Proceedings of the 41st IEEE
Conference on Decision and Control
Las Vegas, Nevada USA, December 2002

Hybrid Verification of an Interface for an Automatic Landing[1]

Meeko Oishi, Ian Mitchell, Alexandre Bayen, Claire Tomlin
Hybrid Systems Lab, Stanford University, Stanford, CA
moishi,imitchell,bayen,tomlin@stanford.edu

Asaf Degani
NASA Ames Research Center, Moffett Field, CA
adegani@mail.arc.nasa.gov

Abstract

Modern commercial aircraft have extensive automation which helps the pilot by performing computations, obtaining data, and completing procedural tasks. The pilot display must contain enough information so that the pilot can correctly predict the aircraft's behavior, while not overloading the pilot with unnecessary information. Human-automation interaction is currently evaluated through extensive simulation. In this paper, using both hybrid and discrete-event system techniques, we show how one could mathematically verify that an interface contains enough information for the pilot to safely and unambiguously complete a desired maneuver. We first develop a nonlinear, hybrid model for the longitudinal dynamics of a large civil jet aircraft in an autoland/go-around maneuver. We find the largest controlled subset of the aircraft's flight envelope for which we can guarantee both safe landing and safe go-around. We abstract a discrete procedural model using this result, and verify a discrete formulation of the pilot display against it. An interface which fails this verification could result in nondeterministic or unpredictable behavior from the pilot's point of view.

1 Introduction

One of the key enabling technologies for increased automation in human-machine systems is *verification*, which allows for heightened confidence that the system will perform as desired. To verify system *safety*, the safety specification is first represented as a desired subset of the state space in which the system should remain. 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. We can restrict system behavior by pruning away system trajectories which lead to unsafe states, to synthesize a controller which, if enforced, guarantees safety.

In the past several years, a method [1] and a numerical tool [2, 3] have been developed for verifying the safety of hybrid systems. Previous work, for example [4], 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 *semi-automated* systems, in which the automaton and a human controller share authority over the control of the system [5]. 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?* We consider this problem within the framework of an example: the automatic landing system (autoland) of a large civil jet airliner.

The autoland system of modern aircraft is one of the most safety-critical components, and is subject to stringent certification criteria [6]. 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. Most of the information is abstracted away, so that only a subset of this information is displayed to the pilot. Here, we are interested in verifying that the cockpit interface provides the pilot with enough information so that the pilot can safely land or safely go-around.

[1]Research supported by a National Science Foundation 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).

0-7803-7516-5/02/$17.00 ©2002 IEEE

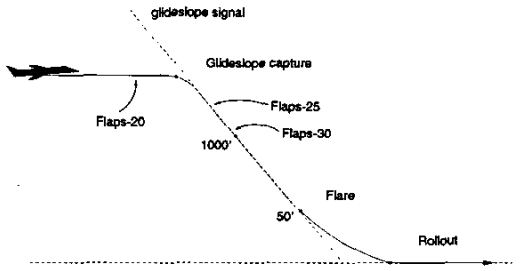

Figure 1: Typical landing scenario.

The pilot's "user model" of the autoland system, based on the pilot's display, manuals, training, and personal experience, is necessarily different from the complete aircraft "truth model" [7]. Discrepancies between these models can result in mode confusion, a potentially unsafe situation in which the system does not behave as the pilot anticipates [5, 8, 9]. Although the human factors community has historically dominated research on human-automation interaction [9, 10, 11, 12], there have recently been efforts by the formal methods community [7, 13, 14, 15, 16] as well as system and control communities [17] to address these safety-critical problems. We build our methodology based on [7], in which user-interfaces are verified for a given task. In [7] the hybrid plant model is represented as an abstracted discrete system in which the system dynamics are modeled as plant-triggered (dynamic) transitions. It is not shown there how the discrete representation with its dynamic transitions are derived. In the present work, we represent the plant model as an explicit hybrid system and show how, with the aid of a control component, the detailed transformation into the equivalent discrete representation is performed.

