output are accessible to the system, the following restrictions arise due to ε events:

$$\begin{array}{rcl} h(q_2) & \neq & h(q_3) \\ h(q_2) & \neq & h(q_4) \\ h(q_2) & \neq & h(q_5) \end{array} \tag{7}$$

and the set of forward-events I_σ^f results in the following non-

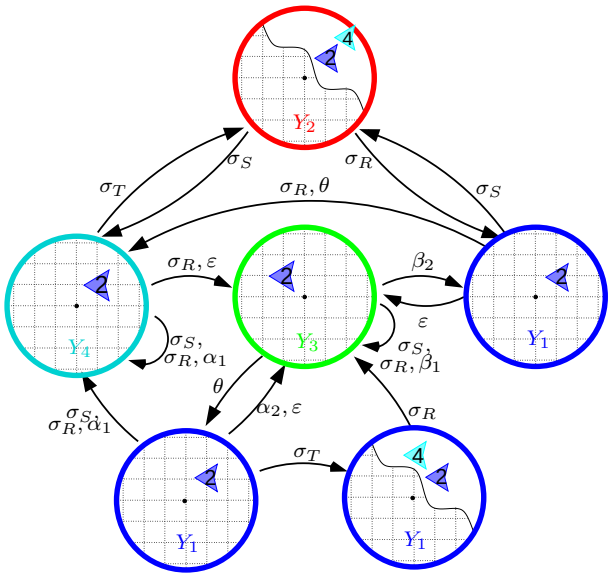

Fig. 5. The minimal output h for the minimal model G, which makes (G, h) immediately observable for Aircraft 2.

singular sets and resultant output constraints:

$$\begin{array}{rcll} I^f_{\sigma_R} & = & \{q_2, q_3, q_5\} & \Rightarrow \quad h(q_2) \neq h(q_3) \neq h(q_5) \\ I^f_{\sigma_S} & = & \{q_1, q_2, q_5\} & \Rightarrow \quad h(q_1) \neq h(q_2) \neq h(q_5) \\ I^f_{\sigma_T} & = & \{q_1, q_6\} & \Rightarrow \quad h(q_1) \neq h(q_6) \\ I^f_{\theta} & = & \{q_4, q_5\} & \Rightarrow \quad h(q_4) \neq h(q_5) \end{array} \tag{8}$$

The restrictions in equations (7) and (8) are indicated in the following table of allowable state pairs.

q_2	$\times$				
q_3	*	$\times$			
q_4	*	$\times$	*		
q_5	$\times$	$\times$	$\times$	$\times$	
q_6	$\times$	*	*	*	*
	q_1	q_2	q_3	q_4	q_5

(9)

Allowable pairs of the reduced model G are marked with (*), unallowable pairs are indicated with ($\times$). The maximum allowables are $\{(q_3, q_4, q_6), (q_5, q_6), (q_2, q_6), (q_1, q_3), (q_1, q_4)\}$. The elements of the binary vector $x = [x_1, x_2, x_3, x_4, x_5, x_6, x_{13}, x_{14}, x_{26}, x_{34}, x_{36}, x_{46}, x_{56}, x_{346}]$ represent each maximum allowable and its subsets. The element $x_{ij} = 1$ if the set containing q_i and q_j is selected, and $x_{ij} = 0$ if it is not. The covering constraints which constrain the minimization of $\mathbf{1}^T x$ are

$$\begin{array}{rcl} x_1 + x_{13} + x_{14} & = & 1 \\ x_2 + x_{26} & = & 1 \\ x_3 + x_{13} + x_{34} + x_{36} + x_{346} & = & 1 \\ x_4 + x_{14} + x_{34} + x_{46} + x_{346} & = & 1 \\ x_5 + x_{56} & = & 1 \\ x_6 + x_{26} + x_{36} + x_{46} + x_{56} + x_{346} & = & 1 \end{array} \tag{10}$$

