Computational Techniques for the Verification of Hybrid Systems

CLAIRE J. TOMLIN, IAN MITCHELL, ALEXANDRE M. BAYEN, AND MEEKO OISHI

Invited Paper

Hybrid system theory lies at the intersection of the fields of engineering control theory and computer science verification. It is defined as the modeling, analysis, and control of systems that involve the interaction of both discrete state systems, represented by finite automata, and continuous state dynamics, represented by differential equations. The embedded autopilot of a modern commercial jet is a prime example of a hybrid system: the autopilot modes correspond to the application of different control laws, and the logic of mode switching is determined by the continuous state dynamics of the aircraft, as well as through interaction with the pilot. To understand the behavior of hybrid systems, to simulate, and to control these systems, theoretical advances, analyses, and numerical tools are needed. In this paper, we first present a general model for a hybrid system along with an overview of methods for verifying continuous and hybrid systems. We describe a particular verification technique for hybrid systems, based on two-person zero-sum game theory for automata and continuous dynamical systems. We then outline a numerical implementation of this technique using level set methods, and we demonstrate its use in the design and analysis of aircraft collision avoidance protocols and in verification of autopilot logic.

Manuscript received August 27, 2002; revised December 13, 2002. This work was supported by the Defense Advanced Research Projects Agency (DARPA) under the Software Enabled Control Program (AFRL Contract F33615-99-C-3014), by the Department of Defense Multidisciplinary University Research Initiative (MURI) program administered by the Office of Naval Research under Grant N00014-02-1-0720, by the National Aeronautics and Space Administration (NASA) Ames Research Center under Grant NAG 2-1564, and by NASA Ames Research Center under Grant NCC2-798 to the San Jose State University Foundation as part of NASA's base research and technology effort, human-automation theory subelement (RTOP 548-40-12). The work of A. Bayen was supported in part by a graduate fellowship provided by the Délégation Générale pour l'Armement, France. The work of M. Oishi was supported in part by a National Science Foundation Graduate Research Fellowship.

C. J. Tomlin is with the Department of Aeronautics and Astronautics and the Department of Electrical Engineering, Stanford University, Stanford, CA 94305-4035 USA (e-mail: tomlin@stanford.edu).

I. Mitchell is with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720-1770 USA (e-mail: imitchel@eecs.berkeley.edu).

A. M. Bayen is with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305-4035 USA (e-mail: bayen@stanford.edu).

M. Oishi is with the Department of Mechanical Engineering, Stanford University, Stanford, CA 94305-4035 USA (e-mail: moishi@stanford.edu).

Digital Object Identifier 10.1109/JPROC.2003.814621

Keywords*—*Hybrid control, hybrid systems, reachability, verification.

I. Introduction

The field of formal verification in computer science has achieved great success in the analysis of large-scale discrete systems: using temporal logic to express discrete sequences of events, such as *Component A will request data until Component B sends data*, researchers in verification have uncovered design flaws in such safety-critical systems as microprocessors that control aircraft cockpit displays and design standards for a military hardware bus [1]. Discrete analysis, however, is not rich enough to verify systems that evolve according to both continuous dynamics and discrete events. *Embedded systems*, or physical systems controlled by a discrete logic, such as the current autopilot logic for automatically controlling an aircraft, or a future automated protocol for controlling an aircraft in the presence of other aircraft, are prime examples of systems in which event sequences are *determined* by continuous state dynamics. These systems use discrete logic in control because discrete abstractions make it easier to manage system complexity and discrete representations more naturally accommodate linguistic and qualitative information in controller design. While engineering control theory has successfully designed tools to verify and control continuous state systems, these tools do not extend to systems that mix continuous and discrete state, as in the examples above.

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 that 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 [2], [3]; in the continuous domain, control problems of the safety type have been addressed in the context of differential games [4]. For systems involving continuous dynamics, it is very difficult to compute and represent the set of states reachable from some initial set. In this paper, we present recent solutions to the problem, including a method, based on the level set techniques of Osher and Sethian [5], which determines an implicit representation of the boundary of this *reachable set*. This method is based on the theorem, which is proved in [6] using a two-person zero-sum game theory for continuous dynamical systems, that the solution to a particular Hamilton–Jacobi partial differential equation corresponds 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 one 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 [7], the level set implementation of [6] and [8], and the aircraft landing examples of [9] and [10].

II. Hybrid Model and Verification Methodology

A. *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 $\dot{x} = f(\cdot,\cdot,\cdot)$, 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 variables that can be controlled, called *control inputs*, and d represents *disturbance inputs*, which are variables 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] and [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

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

models a system that is a finite set of *discrete state variables* Q, a set of input variables $\Sigma = \Sigma_u \cup \Sigma_d$ that is the union of *control actions* $\sigma_u \in \Sigma_u$ and *disturbance actions* $\sigma_d \in \Sigma_d$, a set of *initial states* $\mathrm{Init} \subseteq Q$, and a *transition relation* $R : Q \times \Sigma \rightarrow 2^Q$ that 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 $q(0) \in \mathrm{Init}$ 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 [3], [14], who originally analyzed the von Neumann–Morgenstern [2] discrete games. A comprehensive reference for modeling and control of discrete state systems is [15].

Control algorithms design 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 that evolve both continuously and according to discrete jumps. A hybrid automaton is defined to be a collection

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

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

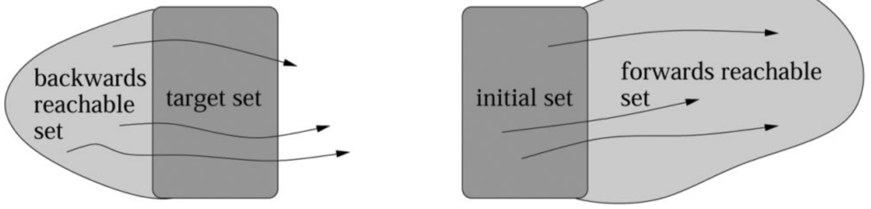

Fig. 1. Difference between backward and forward reachable sets.

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 [7]. This model was developed from the early control work of [16]–[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]–[24], and hybrid input/output automata [25].

B. 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 that 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 in Fig. 1 for generic target and initial sets, in which the arrows represent trajectories of the system. Fig. 2 illustrates how a backward 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,

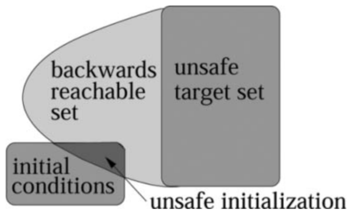

Fig. 2. Using the backward reachable set to verify safety.

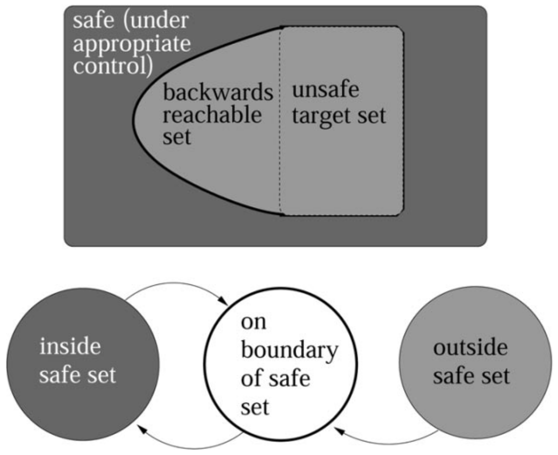

Fig. 3. Discrete abstraction with appropriate control information.

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.

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 backward 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 that must be used on the boundary of the backward 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 Fig. 3). Outside of the backward 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 Fig. 3). When the system state touches the reachable set or unsafe target set boundary, the particular control law that 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 Fig. 3), there is no control law that will guarantee safety; however, application of the particular optimal control law used to compute the boundary may still result in the 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.

