Invariance-Preserving Abstractions of Hybrid Systems: Application to User Interface Design

Meeko Oishi, *Member, IEEE*, Ian Mitchell, *Member, IEEE*, Alexandre M. Bayen, *Member, IEEE*, and Claire J. Tomlin, *Senior Member, IEEE*

***Abstract*—Hybrid systems combine discrete state dynamics which model mode switching, with continuous state dynamics which model physical processes. Hybrid systems can be controlled by affecting both their discrete mode logic and continuous dynamics: in many systems, such as commercial aircraft, these can be controlled both automatically and using manual control. A human interacting with a hybrid system is often presented, through information displays, with a simplified representation of the underlying system. This *user 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. In safety-critical systems, correct and succinct interfaces are paramount: interfaces must provide adequate information and must not confuse the user. We present an invariance-preserving abstraction which generates a discrete event system that can be used to analyze, verify, or design user-interfaces for hybrid human–automation systems. This abstraction is based on hybrid system reachability analysis, in which, through the use of a recently developed computational tool, we find controlled invariant regions satisfying *a priori* safety constraints for each mode, and the controller that must be applied on the boundaries of the computed sets to render the sets invariant. By assigning a discrete state to each computed invariant set, we create a discrete event system representation which reflects the safety properties of the hybrid system. This abstraction, along with the formulation of an interface model as a discrete event system, allows the use of discrete techniques for interface analysis, including existing interface verification and design methods. We apply the abstraction method to two examples: a car traveling through a yellow light at an intersection, and an aircraft autopilot in a landing/go-around maneuver.**

***Index Terms*—Autopilot, discrete abstraction, hybrid systems, pilot displays, reachability, user interface.**

Manuscript received November 28, 2005. Manuscript received in final form April 12, 2007. This work was supported in part by the National Science Foundation under a Graduate Research Fellowship, by DARPA under the Software Enabled Control Program (AFRL Contract F33615-99-C-3014), by the DoD Multidisciplinary University Research Initiative (MURI) Program administered by the Office of Naval Research under Grant N00014-00-1-0637, and by Grant NCC2-798 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).

M. Oishi is with the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC V6T 1Z4, Canada (e-mail: moishi@ece.ubc.ca).

I. M. Mitchell is with the Department of Computer Science, University of British Columbia, Vancouver, BC V6T 1Z4, Canada (e-mail: mitchell@cs.ubc.ca).

A. M. Bayen is with the Department of Civil and Environmental Engineering, University of California, Berkeley, CA 94720-1710 USA (e-mail: bayen@berkeley.edu).

C. J. Tomlin is with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720-1770 USA and also with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305-4035 USA (e-mail: tomlin@eecs.berkeley.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCST.2007.903370

I. Introduction

HUMAN–AUTOMATION interaction is pervasive, occurring in consumer products (alarm clocks, VCRs, cellular phones), transportation systems (automobiles, commercial aircraft, air traffic control), scientific research platforms (unmanned ocean- and aerial-vehicles), and military systems (fleets of semi-autonomous and autonomous aircraft), among others. Often complicated by the underlying dynamics of the physical system, human–automation interaction in aviation has been a controversial topic since the advent of computers and their integration into the cockpit [1]–[3]. The aviation industry has experienced many incidents and some accidents in which the pilot became confused about the current mode or could not anticipate the next mode in the automation [4]–[7]. This potentially dangerous problem has been loosely termed *mode confusion*, and is often addressed in flight when the pilot has the time to devote attention to it, and resolved later with *ad hoc* "fixes." However, mode confusion may occur at critical times of flight: In 1994, all seven people on-board died during a test flight of the A-330 in Toulouse, France [8], [9]. The pilot had attempted to complete a go-around with a simulated engine failure, but an unanticipated combination of aircraft and engine dynamics, flight envelope protection schemes, and confusing interface indications led to the aircraft's stall. The accident involved the aircraft's software, aerodynamics, as well as the pilot's interaction with the combined system.

We focus specifically on an aspect of this problem which we can quantify: the information content presented in the interface. While graphical design of the interface is key in determining how the user processes and interacts with information in the interface, we assume that the user can and does process all information displayed. In human–automation systems, the interface allows observation of information regarding the underlying system dynamics and processes, as well as control over specific behaviors through input devices in the interface. Too much information can overwhelm the user; with too little information the user may not understand the system's behavior or may not be able to perform the desired task. *A key part of the problem of interface design and verification involves the selection of appropriate information from the underlying human–automation system which should be displayed to the human controlling the system.*

Although the engineering psychology community has historically dominated research on human–automation interaction,

there have recently been efforts by the formal methods community [6], [10]–[13] as well as systems and control communities [14], [15] to address these safety-critical problems. Using model checkers, researchers in formal methods have evaluated such interfaces to identify design problems [10], [11], [16] for discrete state models. In [12], [17], and [18], the authors were not only able to verify interfaces for a given task, but additionally formally determine the minimum set of information that must be displayed in the cockpit interface in order to safely complete a given maneuver. We believe that the continuous dynamics plays a crucial role in understanding and designing interfaces, and that it is necessary to introduce both a continuous dynamic component to represent the physical dynamics of the underlying system, and a control component, into previously proposed methodologies to make them sound techniques for physical systems.

We model complex human–automation systems as hybrid systems over which a human shares control with automation. In this framework, which establishes a new way of modeling human interaction with automation, we determine how to abstract, from the hybrid system, a reduced representation of the underlying system. This representation can then be used in existing discrete interface verification or design algorithms. The particular representation which we construct addresses the problem of *system safety*, in which we consider a system to be *safe* if it fulfills a certain mathematical property which can be encoded as a condition on the system's reachable set of states. In aircraft, for example, a safe system is one which remains within its aerodynamic flight envelope. This contribution differs from existing work in hybrid system verification in our treatment of the user's interactions with the hybrid system.

One of the key enabling technologies for human–automation systems is *verification*, which allows for heightened confidence that the system will perform as desired. Verification is defined simply as the process of developing and executing a mathematical proof that a system model satisfies a given specification. Methods and tools to verify systems have become very important as the complexity of automated systems has grown; it is no longer possible to rely on intuition and simulation to test that a system satisfies its specification. Verification tools can aid in drastically reducing time spent on design and validation, but are also crucial in ensuring that safety properties are upheld. In safety-critical, expensive, or high-risk applications such as airbag deployment circuitry, aircraft autopilots, and medical devices, guarantees of safe operation are paramount.

Hybrid reachability analysis and controller synthesis address the problem of guaranteeing system *safety*. Many problems of interest may be posed as reachability specifications. For example, in the problem of verifying system safety, the safety specification is first represented as a desired subset of the state-space in which the system should remain, and then the following reachability question is posed—*Do all trajectories of the system remain within this set of desired states?* The process of verifying safety then involves computing the subset of the state-space which is backwards reachable from this "safe set" of states. If this backwards reachable set intersects any states outside the desired region, then the system is deemed unsafe. *Controller synthesis for safety* is a procedure through which one attempts, by restricting the behavior of the system through a controller, to prune away system trajectories which lead to unsafe states.

In the past several years, methods and tools have been developed for computing reachable sets for continuous and hybrid systems [19]–[23]. Many of these approximate the continuous dynamics and use an over-approximative set representation in order to maintain computational tractability for high dimensional systems. In this paper, we use a time-dependent Hamilton–Jacobi formulation [24] for specifying the reachable set, and the corresponding numerical toolbox based on level set methods [25]–[27] for computing the reachable set. This technique works for general nonlinear dynamics and set representation. Previous work in analyzing hybrid system safety, for example [28], has focused on applications of hybrid system theory to fully automated systems, assuming that the controller itself is an automaton. Here, we consider the problem of controlling *human–automation* systems, in which the automaton and a human controller share authority over the control of the system [7]. The user interacts with the underlying system through an *interface*, a reduced description of the behavior of the system. In particular, we consider the problem of verification of an interface between a semi-automated hybrid system and a human controller, and we pose the question—*Is the information displayed to the human controller about the hybrid system evolution sufficient for the human controller to act in such a way that the system remains safe?*

In previous work [15], we focused on one particular example in which verification that an interface correctly represented the underlying hybrid system was paramount: the automatic landing of a large civil jet aircraft. Here, we generalize the abstraction technique which allowed us to make use of discrete interface verification tools. We analyze the user-interface of a hybrid human–automation system through tools for hybrid system reachability analysis. These tools assume a hybrid model of the human–automation system (which includes how the user interacts with the system). We create a discrete representation of the hybrid model based on the hybrid system reachability result, from which we can then design an appropriate interface for the hybrid system. In cases in which we are given an interface which can be represented as a discrete system, the task, then, is to verify the safety of the discrete interface using the discrete representation of the hybrid human–automation system.

This paper is organized as follows. We first introduce the modeling formalism and describe the particular way in which we model human interaction with a hybrid system. We then describe a method to create the discrete invariance-preserving abstraction, and apply it to two examples, in which correct interface design requires information drawn from the underlying hybrid dynamics. We first present an everyday example: driving on an expressway through a yellow light, and show how, with this method, an interface can be constructed based on the reachability result of the underlying hybrid system. After discussing the abstraction method, we present a more complicated example: the automatic landing of a civil jet aircraft,

without the simplifications taken advantage of in [15]. Last, we offer some conclusions and directions for future work.

II. Problem Description

A. Problem Formulation

Consider a hybrid system $H = (Q, \mathcal{X}, R, f, \Sigma, \mathcal{U})$ defined by the following:

- set of discrete modes Q;
- domain of continuous states $\mathcal{X}$;
- transition function $R : Q \times \Sigma \times \mathcal{X} \to Q \times \mathcal{X}$;
- functions $f_q(x, u) : Q \times \mathcal{X} \times \mathcal{U} \to \mathcal{X}$;
- set of discrete inputs or events $\Sigma = \Sigma_u \cup \Sigma_h \cup \Sigma_d$;
- set of continuous inputs $\mathcal{U}$.

We assume that discrete controlled inputs $\sigma_u \in \Sigma_u$ are initiated by an automated controller (e.g., an automatic transition based on the continuous state), discrete human-initiated events $\sigma_h \in \Sigma_h$ are initiated by the human (e.g., pushing a button, toggling a lever), and discrete disturbance inputs $\sigma_d \in \Sigma_d$ are initiated by something external to the system (e.g., a fault, caused neither by the automatic controller nor the human). We index the continuous dynamics by the current discrete mode, so that $\dot{x} = f_q(x, u)$, with $x \in \mathcal{X} \subseteq \mathbb{R}^n$ and $u \in \mathcal{U} \subseteq \mathbb{R}^m$ in mode $q \in Q$. We assume that the continuous input is controlled automatically. While we anticipate that the method presented in this paper can be extended to treat systems with continuous disturbances or continuous human inputs, these issues are not addressed here. Additionally, the initial set is $(Q_0, \mathcal{X}_0)$ and the constraint set is $\mathcal{C} = (Q, \mathcal{W}_0) \subseteq Q \times \mathcal{X}$. The constraint set represents the desired subset of the state-space in which the system should remain. We index the continuous constraint set in mode q by $\mathcal{W}_0(q)$, i.e., $(q, \mathcal{W}_0(q)) \subseteq \mathcal{C}$.

Let us also consider a discrete event system $\hat{G} = (\hat{Q}, \hat{\Sigma}, \hat{R})$, defined by the following:

- set of discrete modes $\hat{Q}$;
- transition relation $\hat{R} : \hat{Q} \times \hat{\Sigma} \to 2^{\hat{Q}}$;
- set of events $\hat{\Sigma} = \hat{\Sigma}_u \cup \hat{\Sigma}_h \cup \hat{\Sigma}_d$.

The event set contains controlled, human-initiated, and disturbance events. The set of initial modes is $\hat{Q}_0$ and the constraint set is $\hat{\mathcal{C}} \subseteq \hat{Q}$.

Definition 1: A hybrid system H, defined as before, for which $\Sigma_h \neq \emptyset$, is called a *hybrid human–automation system*.

Definition 2: For a trajectory to be *invariant* with respect to a constraint set $\mathcal{C}$, it must begin within, and always remain within $\mathcal{C}$.

Definition 3: For a trajectory to be *user-invariant* with respect to a constraint set $\mathcal{C}$, it must begin within $\mathcal{C}$, and may exit $\mathcal{C}$ *only* under human inputs.

Definition 4: A discrete event system $\hat{G}$ is *invariance-preserving* with respect to the constraint set $\mathcal{C}$ if it accepts only trajectories of $\hat{G}$ that are invariant or user-invariant.

Definition 5: An *invariance-preserving abstraction* $\hat{G}$ is a discrete event system representation of a hybrid system H with constraint set $\mathcal{C}$, for which discrete event system $\hat{G}$ is invariance-preserving.

1) Problem 1: Given a hybrid human–automation system H, and a constraint set $\mathcal{C}$, find an invariance-preserving abstraction $\hat{G}$ of H.

The invariance-preserving abstraction $\hat{G}$ can be compared to an existing discrete interface through a discrete verification procedure; or it can be used to synthesize a discrete interface $\hat{G}_{\text{interface}}$, a reduced version of $\hat{G}$ which a user can use to safely interact with the underlying system H. The majority of work on abstractions (see, for example, [29]–[35]) makes use of discrete abstractions to aid in continuous or hybrid verification, often by exploiting certain properties of the system (for example, polynomial or bounded continuous dynamics). The *invariance-preserving abstraction* we propose here can take place after reachability analysis has been used to determine sets of initial conditions which produce invariant trajectories. Rather than aiding in the verification process, this is a post-processing step in which a discrete automaton is formed based on the verified system.

In our formulation of a hybrid human–automation system, human-initiated input occurs through discrete inputs. The human *may* influence the hybrid system trajectory, but does not have to. For example, consider $(q_{k+1}, x) = R(q_k, x, \sigma_h)$, in which the human may initiate a transition σ_h which switches the mode from q_k to q_{k+1}. The human controls not only *when*, but *if* the transition occurs at all. If the human does not initiate σ_h, then the system will remain in q_k unless other state-based or disturbance transitions exist which will allow the system to exit mode q_k. This nicely models applications such as flight management systems, in which the pilot may initiate high-level mode-changes and the flight management system maintains low-level control.