In this paper, we first develop a model of longitudinal aircraft dynamics in high-lift configurations used during a landing/go-around procedure. Using a computational tool for hybrid systems, we find the largest controllable set for which we can guarantee the aircraft can both safely land and safely go-around. We apply the control law synthesized from this computation, and formulate a new, safe hybrid automaton. From this automaton, we abstract a discrete event system which represents operation in the regions which result in safe landing or go-around maneuvers. We formulate the interface as a discrete event system, as well. Using the verification techniques described in [7], we verify the interface against the abstracted procedural model. Lastly, we discuss implications of our results and directions for future work.

2 Problem Description

In a typical autoland maneuver (Figure 1), the aircraft descends towards the glideslope, an inertial beam which the aircraft can track. With the landing gear down, the pilot sets the flaps at Flaps-20, the first high-lift configuration in the landing sequence. After capturing the glideslope signal, the pilot increases flap deflection, stepping through both Flaps-25 and Flaps-30 by the time the aircraft reaches 1000' altitude. Near 50', the aircraft leaves the glideslope and begins a flare maneuver, which allows the aircraft to touchdown smoothly on the runway with an appropriate descent rate.

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 gusts, for example), the pilot must initiate a go-around maneuver. A go-around can be initiated anytime after the glideslope has been captured and before the aircraft touches down. Pushing the go-around button engages a sequence of events designed to make the aircraft climb as quickly as possible to a dialed-in missed-approach altitude which the pilot usually sets to 2500'.

2.1 Aerodynamic Characteristics

The phases of landing and go-around correspond to fundamentally different operating conditions of the aircraft. We model the nonlinear longitudinal dynamics of a large civil jet aircraft by $\dot{x} = f_i(x, u)$, in which the state $x = [V, \gamma, h] \in \mathbb{R}^3$ includes the aircraft's speed V, flightpath angle γ, and altitude h (see [18]):

$$\begin{bmatrix} m\dot{V} \\ mV\dot{\gamma} \\ \dot{h} \end{bmatrix} = \begin{bmatrix} -D(\alpha, V) + T\cos\alpha - mg\sin\gamma \\ L(\alpha, V) + T\sin\alpha - mg\cos\gamma \\ V\sin\gamma \end{bmatrix} \quad (1)$$

We assume the control input $u = [T, \alpha]$, with aircraft thrust T and angle of attack α. The aircraft has mass $m = 190000$ kg, pitch $\theta = \alpha + \gamma$, and gravitational acceleration is $g = 9.81$ m/s^2. The aircraft's lift $L(\alpha, V) = \frac{1}{2}\rho V^2 S C_L(\alpha)$ and drag $D(\alpha, V) = \frac{1}{2}\rho V^2 S C_D(\alpha)$ 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 [18, 19, 20, 21, 22] (Table 1). $C_{L_\alpha} = 5.105$ in all modes.

2.2 Procedural Automaton

The discrete modes of our hybrid system result from the combination of aircraft dynamics and autopilot modes. We formulate a hybrid *procedural* model based on landing/go-around procedures a pilot is trained to follow. We focus on a small part of the autoland procedure, beginning with the flare maneuver.

i	C_{L_0}	C_{D_0}	K	Flaps Setting	Landing Gear
1	0.4225	0.024847	0.04831	Flaps-20	Down
2	0.7043	0.025151	0.04831	Flaps-25	Down
3	0.8212	0.025455	0.04831	Flaps-30	Down
4	0.4225	0.019704	0.04589	Flaps-20	Up
5	0.7043	0.020009	0.04589	Flaps-25	Up
6	0.8212	0.020313	0.04589	Flaps-30	Up

Table 1: Aerodynamic constants for autoland modes indexed by $\dot{x} = f_i(x, u)$.

