6.43.28.6 VERIFICATION OF HYBRID SYSTEMS[1]

Claire J. Tomlin, Department of Aeronautics and Astronautics, Stanford University, Stanford CA 94305-4035, USA.
Ian Mitchell, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94709, USA.
Alexandre M. Bayen, Department of Aeronautics and Astronautics, Stanford University, Stanford CA 94305-4035, USA.

Keywords: Hybrid Systems; Verification; Reachability; Safety.

Contents

Glossary

Hybrid automaton: A dynamical system which combines continuous-state and discrete-state dynamics, in order to model systems which evolve both continuously and according to discrete jumps.

Safety verification: Given a region of the state space which represents unsafe operation, safety verification is a proof 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.

Backwards reachable set: Given a target set of states, this is the set of states from which trajectories start that can reach the target set.

Controller synthesis for safety: Extraction (from the reachable set computation) of the control law which must be used on the boundary of the reachable set, in order to keep the system state out of this reachable set.

Decidable problem: A problem that may be solved by an algorithm which terminates in a finite number of steps.

Summary

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 which involve the interaction of both discrete state systems, represented by finite automata, and continuous state dynamics, represented by differential equations. The embedded autopi-

[1]This research is supported by DARPA under the Software Enabled Control Program (AFRL contract F33615-99-C-3014), by the DoD Multidisciplinary University Research Initiative (MURI) program administered by the Office of Naval Research under Grant N00014-02-1-0720.

lot 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 chapter, a general model for a hybrid system is first presented, along with an overview of methods for verifying continuous and hybrid systems. Then, a particular verification technique for hybrid systems, based on two-person zero-sum game theory for automata and continuous dynamical systems, is described. A numerical implementation of this technique using level set methods, and its use in the design and analysis of aircraft collision avoidance protocols, and in verification of autopilot logic, is demonstrated.

1. 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 which control aircraft cockpit displays and design standards for a military hardware bus. Discrete analysis, however, is not rich enough to verify systems which 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 which 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 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 chapter, recent research results are summarized, and a detailed set of references is presented, 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; in the continuous domain, control problems of the safety type have been addressed in the context of differential games. For systems involving continuous dynamics, it is very difficult to compute and represent the set of states reachable from some initial set. In this chapter, recent solutions

to the problem are presented, including a method, based on the level set techniques of Osher and Sethian, which determines an implicit representation of the boundary of this *reachable set*. This method is based on the theorem, proved using two-person zero-sum game theory for continuous dynamical systems, that the viscosity solution of a particular Hamilton-Jacobi partial differential equation corresponds exactly to the boundary of the reachable set. In addition, it is shown 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. Two interesting examples, one in the verification of protocols for aircraft collision avoidance, and one in the verification of mode switching logic in autopilots, are presented in this chapter. Other applications that have been studied in this framework are surveyed. This chapter concludes with a discussion of problem complexity.

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 $\dot{x} = f(\cdot, \cdot, \cdot)$, 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 $d(t)$. Sastry and Doyle are recommended 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 which is a finite set of *discrete state variables* Q, a set of input variables $\Sigma = \Sigma_u \times \Sigma_d$ which is the Cartesian product 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 discrete states and inputs, which satisfies the transition relation at each step. The original work of Ramadge and Wonham brought the use of discrete state systems to control, though parallels can be drawn between this work and that of Church, Büchi and Landweber who originally analyzed the von Neumann-Morgenstern discrete games. A comprehensive reference for modeling and control of discrete state systems is Cassandras and Lafortune.

Control algorithms are 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 $S = Q \times \mathbb{R}^n$ is the Cartesian product of discrete and continuous states; $\mathsf{Init} \subseteq S$ is a set of initial states; $In = (\Sigma_u \times \Sigma_d) \times (\mathcal{U} \times \mathcal{D})$ is the set 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 a sequence of continuous flows combined with discrete jumps. 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.

The hybrid automaton model presented above allows for general nonlinear dynamics. This model was developed from those of Brockett, Branicky, Lygeros, and Nerode and Kohn, for which the emphasis was 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 the timed automata of Alur and Dill, linear hybrid automata of Henzinger, and hybrid I/O automata of Lynch.

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, the initial conditions are specified and one seeks to determine the set of all states that can be reached along trajectories that start in that set. Conversely, for a *backward reachable* set, a final or target set of states is specified, and one seeks to determine the set of states from which trajectories start that can reach that target set. For time invariant systems $\dot{x} = f(x)$ without input, it is easy to show that the backwards reachable set is the forwards reachable set of $\dot{x} = -f(x)$. It is interesting to note that the forward and backward reachable sets are not simply time

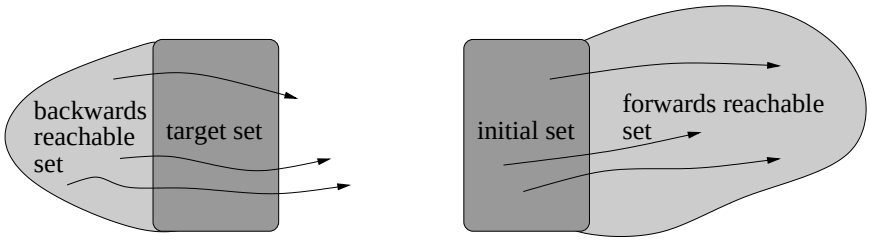

Figure 1: Difference between backwards and forwards reachable sets.