B. Illustration: Advisory System for the Yellow Interval Dilemma

Consider the following scenario. As a single driver on an expressway approaches an intersection, suddenly the light turns yellow. The driver must decide whether to brake and try to stop at the intersection or to continue driving through the intersection before the red light appears. In either case, the driver must avoid a *red light violation*, which occurs whenever the car is in the intersection at any time during a red light. Typically, due to several factors, including accumulated experience about the particular car's braking capabilities, the duration of the yellow light at a given intersection, and the road and weather conditions, the driver has an "intuitive" feel for the correct course of action [36]. At some intersections, despite the driver's best intentions, there are certain combinations of speed and distance from the intersection for which a red light violation is inevitable: this is known as the yellow interval dilemma [37], [38]. We wish to design an add-on advisory system, whose interface would indicate to the driver which action must be taken in order to avoid a red light violation.

We assume a point-mass model of the car without the additional complexity of gearing [39]. With position x from the near-side of the intersection and speed $v = \dot{x}$, braking and acceleration forces enter through $\ddot{x} = u$. The car has limited braking and acceleration capabilities ($u \in [u_{\min}, u_{\max}]$) and

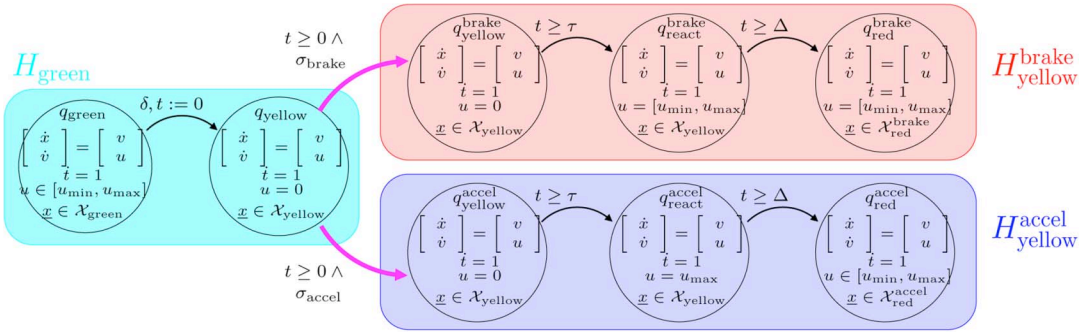

Fig. 1. Hybrid model H_{car} for a car traveling through an intersection, where the input u is the car's acceleration, and $\underline{x} := [x, v, t]$.

must obey the speed limit at all times ($v < v_{\max}$). For this scenario, $u_{\min} = -4$ m/s^2, $u_{\max} = 2$ m/s^2, and $v_{\max} = 24$ m/s. The intersection is 10 m in length and the duration of the yellow light is $\Delta = 4$ s [37], [40]. A typical driver's reaction time after seeing the yellow light is $\tau = 1.5$ s [36], [40], [41], during which time we presume the car travels at its present speed. The advisory system should be designed with the assumption that the driver will take one of two courses of action when a yellow light appears: 1) accelerate through the intersection (without violating the speed limit) or 2) brake to stop at the intersection. The question we wish to answer is: What actions does the driver need to take and when should the driver enact them in order to avoid a red light violation?

Consider the hybrid model H_{car} with continuous state $\underline{x} = [x, v, t]^T$, shown in Fig. 1, in which it follows that:

- $Q_{\text{car}} = \{q_{\text{green}}, q_{\text{yellow}}, q^{\text{brake}}_{\text{yellow}}, q^{\text{brake}}_{\text{react}}, q^{\text{accel}}_{\text{yellow}}, q^{\text{accel}}_{\text{react}}, q^{\text{brake}}_{\text{red}}, q^{\text{accel}}_{\text{red}}\}$
- $\mathcal{X}_{\text{car}} = \mathbb{R} \times \mathbb{R}^+_0 \times \mathbb{R}^+_0$;
- $\mathcal{U}_{\text{car}} = [u_{\min}, u_{\max}]$;
- $\Sigma_{\text{car}} = \Sigma_u \cup \Sigma_h \cup \Sigma_d$, with controlled events $\Sigma_u = \{t \geq 0, t - \tau \geq 0, t - \Delta \geq 0\}$ (which are triggered when the inequality is met), human-initiated events $\Sigma_h = \{\sigma_{\text{brake}}, \sigma_{\text{accel}}\}$, and disturbance event $\Sigma_d = \{\delta\}$.

The event δ represents the light turning yellow. The transition function R_{car} and the continuous dynamics f_{car} are defined as shown in Fig. 1. The initial set is $(q_{\text{green}}, \mathcal{X}_{\text{green}})$, and the constraint set is $\mathcal{C}_{\text{car}} = Q_{\text{car}} \times (\mathcal{W}_{\text{car}})_0$, $(\mathcal{W}_{\text{car}})_0 = \{\mathcal{X}_{\text{green}}, \mathcal{X}_{\text{yellow}}, \mathcal{X}^{\text{accel}}_{\text{red}}, \mathcal{X}^{\text{brake}}_{\text{red}}\}$, with $\mathcal{X}_{\text{green}} = \{\mathbb{R} \times (0, v_{\max}] \times \mathbb{R}^+_0\}$, $\mathcal{X}_{\text{yellow}} = \{\mathbb{R} \times (0, v_{\max}] \times \mathbb{R}^+_0\} \cup \{0 \times 0 \times \mathbb{R}^+_0\}$, $\mathcal{X}^{\text{brake}}_{\text{red}} = \{(-\infty, 0] \times (0, v_{\max}] \times \mathbb{R}^+_0\}$, and $\mathcal{X}^{\text{accel}}_{\text{red}} = \{[L, \infty) \times (0, v_{\max}] \times \mathbb{R}^+_0\}$. The constraint set encapsulates the desired regions of operation in terms of the car's position relative to the intersection and the car's speed: during the red interval, the constraint set is any position *outside* the intersection and any positive speed, plus the position right at the boundary of the intersection with a speed of 0; during the green and yellow intervals the constraint set is any position and positive speed.

1) Problem 2: Find an invariance-preserving abstraction $\hat{G}_{\text{car}}$ of the hybrid human–automation system H_{car}, given the constraint set $\mathcal{C}_{\text{car}}$.

III. Creating an Invariance-Preserving Abstraction

In this section, we propose a three-step algorithm to construct an invariance-preserving abstraction $\hat{G}$ of a hybrid human–automation system H. Each of the three steps in Algorithm 1 will be detailed in the following sections, followed by a proof that the resultant discrete event system $\hat{G}$ is an invariance-preserving abstraction of the hybrid human–automation system H.

Definition 6: For every mode $q \in Q$ in the hybrid human–automation system H, define the set of modes $\text{HumanReach}(q)$ as the set of all modes which could result after any human-controlled transition from q

$$\text{HumanReach}(q) \triangleq \{p | (p, x) \in R(q, x, \sigma_h), \text{ for all } \sigma_h \in \Sigma_h \text{ such that } R(q, x, \sigma_h) \text{ exists}\}. \quad (1)$$

Definition 7: Define the set of *entry modes* $Q_{\text{entry}} = \{\text{HumanReach}(q), q \in Q\} \cup Q_0$ as the union of those modes which result after any discrete human input and those modes in the initial set. Define the cardinality of the entry modes as $N \triangleq \|Q_{\text{entry}}\|$.

Algorithm 1: To create an invariance-preserving discrete abstraction $\hat{G}$ of a hybrid human–automation system H, the following must be done.

1) Separate the hybrid human–automation system H into N hybrid subsystems H_i, using the $\text{HumanReach}(\cdot)$ operator to delineate subsystems which contain no human input.
2) For $i = 1 : N$, compute the invariance-preserving abstraction $\hat{G}_i$ of hybrid subsystem H_i.
3) Combine discrete event systems $\hat{G}_i$ into one discrete event system G.

A. Step 1: Separation Into Hybrid Subsystems

In many human–automation systems, it is important to limit how a human's actions will be prescribed. We therefore wish to minimize the restrictions placed on the human input through our analysis. We begin our analysis of the human–automation system H by first characterizing regions of H in which human

input is impossible. The first step is to decompose H into subsystems which contain *no* discrete human inputs. As noted earlier, it is impossible to guarantee that a trajectory will be invariant when the system contains human-initiated inputs. However, we can determine regions of the state-space for which invariant trajectories are possible because no human-initiated inputs exist. We accomplish this by isolating modes forward-reachable after a human-initiated transition has occurred, and defining those modes as a subsystem of the original system H. Thus, within each subsystem, all of the transitions are controlled or disturbance transitions.

Definition 8: Given an entry mode $q_i \in Q_{\text{entry}}$, the hybrid system $H = (Q, \mathcal{X}, R, f, \Sigma, \mathcal{U})$, and a constraint set $\mathcal{C} = (Q, \mathcal{W}_0)$, define the *hybrid subsystem* $H_i = (Q_i, \mathcal{X}_i, R_i, f_i, \Sigma_i, \mathcal{U}_i)$ as follows:

- Q_i is the set of modes reachable from q_i through any string of automatic and disturbance events (that is, any string which does *not* contain a user-initiated event);
- $R_i(q,x,\sigma) = \begin{cases} R(q,x,\sigma), & \text{if } \sigma \notin \Sigma_h \text{ and } q \in Q_i \\ \{\emptyset\}, & \text{otherwise} \end{cases}$
- $\Sigma_i = \{\sigma | R_i(q,x,\sigma) \text{ exists}, q \in Q_i\}$;

and $\mathcal{X}_i$, f_i, $\mathcal{U}_i$ are subsets of $\mathcal{X}$, f, $\mathcal{U}$ which correspond to the modes in $Q_i \subseteq Q$. Similarly, the constraint set $\mathcal{C}_i \triangleq (Q_i, (\mathcal{W}_i)_0)$ is a subset of $\mathcal{C}$ which corresponds to the modes $Q_i \subseteq Q$.

For ease of notation, we label the entry modes $q_1, \ldots, q_N$. These indices also label the hybrid subsystems, $H_1, \ldots, H_N$. The following indexing function associates each mode $q \in Q$ to the hybrid subsystems it is contained in

$$\text{Ind}(q) \triangleq \{i | q \in Q_i\}, \text{ where } i \in \{1, \ldots, N\}. \tag{2}$$

Note that in general, hybrid human–automation systems will separate into hybrid subsystems which could overlap.

Yellow Interval Example: For the hybrid human–automation system H_{car}, applying Definition 7 results in entry modes $Q_{\text{entry}} = \left\{q_{\text{green}}, q^{\text{brake}}_{\text{yellow}}, q^{\text{accel}}_{\text{yellow}}\right\}$, since $\text{HumanReach}(q_{\text{yellow}}) = \left\{q^{\text{brake}}_{\text{yellow}}, q^{\text{accel}}_{\text{yellow}}\right\}$ and $Q_0 = \{q_{\text{green}}\}$. We then apply Definition 8 to find that H_{car} separates into three hybrid subsystems: H_{green}, $H^{\text{brake}}_{\text{yellow}}$, and $H^{\text{accel}}_{\text{yellow}}$, as shown in Fig. 1. For example, H_{green} contains $Q_{\text{green}} = \{q_{\text{green}}, q_{\text{yellow}}\}$, $\Sigma_{\text{green}} = \{\delta\}$, and R_{green} is graphically depicted in the left-most shaded region of Fig. 1. The constraint set for H_{green} is $\mathcal{C}_{\text{green}} = \{(q_{\text{green}}, (\mathcal{W}_{\text{green}})_0), (q_{\text{yellow}}, (\mathcal{W}_{\text{yellow}})_0)\}$. The index function Ind maps each mode in Q_{car} to either Q_{green}, $Q^{\text{brake}}_{\text{yellow}}$, or $Q^{\text{accel}}_{\text{yellow}}$: for example, $\text{Ind}(q_{\text{yellow}}) = \text{green}$, since $q_{\text{yellow}} \subseteq Q_{\text{green}}$. ■

B. Step 2: Invariance-Preserving Discrete Abstraction

In Step 2 of Algorithm 1, we create an invariance-preserving abstraction of each hybrid subsystem identified in Step 1. Since the hybrid subsystems do not contain any discrete human inputs, standard reachability tools can be used to find invariant sets within each subsystem.

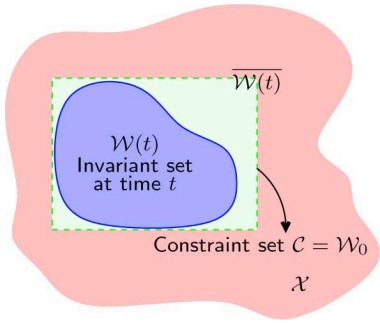

Fig. 2. Constraints on the state-space $\mathcal{X}$ are encapsulated in the constraint set $\mathcal{C} = \mathcal{W}_0$. Applying the reachability computation for t seconds, we obtain the invariant set $\mathcal{W}(t) = \phi(\mathcal{C})$. Trajectories whose initial conditions are contained within $\mathcal{W}(t)$ will remain within $\mathcal{C}$ for t seconds.

Hybrid reachability analysis and controller synthesis provides us with a mathematical guarantee of invariance to within the limits of the hybrid model, through the generation of: 1) the invariant set of states, $(Q, \mathcal{W}_i) \subseteq \mathcal{C}_i$ contained within the constraint set $\mathcal{C}_i$ and 2) a control law which ensures that trajectories which reach the boundary of the invariant set will not be allowed to exit the invariant set (see Fig. 2). For the reachability computations in Problem 1, any tool can be used which can compute the backwards-reachable invariant set for each hybrid subsystem. This flexibility allows for selection of the hybrid reachability tool most appropriate for the particular system at hand [42]–[47]. We use a tool based on level-set methods [26], [27] that computes not only the reachable set, but also the control law necessary to enforce invariance along the boundary of the reachable set. This tool accommodates hybrid systems with general nonlinear continuous dynamics, continuous inputs and disturbances, and discrete inputs and disturbances.

Definition 9: An *invariant set* $\mathcal{W} \subseteq \mathcal{X}$ has the property that all trajectories with initial state $x_0 \in \mathcal{W}$ are invariant with respect to $\mathcal{W}$.

Definition 10: Given a hybrid system H_i with $\Sigma_h = \{\emptyset\}$ (no discrete human input), and a constraint set $\mathcal{C}_i = (Q_i, (\mathcal{W}_i)_0)$, define $\phi : Q_i \times (\mathcal{W}_i)_0 \to Q_i \times (\mathcal{W}_i)_0$ as the result of computing, for t seconds, the invariant set $(Q_i, \mathcal{W}_i)$ by means of a backwards-reachable, over- or convergent-approximative numerical tool: $\phi(\mathcal{C}_i) = (Q_i, \mathcal{W}_i) \subseteq \mathcal{C}_i$.