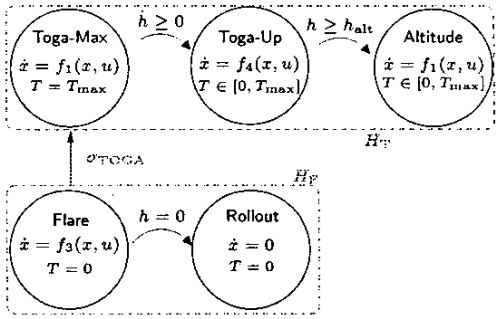

Figure 2: Hybrid procedural automaton $H_{\text{procedure}}$.

The initial state of our procedural model $H_{\text{procedure}}$ (Figure 2) is Flare, with flaps at Flaps-30 and thrust fixed at idle. As instructed, when a pilot initiates a go-around maneuver (often called a "TOGA" due to the "Take-Off/Go-Around" indicator on the pilot controls and display), the pilot changes the flaps to Flaps-20 and the autothrottle forces the thrust to $T_{\max}$ (Toga-Max). When the aircraft obtains a positive rate of climb, the pilot raises the landing gear, and the autothrottle allows $T \in [0, T_{\max}]$ (Toga-Up). The aircraft continues to climb to the missed approach altitude h_{alt}, then switches into an altitude-holding mode, Altitude (with the landing gear down). If a go-around does not occur, the aircraft switches to Rollout when it lands. (We do not model the aircraft's behavior after touchdown.)

Although go-arounds are unpredictable and may be required at any time during the autoland prior to touchdown, σ_{TOGA} is a controlled transition because the pilot must initiate the go-around for it to occur. Certain events occur simultaneously: changing the flaps to Flaps-30 and event σ_{TOGA}, raising the landing gear and $\dot{h} \geq 0$, and lowering the landing gear and $h \geq h_{\text{alt}}$.

2.3 State and Input Bounds

Each mode in the procedural automaton is subject to state and input bounds, due to constraints arising from aircraft aerodynamics and desired aircraft behavior. These bounds, shown in Table 2, form the boundary of the flight envelope $\mathcal{W}_0$. Bounds on V and α are determined by stall speeds and structural limitations for each flap setting [22]. Bounds on γ and T are determined by the desired maneuver [23]. 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.

Mode	V [m/s]	γ [degrees]	α [degrees]
Flare	[55.57, 87.46]	[−6.0°, 0.0°]	[−9°, 15°]
Toga-Max	[63.79, 97.74]	[−6.0°, 0.0°]	[−8°, 12°]
Toga-Up	[63.79, 97.74]	[0.0°, 13.3°]	[−8°, 12°]
Altitude	[63.79, 97.74]	[−0.7°, 0.7°]	[−8°, 12°]

Table 2: State bounds for autoland modes of $H_{\text{procedure}}$.

3 Safety Analysis

The state bounds just described define flight envelopes for each of the discrete modes. These envelopes are not necessarily controlled invariant. Thus, we need to determine what subsets of these envelopes are actually controllable given the input authority available to the autopilot. Because the nonlinear dynamics of our model (1) make analytic determination of the controllable subsets impossible, we employ a previously developed computational algorithm for finding controlled invariant sets for this problem [3].

3.1 Computing Reachable Sets

For each discrete mode of the autoland system, we define the target set as the region outside the flight envelope $\mathcal{W}_0$, denoted $(\mathcal{W}_0)^c$ for the complement of $\mathcal{W}_0$. Given some dynamically evolving system and some target set, we define the backward reachable set $\mathcal{W}^c(t)$ as the set of all system states which reach the target set in time t. The autopilot inputs α and T try to drive the state away from the target set, to keep the aircraft within $\mathcal{W}_0$.

Computing the reachable set in a discrete system with a finite number of states—and hence a finite number of possible transitions—is a straightforward but possibly time consuming task of enumerating all the states which have a path to the target set. Computing reachable sets for a continuous system is a much more difficult undertaking; for example, how should the uncountably many states in any nontrivial target set be represented?