III. Verifying Continuous Systems

Computing reachable sets for safety specifications has been a main focus of the control and computer-aided verification communities for the past several years. In the past three years, several experimental reachability tools have been developed and may be classified according to how sets of states are represented and the assumptions on the dynamics under which states are propagated. We classify as "overapproximative" a group of methods that seek an efficient overapproximation of the reachable set. The tools $\mathbf{d/dt}$ [30], [31] and *Checkmate* [32], [33] represent sets as convex polyhedra and propagate these polyhedra under linear and affine dynamics, which could represent overapproximations of nonlinear dynamics along each surface of the polyhedra. Piecewise affine systems are used in [34], [35]. *VeriSHIFT* [36] uses ellipsoidal overapproximations of reach sets for linear systems with linear input; it implements techniques developed in [37]. Polygonal overapproximations of reachable sets for some classes of nonlinear systems are treated in [38]. The tool *Coho*, developed in [39] and [40], uses as set representation two-dimensional projections of higher dimensional nonconvex polyhedra and evolves these "projectagons" under affine overapproximations of nonlinear dynamics using linear programming. In [41], the authors present a solution using sets specified by linear inequalities, for discrete-time linear dynamics. A recent algorithm [42] proposes to divide the continuous state space into a finite number of sets and then to compute the reachable set using a discrete algorithm. The method works for polynomial dynamics and the subzero level sets of polynomials as set representation: by partitioning the state space into a "cylindrical algebraic decomposition" based on the system polynomials, a discrete approximation of the dynamics can be constructed.

A second group of methods is based on computing "convergent approximations" to reachable sets: here, the goal is to represent as closely as possible the true reachable set. Methods include numerical computation of solutions to static Hamilton–Jacobi equations [43] and to techniques from viability theory and set valued analysis [44]. In our work, we have developed a reachability computation method based on level set techniques [5], [45], [46] and viscosity solutions to Hamilton–Jacobi equations [47], [48] using the ideas presented in [7] and [49]. We represent a set as the zero sublevel set of an appropriate function, and the boundary of this set is propagated under the nonlinear dynamics using a validated numerical approximation of a time-dependent Hamilton–Jacobi–Isaacs (HJI) partial differential equation (PDE) governing system dynamics [6], [50], [51]. These convergent approximative methods allow for both control inputs and disturbance inputs in the problem formulation, and they compute a numerical solution on a fixed grid (the mesh points do not move during the computation).

In most of the overapproximative schemes, the reachable set representation scales polynomially with the continuous state space dimension n. Exceptions include orthogonal polyhedra, which is exponential in n, and the algorithm based on cylindrical algebraic decomposition, in which the representation size depends on the dimension of the polynomials involved. Since algorithm execution time and its memory requirements generally scale linearly with the size of the representation of the reachable set, overapproximative schemes in which the set representation scales polynomially with n have a significant advantage over other schemes. However, these overapproximative schemes are generally too imprecise for problems in which the dynamics are nonlinear and for which the shape of the reachable set is not a polygon or an ellipse. The schemes based on convergent approximations are exponential in n and, thus, are not practical for problems of dimension greater than about five or six. However, these schemes can all handle nonlinear dynamics, they work within a differential game setting, and they make no assumptions about the shape of the reachable set.

In this section, using as motivation a classical pursuit–evasion game involving two identical vehicles, we present our methodology and results for computing reachable sets for continuous systems (1). The material in this section is presented in detail in [6], [8], [52], and [53].

A. A Game of Two Identical Vehicles

As our demonstration example, we will adapt a classical pursuit evasion game involving two identical vehicles (see [53] and [54] for more details). If the vehicles get too close to each other, a collision occurs. One of the vehicles (the *pursuer*) wants to cause a collision, while the other (the *evader*) wants to avoid one. Each vehicle has a three-dimensional (3-D) state vector consisting of a location in the plane and a heading. Isaacs [4] pioneered a framework for solving such games using a method similar to the method of characteristics [55].

We study the problem in relative coordinates (see Fig. 4). We draw our vehicles as aircraft, as we have used the solution to this example as inspiration for verifying two-aircraft tactical conflict avoidance strategies in Air Traffic Control.

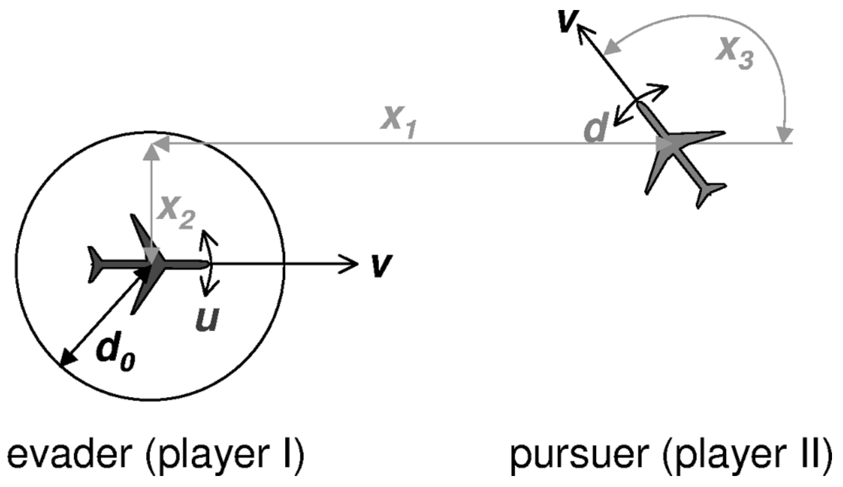

Fig. 4. Relative coordinate system. Origin is located at the center of the evader.

Fixing the evader at the planar origin and facing to the right, the relative model of pursuer with respect to evader is

$$\dot{x} = \frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -v + v\cos x_3 + ux_2 \\ v\sin x_3 - ux_1 \\ d - u \end{bmatrix} = f(x,u,d) \tag{4}$$

where the three state dimensions are relative planar location $[x_1\, x_2]^T \in \mathbb{R}^2$ and relative heading $x_3 \in [0, 2\pi]$, and $v \geq 0$ is the linear velocity of each aircraft. In Fig. 4, the relative heading is measured counterclockwise from the horizontal. The control input is the angular velocity of the evader, $u \in \mathcal{U} = [-1, +1]$, and the disturbance input is $d \in \mathcal{D} = [-1, +1]$, the pursuer's angular velocity. A collision occurs if $\sqrt{x_1^2 + x_2^2} \leq d_0$ for any value of x_3, in $\mathbb{R}^3$ this collision set is a cylinder of radius d_0 centered on the x_3 axis. To solve this pursuit evasion game, we would like to determine the set of initial states from which the pursuer can cause a collision despite the best efforts of the evader.

B. Computing Reachable Sets for Continuous Dynamic Games

The backward 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 backward 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 backward 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 backward reachable set. Let the target set be the collision set

$$\mathcal{G}_0 = \left\{ x \in \mathbb{R}^3 \,|\, \sqrt{x_1^2 + x_2^2} \leq d_0 \right\}. \tag{5}$$

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 (5) 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

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

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 $\mathcal{G}_0$ that 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 [6], we proved that an implicit surface representation of the backward 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

$$\frac{\partial \phi(x,t)}{\partial t} + \min[0, H(x, \nabla\phi(x,t))] = 0 \tag{6}$$

with Hamiltonian

$$H(x,p) = \max_{u \in \mathcal{U}} \min_{d \in \mathcal{D}} p \cdot f(x,u,d) \tag{7}$$

and terminal conditions

$$\phi(x,0) = \phi_0(x). \tag{8}$$

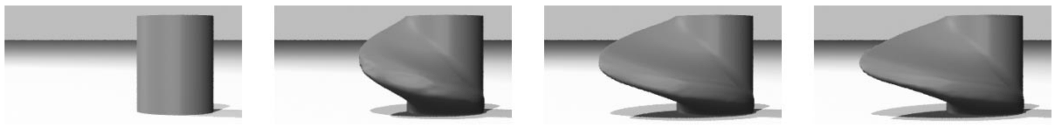
Fig. 5. Growth of the reachable set [6] (animation at [60]).

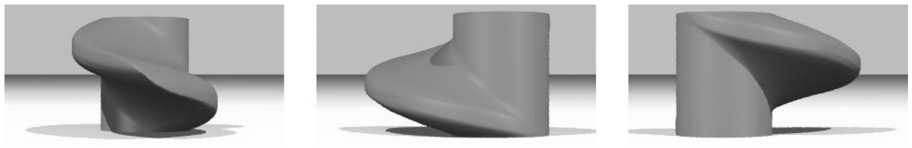
Fig. 6. Other views of the reachable set [6] (animation at [60]).

