Computational Techniques for the Verification and Control of Hybrid Systems

Claire J. Tomlin[1], Ian M. Mitchell, Alexandre M. Bayen[1], and Meeko K. M. Oishi[1]

[1] Hybrid Systems Laboratory, Department of Aeronautics and Astronautics Stanford University, Stanford, CA 94305-4035
[2] Computer Science Department, University of British Columbia, Vancouver, BC, CANADA, V6T 1Z4

1 Introduction

Hybrid systems theory lies at the intersection of the two traditionally distinct fields of computer science verification and engineering control theory. It is loosely defined as the modeling and analysis of systems which involve the interaction of both discrete event systems (represented by finite automata) and continuous time dynamics (represented by differential equations). The goals of this research are in the design of verification techniques for hybrid systems, the development of a software toolkit for efficient application of these techniques, and the use of these tools in the analysis and control of large scale systems. In this paper, we present a summary of recent research results, and a detailed set of references, on the development of tools for the verification of hybrid systems, and on the application of these tools to some interesting examples.

The problem that has received much recent research attention has been the verification of the *safety* property of hybrid systems, which seeks a mathematically precise answer to the question: is a potentially unsafe configuration, or state, reachable from an initial configuration? For discrete systems, this problem has a long history in mathematics and computer science and may be solved by posing the system dynamics as a discrete game [1, 2]; in the continuous domain, control problems of the safety type have been addressed in the context of differential games [3]. For systems involving continuous dynamics, it is very difficult to compute and represent the set of states reachable from some initial set. In this lecture, we present recent solutions to the problem, including a method, based on the level set techniques of Osher and Sethian [4], which determines an implicit representation of the boundary of this *reachable set*. This method is based on the theorem, which is proved in [5] using two-person zero-sum game theory for continuous dynamical systems, that the solution to a particular Hamilton-Jacobi partial differential equation corre-

sponds exactly to the boundary of the reachable set. In addition, we show that useful information for the control of such systems can be extracted from this boundary computation.

Much of the excitement in hybrid system research stems from the potential applications. With techniques such as the above, it is now possible to verify, and design safe, automated control schemes for low dimensional systems. We present two interesting examples in the verification of protocols for aircraft collision avoidance, and of mode switching logic in autopilots. We survey other applications that have been studied in this framework.

We conclude with a discussion of problem complexity and new directions that will enable treatment of problems of higher dimension.

The material in this paper is based on the hybrid system algorithm of [6], the level set implementation of [5, 7], the aircraft landing example of [8], and the interface analysis example of [9, 10]. It was presented as a lecture in the "Summer School Jacques Louis Lions", held during March 17-22 2003 in Montecatini, Italy.

2 Hybrid Model and Verification Methodology

2.1 Continuous, Discrete, and Hybrid Systems

Much of control theory is built around continuous-state models of system behavior. For example, the differential equation model given by

$$\dot{x} = f(x, u, d) \tag{1}$$

describes a system with *state* $x \in \mathbb{R}^n$ that evolves continuously in time according to the dynamical system $f(\cdot, \cdot, \cdot)$, a function of x, $u \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$, $d \in \mathcal{D} \subseteq \mathbb{R}^{n_d}$. In general, u is used to represent parameters that can be controlled, called *control inputs*, and d represents *disturbance inputs*, which are parameters that cannot be controlled, such as the actions of another system in the environment. The initial state $x(0) = x_0$ is assumed to belong to a set $X_0 \subseteq \mathbb{R}^n$ of allowable initial conditions. A *trajectory* of (1) is represented as $(x(t), u(t), d(t))$, such that $x(0) \in X_0$, and $x(t)$ satisfies the differential equation (1) for control and disturbance input trajectories $u(t)$ and $d(t)$. We recommend [11, 12] as current references for continuous-state control systems.

Discrete-state models, such as finite automata, are also prevalent in control. The finite automaton given by

$$(Q, \Sigma, \mathsf{Init}, R) \tag{2}$$

models a system with is a finite set of *discrete state variables* Q, a set of input variables $\Sigma = \Sigma_u \cup \Sigma_d$ which is the union of *control actions* $\sigma_u \in \Sigma_u$ and *disturbance actions* $\sigma_d \in \Sigma_d$, a set of *initial states* $\mathsf{Init} \subseteq Q$, and a *transition relation* $R : Q \times \Sigma \to 2^Q$ which maps the state and input space to subsets of

the state space (2^Q). A trajectory of (2) is a sequence of states and inputs, written as $(q(\cdot), \sigma(\cdot))$, where $q(0) \in \mathsf{Init}$ and $q(i+1) \in R(q(i), \sigma(i))$ for index $i \in \mathbb{Z}$. The original work of Ramadge and Wonham [13] brought the use of discrete state systems to control, though parallels can be drawn between this work and that of Church, Büchi and Landweber [2, 14] who originally analyzed the von Neumann-Morgenstern [1] discrete games. A comprehensive reference for modeling and control of discrete state systems is [15].

Control theory is concerned with the design of a signal, either a continuous or discrete function of time, which when applied to the system causes the system state to exhibit desirable properties. These properties should hold despite possible disruptive action of the disturbance. A concrete example of a continuous-state control problem is in the control of an aircraft: here the state (position, orientation, velocity) of the aircraft evolves continuously over time in response to control inputs (throttle, control surfaces), as well as to disturbances (wind, hostile aircraft).

A *hybrid automaton* combines continuous-state and discrete-state dynamic systems, in order to model systems which evolve both continuously and according to discrete jumps. A hybrid automaton is defined to be a collection:

$$(S, \mathsf{Init}, In, f, \mathsf{Dom}, R) \tag{3}$$

where $S = Q \cup \mathbb{R}^n$ is the union of discrete and continuous states; $\mathsf{Init} \subseteq S$ is a set of initial states; $In = (\Sigma_u \cup \Sigma_d) \cup (\mathcal{U} \cup \mathcal{D})$ is the union of actions and inputs; f is a function which takes state and input and maps to a new state, $f : S \times In \to S$; $\mathsf{Dom} \subseteq S$ is a *domain*; and $R : S \times In \to 2^S$ is a *transition relation*.

The state of the hybrid automaton is represented as a pair (q, x), describing the discrete and continuous state of the system. The continuous-state control system is "indexed" by the mode and thus may change as the system changes modes. Dom describes, for each mode, the subset of the continuous state space within which the continuous state may exist, and R describes the transition logic of the system, which may depend on continuous state and input, as well as discrete state and action. A trajectory of this hybrid system is defined as the tuple: $((q(t), x(t)), (\sigma_u(t), \sigma_d(t)), (u(t), d(t)))$ in which $q(t) \in Q$ evolves according to discrete jumps, obeying the transition relation R; for fixed $q(t)$, $x(t)$ evolves continuously according to the control system $f(q(t), x(t), (\sigma_u(t), \sigma_d(t)), (u(t), d(t)))$. The introduction of disturbance parameters to both the control system defined by f and the reset relation defined by R will allow us to treat uncertainties, environmental disturbances, and actions of other systems.

This hybrid automaton model presented above allows for general nonlinear dynamics, and is a slight simplification of the model used in [6]. This model was developed from the early control work of [16, 17, 18, 19]. The emphasis of this work has been on extending the standard modeling, reachability and stability analyses, and controller design techniques to capture the interaction between the continuous and discrete dynamics. Other approaches to modeling

hybrid systems involve extending finite automata to include simple continuous dynamics: these include timed automata [20], linear hybrid automata [21, 22, 23, 24], and hybrid input/output automata [25].