Yellow Interval Example: We compute the invariant sets for each of the constraint sets in $[x, v]$: $(Q_{\text{green}}, \mathcal{W}) = \phi(\mathcal{C}_{\text{green}})$, $\left(Q^{\text{brake}}_{\text{yellow}}, \mathcal{B}\right) = \phi\left(\mathcal{C}^{\text{brake}}_{\text{yellow}}\right)$, and $\left(Q^{\text{brake}}_{\text{yellow}}, \mathcal{A}\right) = \phi\left(\mathcal{C}^{\text{accel}}_{\text{yellow}}\right)$. In each computed invariant set, we denote the discrete and continuous components as set of pairs of (mode, continuous set): $\mathcal{W} = \{(q_{\text{green}}, \mathcal{W}(q_{\text{green}})), (q_{\text{yellow}}, \mathcal{W}(q_{\text{yellow}})\}$, for example. The reachability computation reveals that $\mathcal{W}(q_{\text{green}}) = \mathcal{W}(q_{\text{yellow}}) = \mathcal{X}_{\text{green}}$. In the braking subsystem $H^{\text{brake}}_{\text{yellow}}$, the invariant region in each of the three modes is: $\mathcal{B}\left(q^{\text{brake}}_{\text{yellow}}\right) = \mathcal{B}\left(q^{\text{brake}}_{\text{react}}\right) = \mathcal{B}\left(q^{\text{brake}}_{\text{red}}\right) = \{1a\}$, the shaded region shown in Fig. 3. In the acceleration subsystem $H^{\text{accel}}_{\text{yellow}}$, the invariant set $\mathcal{A}\left(q^{\text{accel}}_{\text{yellow}}\right)$ is represented by the three shaded regions, labeled $\{2a, 2b, 2c\}$ in Fig. 4. The invariant

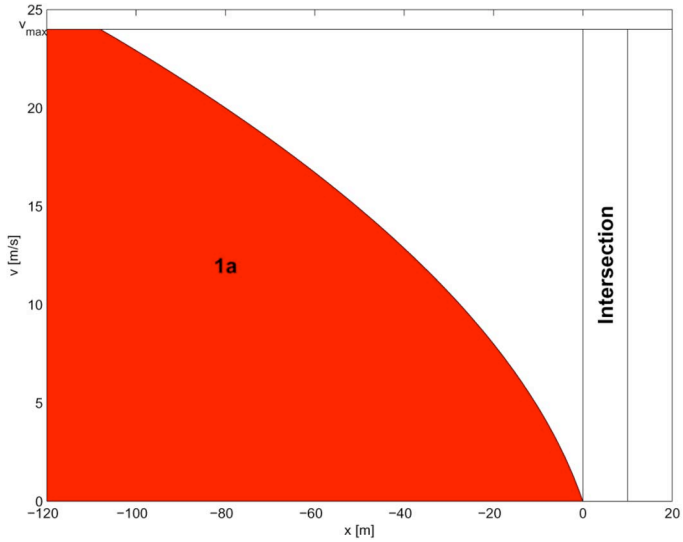

Fig. 3. Invariant region $\mathcal{W}_{\text{yellow}}^{\text{brake}}$ in H_{car} in $[x, v]$.

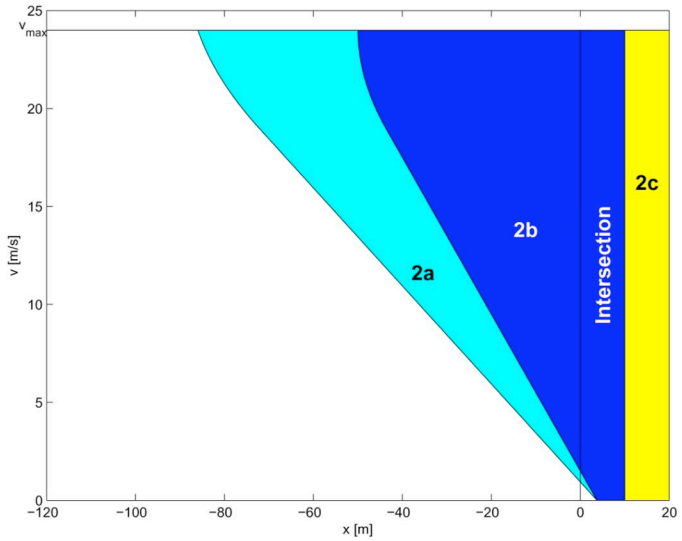

Fig. 4. Invariant set $\mathcal{A}\left(q_{\text{yellow}}^{\text{accel}}\right) \subseteq \phi\left(\mathcal{C}_{\text{yellow}}^{\text{accel}}\right)$ in $[x, v]$.

set $\mathcal{A}\left(q_{\text{react}}^{\text{accel}}\right)$ is represented by $\{2b, 2c\}$; and the invariant set $\mathcal{A}\left(q_{\text{red}}^{\text{accel}}\right)$ is represented by $\{2c\}$. ■

Proposition 1: Given a hybrid subsystem H_i with constraint set C_i, all trajectories whose initial state is contained in the invariant set $(Q_i, \mathcal{W}_i) \subseteq \mathcal{C}_i$ are user-invariant over time period t.

Proof: Proof by contradiction; see Appendix I-A for details. ■

The only way in which a trajectory that begins in $\mathcal{C}_i$ can exit $\mathcal{C}_i$ is through a human-initiated discrete event: a human-initiated event may transition the system into a state *outside* of the invariant set in the new mode, such that for $(p, x) \in \mathcal{C}_i$, $R(p, x, \sigma_h) \not\subseteq \mathcal{C}_j$. Denote the complement of a continuous set $\mathcal{W}_i(p)$ as $\overline{\mathcal{W}_i(p)}$. Additionally, we presume that trajectories which begin *outside* the invariant set in one mode $(p, x) \in (p, \overline{\mathcal{W}_i(p)})$ evolve to states outside the invariant set in the next mode $(q, x') \in (q, \overline{\mathcal{W}_i(q)})$ for controlled or disturbance events in the hybrid system such that $(q, x') = R(p, x, \sigma)$, $\sigma \in \Sigma_u \cup \Sigma_d$.

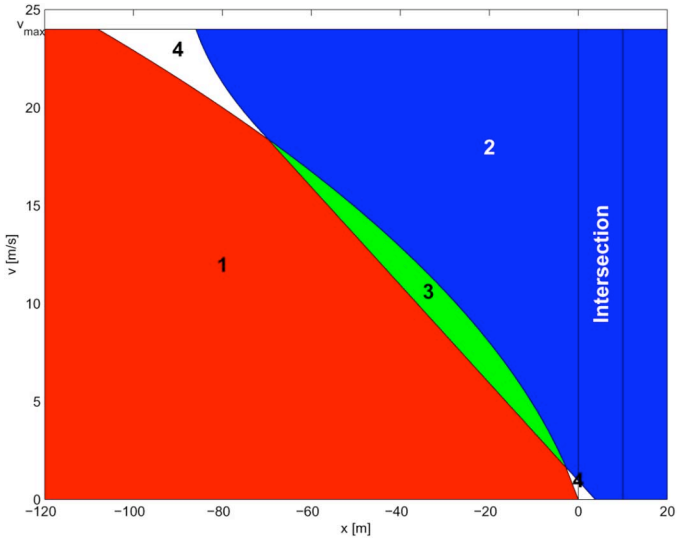

Fig. 5. Intersection of invariant sets $(q_{\text{yellow}}, \mathcal{W})$, $\left(q_{\text{yellow}}^{\text{brake}}, \mathcal{B}\right)$, and $\left(q_{\text{yellow}}^{\text{accel}}, \mathcal{A}_{\text{yellow}}^{\text{accel}}\right)$.

Yellow Interval Example: For any trajectory which begins in the shaded region of Fig. 3, the car will be able to avoid a red light violation by coming to a complete stop before the intersection. For any trajectory which begins in the shaded regions of Fig. 4, the car will be able to pass completely through the intersection before the red light occurs. In both cases, computed control laws to enforce invariance must be applied along the boundaries of the shaded sets. ■

Definition 11: Given a set of k sets $\mathcal{K} = \{\mathcal{W}_1, \ldots, \mathcal{W}_k\}$, $\mathcal{W}_i \subseteq \mathcal{X}$, define a map $\psi : \mathcal{K} \to \hat{Q}$ from the set of k regions in the continuous state-space to the discrete state-space, based on a partition of the continuous domain $\mathcal{X}$. The partition divides $\mathcal{X}$ into 2^k disjoint regions $\{\mathcal{W}_1 \cap \mathcal{W}_2 \cap \cdots \cap \mathcal{W}_k, \overline{\mathcal{W}_1} \cap \mathcal{W}_2 \cap \cdots \cap \mathcal{W}_k, \mathcal{W}_1 \cap \overline{\mathcal{W}_2} \cap \cdots \cap \mathcal{W}_k, \ldots, \overline{\mathcal{W}_1} \cap \overline{\mathcal{W}_2} \cap \cdots \overline{\mathcal{W}_k} \subseteq \mathcal{X}\}$ according to the *intersection* of the sets. Define the set of discrete modes, correspondingly: $\{w_1 w_2 \cdots w_k, \overline{w}_1 w_2 \cdots w_k, w_1 \overline{w}_2 \cdots w_k, \ldots, \overline{w}_1 \overline{w}_2 \cdots \overline{w}_k\}$, so that, for example

$$x \in (\mathcal{W}_1 \cap \overline{\mathcal{W}_2} \cap \cdots \cap \mathcal{W}_k) \xrightarrow{\psi} w_1 \bar{w}_2 \cdots w_k \ni \hat{Q} \quad (3)$$

and more generally, $\hat{Q} = \psi(\mathcal{K})$. The labeling convention of the discrete modes reflects the partitioning of the continuous state-space.

Definition 11 therefore defines a discrete state-space in which the system being in a particular discrete state corresponds precisely to the continuous state being in the corresponding cell of the state-space partition, according to (3).

Yellow Interval Example: Consider the intersection of three invariant sets, $\mathcal{K}_{\text{yellow}} \triangleq \left\{\mathcal{W}(q_{\text{yellow}}), \mathcal{B}\left(q_{\text{yellow}}^{\text{brake}}\right), \mathcal{A}\left(q_{\text{yellow}}^{\text{accel}}\right)\right\}$, representing the mode from which a user-initiated transition is possible, and the two resulting modes. Since $\mathcal{W}(q_{\text{yellow}}) = \mathcal{X}_{\text{green}}$, we construct the intersection of the remaining two sets, producing $2^2 = 4$ disjoint regions in $\mathcal{X}_{\text{green}}$. In Fig. 5, region $\{1\}$ represents $\mathcal{B}\left(q_{\text{yellow}}^{\text{brake}}\right) \cap \overline{\mathcal{A}\left(q_{\text{yellow}}^{\text{accel}}\right)}$; region $\{2\}$

represents $\overline{\mathcal{B}\left(q^{\text{brake}}_{\text{yellow}}\right)} \cap \mathcal{A}\left(q^{\text{accel}}_{\text{yellow}}\right)$; region $\{3\}$ represents $\mathcal{B}\left(q^{\text{brake}}_{\text{yellow}}\right) \cap \overline{\mathcal{A}\left(q^{\text{accel}}_{\text{yellow}}\right)}$; and region $\{4\}$ represents $\overline{\mathcal{B}\left(q^{\text{brake}}_{\text{yellow}}\right)} \cap \overline{\mathcal{A}\left(q^{\text{accel}}_{\text{yellow}}\right)}$. We abstract these four regions to the discrete modes: $wb_1\overline{a_1}$, $w\overline{b_1}a_1$, wb_1a_1, and $w\overline{b_1}\overline{a_1}$, respectively. Let $\hat{Q}_{\text{yellow}} = \{wb_1\overline{a_1}, w\overline{b_1}a_1, wb_1a_1, w\overline{b_1}\overline{a_1}\}$, where $\hat{Q}_{\text{yellow}} = \psi(\mathcal{K}_{\text{yellow}})$. The result is a set of discrete modes which represent the intersection of a set of continuous invariant sets. ■

Definition 12: Define a map $\text{Sel}(\hat{Q}, \mathcal{W}_i)$ which selects, for $\mathcal{W}_i \in \mathcal{K}$, certain modes in $\hat{Q} = \psi(\mathcal{K})$, created from a partition as defined in Definition 11

$$\text{Sel}(\hat{Q}, \mathcal{W}_i)) \triangleq \{p \in \hat{Q} = \psi(\mathcal{K}) \text{ for which } x \in \mathcal{W}_i\}. \quad (4)$$

The result is modes in $\hat{Q}$ which represent continuous states for which $x \in \mathcal{W}_i$.

Yellow Interval Example: Using the map previously defined, consider the case in which the driver should brake to a stop. We want to select the discrete modes in $\hat{Q}_{\text{yellow}}$ which represent the continuous states from which this is possible: $\text{Sel}\left(\hat{Q}_{\text{yellow}}, \mathcal{B}\left(q^{\text{brake}}_{\text{yellow}}\right)\right) = \{wb_1a_1, wb_1\overline{a}_1\}$. We now have a discrete representation of the hybrid states $(q_{\text{yellow}}, \underline{x})$ from which a safe braking maneuver is possible. ■

These definitions are key in creating a discrete abstraction of a hybrid mode. However, beyond constructing the modes of the discrete abstraction (as shown before), we must also construct the discrete transition relation in a way which mimics the behavior of the original hybrid system. Both of these points are addressed in the following algorithm. In the first step, discrete modes are constructed according to the partition defined in Definition 11. The remaining steps define the discrete transition relation for: 1) controlled or disturbance events; 2) human-initiated events; and 3) internal transitions which arise from the continuous dynamics of the original hybrid systems.

Algorithm 2: Given a hybrid subsystem H_i of the hybrid human–automation system H, a constraint set $\mathcal{C}_i$, and an invariant set $(Q_i, \mathcal{W}_i) = \phi(\mathcal{C}_i)$, create a discrete event system $\hat{G}_i$ as follows.

1) For each mode $p \in Q_i$, uniquely map the continuous state-space (partitioned into 2^K cells, where $K = \|\text{Ind}(\text{HumanReach}(p))\|$ is the number of distinct modes possible after any human-initiated transition from mode p) to a set of discrete states $\hat{Q}_p$ corresponding to the hybrid state (p, x)

$$\hat{Q}_p = \psi(\{\mathcal{W}_i(p), \mathcal{W}_k(\text{HumanReach}(p))\}) \quad k \in \text{Ind}(\text{HumanReach}(p)). \quad (5)$$

Here, since H_i has no discrete human input, the only states $p \in Q_i$ for which $\text{HumanReach}(p) \neq \emptyset$ are the "exit" modes from Q_i.