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 (6)–(8) specifies the backward reachable set as

$$\mathcal{G}(\tau) = \{x \in \mathbb{R}^3 \mid \phi(x,-\tau) \le 0\}.$$

Notice that (6) is solved from time $t=0$ backward to some $t=-\tau \le 0$.

There are several interesting points to make about the HJI PDE (6)–(8). First, the $\min[0,H]$ formulation in (6) 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 (7) results in a solution that is not necessarily a "no regret," or saddle, solution to the differential game. By ordering the optimization so that the maximization occurs first, the control input u is effectively "playing" against an unknown disturbance—it is this order that produces a conservative solution appropriate for the application to system verification under uncertainty. Third, it is proven in [6] that out of many possible weak solutions, the viscosity solution [47] of (6)–(8) 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 (6)–(8), we have developed a C++ implementation based on high resolution level set methods (an excellent introduction to these schemes can be found in [46]). We use a fifth-order accurate weighted, essentially nonoscillatory (WENO) stencil [45], [56] to approximate $\nabla\phi(x,t)$, although we have also implemented a basic first-order scheme for speed [5], [57]. We use the well-studied Lax–Friedrichs (LF) approximation [58] to numerically compute Hamiltonian (7). Finally, we treat the time derivative in (6) with the method of lines and a second-order total variation diminishing (TVD) Runge–Kutta scheme [59]. Numerical convergence of our algorithm is demonstrated and validated in [6] and [51].

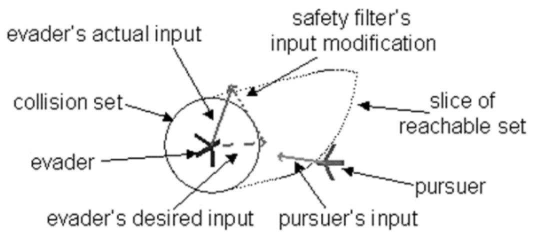

Fig. 7. Annotated frame from collision avoidance example animation.

C. Collision Avoidance Example Results

We can apply our numerical methods to the collision avoidance problem. In Fig. 5, the target set $\mathcal{G}_0$ for the example appears on the far left (the cylinder); the remaining images show how $\mathcal{G}(\tau)$ grows as τ increases from zero. For the parameters chosen in Section III-A, the reachable set converges to a fixed point for $\tau \gtrsim 2.6$. Fig. 6 shows several views of this fixed point. Should the pursuer start anywhere within this reachable set, it can cause a collision by choosing an appropriate input d no matter what input u the evader might choose. Conversely, if the pursuer starts outside this reachable set, then there exists an input u that the evader can choose that will avoid a collision no matter what input d the pursuer might choose. Thus, for initial conditions outside this set, the system can be verified to be safe.

We note that the shape of the reachable set in this example complies with our intuition—the relative heading coordinate x_3 is the vertical coordinate in these figures, so a horizontal slice represents all possible relative planar coordinates of the two vehicles at a fixed relative heading. Consider a slice through the most extended part of the helical bulge (that occurs at the midpoint of the set on the vertical axis). The relative heading for this slice is $x_3=\pi$, which is the case in which the two aircraft have exactly opposite headings. It is not surprising that the reachable set is largest at this relative heading, and smallest for slices at the top and bottom of the reachable set, where $x_3=0$ and, thus, the aircraft have the same heading.

Fig. 7 shows an annotated frame from an animation of the collision avoidance system, and a series of frames from that animation are shown in Fig. 8, progressing from left to right. The evader starts on the left surrounded by the collision

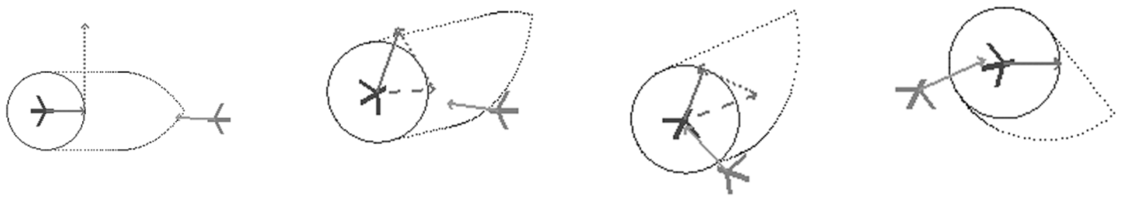

Fig. 8. Evader keeps pursuer from entering reachable set and, hence, avoids collision (animation at [60]).

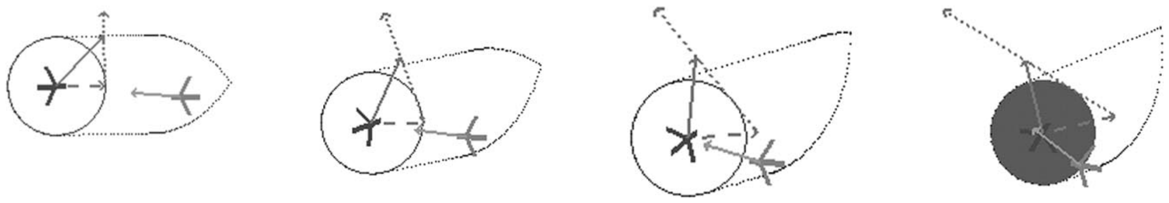

Fig. 9. Pursuer starts within the reachable set, and can thus cause a collision despite the evader's efforts (animation at [60]).

circle, while the pursuer starts on the right. The dotted shape surrounding the evader is the slice of the reachable set for the current relative heading of the two vehicles; for example, in the leftmost figure, the vehicles have relative heading $x_3 \approx \pi$ and so the horizontal midplane slice of the reachable set is shown. The evader wants to continue to the right, and the pursuer simply wants to cause a collision. By choosing its safe input according to (7), as the pursuer approaches the boundary of the reachable set, the evader keeps the pursuer from entering the reachable set and, thus, from causing a collision. Fig. 9 shows a sequence in which the pursuer starts within the reachable set and can cause a collision.

The computations discussed in this section are expensive to perform: they require gridding the state space and, thus, their complexity is exponential in the continuous state dimension. The set in Fig. 6 took about 5 min to compute on a 3-D grid using a standard Pentium III laptop; four-dimensional problems can take a few days to run. In Section VI, we will discuss our current work in computing projective over-approximations to decrease the computation time to achieve a useful result.

IV. Verifying Hybrid Systems

In the previous section, we demonstrated the concept illustrated in Fig. 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 that 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 focused on decidable classes: it was shown that decidability results exist for timed and some classes of linear hybrid automata [61]. Software tools were designed to automatically compute reachable sets for these systems: Uppaal [62] and Kronos [63] for timed automata, and HyTech [64], [65] 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 backward reachable set. The procedure that we describe here was motivated by the work of [66] and [67] for reachability computation and controller synthesis on timed automata, and that of [68] for controller synthesis on linear hybrid automata. Tools based on the analysis of piecewise linear systems, using mathematical programming tools such as CPLEX [34], have found success in several industrial applications.

Our hybrid system analysis algorithm [7] 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 [44], [69], which has extended concepts from viability to hybrid systems [70] 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 $\mathbf{d/dt}$, *Checkmate*, and *VeriSHIFT* have been designed using the different methods of continuous reachable set calculation surveyed in the previous section [30], [31], [33], [36], [41]: 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 [34], [71], experimental industrial batch plants [72], vehicle collision avoidance problems [73], [74], as well as envelope protection problems [9], [75]. The problems that have been solved to date are generally of low dimension: to the best of our knowledge, even the overapproximative methods to date have not been directly applied to systems of continuous dimension greater than six. In the next section, we present results for envelope protection on nonlinear hybrid systems with three continuous dimensions, representing the longitudinal dynamics of jet aircraft under hybrid control.