2.2 Safety Verification

Much of the research in hybrid systems has been motivated by the need to verify the behavior of safety critical system components. The problem of *safety verification* may be encoded as a condition on the region of operation in the system's state space: given a region of the state space which represents unsafe operation, *prove that the set of states from which the system can enter this unsafe region has empty intersection with the system's set of initial states.*

This problem may be posed as a property of the system's *reachable set* of states. There are two basic types of reachable sets. For a *forward reachable* set, we specify the initial conditions and seek to determine the set of all states that can be reached along trajectories that start in that set. Conversely, for a *backward reachable* set we specify a final or target set of states, and seek to determine the set of states from which trajectories start that can reach that target set. It is interesting to note that the forward and backward reachable sets are not simply time reversals of each other. The difference is illustrated

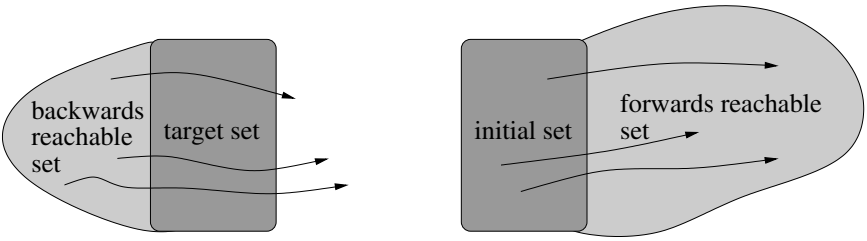

Fig. 1. Difference between backwards and forwards reachable sets.

in Figure 1 for generic target and initial sets, in which the arrows represent trajectories of the system. Figure 2 illustrates how a backwards reachable set may be used to verify system safety.

Powerful software tools for the automatic safety verification of discrete systems have existed for some time, such as Murϕ [26], PVS [27], SMV [28], and SPIN [29]. The verification of hybrid systems presents a more difficult challenge, primarily due to the uncountable number of distinct states in the continuous state space. In order to design and implement a methodology for hybrid system verification, we first need to be able to represent reachable sets of continuous systems, and to evolve these reachable sets according to the system's dynamics.

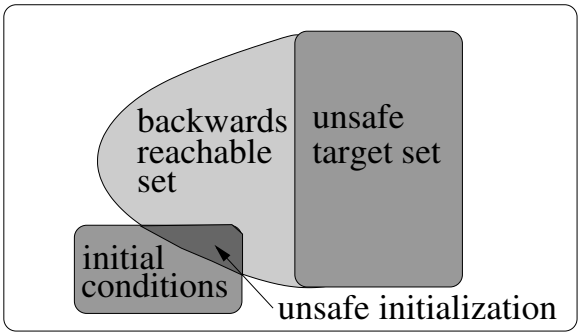

Fig. 2. Using the backwards reachable set to verify safety.

It comes as no surprise that the size and shape of the reachable set depends on the control and disturbance inputs in the system: control variables may be chosen so as to minimize the size of the backwards reachable set from an unsafe target, whereas the full range of disturbance variables must be taken into account in this computation. Thus, the methodology for safety verification has two components. The first involves computing the backward reachable set from an *a priori* specified unsafe target set; the second involves extracting from this computation the control law which must be used on the boundary of the backwards reachable set, in order to keep the system state out of this reachable set. Application of this methodology results in a system description with three simple modes (see Figure 3). Outside of the backwards reachable set, and away from its boundary, the system may use any control law it likes and it will remain safe (labeled as "safe" in Figure 3). When the system state touches the reachable set or unsafe target set boundary, the particular control law which is guaranteed to keep the system from entering the interior of the reachable set must be used. Inside the reachable set (labeled as "outside safe set" in Figure 3), there is no control law which will guarantee safety, however application of the particular optimal control law used to compute the boundary may still result in system becoming safe, if the disturbance is not playing optimally for itself.

In the following section, we first summarize different methods for computing reachable sets for continuous systems. We then provide an overview of our algorithm, which uses an implicit surface function representation of the reachable set, and a differential game theoretic method for its evolution. In the ensuing sections, we illustrate how this reachable set computation may be embedded as the key component in safety verification of hybrid systems.

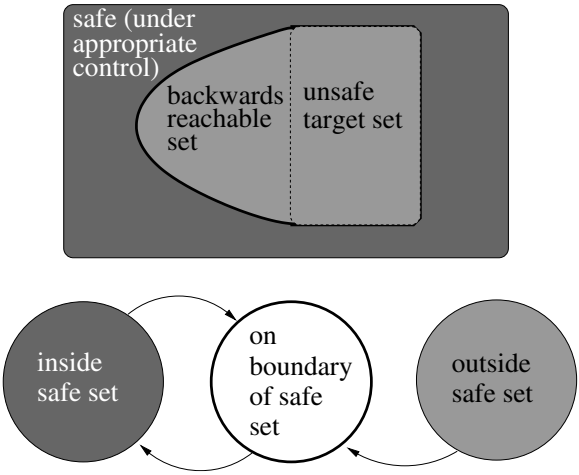

Fig. 3. A discrete abstraction with appropriate control information.

3 Verifying Continuous Systems

In this section, we review our methodology for computing reachable sets for continuous dynamic games. The backwards reachable set is the set of initial conditions giving rise to trajectories that lead to some target set. More formally, let $\mathcal{G}_0$ be the target set, $\mathcal{G}(\tau)$ be the backwards reachable set over finite horizon $\tau < \infty$, $x(\cdot)$ denote a trajectory of the system, and $x(\tau)$ be the state of that trajectory at time τ. Then $\mathcal{G}(\tau)$ is the set of $x(0)$ such that $x(s) \in \mathcal{G}_0$ for some $s \in [0, \tau]$. The choice of input values over time influences how a trajectory $x(t)$ evolves. For systems with inputs, the backwards reachable set $\mathcal{G}(\tau)$ is the set of $x(0)$ such that for every possible control input u there exists a disturbance input d that results in $x(s) \in \mathcal{G}_0$ for some $s \in [0, \tau]$ (where we abuse notation and refer interchangeably to the input signal over time and its instantaneous value).

The solution to the pursuit evasion game described in the previous section is a backwards reachable set. Let the target set be the collision set

$$\mathcal{G}_0 = \left\{ x \in \mathbb{R}^3 | \sqrt{x_1^2 + x_2^2} \leq d_0 \right\}. \tag{4}$$

Then $\mathcal{G}(\tau)$ is the set of initial configurations such that for any possible control input chosen by the evader, the pursuer can generate a disturbance input that leads to a collision within τ time units.

We use the very general implicit surface function representation for the reachable set: for example, consider the cylindrical target set (4) for the collision avoidance example. We represent this set as the zero sublevel set of a scalar function $\phi_0(x)$ defined over the state space

$$\phi_0(x) = \sqrt{x_1^2 + x_2^2} - d_0,$$
$$\mathcal{G}_0 = \left\{x \in \mathbb{R}^3 | \phi_0(x) \leq 0\right\}.$$

Thus, a point x is inside $\mathcal{G}_0$ if $\phi_0(x)$ is negative, outside $\mathcal{G}_0$ if $\phi_0(x)$ is positive, and on the boundary of $\mathcal{G}_0$ if $\phi_0(x) = 0$. Constructing this signed distance function representation for $\mathcal{G}_0$ is straightforward for basic geometric shapes. Using negation, minimum, and maximum operators, we can construct functions $\mathcal{G}_0$ which are unions, intersections, and set differences. For example, if $\mathcal{G}_i$ is represented by $g_i(x)$, then, $\min[g_1(x), g_2(x)]$ represents $\mathcal{G}_1 \cup \mathcal{G}_2$, $\max[g_1(x), g_2(x)]$ represents $\mathcal{G}_1 \cap \mathcal{G}_2$, and $\max[g_1(x), -g_2(x)]$ represents $\mathcal{G}_1 \setminus \mathcal{G}_2$.

In [5] we proved that an implicit surface representation of the backwards reachable set can be found by solving a modified HJI PDE. Using $\nabla\phi$ to represent the gradient of ϕ, the modified HJI PDE is

$$\frac{\partial\phi(x,t)}{\partial t} + \min\left[0, H(x, \nabla\phi(x,t))\right] = 0, \tag{5}$$

with Hamiltonian

$$H(x,p) = \max_{u\in\mathcal{U}} \min_{d\in\mathcal{D}} p \cdot f(x,u,d) \tag{6}$$

and terminal conditions

$$\phi(x,0) = \phi_0(x). \tag{7}$$

If $\mathcal{G}_0$ is the zero sublevel set of $\phi_0(x)$, then the zero sublevel set of the viscosity solution $\phi(x,t)$ to (5)–(7) specifies the backwards reachable set as

$$\mathcal{G}(\tau) = \left\{x \in \mathbb{R}^3 | \phi(x, -\tau) \leq 0\right\}.$$

Notice that (5) is solved from time $t = 0$ backwards to some $t = -\tau \leq 0$.