A solution to integer program is $x^*_1 = 1, x^*_2 = 1, x^*_5 = 1, x^*_{346} = 1$ and all other elements of x^* equal to 0. This is shown in Figure 5, with $h(q_1) := Y_2, h(q_2) := Y_3, h(q_5) := Y_4$, and $h(q_3) = h(q_4) = h(q_6) := Y_1$. While there are many choices of outputs which would result in an immediately observable system, this one is minimal in that it results in an output of minimal cardinality $n^*_Y = 4$. The output broadcast to the aircraft is the minimum set of information from which the aircraft can reconstruct its current state (of the reduced model, G). Table I compares the quantities (as determined by set cardinality) of information which must be transmitted to Aircraft 2 using different model formulations. In addition, the minimal model in Figure 5 has a fewer number of issuing instructions (number of arcs) needed to change formation than the original model in Figure 4, i.e., the frequency of communication required is reduced. While there is a significant reduction in the amount of information about the state which must be transmitted, the cardinality of the event set is not necessarily reduced by this synthesis technique. This is an area of future work.

Model	State Information	Event Information	State Information Cardinality
G_{sys}	Q_{sys}	$\Sigma^{\text{obs}}_{\text{sys}}$	12
G	Y	Σ_o	4

TABLE I. Comparison of information cardinality in original model G_{sys} and reduced model (interface) G with synthesized output Y.

V. Conclusion

We have presented conditions for immediate observability, a property for systems (such as user-interfaces) in which *immediate* determination of the actual state of the system is paramount. Our conditions for immediate observability assume that the current state output as well as either the last-occurring or next-occurring event (or both) are known. For the case in which the last event and the current output are known, we used these conditions to synthesize a state-output map which guarantees immediate observability. This synthesis draws from techniques in state reduction, and makes use of integer optimization. We have applied this condition and output synthesis to the problem of user-interface design for remote systems, in which high communication costs inspire minimized information transfer. Each remote agent must be able to reconstruct the minimal set of information which makes up the agent's user-interface. We determined the minimal output map for which a specific aircraft, flying in a five-aircraft formation, will be able to reconstruct information necessary for the pilot of a specific aircraft to maintain proper formation. The state-related information which must be transmitted is significantly reduced through this synthesis.

VI. Acknowledgments

Research supported by NSF grant ECS-9985072, ONR MURI grant N00014-02-1-0720-P0001, NASA Ames Research Center grant NAG 2-1564, and grant NCC 2-798

from NASA Ames Research Center to the San Jose State University Foundation, as part of NASA's base research and technology effort, human-automation theory sub-element (RTOP 548-40-12). We would also like to acknowledge Asaf Degani and Michael Heymann for discussions regarding interface design.

VII. REFERENCES

[1] M. Heymann and A. Degani, "On abstractions and simplifications in the design of human-automation interfaces," NASA Technical Memorandum 211397, NASA Ames Research Center, Moffett Field, CA, 2002.

[2] M. Oishi, I. Mitchell, A. Bayen, C. Tomlin, and A. Degani, "Hybrid verification of an interface for an automatic landing," in *Proceedings of the IEEE Conference on Decision and Control*, (Las Vegas, NV), December 2002.

[3] M. Oishi, A. Degani, and C. Tomlin, "Verification of user-interfaces for hybrid systems," NASA Technical Memorandum, NASA Ames Research Center, Moffett Field, CA. Submitted September 2002.

[4] P. Ramadge, "Observability of discrete event systems," in *Proceedings of the IEEE Conference on Decision and Control*, (Athens, Greece), pp. 1108–1112, December 1986.

[5] C. Özveren and A. Willsky, "Observability of discrete event dynamic systems," *IEEE Transactions on Automatic Control*, vol. 35, pp. 797–830, July 1990.

[6] P. Caines, R. Greiner, and S. Wang, "Dynamical logic observers for finite automata," in *Proceedings of the IEEE Conference on Decision and Control*, (Austin, TX), pp. 226–233, December 1988.