2) Define the relation $\hat{R}_i(\hat{p}, \sigma)$ for each $\hat{p} \in \hat{Q}_p$, corresponding to the hybrid subsystem transition $q = R_i(p, x, \sigma)$, for $p, q \in Q_i, x \in \mathcal{X}, \sigma \in \Sigma_i$ (i.e., for those controlled or disturbance transitions to modes within the hybrid subsystem H_p). For ease of notation, denote $\mathcal{L} \triangleq \mathcal{W}_i(p)$ the invariant set $\mathcal{W}_i$ in mode p, $\mathcal{M} \triangleq \mathcal{W}_i(q)$ the invariant set $\mathcal{W}_i$ in mode q, $\hat{P} \triangleq \hat{Q}_p$ the set of discrete modes which represent the hybrid state (p, x), and $\hat{Q} \triangleq \hat{Q}_q$ the set of discrete modes which represent the hybrid state (q, x)

$$\hat{R}_i(\hat{p}, \sigma) \in \begin{cases} \text{Sel}(\hat{Q}, \mathcal{M}), & \text{for } \hat{p} \in \text{Sel}(\hat{P}, \mathcal{L}) \\ \text{Sel}(\hat{Q}, \overline{\mathcal{M}}), & \text{for } \hat{p} \in \text{Sel}(\hat{P}, \overline{\mathcal{L}}). \end{cases} \quad (6)$$

Because the reachability analysis of Section II ensures that for controlled and disturbance events in the hybrid system H_i, trajectories that begin in the invariant set in mode p evolve to the invariant set in the following mode q, in Step 2, the discrete transition relation $\hat{R}_i$ is defined to reflect this behavior for controlled and disturbance events. See Fig. 6 for clarification.

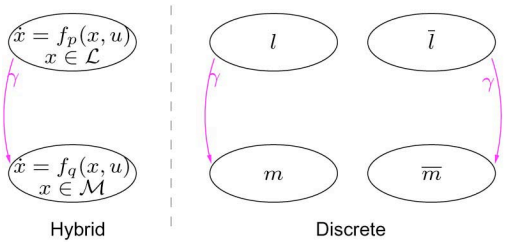

Fig. 6. Discrete abstraction of two-mode hybrid automaton with one *controlled* discrete input γ. Both hybrid modes belong to the same hybrid subsystem H_i, therefore, $p, q \in Q_i$. The invariant set is $\phi(\mathcal{C}_i) = \{(p, \mathcal{L}), (q, \mathcal{M})\}$. In the corresponding discrete abstraction, $\hat{P} = \{l, \bar{l}\}$ and $\hat{Q} = \{m, \overline{m}\}$.

3) Define the relation $\hat{R}_i(\hat{p}, \sigma_h)$ for each $\hat{p} \in \hat{Q}_p$, corresponding to any hybrid subsystem transitions to modes $q \in \text{HumanReach}(p) \neq \emptyset$, for $p \in Q_i, q \in Q_k, k \in \text{Ind}(\text{HumanReach}(p))$, (i.e., for those human-initiated transitions to modes in other hybrid subsystems). For ease of notation, denote $\mathcal{N} \triangleq \mathcal{W}_k(q)$ the invariant set $\mathcal{W}_k$ in mode q

$$\hat{R}(\hat{p}, \sigma_h) \in \begin{cases} \text{Sel}(\hat{Q}, \mathcal{N}), & \text{for } \hat{p} \in \text{Sel}(\hat{P}, \mathcal{N}) \\ \text{Sel}(\hat{Q}, \overline{\mathcal{N}}), & \text{for } \hat{p} \in \text{Sel}(\hat{P}, \overline{\mathcal{N}}). \end{cases} \quad (7)$$

In Step 3, the discrete transition relation $\hat{R}_i$ is defined such that for human-initiated events, trajectories which begin in the invariant set $\phi(C_i)$ in mode p may *or* may not evolve to the invariant set in the following mode q. This reflects the user's prerogative to initiate a transition—whether or not the system will be in the invariant set in the next mode depends on the region of the state-space from which the user enacts the switch. See Fig. 7 for clarification.

4) Define the relations $\hat{R}(\hat{p}, \alpha_q)$ and $\hat{R}(\hat{q}, \beta_q)$ for $\hat{p}, \hat{q} \in \hat{Q}_p$, corresponding to movement of the continuous state in one mode with respect to the boundary of the invariant set in the next mode $q \in \text{HumanReach}(p) \neq 0$, for $p \in Q_i$, $q \in Q_k$, $k \in \text{Ind}(\text{HumanReach}(p))$ (i.e., internal transitions not corresponding directly to any discrete transition in H_p). See Fig. 7 for clarification

$$\hat{R}_i(\hat{p}, \alpha_q) = \hat{q}, \text{ with } \hat{p} \in \text{Sel}(\hat{P}, \mathcal{N}), \quad \hat{q} \in \text{Sel}(\hat{P}, \overline{\mathcal{N}})$$
$$\hat{R}_i(\hat{q}, \beta_q) = \hat{p}. \quad (8)$$

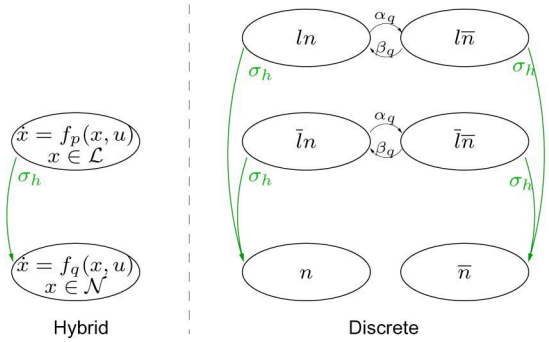

Fig. 7. Discrete abstraction of a two-mode hybrid automaton with *human-initiated* discrete input σ_h. The hybrid modes belong to different hybrid subsystems H_i and H_k, therefore, $p \in Q_i$ and $q \in Q_k$. The invariant sets are $\phi(\mathcal{C}_i) = \{(p, \mathcal{L})\}$ and $\phi(\mathcal{C}_k) = \{(q, \mathcal{N})\}$. In the corresponding discrete abstraction, $\hat{P} = \{ln, \bar{l}n, l\overline{n}, \bar{l}\overline{n}\}$ and $\hat{Q} = \{n, \overline{n}\}$.

The discrete transition relation $\hat{R}_i$ is defined such that for internal events, it reflects the original continuous dynamics that govern the movement between cells of the partition defined in Definition 11. The controlled event α_k occurs when the continuous state *exits* the region in which a transition to mode q would be invariance-preserving: $x \in \mathcal{W}_k(q) \xrightarrow{\alpha_k} x \in \overline{\mathcal{W}_k(q)}$. Conversely, the controlled event β_k occurs when the continuous state *enters* the region in which a transition to mode q would invariance-preserving: $x \in \overline{\mathcal{W}_k(q)} \xrightarrow{\beta_k} x \in \mathcal{W}_k(q)$.

The result is $\hat{G}_i = (\hat{Q}_i, \hat{\Sigma}_i, \hat{R}_i)$, with $\hat{Q}_i = \bigcup_p \hat{Q}_p$, $\hat{\Sigma}_i = \Sigma_i$, and $(\hat{Q}_i)_0 = \hat{Q}_p, p = (Q_i)_0$.

The discrete event system $\hat{G}_i$ reflects, by construction, the evolution of hybrid system trajectories with respect to the hybrid invariant sets.

Proposition 2: The discrete event system $\hat{G}_i$ is invariance-preserving for trajectories whose initial states are contained in $\left(\hat{Q}_i^{\text{inv}}\right)_0 = \text{Sel}(\mathcal{W}_i(p), \hat{Q}_p), p = (Q_i)_0$.

Proof: Proof by induction; see Appendix I-B for details. ■

Yellow Interval Example: Consider subsystem H_{green}. With no user-controlled transitions $\text{HumanReach}(q_{\text{green}}) = \{\emptyset\}$, the state-space for $p = q_{\text{green}}$ is partitioned into 2^1 regions: $\{\mathcal{W}(q_{\text{green}}), \overline{\mathcal{W}(q_{\text{green}})}\}$. The abstraction of these two continuous regions is the modes $\hat{Q}_{\text{green}} = \{w, \overline{w}\}$. Previously we constructed the discrete modes $\hat{Q}_{\text{yellow}} = \{wb_1a_1, wb_1\overline{a_1}, w\overline{b_1}a_1, w\overline{b_1}\overline{a_1}\}$. Since the transition δ from q_{green} to q_{yellow} is a *disturbance* event, we follow Step 2 of Algorithm 2 to determine the transition function in the discrete abstraction. According to (6) since $w = \text{Sel}(\hat{Q}_{\text{green}}, \mathcal{W}(q_{\text{green}}))$ and $\hat{Q}_{\text{yellow}} = \text{Sel}(\hat{Q}_{\text{green}}, \mathcal{W}(q_{\text{yellow}}))$, the transition relation is $\hat{R}(w, \delta) \in \{wb_1a_1, wb_1\overline{a_1}, w\overline{b_1}a_1, w\overline{b_1}\overline{a_1}\}$.

Now consider mode q_{yellow}. Two transitions are possible, both human-initiated events. If braking is applied, the hybrid system switches into mode $q_{\text{yellow}}^{\text{brake}}$; if acceleration is chosen, the hybrid system switches into mode $q_{\text{yellow}}^{\text{accel}}$. In the discrete abstraction, these two modes are represented by $\{b_1, \overline{b_1}\}$ and $\{a_1, \overline{a_1}\}$, respectively, based on reachability analysis performed earlier. Following (7) of Algorithm 2, since we know that the mode $wa_1b_1 \in \text{Sel}\left(\hat{Q}_{\text{yellow}}^{\text{brake}}, \mathcal{B}\left(q_{\text{yellow}}^{\text{brake}}\right)\right)$ is in the region of the state-space that is "safe" in the next mode, $q_{\text{yellow}}^{\text{brake}}$, the corresponding relation in the discrete abstraction is defined as $\hat{R}_{\text{green}}(wb_1a_1, \sigma_{\text{brake}}) \in \text{Sel}\left(\hat{Q}_{\text{yellow}}^{\text{brake}}, \mathcal{B}\left(q_{\text{yellow}}^{\text{brake}}\right)\right)$, which simplifies to $\hat{R}_{\text{green}}(wb_1a_1, \sigma_{\text{brake}}) = b_1$. Similarly, we can find that $\hat{R}_{\text{green}}(w\overline{b_1}a_1, \sigma_{\text{brake}}) = \overline{b_1}$. A total of eight such transitions are defined for the two user-initiated transitions (σ_{brake} and σ_{accel}) possible from each of the four modes in $\hat{Q}_{\text{yellow}}$. The result is depicted in Fig. 8. Additionally, internal transitions are shown graphically in Fig. 8. The transition $\hat{R}_{\text{green}}\left(wb_1a_1, \alpha_{\text{brake}}^{\text{yellow}}\right) = w\overline{b_1}a_1$ corresponds to the continuous state evolving from a region in which braking would be a safe course of action, to one where it would not be safe.

Last, consider one of the remaining hybrid modes: $q_{\text{yellow}}^{\text{brake}}$. No user-initiated transitions are possible, so we abstract this hybrid mode to 2^1 discrete modes, $\hat{Q}_{\text{yellow}}^{\text{brake}} = \{b_1, \overline{b_1}\}$. We again look to (6) of Algorithm 2 to determine the appropriate transition relation in the discrete abstraction: $\hat{R}_{\text{brake}}(b_1, t \geq \tau) = b_2$, $\hat{R}_{\text{brake}}(\overline{b_1}, t \geq \tau) = \overline{b_2}$, where $\hat{Q}_{\text{react}}^{\text{brake}} = \{b_2, \overline{b_2}\}$. The discrete abstraction of the other modes in H_{brake} and H_{accel} proceeds similarly, with $\hat{Q}_{\text{red}}^{\text{brake}} = \{b_3, \overline{b_3}\}$, $\hat{Q}_{\text{yellow}}^{\text{accel}} = \{a_1, \overline{a_1}\}$, $\hat{Q}_{\text{react}}^{\text{accel}} = \{a_2, \overline{a_2}\}$, and $\hat{Q}_{\text{red}}^{\text{accel}} = \{a_3, \overline{a_3}\}$, and similarly defined transition relations in the discrete abstraction, as shown in Fig. 8. ■

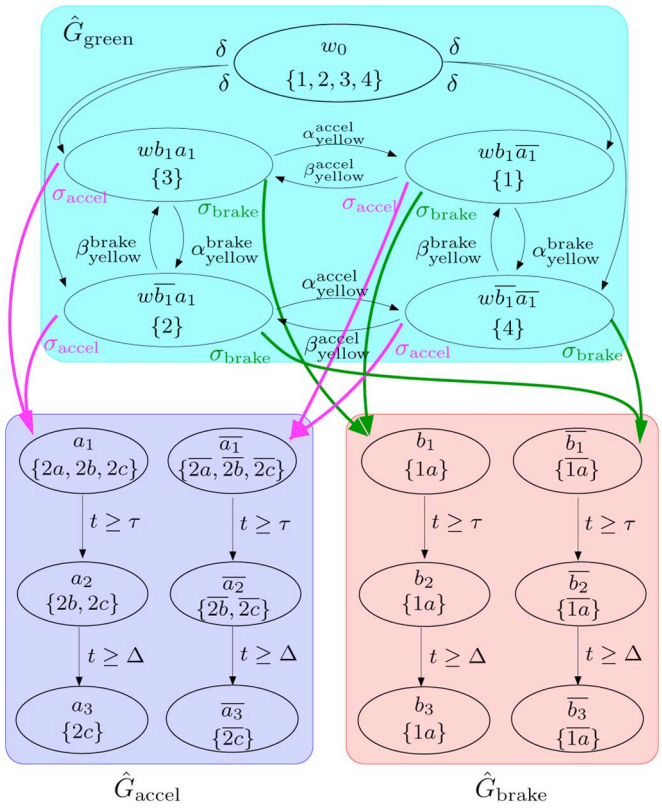

Fig. 8. Discrete event system $\hat{G}_{\text{car}}$ is the invariance-preserving abstraction of H_{car}. In each mode of the abstraction, the name of the discrete mode is listed in the top line. Listed below are the continuous regions the discrete mode corresponds to, as shown in Figs. 3–5 and as appropriate.

C. Step 3: Connecting Systems $\hat{G}_i$

The third step of Algorithm 1 is to combine the N abstractions G_i into a single discrete event system $\hat{G} = (\hat{Q}, \hat{\Sigma}, \hat{R})$.