There are several interesting points to make about the HJI PDE (5)–(7). First, the $\min[0, H]$ formulation in (5) ensures that the reachable set only grows as τ increases. This formulation effectively "freezes" the system evolution when the state enters the target set, which enforces the property that a state which is labeled as "unsafe" cannot become "safe" at a future time. Second, we note that the $\max_u \min_d$ operation in computing the Hamiltonian (6) results in a solution which is not necessarily a "no regret", or saddle, solution to the differential game. By ordering the optimization so that the maximum occurs first, the control input u is effectively "playing" against an unknown disturbance – it is this order which produces a conservative solution, appropriate for the applicaton to system verification under uncertainty. Third, it is proven in [5] that out of many possible weak solutions, the viscosity solution [30] of (5)–(7) yields the reachable set boundary. The significance of this last point is that it enables us to draw from the well developed numerical schemes of the level set literature to compute accurate approximations of $\phi(x,t)$.

To compute numerical approximations of the viscosity solution to (5)–(7), we have developed a C++ implementation based on high resolution level set methods (an excellent introduction to these schemes can be found in [31]). We use a fifth order accurate weighted, essentially non-oscillatory (WENO) stencil [32, 33] to approximate $\nabla\phi(x,t)$, although we have also implemented a basic first order scheme for speed [4, 34]. We use the well studied Lax-Friedrichs (LF) approximation [35] to numerically compute Hamiltonian (6). Finally, we treat the time derivative in (5) with the method of lines and a second order total variation diminishing (TVD) Runge-Kutta scheme [36]. Numerical convergence of our algorithm is demonstrated and validated in [37, 5].

4 Verifying Hybrid Systems

In the previous section, we demonstrated the concept illustrated in Figure 3, in which the problem of verification of safety for continuous systems may be solved by a reachable set computation. This computation abstracts an uncountable number of states into the three classes: *inside safe set*, *boundary of safe set*, and *outside safe set*. We showed that this implicit surface function representation contains information which may be used for designing a safe control law. This safe control law could be used to filter any other control law as the system state approaches the reachable set boundary.

We now consider the problem of computing reachable sets for hybrid systems. Assuming that tools for discrete and continuous reachability are available, computing reachable sets for hybrid systems requires keeping track of the interplay between these discrete and continuous tools. Fundamentally, reachability analysis in discrete, continuous or hybrid systems seeks to partition states into two categories: those that are reachable from the initial conditions, and those that are not. Early work in this area focussed on decidable algorithms: it was shown that decidability results exist for timed and some classes of linear hybrid automata [38]. Software tools were designed to automatically compute reachable sets for these systems: Uppaal [39] and Kronos [40] for timed automata, and HyTech [41, 42] for linear hybrid automata. Some of these tools allow symbolic parameters in the model, and researchers began to study the problem of synthesizing values for these parameters in order to satisfy some kind of control objective, such as minimizing the size of the backwards reachable set. The procedure that we describe here was motivated by the work of [43, 44] for reachability computation and controller synthesis on timed automata, and that of [45] for controller synthesis on linear hybrid automata. Tools based on the analysis of piecewise linear systems, using mathematical programming tools such as CPLEX[46] have found success in several industrial applications.

Our hybrid system analysis algorithm [6] is built upon our implicit reachable set representation and level set implementation for continuous systems.

Thus, we are able to represent and analyze nonlinear hybrid systems, with generally shaped sets. In this sense, our work is related to that of the viability community [47, 48], which has extended concepts from viability to hybrid systems [49]; though the numerical techniques presented here differ from theirs. Other hybrid system reachability algorithms fall within this framework; the differences lie in their discrete and continuous reachability solvers and the types of initial conditions, inputs, invariants and guards that they admit. Tools such as **d/dt**, *Checkmate*, and *VeriSHIFT* have been designed using the different methods of continuous reachable set calculation surveyed in the previous section [50, 51, 52, 53, 54]: the complexity of these tools is essentially the complexity of the algorithm used to compute reachable sets in the corresponding continuous state space.

Methods for hybrid system verification listed above have found application in automotive control problems [46, 55], experimental industrial batch plants [56], vehicle collision avoidance problems [57, 58], as well as envelope protection problems [8, 59]. The problems that have been solved to date are generally of low dimension: to the best of our knowledge, the even the over-approximative methods to date have not been directly applied to systems of continuous dimension greater than 6. In the next section, we present results for envelope projection on nonlinear, hybrid systems with three continuous dimensions, representing the longitudinal dynamics of jet aircraft under hybrid control.

4.1 Computing Reachable Sets for Hybrid Systems

We describe the algorithm first with a picture, and then present the details of a few key components. The full details of the algorithm are in [6], with new implementation results presented in [7].

Consider the sequence of eight diagrams in Figure 4. We draw the hybrid automaton as a set of discrete states $\{q_1, \ldots, q_7\}$ with a transition logic represented by R (the arrows indicate the possible discrete state transitions, the dependence on continuous state and input variables is implied but not shown in the Figure). Associated to each discrete state q_i are the continuous dynamics $\dot{x} = f(q_i, x, (\sigma_u, \sigma_d), (u, d))$ and domain $\mathsf{Dom} \subseteq q_i \times \mathbb{R}^n$, neither of which are shown on the diagram. For illustrative purposes, we consider only one step of our algorithm applied in state q_1, from which there exist transitions to states q_2 and q_3 (shown in diagram 2). We initialize with the unsafe target sets (shown as sets in q_1 and q_2 in diagram 3), and sets which are known to be safe (shown as the "safe" set in q_3 in diagram 4). We augment the unsafe target set in q_1 with states from which there exists an uncontrolled transition to the unsafe set in q_2 (which is represented as a dashed arrow on diagram 5). Uncontrolled transitions may be caused by reset relations affected by disturbance actions. In the absence of other transitions out of state q_1, the set of states backwards reachable from the unsafe target set in q_1 may be computed using the reachable set algorithm of Section 3 on the dynamics

$\dot{x} = f(q_i, x(t), (\sigma_u(t), \sigma_d(t)), (u(t), d(t)))$ (diagram 6). However, there may exist regions of the state space in q_1 from which controllable transitions exist – these transitions could reset the system to a safe region in another discrete state. This is illustrated in diagram 7, with the region in which the system may "escape" to safety from q_1. Thus, the backwards reachable set of interest in this case is the set of states from which trajectories can reach the unsafe target set, without hitting this safe "escape" set first. We call this reachable set the *reach-avoid set*, and it is illustrated in diagram 8.

The algorithm illustrated above is implemented in the following way. The target set $\mathcal{G}_0 \subseteq Q \times \mathbb{R}^n$ can include different subsets of the continuous state space for each discrete mode:

$$\mathcal{G}_0 = \{(q, x) \in Q \times \mathbb{R}^n | g(q, x) \leq 0\} \tag{8}$$

for a level set function $g : Q \times \mathbb{R}^n \to \mathbb{R}$. We seek to construct the largest set of states for which the control, with action/input pair (σ_u, u) can guarantee that the safety property is met despite the disturbance action/input pair (σ_d, d).

For a given set $K \subseteq Q \times \mathbb{R}^n$, we define the *controllable predecessor* $\mathrm{Pre}_u(K)$ and the *uncontrollable predecessor* $\mathrm{Pre}_d(K^c)$ (where K^c refers to the complement of the set K in $Q \times \mathbb{R}^n$) by

$$\begin{aligned} \mathrm{Pre}_u(K) =& \{(q, x) \in K : \exists(\sigma_u, u) \in \Sigma_{\mathbf{u}} \times \mathcal{U} \\ & \forall(\sigma_d, d) \in \Sigma_{\mathbf{d}} \times \mathcal{D}\ R(q, x, \sigma_u, \sigma_d, u, d) \subseteq K\} \\ \mathrm{Pre}_d(K^c) =& \{(q, x) \in K : \forall(\sigma_u, u) \in \Sigma_{\mathbf{u}} \times \mathcal{U} \\ & \exists(\sigma_d, d) \in \Sigma_{\mathbf{d}} \times \mathcal{D}\ R(q, x, \sigma_u, \sigma_d, u, d) \cap K^c \neq \emptyset\} \cup K^c \end{aligned} \tag{9}$$

Therefore $\mathrm{Pre}_u(K)$ contains all states in K for which controllable actions (σ_u, u) can force the state to remain in K for at least one step in the discrete evolution. $\mathrm{Pre}_d(K^c)$, on the other hand, contains all states in K^c, as well as all states from which uncontrollable actions (σ_d, d) may be able to force the state outside of K.