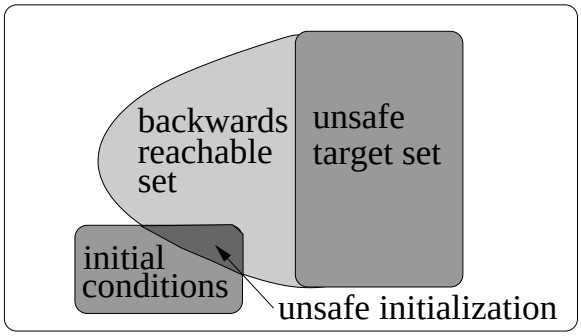

Figure 2: Using the backwards reachable set to verify safety.

reversals of each other. The difference is illustrated 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 (Dill), PVS, SMV, and SPIN. 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, it is necessary 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 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 the system becoming safe, if the disturbance is

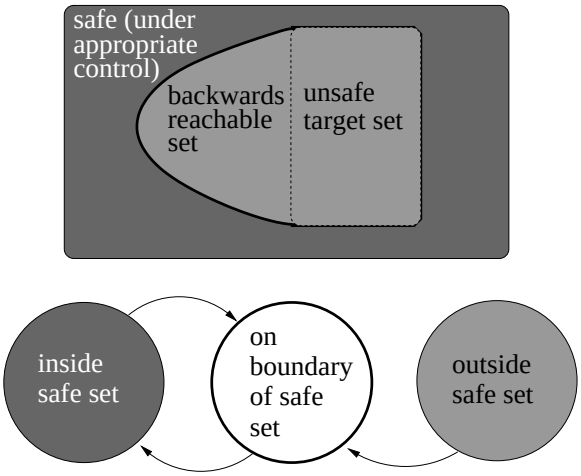

Figure 3: A discrete abstraction with appropriate control information.

not playing optimally for itself.

3. 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. A group of methods which seek an efficient overapproximation of the reachable set is classified as "overapproximative". The tools **d/dt** (Maler) and *Checkmate* (Krogh) 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. *VeriSHIFT* uses ellipsoidal overapproximations of reach sets for linear systems with linear input; it implements techniques developed by Kurzhanski and Varaiya. The tool *Coho*, developed by Greenstreet and Mitchell uses as set representation two dimensional projections of higher dimensional non-convex polyhedra, and evolves these "projectagons" under affine over-approximations of nonlinear dynamics using linear programming. A recent algorithm by Tiwari and Khanna 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 static Hamilton-Jacobi equations and to techniques from viability theory and set valued analysis . In our work, we have developed a reachability computation method based on level set techniques and viscosity solutions to Hamilton-Jacobi equations. A set is represented 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. These convergent approximative methods allow for both control inputs and disturbance inputs in the problem formulation, and they compute a

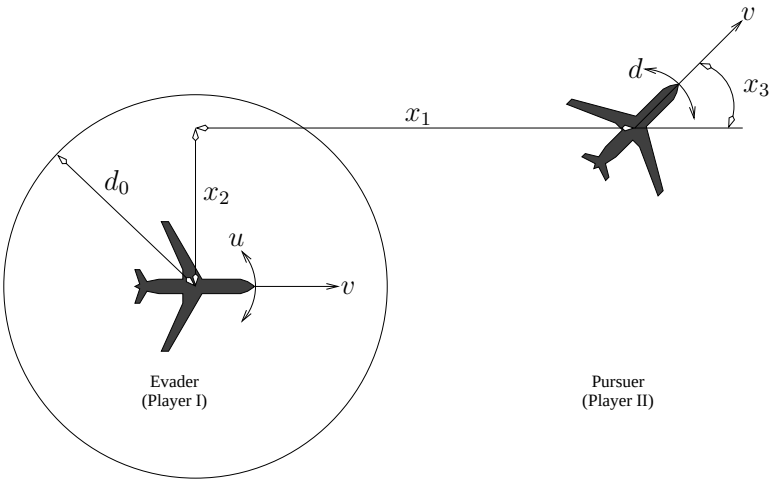

Figure 4: Relative coordinate system. Origin is located at the center of the evader.

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, methodology and results for computing reachable sets for continuous systems (1) are presented. The material in this section is presented in detail in the Ph.D. dissertation of Ian Mitchell.

3.1 A Game of Two Identical Vehicles

Consider as a demonstration example, a classical pursuit evasion game involving two identical vehicles (see Merz for more details). If the vehicles get too close together, 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 state vector consisting of a location in the plane and a heading. Isaacs pioneered a framework for solving such games, using a method similar to the method of characteristics.

The problem is studied in relative coordinates (see Figure 4). The vehicles are drawn as aircraft, as this example has been used as inspiration for verifying two-aircraft tactical conflict avoidance strategies in Air Traffic Control.

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 $[x_1 \quad x_2]^T \in \mathbb{R}^2$ and relative heading $x_3 \in [0, 2\pi]$, and $v \geq 0$ is the linear velocity of each aircraft. In Figure 4 the relative heading is measured counter clockwise 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, the set of initial states from which the pursuer can cause a collision despite the best efforts of the evader must be determined.