Definition 13: We define $\hat{Q} = \bigcup_i \hat{Q}_i$, $\hat{\Sigma} = \bigcup_i \hat{\Sigma}_i$, and $\hat{R}(\hat{q}_i, \sigma) = \hat{R}_i(\hat{q}, \sigma), \sigma \in \hat{\Sigma}_i, \hat{q} \in \hat{Q}_i$ with $i \in \mathrm{Ind}(Q_{\mathrm{entry}})$.

Yellow Interval Example: The invariance-preserving discrete abstraction $\hat{G}_{\mathrm{car}}$ of the hybrid system H_{car} (see Fig. 1) is shown graphically in Fig. 8. ■

Proposition 3: The discrete event system $\hat{G} = (\hat{Q}, \hat{\Sigma}, \hat{R})$ with initial modes $\hat{Q}_0$, constructed according to the three-step procedure in Algorithm 1 from the hybrid system $H = (Q, \mathcal{X}, R, f, \Sigma, \mathcal{U})$ with constraint set $\mathcal{C} \in Q \times \mathcal{X}$, is an invariance-preserving abstraction of H.

Proof: Proof by contradiction; see Appendix I-C for details. ■

D. Using the Discrete Abstraction

An advisory system can then be deduced from the invariance-preserving abstraction $\hat{G}$ using existing discrete reduction techniques, not covered here. See [48]–[51] for information on reduction of finite-state machines, [52]–[56] for computational techniques to accomplish discrete-state reduction, and [17], [18] for application of discrete-state reduction to the problem of interface design. Using these techniques, we can design an interface, or advisory system, which will maintain the invariance-preserving properties of the abstraction $\hat{G}$.

In some situations, an interface may already exist and may be designed by different people than those who designed the safety control laws for the system. In this case, it is important to verify that the interface correctly represents the underlying, hybrid system. The authors in [12] and [17] developed a method to formally verify that one discrete system (such as an interface) adequately represents another discrete system (such as a "truth" model of a system). Using these methods, the existing interface can be verified against the invariance-preserving discrete abstraction developed here, according to the method in [12] and [17]. (We demonstrated this on the autoland example in [15].)

The same authors have used techniques for state reduction of deterministic, incompletely-specified finite state machines, as a tool for interface design [18]. Given a finite-state machine which represents a "truth" model of the actual system and an output function defined for every state, the resultant reduced model, created through state reduction, represents the interface for the original finite-state machine. This technique can be extended for the types of nondeterministic systems particular to the abstraction technique presented here, which may contain far more information than the user needs in order to accurately interact with the system.

Yellow Interval Example: An interface for the yellow interval advisory system, constructed through discrete state-reduction techniques, is shown in Fig. 9. We associate the modes of $\hat{G}_{\mathrm{car}}$ in Fig. 8 to five advisories

Continue Driving	$\{w_0, a_3\}$
Slow to Stop	$\{wb_1\overline{a_1}, b_1, b_2, b_3\}$
Accelerate	$\{w\overline{b_1}a_1, a_1, a_2\}$
Accelerate or Brake	$\{wb_1a_1\}$
Unsafe	$\{w\overline{b_1}\,\overline{a_1}, \overline{b_1}, \overline{b_2}, \overline{b_3}, \overline{a_1}, \overline{a_2}, \overline{a_3}\}$.

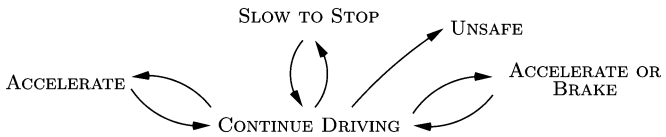

Fig. 9. Interface indications for the dashboard device, as determined from Figs. 3, 4, and 8.

The advisory system provides the reduced set of control instructions to guide the user successfully in navigating the intersection, avoiding a red light violation. The process of interface design through discrete state reduction is closely related to the topic of discrete observability: this relationship is investigated in [57]. ■

IV. Application: Automatic Landing of a Large Civil Jet Aircraft

Autoland systems are complex, safety-critical systems, subject to stringent certification criteria [58]. Modeling the aircraft's behavior, which incorporates logic from the autopilot as well as inherently complicated aircraft dynamics, results in a high-dimensional hybrid system with many continuous and discrete states. Naturally, only a subset of this information is displayed to the pilot. We want to design a cockpit interface which provides the pilot with enough information so that the pilot can safely land or safely go-around. In previous work [15], [59], we introduced a model of an autoland system for a large civil jet aircraft which made many simplifying assumptions regarding the pilot's input. This work demonstrated the use of a hybrid reachability tool [26], [27] to aid in the problem of user-interface verification. We now introduce a more realistic model of the autoland scenario and demonstrate the general method developed in this paper to create an invariance-preserving abstraction. The new model incorporates the multitude of options a pilot has at his disposal in the event of an aborted landing, known as a go-around. The autoland example is derived from publicly available aerodynamic data and actual flight management systems onboard a commercial jet aircraft.

During a typical autoland, the pilot and the autopilot share control over the aircraft. The pilot controls the aircraft's flaps setting and landing gear, which affect the aircraft's aerodynamics. The flaps can be set at Flaps-20, Flaps-25, or Flaps-30, in increasing deflections; the landing gear can be either up or down. The autopilot controls the thrust and angle of attack in order to guide the aircraft to a smooth touchdown with an appropriate descent rate—this is known as a "flare" maneuver.

However, if for any reason the pilot or air traffic controller deems the landing unacceptable (debris on the runway, a potential conflict with another aircraft, or severe wind shear near the runway, for example), the pilot must initiate a go-around maneuver. The pilot initiates a go-around maneuver at any time before the aircraft touches down, by toggling the "TO/GA" (Take-Off/Go-Around) lever. We, therefore, model the decision to go-around as a human-initiated event σ_{TOGA}.

We model the nonlinear longitudinal dynamics of a large civil jet aircraft by $\dot{x} = f_i(x, u)$, in which the state $x = [V, \gamma, h] \in$

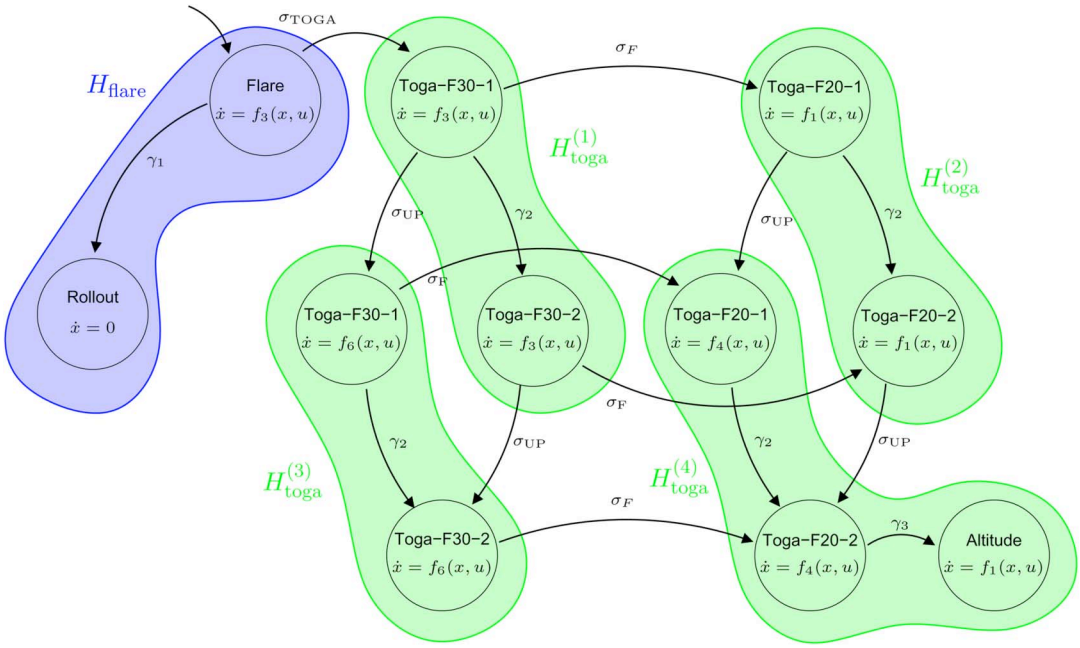

Fig. 10. Autoland/go-around automaton H_{autoland}. The mode labels refer to the particular combination of mode-logic (i.e., desired trajectory the aircraft autopilot is tracking) as well as the dynamics of the particular mode. The first part of the TO/GA maneuver (Toga-Fxx-1) occurs with input $T = T_{\max}$, the second part of the TO/GA maneuver (Toga-Fxx-2) occurs with input $T \in [T_{\min}, T_{\max}]$. The automatic transition γ_1 occurs when $h = 0$, γ_2 occurs when $\dot{h} \geq 0$, and γ_3 occurs when $h \geq h_{\text{alt}}$.

TABLE I
AERODYNAMIC CONSTANTS FOR AUTOLAND MODES

Dynamics	C_{L_0}	C_{D_0}	K	Flaps Setting	Landing Gear
$\dot{x} = f_1(x,u)$	0.4225	0.024847	0.04831	Flaps-20	Down
$\dot{x} = f_3(x,u)$	0.8212	0.025455	0.04831	Flaps-30	Down
$\dot{x} = f_4(x,u)$	0.4225	0.019704	0.04589	Flaps-20	Up
$\dot{x} = f_6(x,u)$	0.8212	0.020313	0.04589	Flaps-30	Up

$\mathbb{R}^3$ includes the aircraft's speed V, flightpath angle γ, and altitude h (see [44] and [60])

$$\begin{bmatrix} m\dot{V} \\ mV\dot{\gamma} \\ \dot{h} \end{bmatrix} = \begin{bmatrix} -D(\alpha,V) + T\cos\alpha - mg\sin\gamma \\ L(\alpha,V) + T\sin\alpha - mg\cos\gamma \\ V\sin\gamma \end{bmatrix}. \quad (9)$$

We assume the control input is $u = [T, \alpha]$, with aircraft thrust T and angle of attack α. The aircraft has mass $m = 190\,000$ kg, pitch $\theta = \alpha + \gamma$, and gravitational acceleration is $g = 9.81$ m/s^2. The aircraft's lift $L(\alpha,V) = (1/2)\rho V^2 S C_L(\alpha)$ and drag $D(\alpha,V) = (1/2)\rho V^2 S C_D(\alpha)$ depend on air density $\rho = 1.225$ kg/m^3, wing surface area $S = 427.80$ m^2, and the coefficients of lift and drag, $C_L(\alpha) = C_{L_0} + C_{L_\alpha}\alpha$ and $C_D(\alpha) = C_{D_0} + KC_L^2(\alpha)$. The constants C_{L_0}, C_{D_0}, and K were determined for the particular combinations of flap settings and landing gear in an autoland/go-around scenario [60]–[64] (see Table I). $C_{L_\alpha} = 5.105$ in all modes.

The initial state is the Flare mode (see Fig. 10), in which the flaps are at Flaps-30 and the thrust is fixed at idle. During a normal automatic landing, upon touchdown, the aircraft switches to Rollout mode. We model this event through γ_1, which occurs when $h = 0$. We do not model the dynamics of the aircraft as it rolls along the runway.

If a go-around is required, the pilot immediately changes the flaps to Flaps-20 and the autothrottle forces the thrust to $T_{\max}$. When the aircraft obtains a positive rate of climb, the pilot raises the landing gear, and the autothrottle allows $T \in [0, T_{\max}]$.

TABLE II
STATE BOUNDS FOR AUTOLAND MODES OF H_{autoland}

Mode	V [m/s]	γ [degrees]	α [degrees]	T [N]
Flare	[55.57, 87.46]	$[-6.0°, 0.0°]$	$[-9°, 15°]$	0
Toga-F30-1	[55.57, 87.46]	$[-6.0°, 0.0°]$	$[-9°, 15°]$	$T_{\max}$
Toga-F30-2	[55.57, 87.46]	$[0.0°, 15.7°]$	$[-9°, 15°]$	$[0, T_{\max}]$
Toga-F20-1	[63.79, 97.74]	$[-6.0°, 0.0°]$	$[-8°, 12°]$	$T_{\max}$
Toga-F20-2	[63.79, 97.74]	$[0.0°, 13.3°]$	$[-8°, 12°]$	$[0, T_{\max}]$
Altitude	[63.79, 97.74]	$[-0.7°, 0.7°]$	$[-8°, 12°]$	$[0, T_{\max}]$

TABLE III
COMPUTATIONAL DOMAIN FOR AUTOLAND HYBRID SUBSYSTEMS

Subsystem	V [m/s]	γ [degrees]	h [m]
H_{flare}	[50, 100]	[-8, 17]	[-5, 20]
$H^{(i)}_{\text{toga}}$	[50, 100]	[-8, 17]	[-5, 20]

The user-controlled transition σ_F occurs when the pilot selects Flaps-20, and the automatic transition γ_2 occurs when $\dot{h} \geq 0$. The aircraft continues to climb to the missed approach altitude, h_{alt}, then automatically switches into an altitude-holding mode, Altitude. We model this event as an automatic transition γ_3 which occurs when $h \geq h_{\text{alt}}$.

While standard procedure calls for a flap change simultaneously when the go-around is initiated, the pilot may complete the flap change *after* toggling the TO/GA lever. While standard procedure indicates a strict order of events, in practice the pilot has more flexibility. For example, the pilot may raise the landing gear before *or* after obtaining a positive rate of climb. After initiating a go-around, three changes must occur, but can occur in any order: 1) the pilot changes the flaps from Flaps-30 to Flaps-20 (σ_F); 2) the pilot raises the landing gear (σ_{UP}); and 3) the aircraft obtains a positive rate of climb (γ_2).

State and input bounds due to constraints arising from aircraft aerodynamics and desired aircraft behavior, are summarized in

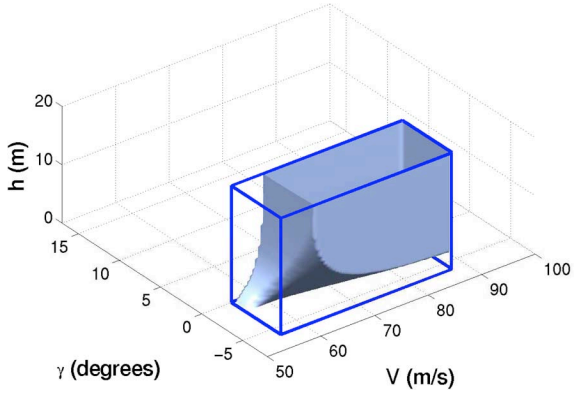

Fig. 11. Constraint set $(\mathcal{W}_{\text{flare}})_0$ (wireframe box) and invariant set $\mathcal{W}_{\text{flare}}$ (solid).