Consider two subsets $G \subseteq Q \times \mathbb{R}^n$ and $E \subseteq Q \times \mathbb{R}^n$ such that $G \cap E = \emptyset$. The reach-avoid operator is defined as:

$$\begin{aligned} \mathrm{Reach}(G, E) = \{(q, x) \in Q \times \mathbb{R}^n \mid & \forall u \in \mathcal{U}\ \exists d \in \mathcal{D} \text{ and } t \geq 0 \text{ such that} \\ & (q, x(t)) \in G \text{ and } (q, x(s)) \in \mathsf{Dom} \setminus E \text{ for } s \in [0, t]\} \end{aligned} \tag{10}$$

where $(q, x(s))$ is the continuous state trajectory of $\dot{x}(s) = f(q, x(s), \sigma_u, \sigma_d, u(s), d(s))$ starting at (q, x).

Now, consider the following algorithm:
initialization: $W^0 = \mathcal{G}_0^c$, $W^{+1} = \emptyset$, $i = 0$
while $W^i \neq W^{i+1}$ **do**
$W^{i-1} = W^i \setminus \mathrm{Reach}\left(\mathrm{Pre}_d((W^i)^c), \mathrm{Pre}_u(W^i)\right)$
$i = i - 1$
end while

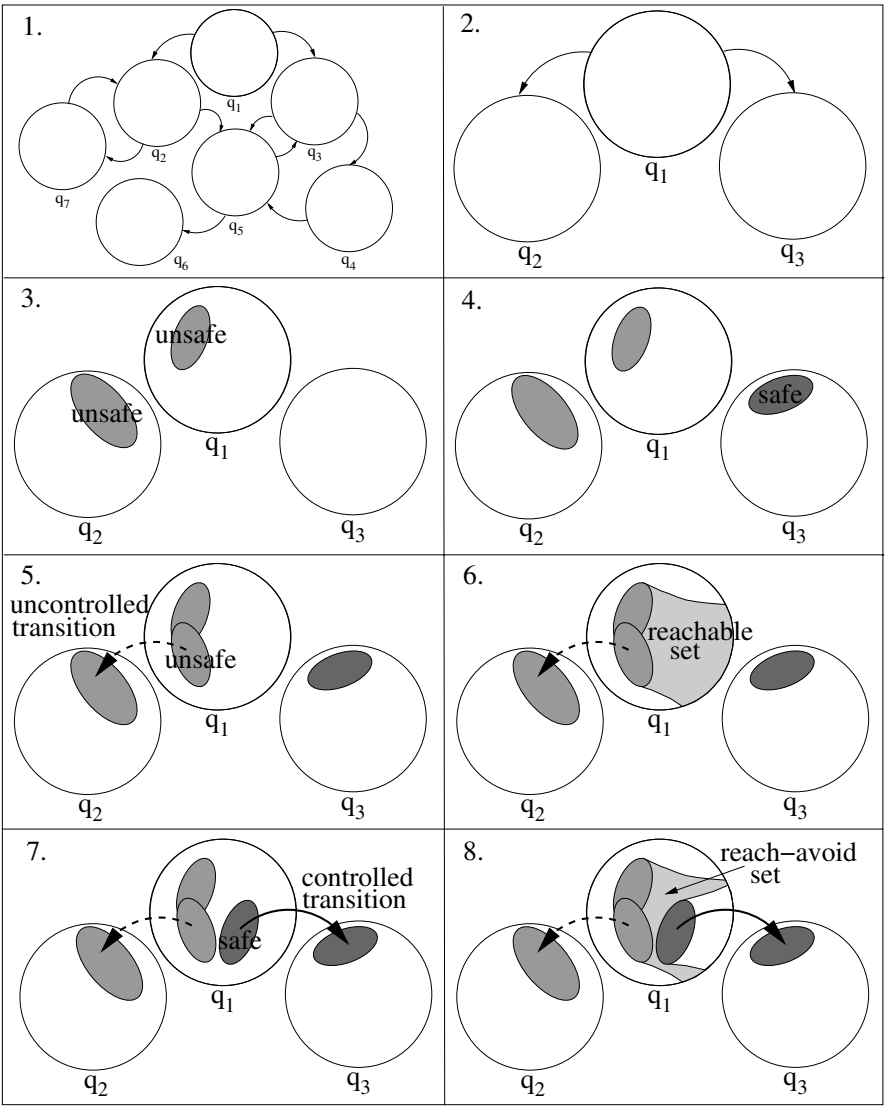

Fig. 4. An illustration of our algorithm for computing reachable sets for hybrid systems.

In the first step of this algorithm, we remove from $\mathcal{G}_0^c$ (the complement of $\mathcal{G}_0$), all states from which a disturbance forces the system either outside $\mathcal{G}_0^c$ or to states from which a disturbance action may cause transitions outside $\mathcal{G}_0^c$, without first touching the set of states from which there is a control action keeping the system inside $\mathcal{G}_0^c$. Since at each step, $W^{i-1} \subseteq W^i$, the set W^i decreases monotonically in size as i decreases. If the algorithm terminates, we denote the fixed point as W^*. The set W^* is used to verify the safety of the system. Recall once more from Figure 3: if the system starts inside W^*, then there exists a control law, extractable from our computational method, for which the system is guaranteed to be safe.

Returning to our pictorial description of the algorithm in Figure 4, and concentrating on the result of one step of the algorithm detailed in Figure 5, we note that, for iteration i: $\mathrm{Pre}_d((W^i)^c) = G_1 \cup G_2$, $E_1 \subset \mathrm{Pre}_u(W^i)$, and $\mathrm{Reach}\left(\mathrm{Pre}_d((W^i)^c), \mathrm{Pre}_u(W^i)\right) = G_3$.

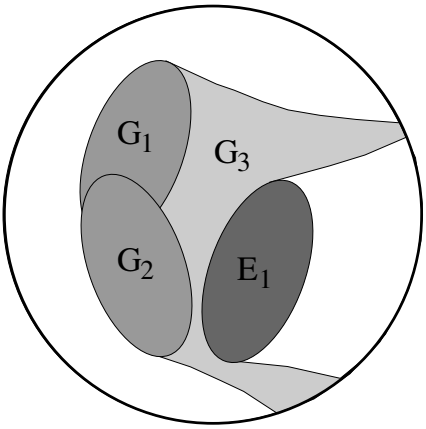

Fig. 5. Detail of the reach-avoid set from diagram 8 of Figure 4.

To implement this algorithm, we need to compute Pre_u, Pre_d, and Reach. The computation of Pre_u and Pre_d requires inversion of the transition relation R subject to the quantifiers $\exists$ and $\forall$; existence of this inverse can be guaranteed subject to conditions on the map R. In our examples, we perform this inversion by hand. The algorithm for computing $\mathrm{Reach}(G, E)$ is a direct modification of the reachable set calculation of Section 3, the details are presented in [7].

5 Flight Management System Example

In this section, we demonstrate our hybrid systems analysis on an interesting and current example, the landing of a civilian aircraft. This example is discussed in detail in [8] and [10]. In addition to the examples presented here,

we have solved a range of multi-mode aircraft collision avoidance examples. Please refer to [7, 57] for these examples.

The autopilots of modern jets are highly automated systems which assist the pilot in constructing and flying four-dimensional trajectories, as well as altering these trajectories online in response to Air Traffic Control directives. The autopilot typically controls the throttle input and the vertical and lateral trajectories of the aircraft to automatically perform such functions as: acquiring a specified altitude and then leveling, holding a specified altitude, acquiring a specified vertical climb or descend rate, automatic vertical or lateral navigation between specified way points, or holding a specified throttle value. The combination of these throttle-vertical-lateral modes is referred to as the *flight mode* of the aircraft. A typical commercial autopilot has several hundred flight modes – it is interesting to note that these flight modes were designed to automate the way pilots fly aircraft manually: by controlling the lateral and vertical states of the aircraft to set points for fixed periods of time, pilots simplify the complex task of flying an aircraft. Those autopilot functions which are specific to aircraft landing are among the most safety critical, as reliable automation is necessary when there is little room for altitude deviations. Thus, the need for automation designs which guarantee safe operation of the aircraft has become paramount. Testing and simulation may overlook trajectories to unsafe states: "automation surprises" have been extensively studied [60] *after* the unsafe situation occurs, and "band-aids" are added to the design to ensure the same problem does not occur again. We believe that the computation of accurate reachable sets inside the aerodynamic flight envelope may be used to influence flight procedures and may help to prevent the occurrence of automation surprises.