A. 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 [7], with new implementation results presented in [8].

Consider the sequence of eight diagrams in Fig. 10. 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 $\mathrm{Dom} \subseteq q_i \times \mathbb{R}^n$, 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 that 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 (that 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 backward reachable from the unsafe target set in q_1 may be computed using the reachable set algorithm of Section III on the dynamics $\dot{x} = f(q_1, x(t), (\sigma_u(t), \sigma_d(t)), (u(t), d(t)))$ (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 backward 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

$$\mathcal{G}_0 = \{(q, x) \in Q \times \mathbb{R}^n \mid g(q, x) \leq 0\} \tag{9}$$

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

$$\begin{aligned}\mathrm{Pre}_u(K) &= \{(q, x) \in K : \exists(\sigma_u, u) \in \Sigma_u \times \mathcal{U}\ \forall(\sigma_d, d) \in \\ &\quad \Sigma_d \times \mathcal{D}\ R(q, x, \sigma_u, \sigma_d, u, d) \subseteq K\} \\ \mathrm{Pre}_d(K^c) &= \{(q, x) \in K : \forall(\sigma_u, u) \in \Sigma_u \times \mathcal{U}\ \exists(\sigma_d, d) \in \\ &\quad \Sigma_d \times \mathcal{D}\ R(q, x, \sigma_u, \sigma_d, u, d) \cap K^c \neq \emptyset\} \cup K^c \end{aligned} \tag{10}$$

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

$$\begin{aligned}&\text{Reach}\ (G, E) \\ &\quad = \{(q, x) \in Q \times \mathbb{R}^n \mid \forall u \in \mathcal{U}\ \exists d \in \mathcal{D} \text{ and } t \geq 0 \\ &\qquad \text{such that} \\ &\qquad (q, x(t)) \in G \text{ and } (q, x(s)) \in \mathrm{Dom} \setminus E \text{ for } s \in [0, t]\}\end{aligned} \tag{11}$$