An algorithm has been developed for computing the reachable sets of continuous nonlinear systems, based on a time dependent Hamilton-Jacobi (HJ) partial differential equation (PDE) [2, 3]. For $\dot{x} = f(x, u)$, $x \in \mathbb{X}$, input $u \in \mathcal{U}$ tries to keep the system from reaching the target set. Define a continuous function $J_0 : \mathbb{X} \to \mathbb{R}$ such that $(\mathcal{W}_0)^c = \{x \in \mathbb{X} | J_0(x) \leq 0\}$. As shown in [2],

by solving the terminal value HJ PDE

$$D_t J(x,t) + \min[0, H(x, D_x J(x,t))] = 0 \quad \text{for } t < 0;$$
$$J(x,0) = J_0(x) \quad \text{for } t = 0; \tag{2}$$

where $H(x,p) = \max_{u \in \mathcal{U}} p^T f(x,u)$, for the function $J : \mathbb{X} \times (-\infty, 0] \to \mathbb{R}$, we obtain an implicit representation of the reachable set $\mathcal{W}^c(t) = \{x \in \mathbb{X} | J(x,-t) \leq 0\}$. The state-dependent control synthesized from this calculation is $u^*(x) = \arg\max_{u \in \mathcal{U}} p^T f(x,u)$.

Analytically solving (2) for a general $J_0(x)$ and $f(x,u)$ is likely to be impossible. Computational algorithms are complicated by the fact that for even smooth $J_0(x)$ and $f(x,u)$, the solution $J(x,t)$ can develop discontinuities in its derivatives after finite time, and hence cease to solve (2) in a classical sense. The appropriate weak solution is the viscosity solution [24], and level set algorithms [25] are numerical techniques developed to compute such solutions. A set of high resolution schemes [26] have been designed and implemented [3] to compute $J(x,t)$, and hence the boundary of the reachable set $\mathcal{W}^c(t)$, very accurately.

3.2 Computing Controllable Flight Envelopes

In any given mode of $H_{\text{procedure}}$, the aircraft should remain within its flight envelope $\mathcal{W}_0$. To determine the maximal controllable subset $\mathcal{W}$ of $\mathcal{W}_0$, we run a reachable set computation. The reachable set typically converges to a fixed point: $\mathcal{W}^c(t) \to \mathcal{W}^c$ as $t \to +\infty$. We call $\mathcal{W}$ the *safe* flight envelope. Yet the full autopilot system contains transitions between modes, and so we cannot examine any mode in isolation.

We separate the hybrid procedural model across the user-controlled switch σ_{TOGA} into two hybrid subsystems, H_{F} and H_{T}, shown in Figure 2. Computationally, automatic transitions are smoothly accomplished by concatenating modes across the switch, so that the change in dynamics across the switching surface is modeled as another nonlinearity in the dynamics. Additionally, we assume in H_{T} that if the aircraft leaves the top of the computational domain ($h = 20$ m) without exceeding its flight envelope, it is capable of reaching Altitude mode, which we consider to be completely safe.

The initial flight envelopes $(\mathcal{W}_{\text{F}})_0$ and $(\mathcal{W}_{\text{T}})_0$ are determined by state bounds on each mode given in Table 2. We perform the reachability computation on H_{F} and H_{T} to obtain the safe flight envelopes $\mathcal{W}_{\text{F}}$ and $\mathcal{W}_{\text{T}}$. Figure 3 shows $\mathcal{W}_{\text{F}}$, and Figure 4 shows $\mathcal{W}_{\text{T}}$ in Toga-Up and Toga-Max modes. (Note that the boundary of $\mathcal{W}_{\text{F}}$ along $\gamma = 0$ corresponds with the transition boundary of $\mathcal{W}_{\text{T}}$ between Toga-Up and Toga-Max, $\dot{h} = 0$.)

Figure 5 shows the continuous region $\mathcal{W}_{\text{F}} \cap \mathcal{W}_{\text{T}}$ from which we can guarantee both a safe landing and a safe go-around. Notice that this set is smaller than $\mathcal{W}_{\text{F}}$, the region from which a safe landing is possible: the pilot is further restricted in executing a go-around. There are states from which a safe landing is possible, but a safe go-around is not.