5.1 Flap Deflection in a Landing Aircraft

In this example, we examine a landing aircraft, and we focus our attention on the flap setting choices available to the pilot. While flap extension and retraction are physically continuous operations, the pilot is presented with a button or lever with a set of discrete settings and the dynamic effect of deflecting flaps is assumed to be minor. Thus, we choose to model the flap setting as a discrete variable. The results in this section are taken from [37].

A simple point mass model for aircraft vertical navigation is used, which accounts for lift L, drag D, thrust T, and weight mg (see [61] and references therein). We model the nonlinear longitudinal dynamics

$$\begin{bmatrix} m\dot{V} \\ mV\dot{\gamma} \\ \dot{h} \end{bmatrix} = \begin{bmatrix} -D(\alpha, V) + T\cos\alpha - mg\sin\gamma \\ L(\alpha, V) + T\sin\alpha - mg\cos\gamma \\ V\sin\gamma \end{bmatrix} \tag{11}$$

in which the state $x = [V, \gamma, h] \in \mathbb{R}^3$ includes the aircraft's speed V, flight path angle γ, and altitude h. We assume the control input $u = [T, \alpha]$, with

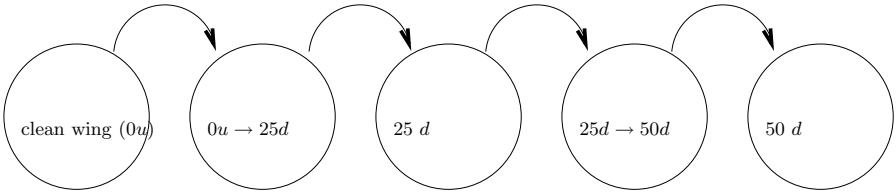

Fig. 6. Discrete transition diagram of flap deflection settings. Clean wing represents no deflection, 25 d represents a deflection of 25°, and 50 d, a deflection of 50°. The modes $0u \rightarrow 25d$ and $25d \rightarrow 50d$ are timed modes to reflect deflection time: if the pilot selects mode 25 d from clean wing, for example, the model will transition into an "intermediate" mode for 10 seconds, before entering 25 d. Thus, the transitions from clean wing to $0u \rightarrow 25d$ and from 25 d to $25d \rightarrow 50d$ are controlled transitions (σ_u) in our analysis, the others are uncontrolled transitions (σ_d).

aircraft thrust T and angle of attack α. The mass of the aircraft is denoted m. The functions $L(\alpha, V)$ and $D(\alpha, V)$ are modeled based on empirical data [62] and Prandtl's lifting line theory [63]:

$$L(\alpha, V) = \tfrac{1}{2}\rho S V^2 C_L(\alpha), \qquad D(\alpha, V) = \tfrac{1}{2}\rho S V^2 C_D(\alpha) \tag{12}$$

where ρ is the density of air, S is wing area, and $C_L(\alpha)$ and $C_D(\alpha)$ are the dimensionless lift and drag coefficients.

In determining $C_L(\alpha)$ we will follow standard autoland procedure and assume that the aircraft switches between three fixed flap deflections $\delta = 0°$, $\delta = 25°$ and $\delta = 50°$ (with slats either extended or retracted), thus constituting a hybrid system with different nonlinear dynamics in each mode. This model is representative of current aircraft technology; for example, in civil jet cockpits the pilot uses a lever to select among four predefined flap deflection settings. We assume a linear form for the lift coefficient $C_L(\alpha) = h_\delta + 4.2\alpha$, where parameters $h_{0°} = 0.2$, $h_{25°} = 0.8$ and $h_{50°} = 1.25$ are determined from experimental data for a DC9-30 [62]. The value of α at which the vehicle stalls decreases with increasing flap deflection: $\alpha_{0°}^{\max} = 16°$, $\alpha_{25°}^{\max} = 13°$, $\alpha_{50°}^{\max} = 11°$; slat deflection adds 7° to the $\alpha^{\max}$ in each mode. The drag coefficient is computed from the lift coefficient as [63] $C_D(\alpha) = 0.041 + 0.045 C_L^2(\alpha)$ and includes flap deflection, slat extension and gear deployment corrections. Thus, for a DC9-30 landing at sea level and for all $\alpha \in [-5°, \alpha_\delta^{\max}]$, the lift and drag terms in (11) are given by

$$L(\alpha, V) = 68.6\ (h_\delta + 4.2\alpha)V^2 \qquad D(\alpha, V) = (2.7 + 3.08\ (h_\delta + 4.2\alpha)^2)V^2$$

In our implementation, we consider three operational modes: $0u$, which represents $\delta = 0°$ with undeflected slats, $25d$, which represents $\delta = 25°$ with deflected slats, and $50d$, for $\delta = 50°$ with deflected slats.

Approximately 10 seconds are required for a 25° degree change in flap deflection. For our implementation, we define transition modes $0u \rightarrow 25d$ and

$25d \rightarrow 50d$ with timers, in which the aerodynamics are those of (11) with coefficients which interpolate those of the bounding operational modes. The corresponding discrete automaton is shown in Figure 6. Transition modes have only a timed switch at $t = t_{\text{delay}}$, so controlled switches will be separated by at least t_{delay} time units and the system is nonzeno. For the executions shown below, $t_{\text{delay}} = 10$ seconds.

The aircraft enters its final stage of landing close to 50 feet above ground level ([62, 64]). Restrictions on the flight path angle, aircraft velocity and touchdown (TD) speed are used to determine the initial safe set W_0:

$$\begin{cases} h \leq 0 & \text{landing or has landed} \\ V > V_{\delta}^{\text{stall}} & \text{faster than stall speed} \\ V < V^{\max} & \text{slower than limit speed} \\ V \sin\gamma \geq \dot{z}_0 & \text{limited TD speed} \\ \gamma \leq 0 & \text{monotonic descent} \end{cases} \cup \begin{cases} h > 0 & \text{aircraft in the air} \\ V > V_{\delta}^{\text{stall}} & \text{faster than stall speed} \\ V < V^{\max} & \text{slower than limit speed} \\ \gamma > -3^{\circ} & \text{limited descent flight path} \\ \gamma \leq 0 & \text{monotonic descent} \end{cases} \tag{13}$$

We again draw on numerical values for a DC9-30 [62]: stall speeds $V_{0u}^{\text{stall}} = 78$ m/s, $V_{25d}^{\text{stall}} = 61$ m/s, $V_{50d}^{\text{stall}} = 58$ m/s, maximal touchdown speed $\dot{h}_0 = 0.9144$ m/s, and maximal velocity $V^{\max} = 83$ m/s. The aircraft's input range is restricted to a fixed thrust at 20% of its maximal value $T = 32KN$, and $\alpha \in [0^{\circ},\ 10^{\circ}]$.

The results of our fixed point computation are shown in Figures 7 and 8. The interior of the surface shown in the first row of Figure 7 represents the initial envelopes W_0 for each of the $0u$, $25d$ and $50d$ modes. The second row of the figure shows the maximally controllable subset of the envelope for each mode individually, as determined by the reachable set computation for continuous systems. The clean wing configuration $0u$ becomes almost completely uncontrollable, while the remaining modes are partially controllable. The subset of the envelope that cannot be controlled in these high lift/high drag configurations can be divided into two components. For low speeds, the aircraft will tend to stall. For values of h near zero and low flight path angles γ, the aircraft cannot pull up in time to avoid landing gear damage at touchdown. The third row shows the results for the hybrid reachable set computation. Here, both modes $0u$ and $25d$ are almost completely controllable, since they can switch instantaneously to the fully deflected mode $50d$. However, no mode can control the states h near zero and low γ, because no mode can pull up in time to avoid landing gear damage. Figure 8 shows a slice through the reach and avoid sets for the hybrid analysis at a fixed altitude of $h = 5$m, for each of the $0u$, $25d$ and $50d$ modes. Here, the grey-scale represents the following: dark grey is the subset of the initial escape set that is also safe in the current mode, mid-grey is the initial escape set, light grey is the known unsafe set, and white is the computed reach set, or those states from which the system can neither remain in the same mode nor switch to safety.