[7] F. Lin and W. Wonham, "On observablity of discrete event systems," *Information Sciences*, vol. 44, pp. 173–198, 1988.

[8] M. Paull and S. Unger, "Minimizing the number of states in incompletely specified sequential switching functions," *IRE Transactions on Electronic Computers*, pp. 356–367, September 1959.

[9] S. Ginsburg, "A technique for the reduction of a given machine to a minimal-state machine," *IRE Transactions on Electronic Computers*, pp. 346–355, September 1959.

[10] Z. Kohavi, *Switching and Finite Automata Theory*. New York: McGraw-Hill, 1978.

[11] S. Unger, *Asynchronous Sequential Switching Circuits*. New York: Wiley-Interscience, 1969.

[12] A. Grasselli and F. Luccio, "A method for minimizing the number of internal states in incompletely specified sequential networks," *IEEE Transactions on Electronic Computers*, pp. 350–359, June 1965.

[13] T. Kam, T. Villa, R. Brayton, and A. Sangiovanni-Vincentelli, "Implicit computation of compatible sets for state minimization of ISFSM's," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, pp. 657–676, July 1997.

[14] J. Rho, G. Hachtel, F. Somenzi, and R. Jacoby, "Exact and heuristic algorithms for the minimization of incompletely specified state machines," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, pp. 167–177, February 1994.

[15] T. Kam, T. Villa, R. Brayton, and A. Sangiovanni-Vincentelli, "Theory and algorithms for state minimization of nondeterministic FSM's," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, pp. 1311–1322, November 1997.

[16] R. Puri and J. Gu, "An efficient algorithm to search for minimal closed covers in sequential machines," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 6, pp. 737–745, June 1993.

[17] J. Pena and A. Oliveira, "A new algorithm for exact reduction of incompletely specified finite state machines," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, pp. 1619–1632, November 1999.

[18] M. Damiani, "The state reduction of nondeterministic finite-state machines," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, pp. 1278–1291, November 1997.

[19] N. Sarter, D. Woods, and C. Billings, "Automation surprises," in *Handbook of Human Factors and Ergonomics*, pp. 1295–1327, NY: John Wiley and Sons, Inc., 1999.

[20] A. Degani, M. Heymann, G. Meyer, and M. Shafto, "Some formal aspects of human-automation interaction," NASA Technical Memorandum 209600, NASA Ames Research Center, Moffett Field, CA, April 2000.

[21] J. Crow, D. Javaux, and J. Rushby, "Models and mechanized methods that integrate human factors into automation design," in *International Conference on Human-Computer Interaction in Aeronautics*, (Toulouse, France), September 2000.

[22] N. Leveson and E. Palmer, "Designing automation to reduce operator errors," in *In the Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, (Orlando, FL), pp. 1144–1150, 1997.

[23] R. Butler, S. Miller, J. Potts, and V. Carreno, "A formal methods approach to the analysis of mode confusion," in *Proceedings of the AIAA/IEEE Digital Avionics Systems Conference*, pp. C41/1–C41/8, 1998.

[24] A. Schrijver, *Theory of Linear and Integer Programming*. New York: Wiley-Interscience, 1998.

[25] G. Nemhauser and L. Wolsey, *Integer and Combinatorial Optimization*. New York: John Wiley and Sons, Inc., 1988.

[26] D. Stipanovic, G. Inalhan, R. Teo, and C. Tomlin, "Decentralized overlapping control of a formation of unmanned aerial vehicles," in *Proceedings of the IEEE Conference on Decision and Control*, (Las Vegas, NV), December 2002.

[27] J. Shin, L. Guibas, and F. Zhao, "A distributed algorithm for managing multi-target identities in wireless ad-hoc sensor networks," in *International Workshop on Information Processing in Sensor Networks*, April 2003.