Table II [64], [65]. Bounds on γ and T are determined by the desired maneuver [66], [67], with $T_{\max} = 686\,700$ N. Additionally, at touchdown, $\theta \in [0°, 12.9°]$ to prevent a tail strike, and $\dot{h} \geq -1.829$ m/s to prevent damage to the landing gear. These constraints, in addition to those listed in Table III, form the constraint set $\mathcal{C}_{\text{autoland}}$.

A. Reachability Analysis of Hybrid Subsystems

We separate the hybrid model H_{autoland} into five hybrid subsystems with no human-initiated events, as shown in Fig. 10. We perform standard reachability analysis on each system using the level-set-based tool [26], [27]. Computationally, automatic transitions are smoothly accomplished by modeling the change in dynamics across the switching surface as another nonlinearity in the dynamics. Additionally, we assume in $H^{(i)}_{\text{toga}}$ that if the aircraft exits the top of the computational domain ($h = 20$ m) without exceeding its flight envelope, it is capable of safely achieving Altitude mode. The computational domain is indicated in Table III.

Invariant sets are computed with a level-set-based reachability tool [27]. While coarse computations can be accomplished in under an hour, computations on a finer grid ($\overline{n} = 100$) such as those shown in Figs. 11–13 can take as long as a day. Fig. 11 depicts the constraint set $(\mathcal{W}_{\text{flare}})_0$ (shown as the wireframe box) as well as the invariant set $\mathcal{W}_{\text{flare}}$ (solid). Fig. 12 depicts the constraint set $\left(\mathcal{W}^{(4)}_{\text{toga}}\right)_0$ (wireframe box), as well as the invariant set $\mathcal{W}^{(4)}_{\text{toga}}$ (solid). The computed invariant set $\mathcal{W}^{(2)}_{\text{toga}}$ is indistinguishable from $\mathcal{W}^{(4)}_{\text{toga}}$ since the dynamics $\dot{x} = f_1(x,u)$ differ from $\dot{x} = f_4(x,u)$ only slightly, in the drag term C_{D_0}. (See Fig. 10 and Table I). Fig. 13 depicts the constraint set $\left(\mathcal{W}^{(1)}_{\text{toga}}\right)_0$ (wireframe box), as well as the invariant set $\mathcal{W}^{(1)}_{\text{toga}}$ (solid). The computed invariant set $\mathcal{W}^{(1)}_{\text{toga}}$ is indistinguishable from $\mathcal{W}^{(3)}_{\text{toga}}$ since the dynamics $\dot{x} = f_3(x,u)$ and $\dot{x} = f_6(x,u)$ also differ only slightly, in C_{D_0}.

The intersections of these sets must be computed for the pilot to be able to safely control the aircraft. For example, for the pilot to safely switch from Toga-F30-1 or Toga-F30-2 to Toga-F20-1 or Toga-F20-2, respectively, by enacting σ_F, the pilot must have information at his disposal regarding the intersection of $\mathcal{W}^{(3)}_{\text{toga}}$ and $\mathcal{W}^{(4)}_{\text{toga}}$. The intersections of $\mathcal{W}^{(1)}_{\text{toga}}$ and $\mathcal{W}^{(3)}_{\text{toga}}$; of $\mathcal{W}^{(2)}_{\text{toga}}$ and $\mathcal{W}^{(4)}_{\text{toga}}$; of $\mathcal{W}^{(1)}_{\text{toga}}$, $\mathcal{W}^{(2)}_{\text{toga}}$, and $\mathcal{W}^{(3)}_{\text{toga}}$; and of $\mathcal{W}_{\text{flare}}$ and $\mathcal{W}^{(1)}_{\text{toga}}$ must be computed.

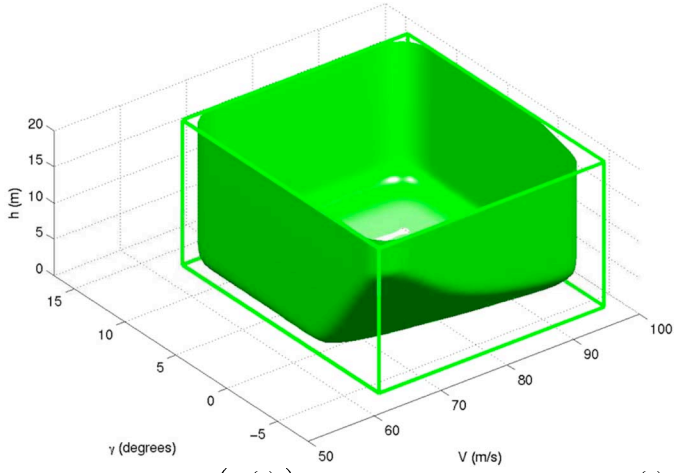

Fig. 12. Constraint set $\left(\mathcal{W}^{(2)}_{\text{toga}}\right)_0$ (wireframe box) and invariant set $\mathcal{W}^{(2)}_{\text{toga}}$ (solid), which are computationally indistinguishable from the constraint set $\left(\mathcal{W}^{(4)}_{\text{toga}}\right)_0$ and invariant set $\left(\mathcal{W}^{(4)}_{\text{toga}}\right)$, respectively.

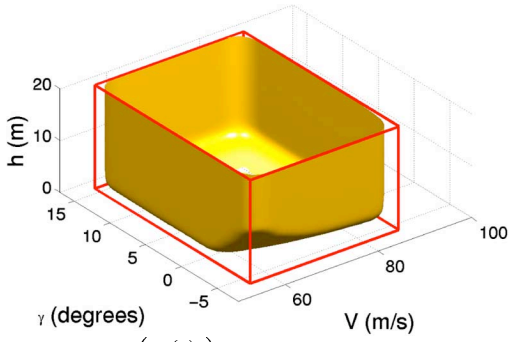

Fig. 13. Constraint set $\left(\mathcal{W}^{(1)}_{\text{toga}}\right)_0$ (wireframe box) and invariant set $\mathcal{W}^{(1)}_{\text{toga}}$ (solid), which are computationally indistinguishable from the constraint set $\left(\mathcal{W}^{(3)}_{\text{toga}}\right)_0$ and invariant set $\mathcal{W}^{(3)}_{\text{toga}}$, respectively.

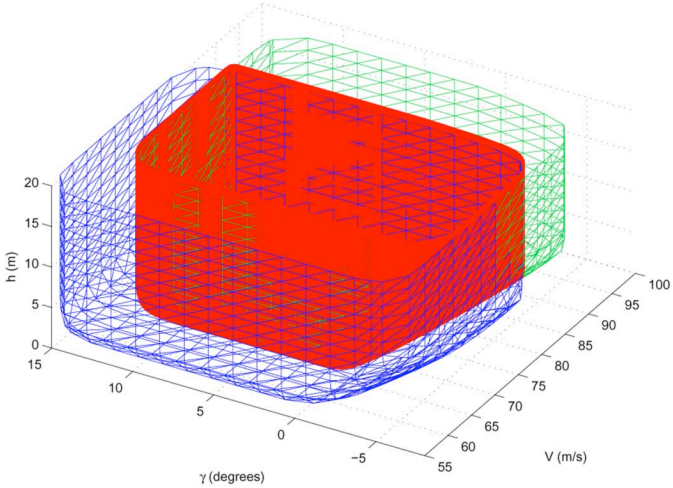

Fig. 14. Invariant sets $\mathcal{W}^{(3)}_{\text{toga}}$ [dark mesh (blue)], $\mathcal{W}^{(4)}_{\text{toga}}$ [light mesh (green)], and their intersection [solid (red)]. The intersection represents the set of continuous states in $\mathcal{W}^{(3)}_{\text{toga}}$ from which the pilot can safely change the flaps (σ_F).

The intersection of $\mathcal{W}^{(3)}_{\text{toga}}$ and $\mathcal{W}^{(4)}_{\text{toga}}$ in Fig. 14 is the set of states from which the aircraft can safely remain in $H^{(3)}_{\text{toga}}$ and from which the aircraft is safe to switch to $H^{(4)}_{\text{toga}}$. States in $\mathcal{W}^{(3)}_{\text{toga}}$ which are outside of this intersection are states from

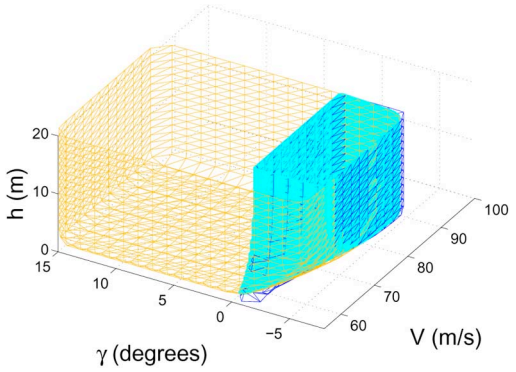

Fig. 15. Intersection $\mathcal{W}_{\text{flare}} \cap \mathcal{W}^{(1)}_{\text{toga}}$ [light solid (cyan)] represents the set of continuous states in Flare in which the pilot can safely enact the transition σ_{TOGA}.

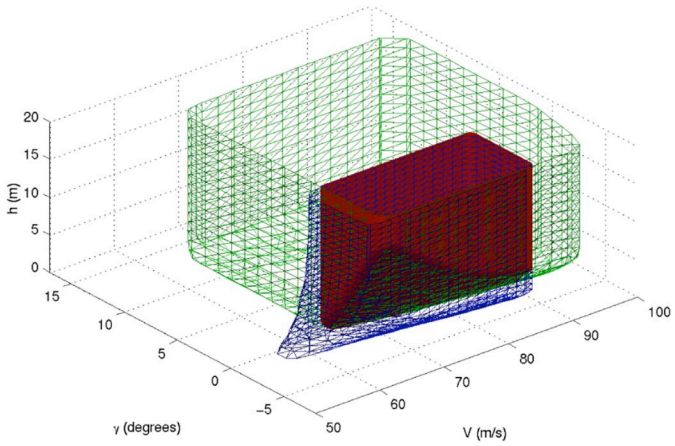

Fig. 16. In a simplified version of H_{autoland} [15], σ_{TOGA} and σ_F are assumed to be concurrent. This results in a significantly reduced region in Flare from which a safe go-around is possible.

which the aircraft can safely remain in $H^{(3)}_{\text{toga}}$, but will become unsafe if the pilot switches the aircraft to $H^{(4)}_{\text{toga}}$ (by raising the flaps from Flaps-30 to Flaps-20). Similarly, the intersection of $\mathcal{W}_{\text{flare}}$ and $\mathcal{W}^{(1)}_{\text{toga}}$ in Fig. 15 is the set of states from which the aircraft can safely land, and alternatively, from which the aircraft can safely switch into the first Toga mode. The analysis shows that there are regions from which a safe landing is possible, but a safe go-around is not: for $x \notin \left(\mathcal{W}_{\text{flare}} \cap \mathcal{W}^{(1)}_{\text{toga}}\right)$ in Flare mode, the aircraft will become unsafe when the pilot initiates a go-around by enacting σ_{TOGA}. This region is necessary for the aircraft to be able to complete a landing, however is problematic in the event of a go-around. The region in Flare from which a safe go-around is possible is considerably larger than the result in [15], shown in Fig. 16: expanding the system to account for all user options results in an increased safe region of operation, since the pilot can allow the aircraft to gain sufficient speed in Toga-F30-1 before changing to Flaps-20.

B. Invariance-Preserving Discrete Abstraction

Using the reachability computations, we can now create an invariance-preserving discrete abstraction of H_{autoland}. In most commercial aircraft, the low-level control is performed by the autopilot, which has authority over small control surface movement. The details of the low-level control are hidden from the pilot, who anticipates system behavior by understanding the behavior of each autopilot mode. We, therefore, assume an automated controller enforces $u \in u^*(x)$ along the boundary of the controllable sets, but leave it to the pilot to enforce any discrete switches necessary to maintain safety. By doing so, we mimic the supervisory role pilots have in highly automated aircraft, including the prerogative not to enforce a recommended switch.

In previous work [15], we used the discrete composition method of Heymann and Degani [17], [18] to determine whether or not a given interface adequately and unambiguously represents $\hat{G}_{\text{autoland}}$, the invariance-preserving abstraction of H_{autoland}. The hybrid model and discrete interface in [15] were both simplified systems with a relatively small number of discrete modes. In general, more complex systems (such as in Fig. 10) will result in discrete systems with a much higher number of discrete modes. The mode explosion which will result from the abstraction of Fig. 10 is necessary to determine the interaction of the user's actions and maintenance of the aircraft within its flight envelope; it also motivates the use of a method, as in [17] and [18], for the verification and design of interfaces too large to be accurately studied in an *ad hoc* way.

C. Results

The discrete abstraction $\hat{G}_{\text{autoland}}$ has 111 states. With the information contained in the abstracted model $\hat{G}_{\text{autoland}}$, it is possible to determine what information the pilot needs in order to steer the aircraft to safe regions of operation. We reduce $\hat{G}_{\text{autoland}}$ using discrete state reduction techniques (see Fig. 17), and propose an interface for H_{autoland} by relabeling the modes of this reduced automaton to indicate possible actions to the pilot. The result $\hat{G}_{\text{interface}}$ is shown in Fig. 18.

We validated the use of the invariance-preserving abstraction through tests in an actual commercial aircraft flight simulator. Using the abstraction method as a tool for user-interface verification, we successfully predicted problematic behaviors in human–automation interaction.

V. Conclusion

Human-automation systems are ubiquitous, from common consumer devices (an indoor thermostat) to extremely complex, specialized systems (modern aircraft autopilots). As the use of human–automation systems inevitably grows, verification of how users interact and supervise such systems becomes crucial. This is especially true when the applications involve safety-critical, expensive, or high-risk systems, but is also applicable to simpler, less-critical systems which can cause pointless frustration. While some systems can be reasoned through in an *ad hoc* way, systems which have nontrivial continuous dynamics or many modes require a methodical approach.