3.2 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 $s \in [0, \tau]$.

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

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

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.

A very general implicit surface function representation for the reachable set is used: 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

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

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, functions $\mathcal{G}_0$ which are unions, intersections, and set differences can be constructed. 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 $\mathcal{G}_1 \setminus \mathcal{G}_2$.

The main result of Mitchell's work is to show that an implicit surface representation of the

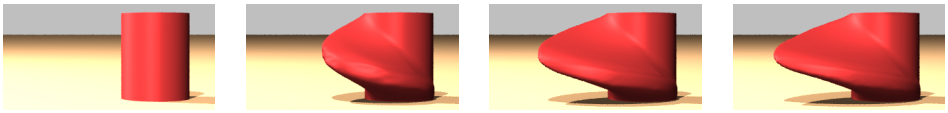

Figure 5: Growth of the reachable set.

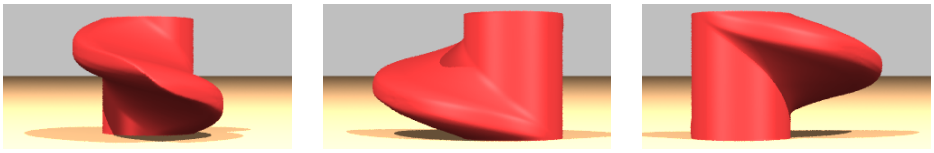

Figure 6: Other views of the reachable set.

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

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

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

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

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 backwards reachable set as

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

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

There are a few 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, note that the $\max_u \min_d$ operation in computing the Hamiltonian (7) 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 application to system verification under uncertainty.

3.3 Collision Avoidance Example Results

This numerical method can be applied to the collision avoidance problem. In Figure 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 earlier in this section, the reachable set converges to a fixed point for $\tau \gtrsim 2.6$. Figure 6 shows several

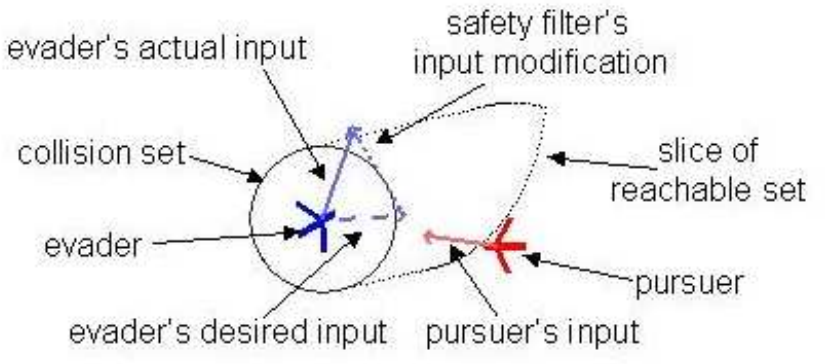

Figure 7: Annotated frame from collision avoidance example animation.

Figure 8: Evader keeps pursuer from entering reachable set, and hence avoids collision.

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.

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 (which 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.

Figure 7 shows an annotated frame from an animation of the collision system, and a series of frames from that animation are shown in Figure 8, progressing from left to right. The evader starts on the left surrounded by the collision circle, while the pursuer starts on the right. The dotted shape surrounding the evader is the slice of the reachable set for the

Figure 9: Pursuer starts within the reachable set, and can thus cause a collision despite the evader's efforts.

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. Figure 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 Figure 6 took about 5 minutes to compute on a three dimensional grid using a standard Pentium III laptop; four dimensional problems can take anywhere from a few hours to a few days to run. In the final section of this chapter, current work in computing projective overapproximations to decrease the computation time to achieve a useful result are discussed.

4. Verifying Hybrid Systems

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.

4.1 Background

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 problem classes which could be solved in a finite number of steps: it was shown that decidability results exist for timed and some classes of linear hybrid automata. Software tools were designed to automatically compute reachable sets for these systems: Uppaal and Kronos for timed automata, and HyTech 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 described here was motivated by work on reachability computation and controller synthesis on timed automata, and that for controller synthesis on linear hybrid automata. Tools based on the analysis of piecewise linear systems, using mathematical programming tools such as CPLEX have found success in several industrial applications.

The hybrid system analysis algorithm presented here is built upon the implicit reachable set representation and level set implementation for continuous systems, allowing representation and analysis of nonlinear hybrid systems, with generally shaped sets. In this sense, this work is related to that of the viability community , which has extended these concepts to hybrid systems; 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: 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, experimental industrial batch plants, vehicle collision avoidance problems, as well as envelope protection. The problems that have been solved to date are generally of low dimension: even the overapproximative 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 protection on nonlinear, hybrid systems with three continuous dimensions, representing the longitudinal dynamics of jet aircraft under hybrid control.

4.2 Computing Reachable Sets for Hybrid Systems

The algorithm is first described with a picture, and then the details of a few key components are presented.