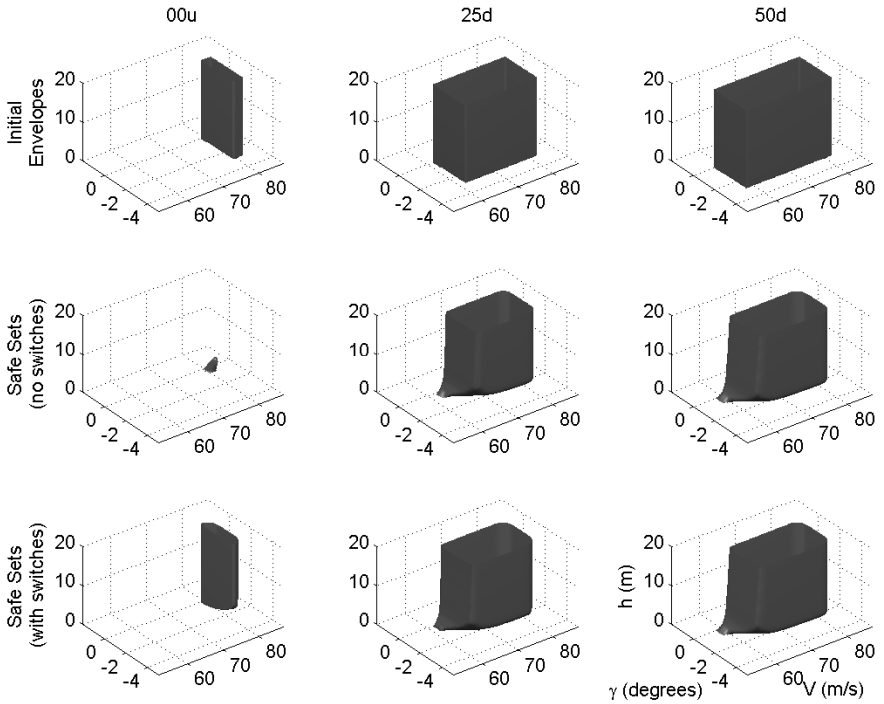

Fig. 7. Maximally controllable safe envelopes for the multimode landing example. From left to right the columns represent modes $0u$, $25d$ and $50d$.

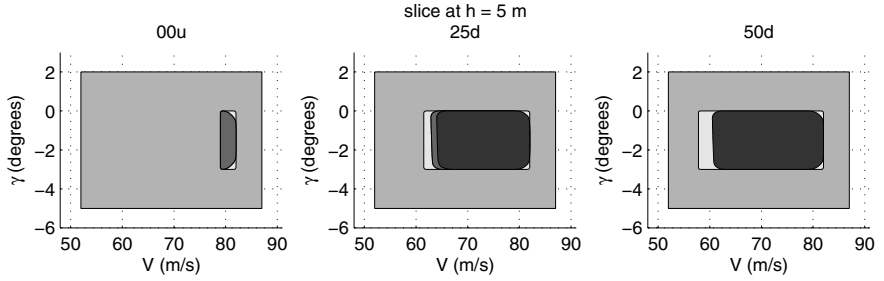

Fig. 8. Slices through the reach and avoid sets for the hybrid analysis at a fixed altitude of $h = 5$m. From left to right the columns represent modes $0u$, $25d$ and $50d$.

5.2 Take Off / Go Around Interface Analysis

We now examine another aircraft landing example with the goal of using hybrid system verification in order to prove desirable qualities about the pilot's display. Naturally, only a subset of all information about the aircraft is displayed to the pilot – but how much information is enough? When the pilot does not have the required information at his disposal, and becomes confused by the cockpit automation, automation surprises and mode confusion can occur. Automation surprises are contributing factors in many aircraft incidents, commonly regarded as indicators of future aircraft accidents. Currently, extensive flight simulation and testing are used to validate autopilot systems and their displays. However, discovering design errors as early as possible in the design process is important for aircraft manufacturers as well as pilots, and hybrid verification tools can aid in this process. The results in this section are taken from [9], which uses the same form of longitudinal dynamic model (11) as the previous section, with new parameters for a large commercial aircraft [9].

In modeling $C_L(\alpha)$ and $C_D(\alpha)$ as in (12), we define $C_L(\alpha) = C_{L_0} + C_{L_\alpha}\alpha$ and $C_D(\alpha) = C_{D_0} + KC_L^2(\alpha)$. The constants C_{L_0}, C_{D_0}, and K represent a particular aircraft configuration, as indicated in Table 1. $C_{L_\alpha} = 5.105$ in all modes. The aircraft has mass $m = 190000$ kg, wing surface area $S = 427.80$ m/s^2, and maximum thrust $T_{\max} = 686700$ N.

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

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

The model for this example also varies from the previous example in that we directly account for the user's actions in the hybrid system. We assume that the pilot operates the aircraft according to strict procedure, shown in Figure 9. During landing, 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. A go-around can be initiated at any time 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 preset missed-approach altitude, $h_{\text{alt}} = 2500$ feet.

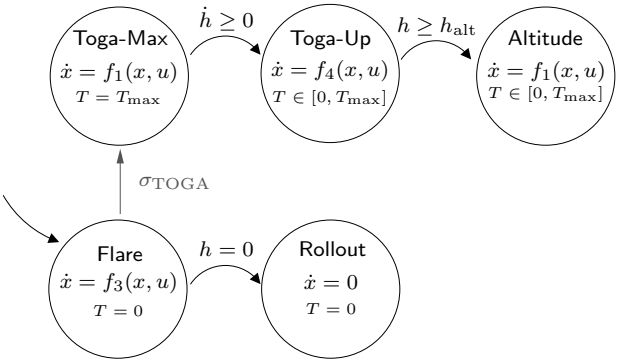

Fig. 9. Hybrid procedural automaton $H_{\text{procedure}}$. The dynamics $f_i(x, u) = f(q_i, x, u)$ differ in the values of aerodynamic coefficients affecting lift and drag.

The initial state of the procedural model $H_{\text{procedural}}$ (Figure 9) is Flare, with flaps at Flaps-30 and thrust fixed at idle. When a pilot initiates a go-around maneuver (often called a "TOGA" due to the "Take-Off/Go-Around" indicator on the pilot 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 automatically switches into an altitude-holding mode, Altitude, to prepare for the next approach (with the landing gear down). If a go-around is not initiated from Flare, 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, we model σ_{TOGA} as 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 $h \geq h_{\text{alt}}$.

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 initial envelope W_0. Bounds on V and α are determined by stall speeds and structural limitations for each flap setting. Bounds on γ and T are determined by the desired maneuver [65]. 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.

We separate the hybrid procedural model (Figure 9) across the user-controlled switch σ_{TOGA}, into two hybrid subsystems: H_{F} and H_{T}. $H_{\mathbf{F}}$ encompasses Flare and Rollout, $H_{\overset{\circ}{T}}$ encompasses Toga-Max, Toga-Up, and Altitude. Computationally, automatic transitions are smoothly accomplished by concatenating modes across automatic transitions, so that the change in dy-

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

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]	$[-0.7°, 0.7°]$	$[-8°, 12°]$

namics across the switching surface is modeled as another nonlinearity in the dynamics. Additionally, we assume in $H_{\underset{T}{\circ}}$ 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 for H_{F} and H_{T}, $(W_{\mathbf{F}})_0$ and $(W_{\text{T}})_0$, are determined by state bounds on each mode given in Table 2. We perform the reachable set computation on $H_{\mathbf{F}}$ and H_{T} separately to obtain the safe flight envelopes $W_{\mathbf{F}}$ and W_T. Figure 10 shows $W_{\mathbf{F}}$, and Figure 11 shows W_{T} in Toga-Up and Toga-Max modes. (Note that the boundary of $W_{\mathbf{F}}$ along $\gamma = 0$ corresponds with the transition boundary of W_{T} between Toga-Up and Toga-Max, $\dot{h} = 0$.)