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**
$W^{i-1} = W^i \setminus \mathrm{Reach}(\mathrm{Pre}_d((W^i)^c), \mathrm{Pre}_u(W^i))$
$i = i - 1$
**end while**
```

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 Fig. 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 Fig. 10, and concentrating on the result of one step of

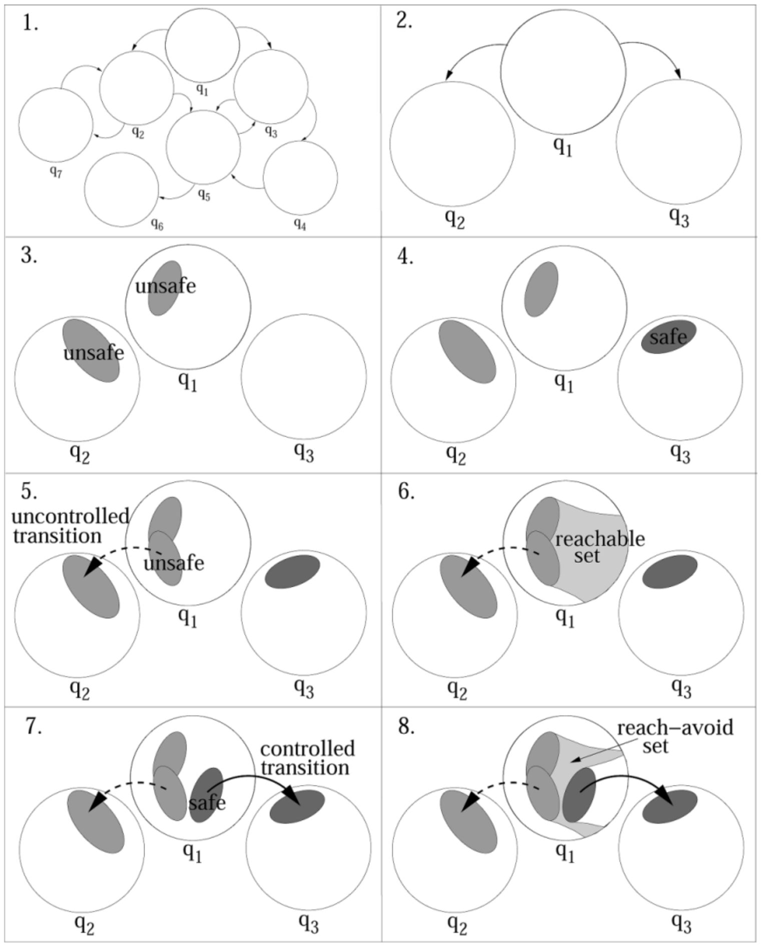

Fig. 10. Illustration of our algorithm for computing reachable sets for hybrid systems.

the algorithm detailed in Fig. 11, 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 $\mathrm{Reach}(\mathrm{Pre}_d((W^i)^c), \mathrm{Pre}_u(W^i)) = G_3$.

To implement this algorithm, we need to compute $\mathrm{Pre}_u, \mathrm{Pre}_d$, and Reach. The computation of Pre_u and Pre_d requires inversion of the transition relation R subject to the quantifiers $\exists$ and $\forall$; the 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 III; the details are presented in [8]. Finally, we remark that this algorithm is semidecidable when the operators $\mathrm{Pre}_u, \mathrm{Pre}_d$, and Reach are computable: when the continuous state dynamics are constant and the guards and resets are polyhedra, then the algorithm reduces to that for linear hybrid automata [68].

V. 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 [9] and [10]. In addition to the examples presented here, we have solved a range of multimode aircraft collision avoidance examples. Please refer to [8], [73] for these examples.

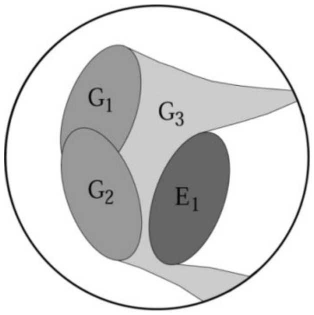

Fig. 11. Detail of the reach-avoid set from diagram 8 of Fig. 10.

The autopilots of modern jets are highly automated systems that 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

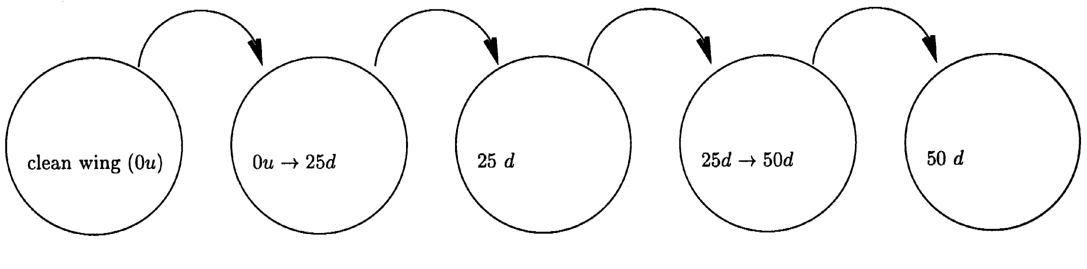

Fig. 12. Discrete transition diagram of flap deflection settings. Clean wing represents no deflection, $25d$ represents a deflection of 25°, and $50d$ represents a deflection of 50°. The modes $0u \rightarrow 25d$ and $25d \rightarrow 50d$ are timed modes to reflect deflection time: if the pilot selects mode $25d$ from clean wing, for example, the model will transition into an "intermediate" mode for 10 s, before entering $25d$. Thus, the transitions from clean wing to $0u \rightarrow 25d$ and from $25d$ to $25d \rightarrow 50d$ are controlled transitions (σ_u) in our analysis; the others are uncontrolled transitions (σ_d).

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 that 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 that guarantee safe operation of the aircraft has become paramount. Testing and simulation may overlook trajectories to unsafe states: "automation surprises" have been extensively studied [76] *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.

A. 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 [51].

A simple point mass model for aircraft vertical navigation is used, which accounts for lift L, drag D, thrust T, and weight mg (see [77] and references therein). We model the nonlinear longitudinal dynamics

$$\begin{bmatrix} m\dot{V} \\ mV\dot{\gamma} \\ \dot{h} \end{bmatrix} = \begin{bmatrix} -D(\alpha, V) + T\cos\alpha - mg\sin\gamma \\ L(\alpha, V) + T\sin\alpha - mg\cos\gamma \\ V\sin\gamma \end{bmatrix} \qquad (12)$$

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 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 [78] and Prandtl's lifting line theory [79]

$$L(\alpha, V) = \frac{1}{2}\rho S V^2 C_L(\alpha), \quad D(\alpha, V) = \frac{1}{2}\rho S V^2 C_D(\alpha) \qquad (13)$$

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 $h_{0^\circ} = 0.2, h_{25^\circ} = 0.8$, and $h_{50^\circ} = 1.25$ are determined from experimental data for a DC9-30 [78]. The value of α at which the vehicle stalls decreases with increasing flap deflection: $\alpha_{0^\circ}^{\max} = 16°, \alpha_{25^\circ}^{\max} = 13°, \alpha_{50^\circ}^{\max} = 11°$; slat deflection adds 7° to the $\alpha^{\max}$ in each mode. The drag coefficient is computed from the lift coefficient as [79] $C_D(\alpha) = 0.041 + 0.045C_L^2(\alpha)$ 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 (12) are given by

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

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 s are required for a 25° 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 (12) with coefficients that interpolate those of the bounding operational modes. The corresponding discrete automaton is shown in Fig. 12. Transition modes have only a timed switch at $t = t_{\text{delay}}$, so controlled switches will be separated by at least t_{delay} time

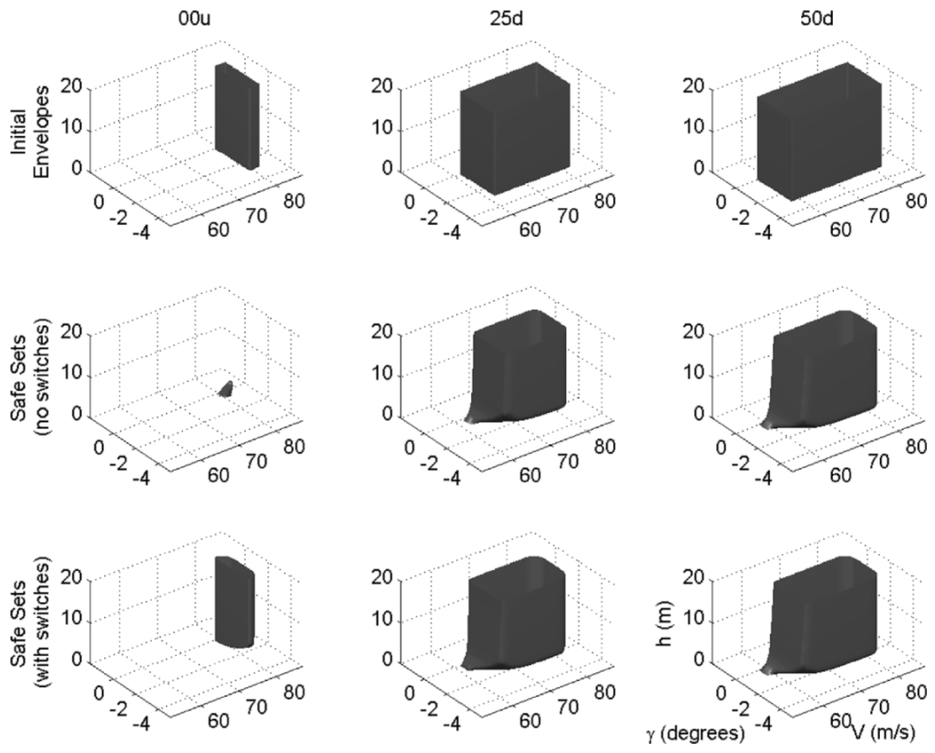

Fig. 13. Maximally controllable safe envelopes for the multimode landing example. From left to right, the columns represent modes $0u$, $25d$, and $50d$.

units and the system is nonzeno. For the executions shown below, $t_{\text{delay}} = 10$ s.

The aircraft enters its final stage of landing close to 50 feet above ground level ([78], [80]). Restrictions on the flight path angle, aircraft velocity, and touchdown (TD) speed are used to determine the initial safe set W_0

$$\begin{cases} h \le 0 & \text{landing or has landed} \\ V > V_\delta^{\text{stall}} & \text{faster than stall speed} \\ V < V^{\max} & \text{slower than limit speed} \\ V \sin\gamma \ge \dot{z}_0 & \text{limited TD speed} \\ \gamma \le 0 & \text{monotonic descent} \end{cases}$$
$$\cup \begin{cases} h > 0 & \text{aircraft in the air} \\ V > V_\delta^{\text{stall}} & \text{faster than stall speed} \\ V < V^{\max} & \text{slower than limit speed} \\ \gamma > -3^\circ & \text{limited descent flight path} \\ \gamma \le 0 & \text{monotonic descent.} \end{cases} \quad (14)$$

We again draw on numerical values for a DC9-30 [78]: 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 $\alpha \in [0^\circ,\ \alpha_\delta^{\max}]$.

The results of our fixed point computation are shown in Figs. 13 and 14. The interior of the surface shown in the first row of Fig. 13 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 quickly 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. Fig. 14 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 grayscale represents the following: dark gray is the subset of the initial escape set that is also safe in the current mode, mid-gray is the initial escape set, light gray 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.

B. Take Off/Go Around 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, automation surprises and mode confusion can occur. Currently, extensive flight simulation and testing are used to validate autopilot systems and their displays. However, discovering

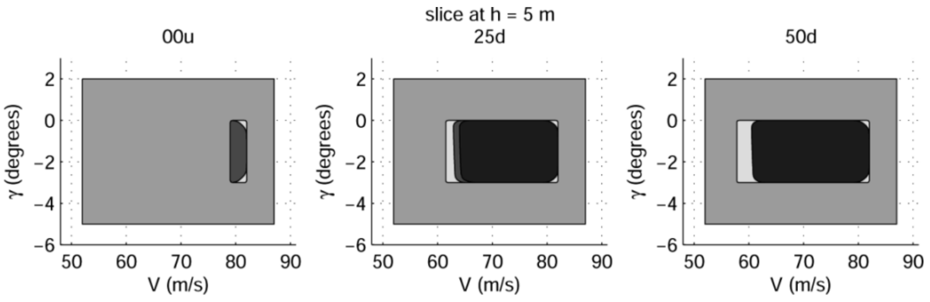

Fig. 14. Slices through the reach and avoid sets for the hybrid analysis at a fixed altitude of $h = 5$ m. From left to right, the columns represent modes $0u$, $25d$, and $50d$.

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

design errors as early as possible in the design process is important, and hybrid verification tools can aid in this process. The results in this section are taken from [10], which uses the same form of longitudinal dynamic model (12) as the previous section, with new parameters for a large commercial aircraft.