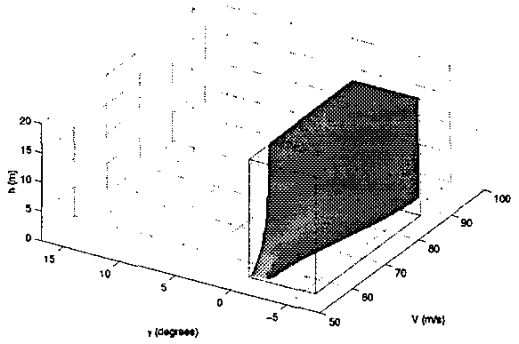

Figure 3: Safe region $\mathcal{W}_F$; the outer box is $(\mathcal{W}_F)_0$.

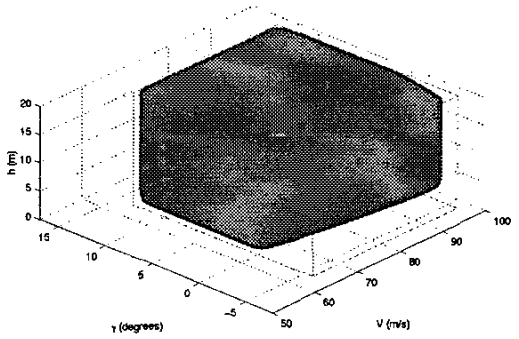

Figure 4: Safe region $\mathcal{W}_T$: the outer box is $(\mathcal{W}_T)_0$.

4 Interface Verification

A general verification technique for analyzing interfaces has been sought for many years. The need was motivated by serious incidents and accidents, involving human interaction with complex automated systems (e.g., cockpit automation). Recently, a theory, methodology, and a detailed verification procedure was developed by researchers at NASA [7, 16]. The methodology considers four elements: the machine model, user model, interface model, and the task specification (e.g., safe/unsafe, multiple modes). In this section we use the methodology of [7] in the context of this hybrid system example.

In most commercial aircraft, the low-level control is performed by the autopilot, and the pilot anticipates system behavior by understanding the behavior of each autopilot mode. We assume an automated controller enforces $u = u^*(x)$ within each hybrid subsystem. By doing so, we mimic the supervisory role pilots have in

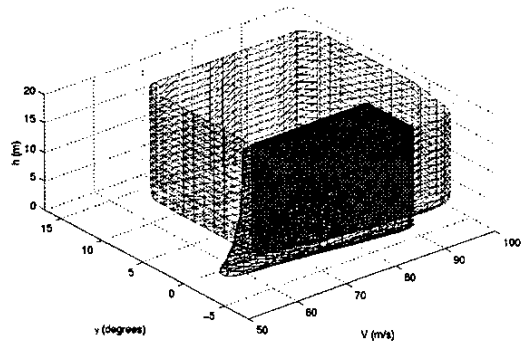

Figure 5: The solid shape is the safe region $\mathcal{W}_F \cap \mathcal{W}_T$, from which safe landing and safe go-around is possible. The meshes depict $\mathcal{W}_F$ and $\mathcal{W}_T$.

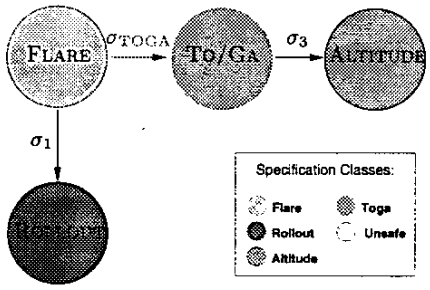

Figure 6: $G_{\text{interface}}$ for autoland/go-around maneuver. Event σ_1 occurs when $h = 0$, σ_3 when $h \geq h_{\text{alt}}$.

highly automated aircraft, including the option not to enforce a recommended switch.