Consider the sequence of eight diagrams in figure 10. 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, 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). 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). 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 the previous section 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. Call this reachable set the *reach-avoid set*, 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:

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

for a level set function $g : Q \times \mathbb{R}^n \to \mathbb{R}$. Now, one would like 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

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

Therefore $\mathrm{Pre}_u(K)$ contains all states in K for which controllable actions (σ_u, u) can force

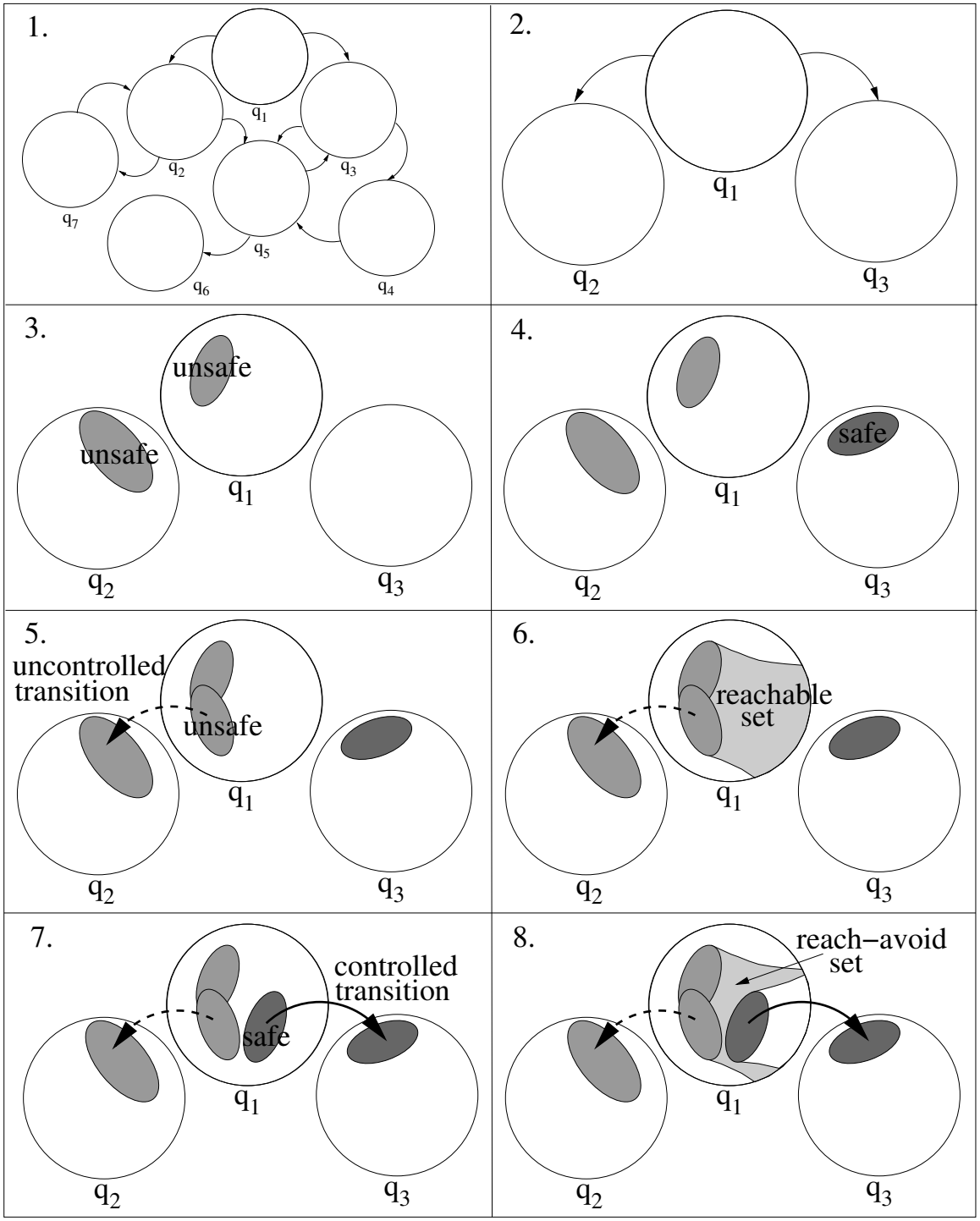

Figure 10: An illustration of the algorithm for computing reachable sets for hybrid systems.

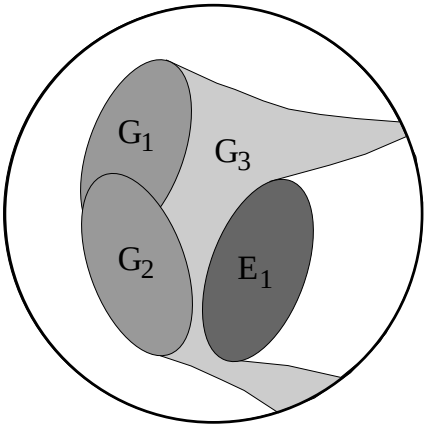

Figure 11: Detail of the reach-avoid set from diagram 8 of Figure 10.

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:

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

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

In the first step of this algorithm, 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 this computational method, for which the system is guaranteed to be safe.

Returning to the pictorial description of the algorithm in Figure 10, and concentrating on the result of one step of the algorithm detailed in Figure 11, 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 $\mathrm{Reach}\left(\mathrm{Pre}_d((W^i)^c), \mathrm{Pre}_u(W^i)\right) = G_3$.

To implement this algorithm, Pre_u, Pre_d, and Reach must be computed. 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 examples, this inversion may be performed by hand. The algorithm for computing Reach(G, E) is a direct modification of the reachable set calculation of the previous section, the details are presented in Ian Mitchell's dissertation.