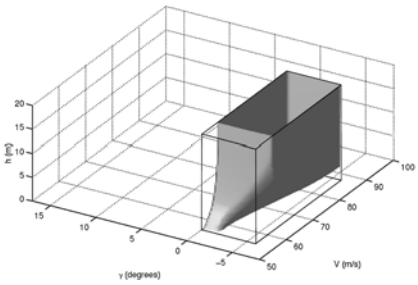

Fig. 10. Safe region W_F; the outer box is $(W_F)_0$.

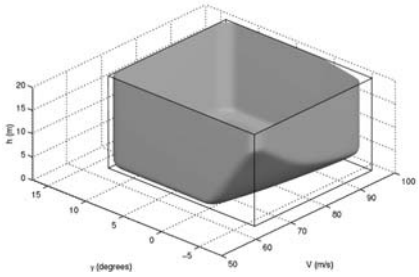

Fig. 11. Safe region W_T: the outer box is $(W_T)_0$.

Figure 12 shows the continuous region $W_{\mathbf{F}} \cap W_{\text{T}}$ from which we can guarantee both a safe landing and a safe go-around. Notice that this set is smaller than $W_{\mathbf{F}}$, 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.

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 should have enough information to know before entering a go-around maneuver whether or not the aircraft will remain safe: thus the

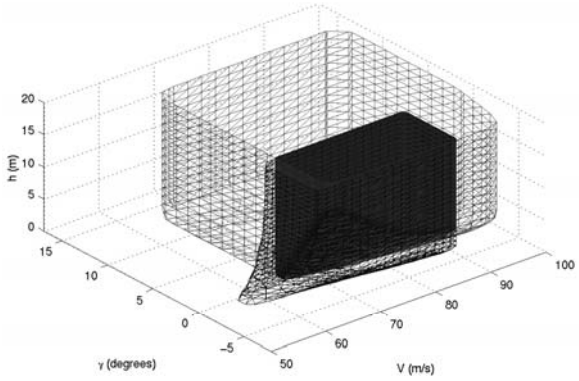

Fig. 12. The solid shape is the safe region $W_F \cap W_T$, from which safe landing and safe go-around is possible. The meshes depict $W_{\mathbf{F}}$ and W_{T}.

pilot could respond to this information by increasing speed, decreasing ascent rate, or decreasing angle of attack.

Further details on how hybrid system verification is used to verify information contained in user-interfaces can be found in [9].

6 Summary

We have presented a method and algorithm for hybrid systems analysis, specifically, for the verification of safety properties of hybrid systems. We have also given a brief summary of other available methods. All techniques rely on the ability to compute reachable sets of hybrid systems, and they differ mainly in the assumptions made about the representation of sets, and evolution of the continuous state dynamics. We have described and demonstrated our algorithm, which represents a set implicitly as the zero sublevel set of a given function, and computes its evolution through the hybrid dynamics using a combination of constrained level set methods and discrete mappings through transition functions.

Many directions for further work are available, and we are pursuing several of them. Our algorithm is currently constrained by computational complexity: examples with four continuous dimensions take several days to run on our standard desktop computers, five dimensions takes weeks. We are working on a variant of our algorithm which first projects the high dimensional target into a set of lower dimensional subspaces of the state space, computes the reachable sets of these projections (quickly, as they are in low dimensions), and then "backprojects" these sets to form, in high dimensions, an overapproximation of the actual reachable set. The actual reachable set need never be computed, and overapproximations of unsafe sets can be used to verify safety, as we showed in the last lecture. We are also developing methods to compute tight polyhedral

overapproximations of the reachable set for general nonlinear hybrid systems [66] – these methods scale well in continuous dimension, as they do not require gridding the state space.

Acknowledgments

We would like to thank John Lygeros and Shankar Sastry, who are coauthors of the hybrid system algorithm and provided valuable insight into the more recent viscosity solution proofs. We thank Ron Fedkiw and Stan Osher for their help and discussions regarding level set methods; the rendering software that we used in Section 3 was written by Ron Fedkiw.

References

1. J. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*. Princeton University Press, 1947.
2. A. Church, "Logic, arithmetic, and automata," in *Proceedings of the International Congress of Mathematicians*, pp. 23–35, 1962.
3. R. Isaacs, *Differential Games*. John Wiley, 1967.
4. 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.
5. 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*, December 2001. Submitted.
6. C. J. 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, July 2000.
7. I. Mitchell, *Application of Level Set Methods to Control and Reachability Problems in Continuous and Hybrid Systems*. PhD thesis, Scientific Computing and Computational Mathematics, Stanford University, August 2002.
8. A. M. Bayen, I. Mitchell, M. Oishi, and C. J. Tomlin, "Automatic envelope protection and cockpit interface analysis of an autoland system using hybrid system theory." Submitted to the AIAA Journal of Guidance, Control, and Dynamics, December 2002.
9. M. Oishi, I. Mitchell, A. M. Bayen, C. J. Tomlin, and A. Degani, "Hybrid verification of an interface for an automatic landing," in *Proceedings of the IEEE Conference on Decision and Control*, (Las Vegas, NV), December 2002.
10. M. Oishi, C. J. Tomlin, and A. Degani, "Verification of user interfaces for hybrid systems." Submitted as a NASA Technical Memorandum, Ames Research Center, November 2002.
11. S. S. Sastry, *Nonlinear Systems: Analysis, Stability and Control*. New York: Springer Verlag, 1999.
12. J. Doyle, B. Francis, and A. Tannenbaum, *Feedback Control Theory*. New York: Macmillan, 1992.

13. P. J. G. Ramadge and W. M. Wonham, "The control of discrete event dynamical systems," *Proceedings of the IEEE*, vol. Vol.77, no. 1, pp. 81–98, 1989.
14. J. R. Büchi and L. H. Landweber, "Solving sequential conditions by finite-state operators," in *Proceedings of the American Mathematical Society*, pp. 295–311, 1969.
15. C. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Boston: Kluwer, 1999.
16. R. Brockett, "Hybrid models for motion control systems," in *Perspectives in Control* (H. Trentelman and J. Willems, eds.), pp. 29–54, Boston: Birkhauser, 1993.
17. M. S. Branicky, *Control of Hybrid Systems*. PhD thesis, Department of Electrical Engineering and Computer Sciences, Massachusetts Institute of Technology, 1994.
18. J. Lygeros, *Hierarchical, Hybrid Control of Large Scale Systems*. PhD thesis, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, 1996.
19. A. Nerode and W. Kohn, "Models for hybrid systems: Automata, topologies, controllability, observability," in *Hybrid Systems* (R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, eds.), LNCS 736, pp. 317–356, New York: Springer Verlag, 1993.
20. R. Alur and D. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, pp. 183–235, 1994.
21. R. Alur, C. Courcoubetis, T. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," in *Proceedings of the 11th International Conference on Analysis and Optimization of Systems: Discrete-event Systems* (G. Cohen and J.-P. Quadrat, eds.), no. 199 in LNCS, pp. 331–351, Springer Verlag, 1994.
22. T. Henzinger, "The theory of hybrid automata," in *Proceedings of the 11th Annual Symposium on Logic in Computer Science*, pp. 278–292, IEEE Computer Society Press, 1996.
23. A. Puri and P. Varaiya, "Decidability of hybrid systems with rectangular differential inclusions," in *CAV94: Computer-Aided Verification*, no. 818 in LNCS, pp. 95–104, Stanford, CA: Springer Verlag, 1995.
24. O. Shakernia, G. J. Pappas, and S. S. Sastry, "Decidable controller synthesis for classes of linear systems," in *Hybrid Systems: Computation and Control* (B. Krogh and N. Lynch, eds.), LNCS 1790, pp. 407–420, Springer Verlag, 2000.
25. N. Lynch, R. Segala, and F. Vaandraager, "Hybrid I/O automata," 2002. Submitted. Also, Technical Report MIT-LCS-TR-827b, MIT Laboratory for Computer Science, Cambridge, MA 02139.
26. D. L. Dill, "The Murϕ verification system," in *Conference on Computer-Aided Verification*, LNCS, pp. 390–393, Springer-Verlag, July 1996.
27. S. Owre, J. M. Rushby, and N. Shankar, "PVS: A prototype verification system," in *11th International Conference on Automated Deduction (CADE)* (D. Kapur, ed.), vol. 607 of *Lecture Notes in Artificial Intelligence*, (Saratoga, NY), pp. 748–752, Springer-Verlag, June 1992.
28. J. Burch, E. M. Clarke, K. McMillan, D. Dill, and L. Hwang, "Symbolic model checking: 10^{20} states and beyond," *Information and Computation*, vol. 98, pp. 142–170, June 1992.