The pilot activates various knobs, buttons, and toggles to change the system's mode. Interaction between the pilot's actions and the system's modes are encapsulated by a finite-state machine representation of the interface $G_{\text{interface}} = (Q_{\text{interface}}, \Sigma_{\text{interface}}, \delta_{\text{interface}})$. Modes $Q_{\text{interface}}$ are determined by the indications on the display; events $\Sigma_{\text{interface}}$ are determined by internal transitions in the system, or by the pilot's actions. The transition function is $\delta_{\text{interface}}$. The interface for an autoland/go-around is shown in Figure 6.

To compare the interface against the procedural model, we implement the controller for safety $u^*(x)$ in $H_{\text{procedure}}$ and create a discrete abstraction $G^*_{\text{procedure}}$ based on the resultant closed-loop hybrid system. We partition the state-space in each mode into the interior, boundary, and complement of the safe flight envelope in that particular mode. Across the user-controlled switch σ_{TOGA}, we partition the state space according to the intersection of $\mathcal{W}_F$ in Flare and $\mathcal{W}_T$ in Toga-Up, resulting in nine regions in each mode. Across all other switches in H_F and H_T, we enforce safety by implementing $u^*(x)$ so that trajectories which begin inside or on the boundary of the safe flight envelope in one mode will remain within or on the boundary of the safe flight envelope in all other modes in that hybrid subsystem. Only across user-controlled switches can the system become unsafe, because we can make no guarantees about the user's actions. $G^*_{\text{procedure}}$ has modes $Q^*_{\text{procedure}}$, events $\Sigma^*_{\text{procedure}}$, and transition function $\delta^*_{\text{procedure}}$.

We verify the correspondence between $G_{\text{interface}}$ and $G^*_{\text{procedure}}$ according to the verification methodology in [7]. We associate each mode in $Q_{\text{interface}}$ and $Q^*_{\text{procedure}}$ to a certain *specification class* [7]. Specification classes are a way of indicating a type of behavior or quality of the system – for example, modes which the system should avoid belong to a specification class **Unsafe**.

The interface and the abstracted procedural model are related through their events: events in $\Sigma^*_{\text{procedure}}$ map to events in $\Sigma_{\text{interface}}$. We define the map through $\Sigma^*_{\text{procedure}} \xrightarrow{\pi} \Sigma_{\text{interface}}$, by examining the events in each set and creating a correspondence between them by hand [7]. Events in $\Sigma^*_{\text{procedure}}$ which do not have a corresponding transition in $\Sigma_{\text{interface}}$ map to the empty event ε [7].

The two systems are verified through the creation of a composition, defined by the map π. The composition $G_{\text{composition}}$ allows us to keep track of the modes and events in both systems ($G_{\text{interface}}$ and $G^*_{\text{procedure}}$) at the same time. The process of creating the composition uncovers possible problems: *error* states, *blocking* states, and *augmented* states [7].

The composition begins with each initial state in each system for a given specification class, and is repeated for each pair of initial states. If each event α in $G^*_{\text{procedure}}$ such that $p \xrightarrow{\alpha} p'$ has a corresponding event $\pi(\alpha)$ in $G_{\text{interface}}$ such that $q \xrightarrow{\pi(\alpha)} q'$, then the composite state $(p, q) \xrightarrow{\pi(\alpha)} (p', q')$ exists. If p and q have the same specification class, and p' and q' have the same specification class, then the composition continues throughout the model. An *error state* exists when p' and q' have different specification classes [7].

Other problems occur when the composition fails. If for a transition $\alpha \in \Sigma^*_{\text{procedure}}$ from $p \xrightarrow{\alpha} p'$ there is no corresponding transition $q \xrightarrow{\pi(\alpha)} q'$, then the composition has reached a *blocking state* [7]. (The interface blocks a transition which occurs in the abstracted procedural model.) Alternatively, if there is a transition $\pi(\alpha) \in \Sigma_{\text{interface}}$ from $q \xrightarrow{\pi(\alpha)} q'$ but no corresponding transition $\alpha \in \Sigma^*_{\text{procedure}}$ from $p \xrightarrow{\alpha} p'$, then the composition has reached an *augmented state* [7]. (The interface indicates a transition which is not possible in the abstracted procedural model.)