In modeling $C_L(\alpha)$ and $C_D(\alpha)$ as in (13), 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 $m = 190\,000$ kg, wing surface area $S = 427.80$ m/s^2, and maximum thrust $T_{\max} = 686\,700$ N. Here, the symbology Flaps-20, Flaps-25, Flaps-30 represents an increasing flap deflection: in the determination of the aerodynamic coefficients, we used 30°, 40°, and 50°, respectively, for these values.

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 Fig. 15. 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.

The initial state of the procedural model $H_{\text{procedural}}$ (Fig. 15) 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" in-

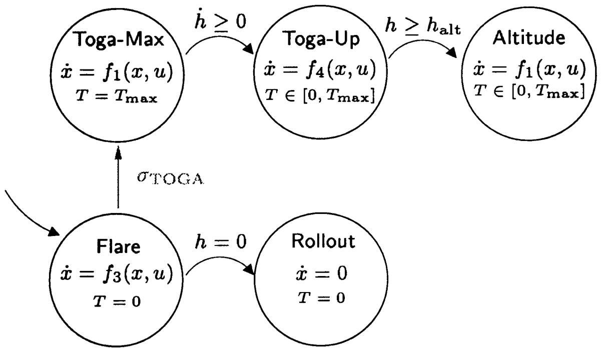

Fig. 15. Hybrid procedural automaton $H_{\text{procedural}}$. The dynamics $f_i(x, u) = f(q_i, x, u)$ differ in the values of aerodynamic coefficients affecting lift and drag.

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]	$[-0.7°, 0.7°]$	$[-8°, 12°]$

dicator 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 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

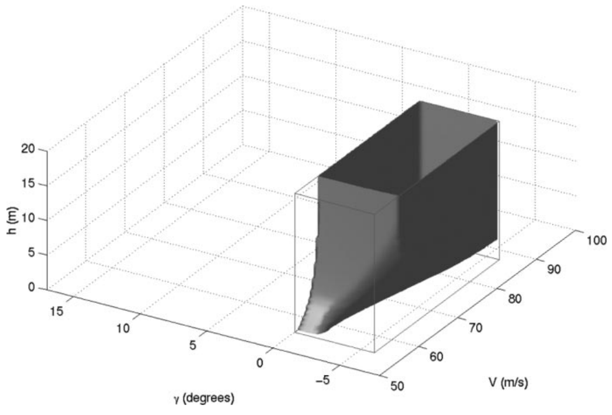

Fig. 16. Safe region W_F; the outer box is $(W_F)_0$.

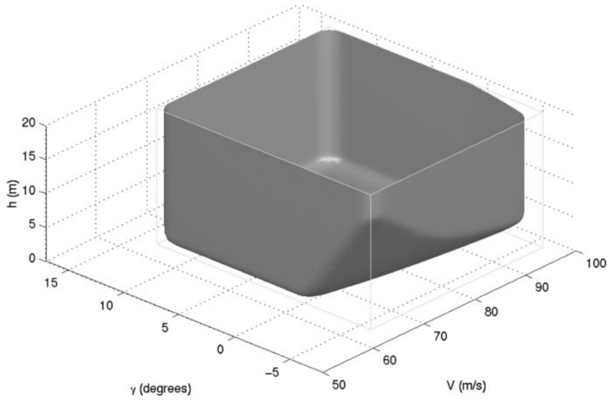

Fig. 17. Safe region W_T; the outer box is $(W_T)_0$.

structural limitations for each flap setting. Bounds on γ and T are determined by the desired maneuver [81]. 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 (Fig. 15) across the user-controlled switch σ_{TOGA}, into two hybrid subsystems: H_{F} and H_{T}. H_{F} encompasses Flare and Rollout, and H_{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 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 for H_{F} and $H_{\text{T}}, (W_{\text{F}})_0$ 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_{F} and H_{T} separately to obtain the safe flight envelopes W_{F} and W_T. Fig. 16 shows W_{F}, and Fig. 17 shows W_{T} in Toga-Up and Toga-Max modes. (Note that the boundary of W_{F} along $\gamma = 0$ corresponds with the transition boundary of W_{T} between Toga-Up and Toga-Max, $\dot{h} = 0$.)

Fig. 18 shows the continuous region $W_{\text{F}} \cap W_{\text{T}}$ from which we can guarantee both a safe landing and a safe go-around. Notice that this set is smaller than W_{F}, the region from which a safe landing is possible: the pilot is further restricted in

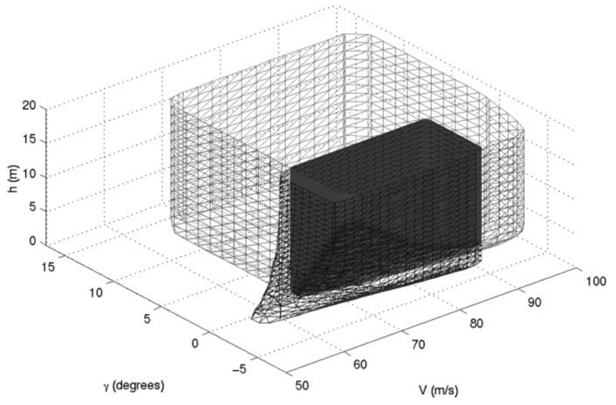

Fig. 18. Solid shape is the safe region $W_F \cap W_T$, from which safe landing and safe go-around is possible. The meshes depict W_{F} and W_{T}.

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

VI. 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, while five dimensional problems take weeks. We are working on a variant of our algorithm that 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. Our initial algorithmic and experimental results are presented in [52], in which we show that a fairly tight overapproximation of the set shown in Fig. 6 may be computed in less than 1 min (compared to 5 min for the full set). We are also developing methods to compute tight polyhedral overapproximations of the reachable set for general nonlinear hybrid systems [38]—these methods scale well in continuous dimension, as they do not require gridding the state space.

Perhaps more exciting are the applications that we are currently working on: we have used reachable set computations to design safe emergency escape maneuvers for dual aircraft closely spaced parallel approaches [82], and we are currently working with Boeing St. Louis on a real-time application of our pursuit-evasion game (under the DARPA Software Enabled Control program). Also, we believe that these algorithms have great potential for human–automation interface design.

ACKNOWLEDGMENT

The authors would like to thank J. Lygeros and S. Sastry, who are coauthors of the hybrid system algorithm and provided valuable insight into the more recent viscosity solution proofs. They thank R. Fedkiw and S. Osher for their help and discussions regarding level set methods; the rendering software used in Section III was written by Fedkiw. Finally, the authors thank A. Degani and M. Heymann for their collaboration and helpful comments on the autoland analysis work.

REFERENCES

[1] J. Glanz, "Mathematical logic flushes out the bugs in chip designs," *Science*, vol. 267, pp. 332–333, Jan. 1995.

[2] J. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*. Princeton, NJ: Princeton Univ. Press, 1947.

[3] A. Church, "Logic, arithmetic, and automata," in *Proc. Int. Congr. Mathematicians*, 1962, pp. 23–35.

[4] R. Isaacs, *Differential Games*. New York: Wiley, 1967.

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

[6] I. Mitchell, A. M. Bayen, and C. J. Tomlin, "Computing reachable sets for continuous dynamic games using level set methods," *IEEE Trans. Automat. Contr.*, submitted for publication.

[7] C. J. Tomlin, J. Lygeros, and S. Sastry, "A game theoretic approach to controller design for hybrid systems," *Proc. IEEE*, vol. 88, pp. 949–970, July 2000.

[8] I. Mitchell, "Application of level set methods to control and reachability problems in continuous and hybrid systems," Ph.D. dissertation, Dept. Scientific Computing and Computational Mathematics, Stanford Univ., Stanford, CA, Aug. 2002.

[9] 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," *AIAA J. Guidance, Contr., Dyn.*, submitted for publication.

[10] M. Oishi, I. Mitchell, A. M. Bayen, C. J. Tomlin, and A. Degani, "Hybrid verification of an interface for an automatic landing," presented at the Proc. IEEE Conf. Decision and Control, Las Vegas, NV, Dec. 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," *Proc. IEEE*, vol. 77, pp. 81–98, Jan. 1989.

[14] J. R. Büchi and L. H. Landweber, "Solving sequential conditions by finite-state operators," *Proc. Amer. Math. Soc.*, pp. 295–311, 1969.

[15] C. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Boston, MA: Kluwer, 1999.

[16] R. Brockett, "Hybrid models for motion control systems," in *Perspectives in Control*, H. Trentelman and J. Willems, Eds. Boston, MA: Birkhauser, 1993, pp. 29–54.

[17] M. S. Branicky, "Control of hybrid systems," Ph.D. dissertation, Dept. Electrical Engineering and Computer Sciences, Massachusetts Inst. Technol., Cambridge, 1994.

[18] J. Lygeros, "Hierarchical, hybrid control of large scale systems," Ph.D. dissertation, Dept. Electrical Engineering and Computer Sciences, Univ. California, Berkeley, 1996.

[19] A. Nerode and W. Kohn, "Models for hybrid systems: Automata, topologies, controllability, observability," in *Lecture Notes in Computer Science, Hybrid Systems*, R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, Eds. New York: Springer-Verlag, 1993, vol. 736, pp. 317–356.

[20] R. Alur and D. Dill, "A theory of timed automata," *Theor. Comput. Sci.*, 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 *Lecture Notes in Computer Science, 11th International Conference on Analysis and Optimization of Systems: Discrete-Event Systems*, G. Cohen and J.-P. Quadrat, Eds. New York: Springer-Verlag, 1994, vol. 199, pp. 331–351.

[22] T. Henzinger, "The theory of hybrid automata," in *Proc. 11th Annu. Symp. Logic in Computer Science*. Los Alamitos, CA: IEEE Comput. Soc. Press, 1996, pp. 278–292.

[23] A. Puri and P. Varaiya, "Decidability of hybrid systems with rectangular differential inclusions," in *Lecture Notes in Computer Science, CAV94: Computer-Aided Verification*. New York: Springer-Verlag, 1995, vol. 818, pp. 95–104.