29. G. Holzmann, "The model checker Spin," *IEEE Transactions on Software Engineering*, vol. 23, pp. 279–295, May 1997. Special issue on Formal Methods in Software Practice.
30. 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.
31. S. Osher and R. Fedkiw, *The Level Set Method and Dynamic Implicit Surfaces*. Springer-Verlag, 2002.
32. G.-S. Jiang and D. Peng, "Weighted ENO schemes for Hamilton-Jacobi equations," *SIAM Journal on Scientific Computing*, vol. 21, pp. 2126–2143, 2000.
33. S. Osher and C.-W. Shu, "High-order essentially nonoscillatory schemes for Hamilton-Jacobi equations," *SIAM Journal of Numerical Analysis*, vol. 28, no. 4, pp. 907–922, 1991.
34. J. A. Sethian, *Level Set Methods and Fast Marching Methods*. New York: Cambridge University Press, 1999.
35. M. G. Crandall and P.-L. Lions, "Two approximations of solutions of Hamilton-Jacobi equations," *Mathematics of Computation*, vol. 43, no. 167, pp. 1–19, 1984.
36. C.-W. Shu and S. Osher, "Efficient implementation of essentially non-oscillatory shock-capturing schemes," *Journal of Computational Physics*, vol. 77, pp. 439–471, 1988.
37. I. Mitchell, A. M. Bayen, and C. J. Tomlin, "Validating a Hamilton-Jacobi approximation to hybrid system reachable sets," in *Hybrid Systems: Computation and Control* (M. D. D. Benedetto and A. Sangiovanni-Vincentelli, eds.), LNCS 2034, pp. 418–432, Springer Verlag, 2001.
38. T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata," in *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, 1995.
39. K. Larsen, P. Pettersson, and W. Yi, "Uppaal in a nutshell," *Software Tools for Technology Transfer*, vol. 1, 1997.
40. S. Yovine, "Kronos: A verification tool for real-time systems," *Software Tools for Technology Transfer*, vol. 1, pp. 123–133, 1997.
41. R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho, "Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems," in *Hybrid Systems* (R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, eds.), LNCS, pp. 366–392, New York: Springer Verlag, 1993.
42. T. A. Henzinger, P. Ho, and H. Wong-Toi, "HyTech: A model checker for hybrid systems," *Software Tools for Technology Transfer*, vol. 1, pp. 110–122, 1997.
43. O. Maler, A. Pnueli, and J. Sifakis, "On the synthesis of discrete controllers for timed systems," in *STACS 95: Theoretical Aspects of Computer Science* (E. W. Mayr and C. Puech, eds.), no. 900 in LNCS, pp. 229–242, Munich: Springer Verlag, 1995.
44. E. Asarin, O. Maler, and A. Pnueli, "Symbolic controller synthesis for discrete and timed systems," in *Proceedings of Hybrid Systems II, Volume 999 of LNCS* (P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, eds.), Cambridge: Springer Verlag, 1995.
45. H. Wong-Toi, "The synthesis of controllers for linear hybrid automata," in *Proceedings of the IEEE Conference on Decision and Control*, (San Diego, CA), 1997.

46. A. Bemporad and M. Morari, "Verification of hybrid systems via mathematical programming," in *Hybrid Systems: Computation and Control* (F. Vaandrager and J. H. van Schuppen, eds.), no. 1569 in LNCS, pp. 30–45, Berlin: Springer Verlag, 1999.
47. P. Cardaliaguet, M. Quincampoix, and P. Saint-Pierre, "Set-valued numerical analysis for optimal control and differential games," in *Stochastic and Differential Games: Theory and Numerical Methods* (M. Bardi, T. Parthasarathy, and T. E. S. Raghavan, eds.), vol. 4 of *Annals of International Society of Dynamic Games*, Birkhäuser, 1999.
48. J.-P. Aubin, J. Lygeros, M. Quincampoix, S. Sastry, and N. Seube, "Impulse differential inclusions: A viability approach to hybrid systems," *IEEE Transactions on Automatic Control*, vol. 47, no. 1, pp. 2–20, 2002.
49. A. M. Bayen, E. Crück, and C. J. Tomlin, "Guaranteed overapproximation of unsafe sets for continuous and hybrid systems: Solving the Hamilton-Jacobi equation using viability techniques," in *Hybrid Systems: Computation and Control* (C. J. Tomlin and M. R. Greenstreet, eds.), LNCS 2289, pp. 90–104, Springer Verlag, 2002.
50. T. Dang, *Vérification et synthèse des systèmes hybrides.* PhD thesis, Institut National Polytechnique de Grenoble (Verimag), 2000.
51. E. Asarin, O. Bournez, T. Dang, and O. Maler, "Approximate reachability analysis of piecewise-linear dynamical systems," in *Hybrid Systems: Computation and Control* (B. Krogh and N. Lynch, eds.), LNCS 1790, pp. 21–31, Springer Verlag, 2000.
52. A. Chutinan and B. H. Krogh, "Verification of infinite-state dynamic systems using approximate quotient transition systems," *IEEE Transactions on Automatic Control*, vol. 46, no. 9, pp. 1401–1410, 2001.
53. O. Botchkarev and S. Tripakis, "Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations," in *Hybrid Systems: Computation and Control* (B. Krogh and N. Lynch, eds.), LNCS 1790, pp. 73–88, Springer Verlag, 2000.
54. R. Vidal, S. Schaffert, J. Lygeros, and S. S. Sastry, "Controlled invariance of discrete time systems," in *Hybrid Systems: Computation and Control* (B. Krogh and N. Lynch, eds.), LNCS 1790, pp. 437–450, Springer Verlag, 2000.
55. A. Balluchi, M. D. Benedetto, C. Pinello, C. Rossi, and A. Sangionvanni-Vincentelli, "Hybrid control for automotive engine management: The cut-off case," in *Hybrid Systems: Computation and Control* (T. Henzinger and S. Sastry, eds.), no. 1386 in LNCS, pp. 13–32, New York: Springer Verlag, 1998.
56. Esprit, "Verification of Hybrid Systems: Results of a European Union Esprit Project," in *European Journal of Control* (O. Maler, ed.), Vol. 7, Issue 4, 2001.
57. C. J. Tomlin, I. Mitchell, and R. Ghosh, "Safety verification of conflict resolution maneuvers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 2, no. 2, pp. 110–120, 2001. June.
58. M. Oishi, C. J. Tomlin, V. Gopal, and D. Godbole, "Addressing multiobjective control: Safety and performance through constrained optimization," in *Hybrid Systems: Computation and Control* (M. D. D. Benedetto and A. Sangiovanni-Vincentelli, eds.), LNCS 2034, pp. 459–472, Springer Verlag, 2001.
59. T. Dang and O. Maler, "Reachability analysis via face lifting," in *Hybrid Systems: Computation and Control* (S. Sastry and T. Henzinger, eds.), no. 1386 in LNCS, pp. 96–109, Springer Verlag, 1998.

60. A. Degani, *Modeling Human-Machine Systems: On Modes, Error, and Patterns of Interaction*. PhD thesis, Department of Industrial and Systems Engineering, Georgia Institute of Technology, 1996.
61. C. J. Tomlin, *Hybrid Control of Air Traffic Management Systems*. PhD thesis, Department of Electrical Engineering, University of California, Berkeley, 1998.
62. I. M. Kroo, *Aircraft Design: Synthesis and Analysis*. Stanford, California: Desktop Aeronautics Inc., 1999.
63. J. Anderson, *Fundamentals of Aerodynamics*. New York: McGraw Hill Inc., 1991.
64. United States Federal Aviation Administration, *Federal Aviation Regulations*, 1990. Section 25.125 (landing).
65. T. Lambregts, "Automatic flight control: Concepts and methods." FAA National Resource Specialist, Advanced Controls, 1995.
66. I. Hwang, D. Stipanovic, and C. Tomlin, "Polyhedral reachable sets for continuous and hybrid systems." Submitted to the 2003 American Control Conference, August 2002.

Part II

Case Studies