5. Flight Management System Example

In this section, the hybrid systems analysis is demonstrated on an interesting and current example, the landing of a civilian aircraft.

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 *after* the unsafe situation occurs, and "band-aids" are added to the design to ensure the same problem does not occur again. It is possible 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.

In this example, a landing aircraft is examined, attention is focussed 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, the flap setting as a discrete variable.

A simple point mass model for aircraft vertical navigation is used, which accounts for lift L, drag D, thrust T, and weight mg. The nonlinear longitudinal dynamics are modeled as

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

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 and Prandtl's lifting line theory :

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

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.

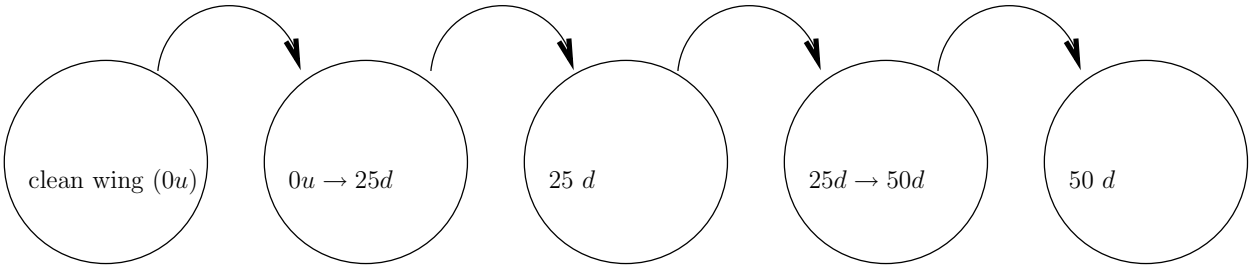

Figure 12: Discrete transition diagram of flap deflection settings. Clean wing represents no deflection, 25 d represents a deflection of 25°, and 50 d, a deflection of 50°. The modes $0u \to 25d$ and $25d \to 50d$ are timed modes to reflect deflection time: if the pilot selects mode 25 d from clean wing,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

In determining $C_L(\alpha)$ standard autoland procedures are assumed: 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. A linear form for the lift coefficient $C_L(\alpha) = h_\delta + 4.2\alpha$ is assumed, where parameters $h_{0°} = 0.2$, $h_{25°} = 0.8$ and $h_{50°} = 1.25$ are determined from experimental data for a DC9-30. 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 $C_D(\alpha) = 0.041 + 0.045C_L^2(\alpha)$ 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

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

In this implementation, three operational modes are considered: $0u$, 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 this implementation, transition modes $0u \to 25d$ and $25d \to 50d$ are defined with timers, in which the aerodynamics are those of (12) with coefficients which interpolate those of the bounding operational modes. The corresponding discrete automaton is shown in Figure 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 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. Restrictions on the flight path angle, aircraft velocity and $t$ouchdown (TD) speed are used to determine the initial safe set W_0:

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

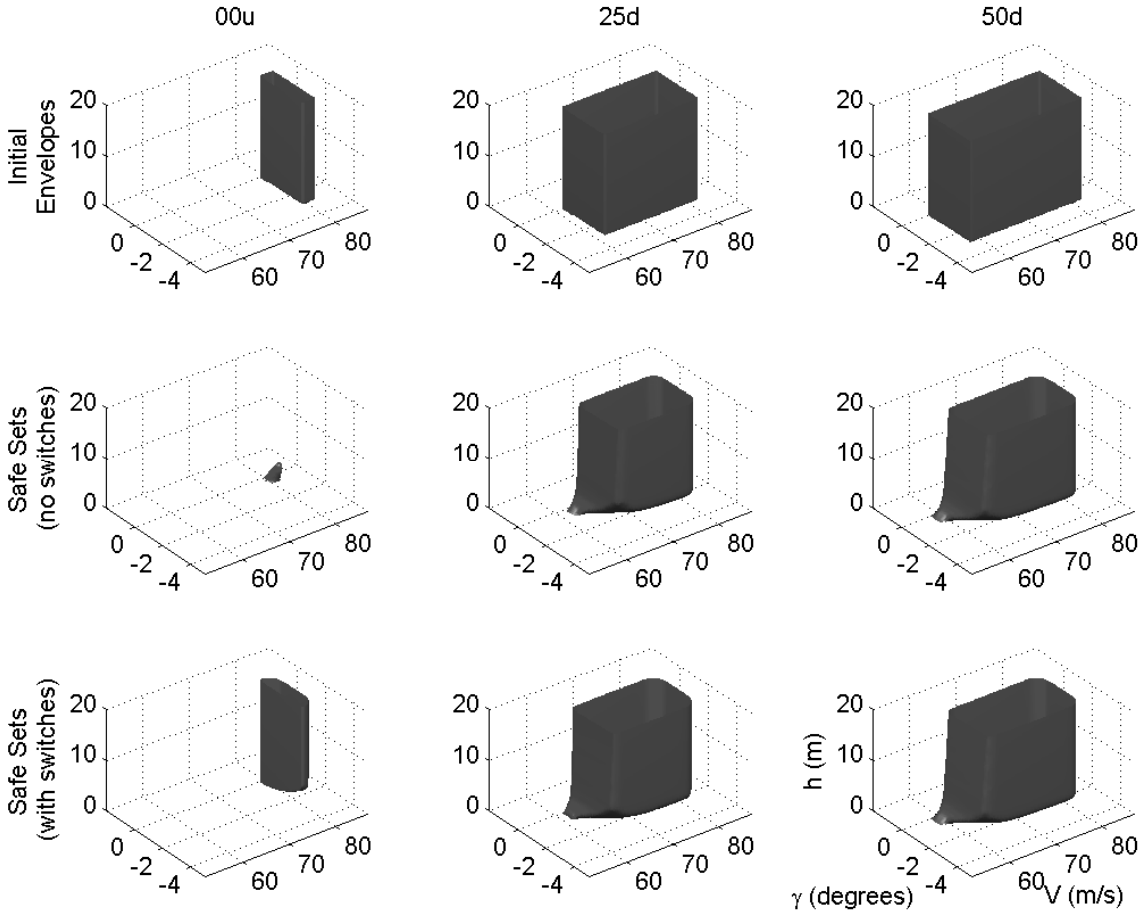

Figure 13: Maximally controllable safe envelopes for the multimode landing example. From left to right the columns represent modes $0u$, $25d$ and $50d$.

Numerical values for a DC9-30 are used: 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 (to a value of $T = 32$kN), and $\alpha \in [0^\circ,\ 10^\circ]$.

The results of the fixed point computation are shown in Figures 13 and 14. The interior of the surface shown in the first row of Figure 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 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 14 shows a slice through the reach and avoid sets for the hybrid analysis

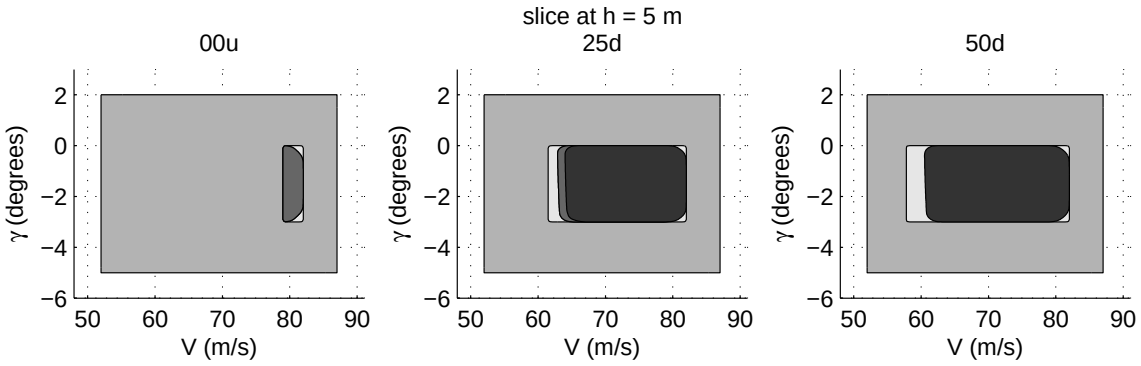

Figure 14: Slices through the reach and avoid sets for the hybrid analysis at a fixed altitude of $h = 5$m. From left to right the columns represent modes $0u$, $25d$ and $50d$.

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.

6. Conclusions

In this chapter, a method and algorithm for hybrid systems analysis, specifically, for the verification of safety properties of hybrid systems, are presented. This algorithm 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. Other available methods are briefly summarized. 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.

Acknowledgments

The authors 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. Meeko Oishi provided valuable insights into the use of hybrid analysis of flight management systems to inspire interface design. The authors also acknowledge the help of Ron Fedkiw and Stan Osher regarding level set methods; the rendering software that was used in this chapter was written by Ron Fedkiw.

References

[Alur et al., 1993] Alur, R., Courcoubetis, C., Henzinger, T. A., and Ho, P.-H. (1993). Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In Grossman, R. L., Nerode, A., Ravn, A. P., and Rischel, H., editors, *Hybrid Systems*, LNCS, pages 366–392. Springer Verlag, New York.

[Alur and Dill, 1994] Alur, R. and Dill, D. (1994). A theory of timed automata. *Theoretical Computer Science*, 126:183–235.

[Anderson, 1991] Anderson, J. (1991). *Fundamentals of Aerodynamics*. McGraw Hill Inc., New York.

[Asarin et al., 2000] Asarin, E., Bournez, O., Dang, T., and Maler, O. (2000). Approximate reachability analysis of piecewise-linear dynamical systems. In Krogh, B. and Lynch, N., editors, *Hybrid Systems: Computation and Control*, LNCS 1790, pages 21–31. Springer Verlag.

[Aubin et al., 2002] Aubin, J.-P., Lygeros, J., Quincampoix, M., Sastry, S., and Seube, N. (2002). Impulse differential inclusions: A viability approach to hybrid systems. *IEEE Transactions on Automatic Control*, 47(1):2–20.

[Balluchi et al., 1998] Balluchi, A., Benedetto, M. D., Pinello, C., Rossi, C., and Sangionvanni-Vincentelli, A. (1998). Hybrid control for automotive engine management: The cut-off case. In Henzinger, T. and Sastry, S., editors, *Hybrid Systems: Computation and Control*, number 1386 in LNCS, pages 13–32. Springer Verlag, New York.