We presented a method for the synthesis of an invariance-preserving abstraction of a hybrid human–automation system. The method presented here involves three steps: 1) separation of the hybrid system into hybrid subsystems with no discrete human inputs; 2) abstraction of each hybrid subsystem into a discrete

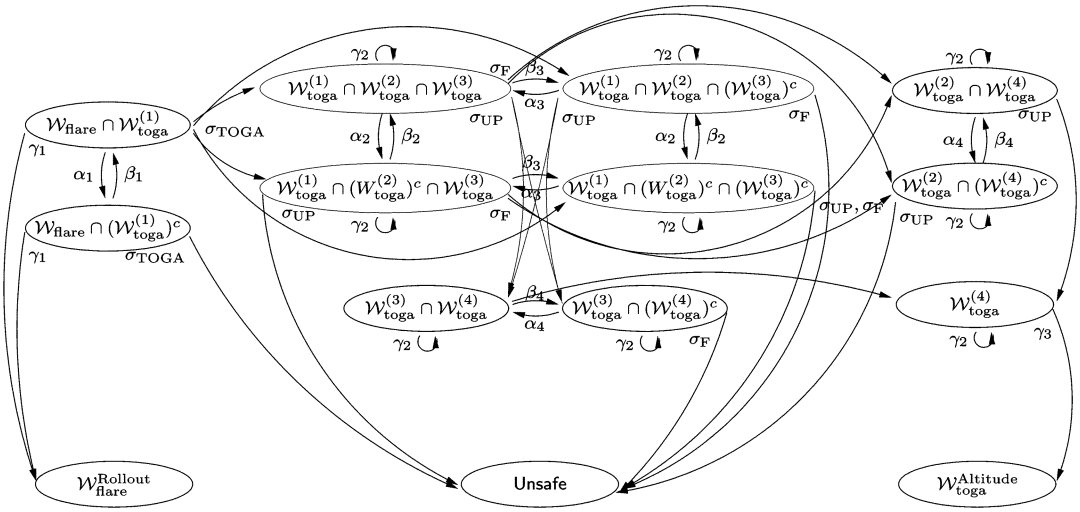

Fig. 17. Reduced version of discrete event system $\hat{G}_{\text{autoland}}$ for autoland/go-around maneuver.

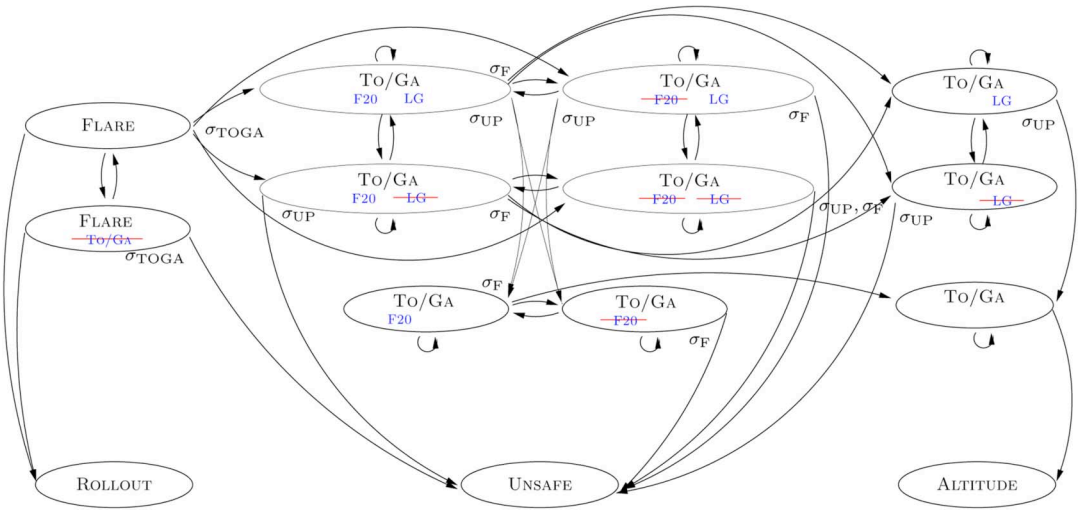

Fig. 18. Proposed interface $\hat{G}_{\text{interface}}$ for autoland/go-around maneuver H_{autoland}, based on reduced version of $\hat{G}_{\text{autoland}}$ (see Fig. 17). The pilot's presumed and recommended actions are indicated in each mode. Events which will transition the system to unsafety are struck through.

system on the basis of information procured through hybrid reachability analysis and controller synthesis of each hybrid subsystem; and 3) connection of each of the resultant discrete subsystems into one discrete event system which is an invariance-preserving abstraction of the original hybrid system. The key contribution of this work is a systematic way to create an abstraction of hybrid systems based on a hybrid reachability result: the resultant discrete system is one for which existing interface analysis, verification, and design techniques can be implemented.

The resulting discrete event system preserves information regarding the invariance of the underlying hybrid system and the potential effect of the human's input on maintaining invariance. Applying discrete state reduction techniques to this invariance-preserving abstraction results in a reduced discrete event system (interface) with which the human can effectively interact. The advantage of using this technique is that interfaces designed through this method will contain information about the invariance of the underlying hybrid system—this information would not otherwise have been incorporated through standard discrete event system modeling and analysis techniques. In safety-critical systems, such as civil jet aircraft automation, information about the effect of the human's actions in system invariance is vital for safe operation. In autoland/go-around scenario, this information results in a interface which provides the minimal information necessary for the pilot to safely complete a go-around maneuver. The two examples presented, an add-on dashboard device for yellow interval guidance and an aircraft autopilot, served to demonstrate the abstraction algorithm and its application to a wide range of problems.

APPENDIX

Proof of Proposition 1: Proof by contradiction. Assume there exists a trajectory with initial state $(q_0, x_0) \in \phi(\mathcal{C}_i) = (Q_i, \mathcal{W}_i)$ which is not user-invariant. From Definition 3, this means that either (a) $(q_0, x_0) \notin \mathcal{C}_i$ or (b) the trajectory may exit $\mathcal{C}_i$ under disturbance or controlled inputs. Since $\phi(\mathcal{C}_i) \subseteq \mathcal{C}_i$, we know $(q_0, x_0) \in \mathcal{C}_i$ which contradicts the first point (a).

In addition, since all trajectories which begin in $\phi(\mathcal{C}_i)$, will by construction (Definition 10), remain in $\mathcal{C}_i$ under controlled and disturbance inputs, this contradicts the second point (b). ■

Proof of Proposition 2: Proof by induction. 1) Begin with the $k = 0$ mode in the initial set $q_0 \in \left(\hat{Q}_i^{\text{inv}}\right)_0$. By definition, the initial set contains only modes corresponding to hybrid states $(q_0, x) \in \mathcal{W}_i$ within the invariant set. 2) Consider a generic state q_k resultant from a string of k events. (a) For any controlled or disturbance input σ for which $R_i(q_k, x, \sigma) = (q_{k+1}, x)$, implementing the control law arising from the reachability calculation assures that $(q_k, x) \in \mathcal{W}_i \longrightarrow (q_{k+1}, x) \in \mathcal{W}_i$, and (6) assures that this is reflected in the abstracted system G_i, as well. According to the discrete transition function formed in (6), $q_k \in \text{Sel}(\mathcal{W}_i(q_k), \hat{Q}_{q_k})$ implies that $q_{k+1} \in \hat{R}(q_0, \sigma) \in \text{Sel}(\mathcal{W}_i(q_{k+1}), \hat{Q}_{q_{k+1}})$, so that the set of discrete modes corresponding to (q_{k+1}, x) are contained within the discrete representation of $\mathcal{W}_i$. (b) For any human input σ_h such that $R(q_k, x, \sigma_h) = (q_{k+1}, x)$, the hybrid state may transition to a state outside of the invariant set $(q_{k+1}, x) \in \{\mathcal{W}_j, \overline{\mathcal{W}_j}\}$, depending from which regions of the hybrid state space the human enacts σ_h, and (7) reflects this same phenomenon in the abstracted system G_i as well. Similarly, in the abstracted system G_i, although $q_k \in \overline{\text{Sel}(\mathcal{W}_i(q_k), \hat{Q}_{q_k})}$, $q_k \in \{\text{Sel}(\mathcal{W}_i(q_{k+1}), \hat{Q}_{q_{k+1}}), \overline{\text{Sel}(\mathcal{W}_j(q_{k+1}), \hat{Q}_{q_{k+1}})}\}$, as indicated in (7): discrete trajectories are allowed to transition into discrete states which correspond to the complement of the invariant set.

From the two previous points, we can conclude that trajectories beginning from Q_i^{inv} will remain in discrete modes corresponding to the hybrid region $\mathcal{W}_i$ for any event strings consisting only of controlled or disturbance inputs. For any discrete human input to a state outside of H_i, the resultant discrete state will not necessarily correspond to a hybrid state within $\mathcal{W}_j$. Thus G_i with initial mode Q_i^{inv} accepts only invariant or user-invariant trajectories, and is therefore invariance-preserving. ■

Proof of Proposition 3: Proof by contradiction. Assume that $\hat{G}$ accepts trajectories which are *not* user-invariant. Then there exists a trajectory with initial state $q \in Q_0$ which is not invariant and not user-invariant. Therefore, this trajectory must enter a discrete state corresponding to $\overline{\mathcal{W}}$, where $\mathcal{W} = \bigcup_i \mathcal{W}_i \subseteq \mathcal{C}$, $\mathcal{W}_i = \phi(\mathcal{C}_i)$ for each subsystem H_i, $i \in \{1, \ldots, n\}$, through either a controlled or disturbance discrete input.

From Proposition 2, we know that each subsystem $\hat{G}_i$ is an invariance-preserving abstraction of its corresponding hybrid subsystem H_i for trajectories whose initial states are contained within $\left(\hat{Q}_i^{\text{inv}}\right)_0$. From Definition 2, this means that only invariant or user-invariant trajectories are accepted by $\hat{G}_i$. This contradicts the existence of the trajectory assumed previously, which is neither invariant nor user-invariant.

In addition, the only transitions possible between subsystems (i.e., from modes in $\hat{G}_i$ to modes in $\hat{G}_j$) are discrete human inputs. Trajectories corresponding to hybrid states in the invariant set $\mathcal{W}_i$ can transition into modes corresponding to hybrid states *outside* the invariant set $\mathcal{W}_j$, as allowed by user-invariant trajectories. Therefore, all trajectories whose state may transition from one subsystem to another subsystem are user-invariant. This contradicts the existence of the trajectory assumed above, which is neither invariant nor user-invariant. ■

ACKNOWLEDGMENT

The authors would like to thank A. Degani and M. Heymann for their contributions to the interface analysis and verification methods which inspired this work. They would also like to thank D. Austin, R. Mumaw, and C. Hynes for their help regarding the aircraft autoland scenario.

REFERENCES

[1] C. Billings, *Aviation Automation: The Search for a Human-Centered Approach*. Hillsdale, NJ: Erlbaum, 1997.
[2] E. Wiener and R. Curry, "Flight-deck automation: promises and problems," NASA Ames Research Center, Moffett Field, CA, Tech. Memo. 81206, 1980.
[3] R. Parasuraman, T. Sheridan, and C. Wickens, "A model for types and levels of human interaction with automation," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 30, no. 3, pp. 286–297, May 2000.
[4] N. Sarter, D. Woods, and C. Billings, "Automation surprises," in *Handbook of Human Factors and Ergonomics*. New York: Wiley, 1999, pp. 1295–1327.
[5] E. Palmer, "Oops, it didn't arm—A case study of two automation surprises," presented at the 8th Int. Symp. Aviation Psych. Conf., Columbus, OH, 1995.
[6] N. Leveson and E. Palmer, "Designing automation to reduce operator errors," in *Proc. IEEE Conf. Syst., Man, Cybern.*, 1997, pp. 1144–1150.
[7] A. Degani, M. Shafto, and A. Kirlik, "Modes in human-machine systems: Constructs, representation, and classification," *Int. J. Aviation Psych.*, vol. 9, no. 2, pp. 125–138, 1999.
[8] K. Abbott, S. Slotte, and D. Stimson, "The interfaces between flightcrews and modern flight deck systems," Federal Aviation Administration, Human Factors Team Report, Washington, DC, 1996.
[9] C. Hynes, G. Hardy, and L. Sherry, *Synthesis From Design Requirements of a Hybrid System for Transport Aircraft Longitudinal Control*. Moffett Field, CA: NASA Ames Research Center, 2001.
[10] J. Rushby, "Using model checking to help discover mode confusions and other automation surprises," *Reliab. Eng. Syst. Safety*, vol. 75, no. 2, pp. 167–177, 1999.
[11] R. Butler, S. Miller, J. Potts, and V. Carreno, "A formal methods approach to the analysis of mode confusion," in *Proc. AIAA/IEEE Digit. Avionics Syst. Conf.*, 1998, pp. C41/1–C41/8.
[12] A. Degani, M. Heymann, G. Meyer, and M. Shafto, "Some formal aspects of human–automation interaction," NASA Ames Research Center, Moffett Field, CA, Tech. Memo. 209600, 2000.
[13] N. Leveson, L. Pinnel, S. Sandys, S. Koga, and J. Reese, C. Johnson, Ed., "Analyzing software specifications for mode confusion potential," in *Proc. Workshop Human Error Syst. Develop.*, 1997, pp. 132–146.
[14] S. Vakil, A. Midkiff, T. Vaneck, and R. Hansman, "Mode awareness in advanced autoflight systems," presented at the 6th IFAC/IFIP/IFORS/IEA Symp. Anal., Des., Evaluation Man-Mach. Syst. Conf., Cambridge, MA, 1995.
[15] M. Oishi, I. Mitchell, A. Bayen, C. Tomlin, and A. Degani, "Hybrid verification of an interface for an automatic landing," in *Proc. IEEE Conf. Dec. Control*, 2002, pp. 1607–1613.
[16] J. Crow, D. Javaux, and J. Rushby, "Models and mechanized methods that integrate human factors into automation design," presented at the Int. Conf. Human-Comput. Interaction Aeronautics, Toulouse, France, 2000.
[17] A. Degani and M. Heymann, "Formal verification of human–automation interaction," *Human Factors*, vol. 44, no. 1, pp. 28–43, 2002.
[18] M. Heymann and A. Degani, "On abstractions and simplifications in the design of human–automation interfaces," NASA Ames Research Center, Moffett Field, CA, Tech. Memo. 211397, 2002.
[19] A. Chutinan and B. H. Krogh, "Verification of infinite-state dynamics systems using approximate quotient transition systems," *IEEE Trans. Autom. Control*, vol. 46, no. 9, pp. 1401–1410, Sep. 2001.
[20] G. Frehse, "PHAVer: Algorithmic verification of hybrid systems past HyTech," in *Hybrid Systems: Computation and Control*, ser. LNCS 34143, M. Morari and L. Thiele, Eds. New York: Springer Verlag, 2005, pp. 258–273.
[21] E. Asarin, "Reachability analysis of nonlinear systems using conservative approximation," in *Hybrid Systems: Computation and Control*, ser. LNCS 2623, T. Dang, A. Girard, O. Maler, and A. Pneuli, Eds. New York: Springer Verlag, 2003, pp. 20–35.
[22] A. B. Kurzhanski, "Ellipsoidal techniques for hybrid dynamics: The reachability problem," in *New Directions and Applications in Control Theory*, ser. Lecture Notes in Control and Information Sciences 321, W. Dayawansa, A. Lindquist, Y. Zhou, and P. Varaiya, Eds. New York: Springer Verlag, 2005, pp. 193–205.

