Multiobjective Hybrid Controller Synthesis *

John Lygeros, Claire Tomlin and Shankar Sastry

Intelligent Machines and Robotics Laboratory
University of California, Berkeley
Berkeley, CA 94720
lygeros, clairet, sastry@eecs.berkeley.edu

Abstract. The problem of systematically synthesizing hybrid controllers that satisfy multiple requirements is considered. We present a technique, based on the principles of optimal control, for determining the class of least restrictive controllers that satisfy the most important requirement (which we refer to as safety). The system performance with respect to the lower priority requirement (which we refer to as efficiency) can then be optimized within this class. We motivate our approach by three examples, one purely discrete (the problem of reachability in finite automata) one hybrid (the steam boiler problem) and one primarily continuous (a flight vehicle management system).

1 Introduction

In this paper we concentrate on the problem of controlling hybrid systems, that is "steering" them using continuous and discrete inputs in an attempt to ensure that the system behavior satisfies certain requirements. For most real systems multiple requirements are imposed on the design. For example, for purely discrete systems the requirements usually considered are those of safety (encoded by requirements over the finite runs of the system) and liveness or fairness (encoded by requirements over the infinite runs), while for conventional control problems the requirements considered are usually safety (encoded by stability or constraints on the system trajectories) and efficiency (the requirement for small inputs or bounds on the speed of convergence for example). In such a multi-objective setting some of the requirements are usually assumed to be more important than others, either explicitly or implicitly. This priority is important from the point of view of controller synthesis, as one would like to ensure that high priority specifications are not violated in favor of low priority ones.

We present a methodology for designing hybrid controllers for hybrid systems in such a multi-objective setting. For simplicity we restrict our attention to two performance criteria and will use *safety* to refer to the high priority criterion and *efficiency* to refer to the low priority one. Using optimal control tools we attempt to classify the controllers that can be used to guarantee safety. Efficiency

* Research supported by the Army Research Office under grant DAAH 04-95-1-0588, the PATH program, Institute of Transportation Studies, University of California, Berkeley, under MOU-238, and by NASA under grant NAG 2-1039.

can then be optimized within this class of *least restrictive safe controls*. The resulting controller will typically be hybrid (even if the plant dynamics are purely continuous) as it involves switching between the safe and efficient controllers.

Our analysis is based on the hybrid system model introduced in [1], which is outlined in Section 2. The theoretical framework (presented in Section 3) is motivated by three examples. The first is purely discrete and involves the control of finite automata. The second is the well known steam boiler benchmark problem [2]. This is a hybrid problem in that a continuous process (the level of water in the boiler) is to be controlled using discrete controls (pumps being switched on and off). Finally, the third example is continuous and is motivated by the design of a flight vehicle management system.

2 Hybrid System Modeling

The basic entity of our models will be the *hybrid dynamical system* or *hybrid automaton* (the terms will be used interchangeably). Hybrid automata are convenient abstractions of systems with phased operation and they appear extensively in the literature [3, 4, 5, 6]. The model we consider will be similar to models used primarily in computer science; we take an input/output approach, along the lines of the reactive module paradigm [7]. For an overview of hybrid models from the dynamical systems point of view see [8].

We consider a finite collection of variables of two distinct kinds, discrete and continuous. A variable is called *discrete* if it takes values in a countable set and it is called *continuous* otherwise. We will assume no special algebraic structure for the values of the discrete variables. The only operations we will allow are assigning a value to a variable and checking whether the value of a variable and a member of the value set (or the values of two variables that take values in the same set) are equal. We assume that continuous variables take values in subsets of $\mathbb{R}^n$ for some value of n. The variables in our model will be split into three classes: *inputs*, *outputs* and *states*. We will denote the input space (set where the input variables take values) by $U = U_D \times U_C$, the output space by $Y = Y_D \times Y_C$ and the state space by $X = X_D \times X_C$. The subscripts D and C indicate whether the variable is discrete or continuous. To avoid unnecessary subscripts we denote an element of U by u, an element of Y by y and an element of X by (q, x). To simplify the notation we will omit X_D and q when there is only one discrete state and X_C and x when there are no continuous states.

Our model evolves in continuous time, so we will assume a set of times of interest of the form $T = [t_i, t_f] \subset \mathbb{R}$. The variables will evolve either continuously as a function of time or in instantaneous jumps. Therefore the evolution of the system will be over sets of the form:

$$\mathcal{T} = \{[\tau_0', \tau_1][\tau_1', \tau_2], \ldots [\tau_{n-1}', \tau_n]\} \tag{1}$$

with $\tau_i \in T$ for all i, $\tau_0' = t_i, \tau_n = t_f$ and $\tau_i = \tau_i' \leq \tau_{i+1}$ for all $i = 1, 2, \ldots, n-1$. The implication is that τ_i are the times at which discrete jumps of the state or input occur. We will use τ to denote an element of $\mathcal{T}$.