[24] O. Shakernia, G. J. Pappas, and S. S. Sastry, "Decidable controller synthesis for classes of linear systems," in *Lecture Notes in Computer Science, Hybrid Systems: Computation and Control*, B. Krogh and N. Lynch, Eds. New York: Springer-Verlag, 2000, vol. 1790, pp. 407–420.

[25] N. Lynch, R. Segala, and F. Vaandraager, "Hybrid I/O automata," MIT Laboratory for Computer Science, Cambridge, MA, Tech. Rep. MIT-LCS-TR-827b, 2002.

[26] D. L. Dill, "The Murϕ verification system," in *Lecture Notes in Computer Science, Conference on Computer-Aided Verification*. New York: Springer-Verlag, 1996, pp. 390–393.

[27] S. Owre, J. M. Rushby, and N. Shankar, "PVS: A prototype verification system," in *Lecture Notes in Artificial Intelligence, 11th International Conference on Automated Deduction (CADE)*, D. Kapur, Ed. New York: Springer-Verlag, 1992, vol. 607, pp. 748–752.

[28] J. Burch, E. M. Clarke, K. McMillan, D. Dill, and L. Hwang, "Symbolic model checking: 10^{20} states and beyond," *Inform. Comput.*, vol. 98, pp. 142–170, June 1992.

[29] G. Holzmann, "The model checker spin," *IEEE Trans. Software Eng. (Special Issue on Formal Methods in Software Practice)*, vol. 23, pp. 279–295, May 1997.

[30] T. Dang, "Vérification et synthèse des systèmes hybrides," Ph.D. dissertation, Inst. National Polytechnique de Grenoble (Verimag), 2000.

[31] E. Asarin, O. Bournez, T. Dang, and O. Maler, "Approximate reachability analysis of piecewise-linear dynamical systems," in *Lecture Notes in Computer Science, Hybrid Systems: Computation and Control*, B. Krogh and N. Lynch, Eds. New York: Springer-Verlag, 2000, vol. 1790, pp. 21–31.

[32] A. Chutinan and B. H. Krogh, "Approximating quotient transition systems for hybrid systems," in *Proc. Amer. Control Conf.*, Chicago, IL, 2000, pp. 1689–1693.

[33] ——, "Verification of infinite-state dynamic systems using approximate quotient transition systems," *IEEE Trans. Automat. Contr.*, vol. 46, pp. 1401–1410, Sept. 2001.

[34] A. Bemporad and M. Morari, "Verification of hybrid systems via mathematical programming," in *Lecture Notes in Computer Science, Hybrid Systems: Computation and Control*, F. Vaandrager and J. H. van Schuppen, Eds. Berlin, Germany: Springer-Verlag, 1999, vol. 1569, pp. 30–45.

[35] A. Bemporad, F. D. Torrisi, and M. Morari, "Optimization-based verification and stability characterization of piecewise affine and hybrid systems," in *Lecture Notes in Computer Science, Hybrid Systems: Computation and Control*, B. Krogh and N. Lynch, Eds. Berlin, Germany: Springer-Verlag, 2000, vol. 1790, pp. 45–59.

[36] O. Botchkarev and S. Tripakis, "Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations," in *Lecture Notes in Computer Science, Hybrid Systems: Computation and Control*, B. Krogh and N. Lynch, Eds. Berlin, Germany: Springer-Verlag, 2000, vol. 1790, pp. 73–88.

[37] A. B. Kurzhanski and P. Varaiya, "Ellipsoidal techniques for reachability analysis," in *Lecture Notes in Computer Science, Hybrid Systems: Computation and Control*, B. Krogh and N. Lynch, Eds. Berlin, Germany: Springer-Verlag, 2000, vol. 1790, pp. 202–214.

[38] I. Hwang, D. Stipanovic, and C. Tomlin, "Polyhedral reachable sets for continuous and hybrid systems," presented at the 2003 Amer. Control Conf, Denver, CO.

[39] M. Greenstreet and I. Mitchell, "Integrating projections," in *Lecture Notes in Computer Science, Hybrid Systems: Computation and Control*, S. Sastry and T. Henzinger, Eds. New York: Springer-Verlag, 1998, vol. 1386, pp. 159–174.

[40] ——, "Reachability analysis using polygonal projections," in *Lecture Notes in Computer Science, Hybrid Systems: Computation and Control*, F. Vaandrager and J. H. van Schuppen, Eds. New York: Springer-Verlag, 1999, vol. 1569, pp. 103–116.

[41] R. Vidal, S. Schaffert, J. Lygeros, and S. S. Sastry, "Controlled invariance of discrete time systems," in *Lecture Notes in Computer Science, Hybrid Systems: Computation and Control*, B. Krogh and N. Lynch, Eds. New York: Springer-Verlag, 2000, vol. 1790, pp. 437–450.

[42] A. Tiwari and G. Khanna, "Series of abstractions for hybrid automata," in *Lecture Notes in Computer Science, Hybrid Systems: Computation and Control*, C. J. Tomlin and M. R. Greenstreet, Eds. New York: Springer-Verlag, 2002, vol. 2289, pp. 465–478.

[43] M. Bardi and I. Capuzzo-Dolcetta, *Optimal Control and Viscosity Solutions of Hamilton–Jacobi–Bellman Equations*. Boston, MA: Birkhäuser, 1997.

[44] 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. Boston, MA: Birkhäuser, 1999, vol. 4, Annals of International Society of Dynamic Games.

[45] S. Osher and C.-W. Shu, "High-order essentially nonoscillatory schemes for Hamilton–Jacobi equations," *SIAM J. Numer. Anal.*, vol. 28, no. 4, pp. 907–922, 1991.

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

[47] M. G. Crandall, L. C. Evans, and P.-L. Lions, "Some properties of viscosity solutions of Hamilton–Jacobi equations," *Trans. Amer. Math. Soc.*, vol. 282, no. 2, pp. 487–502, 1984.

[48] L. C. Evans and P. E. Souganidis, "Differential games and representation formulas for solutions of Hamilton–Jacobi–Isaacs equations," *Indiana Univ. Math. J.*, vol. 33, no. 5, pp. 773–797, 1984.

[49] J. Lygeros, C. Tomlin, and S. Sastry, "Controllers for reachability specifications for hybrid systems," *Automatica*, vol. 35, no. 3, pp. 349–370, 1999.

[50] I. Mitchell and C. Tomlin, "Level set methods for computation in hybrid systems," in *Lecture Notes in Computer Science, Hybrid Systems: Computation and Control*, B. Krogh and N. Lynch, Eds. New York: Springer-Verlag, 2000, vol. 1790, pp. 310–323.

[51] I. Mitchell, A. M. Bayen, and C. J. Tomlin, "Validating a Hamilton–Jacobi approximation to hybrid system reachable sets," in *Lecture Notes in Computer Science, Hybrid Systems: Computation and Control*, M. D. D. Benedetto and A. Sangiovanni Vincentelli, Eds. New York: Springer Verlag, 2001, vol. 2034, pp. 418–432.

[52] I. Mitchell and C. J. Tomlin, "Overapproximating reachable sets by Hamilton–Jacobi projections," *J. Sci. Comput.*, to be published.

[53] I. Mitchell, "Games of two identical vehicles," Dept. Aeronautics and Astronautics, Stanford Univ., Stanford, CA, Tech. Rep. SUDAAR 740, July 2001.

[54] A. W. Merz, "The game of two identical cars," *J. Optim. Theory Applicat.*, vol. 9, no. 5, pp. 324–343, 1972.

[55] A. M. Bayen and C. J. Tomlin, "A construction procedure using characteristics for viscosity solutions of the Hamilton–Jacobi equation," in *Proc. IEEE Conf. Decision and Control*, Orlando, FL, 2001, pp. 1657–1662.

[56] G.-S. Jiang and D. Peng, "Weighted ENO schemes for Hamilton–Jacobi equations," *SIAM J. Sci. Comput.*, vol. 21, pp. 2126–2143, 2000.

[57] J. A. Sethian, *Level Set Methods and Fast Marching Methods*. Cambridge, U.K.: Cambridge Univ. Press, 1999.

[58] M. G. Crandall and P.-L. Lions, "Two approximations of solutions of Hamilton–Jacobi equations," *Math. Comput.*, vol. 43, no. 167, pp. 1–19, 1984.

[59] C.-W. Shu and S. Osher, "Efficient implementation of essentially nonoscillatory shock-capturing schemes," *J. Comput. Phys.*, vol. 77, pp. 439–471, 1988.

[60] I. Mitchell. (2001). [Online]. Available: http://cherokee.stanford.edu/~mitchell

[61] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata," presented at the Proc. 27th Annu. ACM Symp. Theory of Computing, 1995.