[Bardi and Capuzzo-Dolcetta, 1997] Bardi, M. and Capuzzo-Dolcetta, I. (1997). *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman equations*. Birkhäuser, Boston.

[Bayen et al., 2002a] Bayen, A. M., Crück, E., and Tomlin, C. J. (2002a). Guaranteed overapproximation of unsafe sets for continuous and hybrid systems: Solving the Hamilton-Jacobi equation using viability techniques. In Tomlin, C. J. and Greenstreet, M. R., editors, *Hybrid Systems: Computation and Control*, LNCS 2289, pages 90–104. Springer Verlag.

[Bayen et al., 2002b] Bayen, A. M., Mitchell, I., Oishi, M., and Tomlin, C. J. (2002b). 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.

[Bayen and Tomlin, 2001] Bayen, A. M. and Tomlin, C. J. (2001). A construction procedure using characteristics for viscosity solutions of the Hamilton-Jacobi equation. In *Proceedings of the IEEE Conference on Decision and Control*, pages 1657–1662, Orlando, FL.

[Bemporad and Morari, 1999] Bemporad, A. and Morari, M. (1999). Verification of hybrid systems via mathematical programming. In Vaandrager, F. and van Schuppen, J. H., editors, *Hybrid Systems: Computation and Control*, number 1569 in LNCS, pages 30–45. Springer Verlag, Berlin.

[Botchkarev and Tripakis, 2000] Botchkarev, O. and Tripakis, S. (2000). Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations. In Krogh, B. and Lynch, N., editors, *Hybrid Systems: Computation and Control*, LNCS 1790, pages 73–88. Springer Verlag.

[Branicky, 1994] Branicky, M. S. (1994). *Control of Hybrid Systems*. PhD thesis, Department of Electrical Engineering and Computer Sciences, Massachusetts Institute of Technology.

[Brockett, 1993] Brockett, R. (1993). Hybrid models for motion control systems. In Trentelman, H. and Willems, J., editors, *Perspectives in Control*, pages 29–54. Birkhauser, Boston.

[Büchi and Landweber, 1969] Büchi, J. R. and Landweber, L. H. (1969). Solving sequential conditions by finite-state operators. In *Proceedings of the American Mathematical Society*, pages 295–311.

[Burch et al., 1992] Burch, J., Clarke, E. M., McMillan, K., Dill, D., and Hwang, L. (1992). Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170.

[Cardaliaguet et al., 1999] Cardaliaguet, P., Quincampoix, M., and Saint-Pierre, P. (1999). Set-valued numerical analysis for optimal control and differential games. In Bardi, M., Parthasarathy, T., and Raghavan, T. E. S., editors, *Stochastic and Differential Games: Theory and Numerical Methods*, volume 4 of *Annals of International Society of Dynamic Games*. Birkhäuser.

[Cassandras and Lafortune, 1999] Cassandras, C. and Lafortune, S. (1999). *Introduction to Discrete Event Systems*. Kluwer, Boston.

[Church, 1962] Church, A. (1962). Logic, arithmetic, and automata. In *Proceedings of the International Congress of Mathematicians*, pages 23–35.

[Chutinan and Krogh, 2001] Chutinan, A. and Krogh, B. H. (2001). Verification of infinite-state dynamic systems using approximate quotient transition systems. *IEEE Transactions on Automatic Control*, 46(9):1401–1410.

[Crandall et al., 1984] Crandall, M. G., Evans, L. C., and Lions, P.-L. (1984). Some properties of viscosity solutions of Hamilton-Jacobi equations. *Transactions of the American Mathematical Society*, 282(2):487–502.

[Dang, 2000] Dang, T. (2000). *Vérification et synthèse des systèmes hybrides*. PhD thesis, Institut National Polytechnique de Grenoble (Verimag).

[Dang and Maler, 1998] Dang, T. and Maler, O. (1998). Reachability analysis via face lifting. In Sastry, S. and Henzinger, T., editors, *Hybrid Systems: Computation and Control*, number 1386 in LNCS, pages 96–109. Springer Verlag.

[Dill, 1996] Dill, D. L. (1996). The Murϕ verification system. In *Conference on Computer-Aided Verification*, LNCS, pages 390–393. Springer-Verlag.

[Doyle et al., 1992] Doyle, J., Francis, B., and Tannenbaum, A. (1992). *Feedback Control Theory*. Macmillan, New York.

[Esprit, 2001] Esprit (2001). Verification of hybrid systems: Results of a european union esprit project. In Maler, O., editor, *European Journal of Control*, Vol. 7, Issue 4.

[Greenstreet and Mitchell, 1999] Greenstreet, M. and Mitchell, I. (1999). Reachability analysis using polygonal projections. In Vaandrager, F. and van Schuppen, J. H., editors, *Hybrid Systems: Computation and Control*, number 1569 in LNCS, pages 103–116. Springer Verlag, New York.

[Henzinger, 1996] Henzinger, T. (1996). The theory of hybrid automata. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science*, pages 278–292. IEEE Computer Society Press.

[Henzinger et al., 1997] Henzinger, T. A., Ho, P., and Wong-Toi, H. (1997). HyTech: A model checker for hybrid systems. *Software Tools for Technology Transfer*, 1:110–122.