[23] M. Kvasnica, P. Grieder, M. Baotic, and M. Morari, "Multi-parametric toolbox (MPT)," in *Hybrid Systems: Computation and Control*, ser. LNCS 2993, R. Alur and G. Pappas, Eds. New York: Springer Verlag, 2004, pp. 448–462.
[24] C. Tomlin, J. Lygeros, and S. Sastry, "A game theoretic approach to controller design for hybrid systems," *Proc. IEEE*, vol. 88, no. 7, pp. 949–970, Jul. 2000.
[25] I. Mitchell, "Application of level set methods to control and reachability problems in continuous and hybrid systems," Ph.D. dissertation, Dept. Scientific Comput. Comput. Math., Stanford Univ., Stanford, CA, 2002.
[26] I. Mitchell, A. Bayen, and C. Tomlin, "Validating a Hamilton–Jacobi approximation to hybrid system reachable sets," in *Hybrid Systems: Computation and Control*, ser. LNCS 2034, M. D. Benedetto and A. Sangiovanni-Vincentelli, Eds. New York: Springer Verlag, 2001, pp. 418–432.
[27] I. Mitchell, A "Toolbox of level-set methods, Version 1.1 Beta ed.," University of British Columbia, Vancouver, BC, Canada, 2005 [Online]. Available: http://www.cs.ubc.ca/~mitchell/ToolboxLS/index.html
[28] M. Oishi, C. Tomlin, V. Gopal, and D. Godbole, "Addressing multiobjective control: Safety and performance through constrained optimization," in *Hybrid Systems: Computation and Control*, ser. LNCS 2034, M. Di Benedetto and A. Sangiovanni-Vincentelli, Eds. Philadelphia, PA: Springer-Verlag, 2001, pp. 459–472.
[29] A. Bhatia and E. Frazzoli, "Incremental search methods for reachability analysis of continuous and hybrid systems," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science 2993, R. Alur and G. Pappas, Eds. Philadelphia, PA: Springer-Verlag, 2004, pp. 142–156.
[30] A. Tiwari and G. Khanna, "Nonlinear systems: Approximating reach sets," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science 2993, R. Alur and G. Pappas, Eds. Philadelphia, PA: Springer-Verlag, 2004, pp. 142–156.
[31] R. Alur, F. Ivancic, and T. Dang, "Progress on reachability analysis of hybrid systems using predicate abstraction," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science 2623, F. Wiedijk, O. Maler, and A. Pneuli, Eds. Prague, The Czech Republic: Springer-Verlag, 2003, pp. 4–19.
[32] T. J. Koo and S. Sastry, "Bisimulation based hierarchical system architecture for single-agent multi-modal systems," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science 2623, C. Tomlin and M. Greenstreet, Eds. Stanford, CA: Springer-Verlag, 2002, pp. 281–293.
[33] P. Tabuada, G. Pappas, and P. Lima, "Composing abstractions of hybrid systems," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science 2623, C. Tomlin and M. Greenstreet, Eds. Stanford, CA: Springer-Verlag, 2002, pp. 436–450.
[34] G. Pappas and S. Sastry, "Towards continuous abstractions of dynamical and control systems," in *Hybrid Systems IV*, ser. Lecture Notes in Computer Science 1273, P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, Eds. New York: Springer-Verlag, 1997, pp. 329–341.
[35] A. Puri and P. Varaiya, "Verification of hybrid systems using abstractions," in *Hybrid Systems II*, ser. LNCS. New York: Springer-Verlag, 1995.
[36] A. Crawford, "Driver judgment and error during the amber period at traffic lights," *Ergonomics*, vol. 5, no. 4, pp. 513–532, Oct. 1962.
[37] C. Liu, R. Herman, and D. Gazis, "A review of the yellow interval dilemma," *Transportation Res. A*, vol. 30, no. 5, pp. 333–348, 1996.
[38] D. Gazis, R. Herman, and A. Maradudin, "The problem of the amber signal light in traffic flow," *Oper. Res.*, vol. 8, no. 1, pp. 112–132, Jan./Feb. 1960.
[39] S. Hedlund and A. Rantzer, "Optimal control of hybrid systems," in *Proc. IEEE Conf. Dec. Control*, Phoenix, AZ, 1999, pp. 3972–3977.
[40] ITE Technical Council Task Force 4TF-1, Institute of Transportation Engineers, Washington, D.C., "Determining vehicle signal change and clearance intervals," 1994.
[41] W. Stimpson, P. Zador, and P. Tarnoff, "The influence of the time duration of yellow traffic signals on driver response," *ITE J.*, pp. 22–29, Nov. 1980.
[42] A. Chutinan and B. Krogh, "Computational techniques for hybrid system verification," *IEEE Trans. Autom. Control*, vol. 48, no. 1, pp. 64–75, Jan. 2003.
[43] F. Torrisi and A. Bemporad, "Hysdel—A tool for generating computational hybrid models for analysis and synthesis problems," *IEEE Trans. Control Syst. Technol.*, vol. 12, no. 3, pp. 235–249, Mar. 2004.
[44] C. Tomlin, I. Mitchell, A. Bayen, and M. Oishi, "Computational techniques for the verification of hybrid systems," *Proc. IEEE*, vol. 91, no. 1, pp. 986–1001, Jan. 2003.
[45] E. Asarin, T. Dang, and O. Maler, "D/DT: A verification tool for hybrid systems," in *Proc. IEEE Conf. Dec. Control*, 2001, pp. 2893–2898.
[46] T. Henzinger, P. Ho, and H. Wong-Toi, "A user guide to hytech," in *TACAS 95: Tools and Algorithms for the Construction and Analysis of Systems*, ser. LNCS 1019, E. Brinksma, W. Cleavel, K. Larsen, T. Margaria, and B. Steffen, Eds. New York: Springer-Verlag, 1995, pp. 41–71.
[47] S. Prajna, A. Papachristodoulou, P. Seiler, and P. A. Parrilo, "SOSTOOLS: Sum of squares optimization toolbox for MATLAB," 2004 [Online]. Available: http://www.cds.caltech.edu/sostools; http://www.aut.ee.ethz.ch/~parrilo/sostools
[48] Z. Kohavi, *Switching and Finite Automata Theory*. New York: McGraw-Hill, 1978.
[49] M. Paull and S. Unger, "Minimizing the number of states in incompletely specified sequential switching functions," *IRE Trans. Electron. Comput.*, vol. EC-8, pp. 356–367, Sep. 1959.
[50] S. Ginsburg, "A technique for the reduction of a given machine to a minimal-state machine," *IRE Trans. Electron. Comput.*, vol. EC-8, pp. 346–355, Sep. 1959.
[51] A. Grasselli and F. Luccio, "A method for minimizing the number of internal states in incompletely specified sequential networks," *IEEE Trans. Electron. Comput.*, vol. EC-14, no. 3, pp. 350–359, Jun. 1965.
[52] T. Kam, T. Villa, R. Brayton, and A. Sangiovanni-Vincentelli, "Implicit computation of compatible sets for state minimization of ISFSM's," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 16, no. 7, pp. 657–676, Jul. 1997.
[53] T. Kam, T. Villa, R. Brayton, and A. Sangiovanni-Vincentelli, "Theory and algorithms for state minimization of nondeterministic FSM's," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 16, no. 11, pp. 1311–1322, Nov. 1997.
[54] J. Rho, G. Hachtel, F. Somenzi, and R. Jacoby, "Exact and heuristic algorithms for the minimization of incompletely specified state machines," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 13, no. 2, pp. 167–177, Feb. 1994.
[55] R. Puri and J. Gu, "An efficient algorithm to search for minimal closed covers in sequential machines," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 6, no. 6, pp. 737–745, Jun. 1993.
[56] J. Pena and A. Oliveira, "A new algorithm for exact reduction of incompletely specified finite state machines," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 18, no. 11, pp. 1619–1632, Nov. 1999.
[57] M. Oishi, I. Hwang, and C. Tomlin, "Immediate observability of discrete event systems with application to user-interface design," in *Proc. IEEE Conf. Dec. Control*, 2003, pp. 2665–2672.
[58] Federal Aviation Administration, U.S. Department of Transportation, Washington, DC, "Criteria for approval of category III weather minima for takeoff, landing, and rollout," Advisory Circular 120-28D, 1999.
[59] M. Oishi, A. Degani, and C. Tomlin, "Verification of hybrid systems: Application to user-interfaces," NASA Ames Research Center, Moffett Field, CA, Tech. Memo. 212803, 2003.
[60] A. Bayen and C. Tomlin, "Nonlinear hybrid automaton model for aircraft landing," Dept. Aeronautics and Astronautics, Stanford Univ., Stanford, CA, SUDAAR 737, 2001.
[61] S. Rogers, K. Roth, H. Cao, J. Slotnick, M. Whitlock, S. Nash, and M. Baker, "Computation of viscous flow for a boeing 777 aircraft in landing configuration," presented at the AIAA Conf., Denver, CO, 1992.
[62] J. Roskam and C.-T. Lan, *Airplane Aerodynamics and Performance*. Lawrence, KS: Design, Analysis, and Research Corporation, 1997.
[63] A. Flaig and R. Hilbig, "High-lift design for large civil aircraft," in *AGARD CP 515*, 1992, pp. 31-1–31-12.
[64] L. Jenkinson, P. Simpkin, and D. Rhodes, *Civil Jet Aircraft Design*. Reston, VA: American Institute of Aeronautics and Astronautics, Inc., 1999.
[65] I. Kroo, "Aircraft Design: Synthesis and Analysis, Version 0.99" Desktop Aeronautics, Stanford, CA, 2001.
[66] C. Hynes, Personal communication, Aug. 2001.
[67] T. Lambregts, "Automatic flight control: Concepts and methods," FAA National Resource Specialist, Advanced Controls, 1995.

Meeko Oishi (S'99–M'04) received the M.S. and Ph.D. degrees in mechanical engineering from Stanford University, Stanford, CA, in 2000 and 2004, respectively, and the B.S.E. in mechanical engineering from Princeton University, Princeton, NJ, in 1998.

She is an Assistant Professor with the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC, Canada. She has held postdoctoral positions with Sandia National Laboratories (2005) and with the National Science Foundation (NSF) National Ecological Observatory Network (NEON) (2004).

She has been a Science and Technology Policy Fellow at The National Academies (2004), a Visiting Researcher at NASA Ames Research Center (2001–2003) and Honeywell Technology Center (2000), and a summer intern with Boeing, Intel, and Sandia National Laboratories. Her research interests include nonlinear dynamical systems, hybrid control theory, user-interface analysis, and reachability analysis.

Dr. Oishi was a recipient of the Truman Postdoctoral Fellowship in National Security Science and Engineering in 2005, the NSF Graduate Research Fellowship (1998–1999 2000–2002), and the John Bienkowski Memorial Prize from Princeton University in 1998.

Ian M. Mitchell (S'97–M'03) received the B.A.Sc. in engineering physics and the M.Sc. degree in computer science from the University of British Columbia, Vancouver, BC, Canada, in 1994 and 1997, respectively, and the Ph.D. degree in scientific computing and computational mathematics from Stanford University, Stanford, CA, in 2002.

After spending a year as a Postdoctoral Researcher with the Department of Electrical Engineering and Computer Science, University of California, Berkeley, and the Department of Computer Science, Stanford University, in August, 2003, he became an Assistant Professor with the Department of Computer Science, University of British Columbia. He is the author of the *Toolbox of Level Set Methods*, the first publicly available high accuracy implementation of solvers for dynamic implicit surfaces and the time dependent Hamilton–Jacobi equation that works in arbitrary dimension. His research interests include scientific computing, hybrid systems, verification and robot path planning.

Dr. Mitchell was a recipient of a 1999 SIAM/AAAS Mass Media Fellowship and a 1997–1998 Stanford School of Engineering Graduate Fellowship.

Alexandre M. Bayen (S'02–M'04) received the B.S. degree in applied mathematics from the Ecole Polytechnique, Palaiseau, France, in 1998, the M.S. and the Ph.D. degrees in aeronautics and astronautics from Stanford University, Stanford, CA, in 1999 and 2004, respectively.

Since March 2005, he has been an Assistant Professor with the Department of Civil and Environmental Engineering, University of California at Berkeley, Berkeley. From 2001 to 2003, he was a Visiting Researcher with NASA Ames, Moffett Field, CA. From 2004 to 2005, he held the rank of Major with the Department of Defense, France. During that time, he was the Research Director of the Laboratoire de Navigation Autonome, Laboratoire de Recherches Balistiques et Aérodynamiques, Vernon, France. His research interests include control of distributed parameter systems, combinatorial optimization, hybrid systems and air traffic automation.

Mr. Bayen was a recipient of the Graduate Fellowship of the Délégation Générale pour l'Armement (1998–2002) from France and of the Ballhaus Prize for best doctoral thesis from the Department of Aeronautics and Astronautics, Stanford University (2004).

Claire J. Tomlin (S'93–M'99) received the M.Sc. degree from Imperial College London, London, U.K., in 1993, the B.A.Sc. degree from the University of Waterloo, Waterloo, ON, Canada, in 1992, and the Ph.D. degree from the University of California at Berkeley, Berkeley, in 1998, all in electrical engineering.

She is a Professor with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, and with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA. She has held Visiting Research Positions with NASA Ames, Honeywell Labs, and the University of British Columbia. Her research interests include control systems, specifically hybrid control theory, and she works on air traffic control automation, flight management system analysis and design, and modeling and analysis of biological cell networks.

Prof. Tomlin was a recipient of the MacArthur Fellowship (2006), the Eckman Award of the American Automatic Control Council (2003), the MIT Technology Review's Top 100 Young Innovators Award (2003), the AIAA Outstanding Teacher Award (2001), an NSF Career Award (1999), a Terman Fellowship (1998), and the Bernard Friedman Memorial Prize in Applied Mathematics (1998).