[62] K. Larsen, P. Pettersson, and W. Yi, "Uppaal in a nutshell," *Softw. Tools Technol. Transfer*, vol. 1, 1997.

[63] S. Yovine, "Kronos: A verification tool for real-time systems," *Softw. Tools Technol. Transfer*, vol. 1, pp. 123–133, 1997.

[64] 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 *Lecture Notes in Computer Science, Hybrid Systems*, R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, Eds. New York: Springer-Verlag, 1993, vol. 736, pp. 366–392.

[65] T. A. Henzinger, P. Ho, and H. W. Toi, "HyTech: A model checker for hybrid systems," *Softw. Tools Technol. Transfer*, vol. 1, pp. 110–122, 1997.

[66] O. Maler, A. Pnueli, and J. Sifakis, "On the synthesis of discrete controllers for timed systems," in *Lecture Notes in Computer Science, STACS 95: Theoretical Aspects of Computer Science*, E. W. Mayr and C. Puech, Eds. Berlin, Germany: Springer Verlag, 1995, vol. 900, pp. 229–242.

[67] E. Asarin, O. Maler, and A. Pnueli, "Symbolic controller synthesis for discrete and timed systems," in *Lecture Notes in Computer Science, Hybrid Systems II*, P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, Eds. New York: Springer-Verlag, 1995, vol. 999.

[68] H. W. Toi, "The synthesis of controllers for linear hybrid automata," presented at the IEEE Conf. Decision and Control, San Diego, CA, 1997.

[69] J.-P. Aubin, J. Lygeros, M. Quincampoix, S. Sastry, and N. Seube, "Impulse differential inclusions: A viability approach to hybrid systems," *IEEE Trans. Automat. Contr.*, vol. 47, pp. 2–20, Jan. 2002.

[70] 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 *Lecture Notes in Computer Science, Hybrid Systems: Computation and Control*, C. J. Tomlin and M. R. Greenstreet, Eds. New York: Springer-Verlag, 2002, vol. 2289, pp. 90–104.

[71] A. Balluchi, M. D. Benedetto, C. Pinello, C. Rossi, and A. Sangionvanni-Vincentelli, "Hybrid control for automotive engine management: The cut-off case," in *Lecture Notes in Computer Science, Hybrid Systems: Computation and Control*, T. Henzinger and S. Sastry, Eds. New York: Springer-Verlag, 1998, vol. 1386, pp. 13–32.

[72] *Special Issue on Verification of Hybrid Systems—Results of a European Union Esprit Project, Eur. J. Contr.*, vol. 7, no. 4, 2001.

[73] C. J. Tomlin, I. Mitchell, and R. Ghosh, "Safety verification of conflict resolution maneuvers," *IEEE Trans. Intell. Transport. Syst.*, vol. 2, pp. 110–120, June 2001.

[74] M. Oishi, C. J. Tomlin, V. Gopal, and D. Godbole, "Addressing multiobjective control: Safety and performance through constrained optimization," in *Lecture Notes in Computer Science, Hybrid Systems: Computation and Control*, M. D. D. Benedetto and A. Sangiovanni-Vincentelli, Eds. New York: Springer-Verlag, 2001, vol. 2034, pp. 459–472.

[75] T. Dang and O. Maler, "Reachability analysis via face lifting," in *Lecture Notes in Computer Science, Hybrid Systems: Computation and Control*, S. Sastry and T. Henzinger, Eds. New York: Springer-Verlag, 1998, vol. 1386, pp. 96–109.

[76] A. Degani, "Modeling human-machine systems: On modes, error, and patterns of interaction," Ph.D. dissertation, Dept. Industrial and Systems Engineering, Georgia Inst. Technol., Atlanta, 1996.

[77] C. J. Tomlin, "Hybrid control of air traffic management systems," Ph.D. dissertation, Dept. Electrical Engineering, Univ. California, Berkeley, 1998.

[78] I. M. Kroo, *Aircraft Design: Synthesis and Analysis*. Stanford, CA: Desktop Aeronautics Inc., 1999.

[79] J. Anderson, *Fundamentals of Aerodynamics*. New York: McGraw-Hill, 1991.

[80] *Federal Aviation Regulations*, 1990. Section 25.125 (landing).

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

[82] R. Teo and C. J. Tomlin, “Computing danger zones for provably safe closely spaced parallel approaches,” *J. Guidance, Contr., Dyn.*, vol. 26, pp. 434–443, May/June 2003.

Claire Tomlin received the Ph.D. degree in electrical engineering from the University of California, Berkeley, in 1998.

Since September 1998, she has been an Assistant Professor in the Department of Aeronautics and Astronautics at Stanford University, Stanford, CA, with a courtesy appointment in Electrical Engineering. She was a Graduate Fellow in the Division of Applied Sciences at Harvard University in 1994, and she has been a Visiting Researcher at NASA Ames Research Center during 1994–1998, at Honeywell Technology Center in 1997, and at the University of British Columbia in 1994. She was an invited participant in the National Academy of Engineering’s Frontiers of Engineering Program in 2002, and she is currently a member of DARPA’s Information Systems and Technology (ISAT) study group. Her research interests are in hybrid systems, air traffic control automation, flight management system analysis and design, and modeling and analysis of biological cell networks.

Dr. Tomlin is the recipient of the 2003 Donald P. Eckman Award of the American Automatic Control Council, the AIAA Outstanding Teacher Award, Stanford (2001), NSF Career Award, Stanford (1999), Terman Fellowship, Stanford (1998), the Bernard Friedman Memorial Prize in Applied Mathematics, Berkeley (1998), and the Zonta Amelia Earhart Awards for Aeronautics Research (1996–98).

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

After spending a year as a Postdoctoral Researcher in the Department of Electrical Engineering and Computer Science at the University of California, Berkeley, and the Department of Computer Science at Stanford University, he will join the faculty in the Department of Computer Science at the University of British Columbia as an Assistant Professor. His research interests include scientific computing, hybrid systems, verification, and optimization.

Dr. Mitchell is the recipient of a 1999 SIAM/AAAS Mass Media Fellowship and a 1997–98 Stanford School of Engineering Graduate Fellowship.

Alexandre M. Bayen received the B.S. degree in applied mathematics from the Ecole Polytechnique, Paris, France, in 1998, and the M.S. degree in aeronautics and astronautics from Stanford University, Stanford, CA, in 1999. He is currently working toward the Ph.D. degree in aeronautics and astronautics at Stanford University.

He was a Visiting Researcher at NASA Ames Research Center from 2000 to 2003. His research interests include combinatorial optimization, hybrid systems, air traffic automation, viability theory and optimal control.

Mr. Bayen is the recipient of the Graduate Fellowship of the Délégation Générale pour l’Armement (1998–2002) from France.

Meeko Oishi received the B.S.E. degree in mechanical engineering from Princeton University, Princeton, NJ, in 1998, and the M.S. degree in mechanical engineering from Stanford University, Stanford, CA, in 2000. She is a graduate student in the Hybrid Systems Laboratory at Stanford University, and a Ph.D. candidate in the Mechanical Engineering Department.

She has been a Visiting Researcher at NASA Ames Research Center (2001–2003) and at Honeywell Technology Center (2000), and has held summer internships at Boeing, Intel, and Sandia National Laboratories. Her research interests include hybrid and nonlinear systems, user-interface analysis and design, flight management systems, and theoretical ecology.

Ms. Oishi is the recipient of the NSF Graduate Research Fellowship (1998–1999, 2000–2002) and the John Bienkowski Memorial Prize from the Mechanical and Aerospace Department at Princeton University (1998).