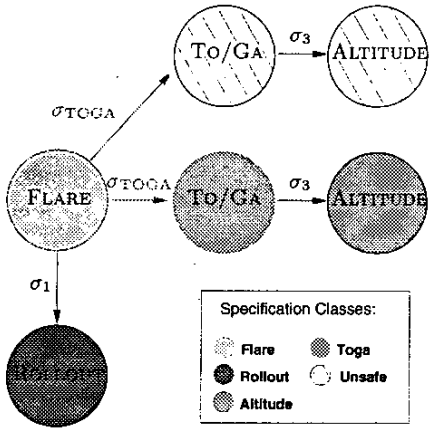

Figure 7: Nondeterministic behavior from the pilot's point of view.

If the composition fails (due to blocking or augmented states), or if the composition contains error states (due to mismatched specification classes) then the interface is *not* an adequate representation of the procedural model [7].

Following this process for the autoland example, we find that the composition $G_{\text{composition}}$ contains error states. For example, if the pilot initiates a go-around when the aircraft is in $\mathcal{W}_F^o \cap \mathcal{W}_T^o$ in FLARE, $G_{\text{composition}}$ reveals specification classes (**Flare,Flare**) $\xrightarrow{\sigma_{TOGA}}$ (**Toga, Toga**). However, if the pilot initiates a go-around from $\mathcal{W}_F^o \cap W_T^c$ in FLARE, $G_{\text{composition}}$ reveals specification classes (**Flare, Flare**) $\xrightarrow{\sigma_{TOGA}}$ (**Unsafe,Toga**). From the pilot's point of view, error states appear as nondeterminism: the aircraft sometimes behaves as the pilots expects, but sometimes does not, as shown in Figure 7.

5 Implications and Conclusion

There is an ongoing debate in aviation, space, and other safety-critical industries about the role of the operator and the extent to which automation can and should be used. This debate has been fueled by incidents and accidents in which pilots were surprised about the behavior of the automation. While the debate will continue, it is clear that some of the problems in human-automation interaction stem from design problems. Interface verification methods are critical for identifying design problems early on in the design phase. Current efforts at NASA are aimed at developing methods for extracting the machine, interface, and user-models from Java code and then applying the interface verification method of [7] to identify error states.

Verification within a hybrid framework allows us to account for the inherently complicated dynamics underlying the simple, discrete representations displayed to the pilot. In this example, in order to safely supervise the system, the pilot must have enough information to know *before* entering a go-around maneuver whether or not the aircraft will remain safe.

The interface verification methodology begins with a procedural model, a hybrid system which incorporates discrete mode logic as well as nonlinear continuous dynamics. The hybrid safety computation provides us with continuous control restrictions, which, if enforced, guarantee that the system will always remain safe. This guarantee holds to within the accuracy of our model. We abstract a discrete event system from this hybrid system with safety restrictions. To do so, we partition the continuous states of the hybrid system with safety restrictions according to their location in safe or unsafe regions in each mode. This abstraction, along with the formulation of the interface model as a discrete event system, allows us to use existing interface verification techniques [7]. We compare the discrete interface and procedural models by analyzing their composition for error, blocking, and augmented states, which result in confusing and unpredictable behavior from the pilot's point of view.

The methodology presented here also extends to systems with disturbances, such as wind or an engine failure. While verification tools can aid design, we also hope to contribute directly to the design problem (as in [27]), within a hybrid framework.

6 Acknowledgements

Thanks to Michael Heymann, David Austin, Randall Mumaw, and Charles Hynes for their invaluable assistance and input.

References

[1] C. Tomlin, J. Lygeros, and S. Sastry, "A game theoretic approach to controller design for hybrid systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 949–970, 2000.