[Holzmann, 1997] Holzmann, G. (1997). The model checker Spin. *IEEE Transactions on Software Engineering*, 23(5):279–295. Special issue on Formal Methods in Software Practice.

[Isaacs, 1967] Isaacs, R. (1967). *Differential Games*. John Wiley.

[Kurzhanski and Varaiya, 2000] Kurzhanski, A. B. and Varaiya, P. (2000). Ellipsoidal techniques for reachability analysis. In Krogh, B. and Lynch, N., editors, *Hybrid Systems: Computation and Control*, LNCS 1790, pages 202–214. Springer Verlag.

[Larsen et al., 1997] Larsen, K., Pettersson, P., and Yi, W. (1997). Uppaal in a nutshell. *Software Tools for Technology Transfer*, 1.

[Lygeros, 1996] Lygeros, J. (1996). *Hierarchical, Hybrid Control of Large Scale Systems*. PhD thesis, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley.

[Lynch et al., 2002] Lynch, N., Segala, R., and Vaandraager, F. (2002). Hybrid I/O automata. Submitted. Also, Technical Report MIT-LCS-TR-827b, MIT Laboratory for Computer Science, Cambridge, MA 02139.

[Merz, 1972] Merz, A. W. (1972). The game of two identical cars. *Journal of Optimization Theory and Applications*, 9(5):324–343.

[Mitchell, 2001] Mitchell, I. (2001). Games of two identical vehicles. Technical report, SUDAAR 740, Stanford University Department of Aeronautics and Astronautics.

[Mitchell, 2002] Mitchell, I. (2002). *Application of Level Set Methods to Control and Reachability Problems in Continuous and Hybrid Systems*. PhD thesis, Scientific Computing and Computational Mathematics, Stanford University.

[Mitchell et al., 2001] Mitchell, I., Bayen, A. M., and Tomlin, C. J. (2001). Validating a Hamilton-Jacobi approximation to hybrid system reachable sets. In Benedetto, M. D. D. and Sangiovanni-Vincentelli, A., editors, *Hybrid Systems: Computation and Control*, LNCS 2034, pages 418–432. Springer Verlag.

[Mitchell et al., 2002] Mitchell, I., Bayen, A. M., and Tomlin, C. J. (2002). Computing reachable sets for continuous dynamic games using level set methods. *IEEE Transactions on Automatic Control*. Submitted.

[Mitchell and Tomlin, 2002] Mitchell, I. and Tomlin, C. J. (2002). Overapproximating reachable sets by Hamilton-Jacobi projections. *Journal of Scientific Computing*. Submitted May 2002. Accepted with minor revisions August 2002.

[Nerode and Kohn, 1993] Nerode, A. and Kohn, W. (1993). Models for hybrid systems: Automata, topologies, controllability, observability. In Grossman, R. L., Nerode, A., Ravn, A. P., and Rischel, H., editors, *Hybrid Systems*, LNCS 736, pages 317–356. Springer Verlag, New York.

[Oishi et al., 2001] Oishi, M., Tomlin, C. J., Gopal, V., and Godbole, D. (2001). Addressing multiobjective control: Safety and performance through constrained optimization. In Benedetto, M. D. D. and Sangiovanni-Vincentelli, A., editors, *Hybrid Systems: Computation and Control*, LNCS 2034, pages 459–472. Springer Verlag.

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

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

[Owre et al., 1992] Owre, S., Rushby, J. M., and Shankar, N. (1992). PVS: A prototype verification system. In Kapur, D., editor, *11th International Conference on Automated Deduction (CADE)*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 748–752, Saratoga, NY. Springer-Verlag.

[Ramadge and Wonham, 1989] Ramadge, P. J. G. and Wonham, W. M. (1989). The control of discrete event dynamical systems. *Proceedings of the IEEE*, Vol.77(1):81–98.

[Sastry, 1999] Sastry, S. S. (1999). *Nonlinear Systems: Analysis, Stability and Control*. Springer Verlag, New York.

[Tiwari and Khanna, 2002] Tiwari, A. and Khanna, G. (2002). Series of abstractions for hybrid automata. In Tomlin, C. J. and Greenstreet, M. R., editors, *Hybrid Systems: Computation and Control*, LNCS 2289, pages 465–478. Springer Verlag.

[Tomlin et al., 2000] Tomlin, C. J., Lygeros, J., and Sastry, S. (July 2000). A game theoretic approach to controller design for hybrid systems. *Proceedings of the IEEE*, 88(7):949–970.

[Tomlin et al., 2001] Tomlin, C. J., Mitchell, I., and Ghosh, R. (2001). Safety verification of conflict resolution maneuvers. *IEEE Transactions on Intelligent Transportation Systems*, 2(2):110–120. June.

[Vidal et al., 2000] Vidal, R., Schaffert, S., Lygeros, J., and Sastry, S. S. (2000). Controlled invariance of discrete time systems. In Krogh, B. and Lynch, N., editors, *Hybrid Systems: Computation and Control*, LNCS 1790, pages 437–450. Springer Verlag.

[von Neumann and Morgenstern, 1947] von Neumann, J. and Morgenstern, O. (1947). *Theory of Games and Economic Behavior*. Princeton University Press.

[Yovine, 1997] Yovine, S. (1997). Kronos: A verification tool for real-time systems. *Software Tools for Technology Transfer*, 1:123–133.