[2] I. Mitchell, A. Bayen, and C. Tomlin, "Validating a Hamilton-Jacobi approximation to hybrid system reachable sets," in *Hybrid Systems: Computation and Control* (M. D. Benedetto and A. Sangiovanni-Vincentelli, eds.), LNCS 2034, pp. 418–432, Springer Verlag, March 2001.

[3] I. Mitchell, A. M. Bayen, and C. J. Tomlin, "Computing reachable sets for continuous dynamic games using level set methods," *IEEE Transactions on Automatic Control*. Submitted, December 2001.

[4] M. Oishi, C. Tomlin, V. Gopal, and D. Godbole, "Addressing multiobjective control: Safety and performance through constrained optimization," in *Hybrid Systems: Computation and Control* (M. D. Benedetto and A. Sangiovanni-Vincentelli, eds.), LNCS 2034, pp. 459–472, Springer Verlag, March 2001.

[5] A. Degani, M. Shafto, and A. Kirlik, "Modes in human-machine systems: Constructs, representation, and classification," *International Journal of Aviation Psychology*, vol. 9, no. 2, pp. 125–138, 1999.

[6] Federal Aviation Administration, "Criteria for approval of Category III weather minima for takeoff, landing, and rollout," Advisory Circular 120-28D, U.S. Department of Transportation, July 1999.

[7] A. Degani and M. Heymann, "Formal verification of human-automation interaction," *Human Factors*, vol. 44, no. 1, pp. 28–43, 2002.

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

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

[10] R. Parasuraman, T. Sheridan, and C. Wickens, "A model for types and levels of human interaction with automation," *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, vol. 30, May 2000.

[11] C. Billings, *Aviation Automation: The Search for a Human-Centered Approach.* Hillsdale, NJ: Erlbaum, 1997.

[12] E. Wiener, "The human factors of advanced technology ("glass cockpit") transport aircraft," NASA Contractor Report 177528, NASA Ames Research Center, Moffett Field, CA, 1989.

[13] J. Rushby, "Using model checking to help discover mode confusions and other automation surprises," in *Proceedings of the Workshop on Human Error, Safety, and System Development (HESSD)*, (Belgium), June 1999.

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

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

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

[17] S. Vakil, A. Midkiff, T. Vaneck, and R. Hansman, "Mode awareness in advanced autoflight systems," in *Proceedings of the 6th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Man-Machine Systems*, (Cambridge, MA), 1995.

[18] A. Bayen and C. Tomlin, "Nonlinear hybrid automaton model for aircraft landing," SUDAAR 737, Dept. of Aeronautics and Astronautics, Stanford University, Stanford, CA, 2001.

[19] 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," in *AIAA Conference Proceedings*, no. 2000-4221, October 1992.

[20] J. Roskam and C.-T. Lan, *Airplane Aerodynamics and Performance.* Lawrence, Kansas: Design, Analysis, and Research Corporation, 1997.

[21] A. Flaig and R. Hilbig, "High-lift design for large civil aircraft," in *AGARD Conference Proceedings 515*, (France), October 1992.

[22] L. Jenkinson, P. Simpkin, and D. Rhodes, *Civil Jet Aircraft Design.* Reston, VA: American Institute of Aeronautics and Astronautics, Inc., 1999. http://www.bh.com/companions/aerodata.

[23] T. Lambregts, "Automatic flight control: Concepts and methods." FAA National Resource Specialist, Advanced Controls, 1995.

[24] M. G. Crandall, L. C. Evans, and P.-L. Lions, "Some properties of viscosity solutions of Hamilton-Jacobi equations," *Transactions of the American Mathematical Society*, vol. 282, no. 2, pp. 487–502, 1984.

[25] S. Osher and J. A. Sethian, "Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations," *Journal of Computational Physics*, vol. 79, pp. 12–49, 1988.

[26] S. Osher and R. Fedkiw, *The Level Set Method and Dynamic Implicit Surfaces.* Springer-Verlag, 2002.

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