Definition 1 *A* **hybrid dynamical system**, H, *is a collection* (X, U, Y, I, f, E, h), *with* $X = X_D \times X_C$, $U = U_D \times U_C$, $Y = Y_D \times Y_C$, $I \subset X$, $f : X \times U \to TX_C$, $E \subset X \times U \times X$ *and* $h : X \times U \longrightarrow Y$. X_C, U_C, Y_C *are respectively open subsets of* $\mathbb{R}^n, \mathbb{R}^m, \mathbb{R}^p$, *for some finite values of* n, m, p *and* X_D, U_D, Y_D *are countable sets.*

Here TX_C represents the tangent space of the space X_C. We assume that f is time invariant[2] and satisfies the standard existence-uniqueness assumptions.

Definition 2 *A* **run** *of the hybrid dynamical system* H *over an interval* $T = [t_i, t_f]$ *consists of a collection* (τ, q, x, y, u) *with* $\tau \in \mathcal{T}$, $q : \tau \to X_D$, $x : \tau \to X_C$, $y : \tau \to Y$ *and* $u : \tau \to U$ *which satisfies the following properties:*

1. **Initial Condition**: $(q(\tau_0'), x(\tau_0')) \in I$.
2. **Discrete Evolution**: $(q(\tau_i), x(\tau_i), u(\tau_i), q(\tau_i'), x(\tau_i')) \in E$, *for all* i.
3. **Continuous Evolution**: *for all* i *with* $\tau_i' < \tau_{i+1}$ *and for all* $t \in [\tau_i', \tau_{i+1}]$:

$$\dot{x}(t) = f(q(t), x(t), u(t)), \quad q(t) = q(\tau_i'), \quad (q(t), x(t), u(t), q(t), x(t)) \in E$$

4. **Output Evolution**: *for all* $t \in \tau$, $y(t) = h(q(t), x(t), u(t))$.

It can be shown [1] that the definitions introduced here are rich enough to model continuous dynamical systems, finite state systems, rectangular automata, autonomous jumps, controlled jumps, etc. Note that the set E summarizes the information contained in the invariants, the transition guards and the transition reset relations that appear in the hybrid models of [3, 4, 5].

A number of operations can be defined on hybrid dynamical systems [1]. Here we restrict our attention to just one, called interconnection, which allows us to form new hybrid systems out of collections of existing ones. Let $\{H_i\}_{i=1}^N$ be a collection of hybrid automata, $H_i = \{X_i, U_i, Y_i, I_i, f_i, E_i, h_i\}$. We can write the inputs and outputs in vector form as $u_i = [u_{i,1} \quad \dots \quad u_{i,m_i}]^T \in U_i$ and $y_i = [y_{i,1} \quad \dots \quad y_{i,p_i}]^T \in Y_i$. Let:

$$\hat{U} = \{(1,1), (1,2), \dots, (1,m_1), (2,1), \dots, (2,m_2), \dots, (N,1), \dots, (N,m_N)\}$$
$$\hat{Y} = \{(1,1), (1,2), \dots, (1,p_1), (2,1), \dots, (2,p_2), \dots, (N,1), \dots, (N,p_N)\}$$

Definition 3 *An* **interconnection**, $\mathcal{I}$, *of a collection of hybrid automata is a partial map* $\mathcal{I} : \hat{U} \longrightarrow \hat{Y}$.

An interconnection of hybrid automata can be thought of as a pairing $(u_{i,j}, y_{k,l})$ of inputs and outputs. An interconnection is only a partial map (some inputs may be left free), need not be surjective (some outputs may be left free) and need not be injective (an output may be paired with more than one input). Let $Pre(\mathcal{I})$ be the subset of $\hat{U}$ for which the partial map $\mathcal{I}$ is defined and let Π_α denote the projection of a vector valued quantity to the element with index α.

[2] With some additional notation the same definitions can be given in terms of the flow of the vector field. The advantage would be that the definitions would directly extend to time varying vector fields, discrete time systems, etc.

Definition 4 *Given a collection of hybrid automata* $\{H_i\}_1^N$ *and an interconnection* $\mathcal{I}$*, the symbolic operation* **substitution**, *denoted by* $\rightsquigarrow$*, assigns to each input,* $u_{i,j}$*, a map on* $X_1 \times \ldots \times X_N \times U_1 \times \ldots \times U_N$*, according to:*

$$u_{i,j} \rightsquigarrow \begin{cases} u_{i,j} & \text{if } (i,j) \notin Pre(\mathcal{I}) \\ h_{\mathcal{I}(i,j)} : X_{\Pi_1(\mathcal{I}(i,j))} \times U_{\Pi_1(\mathcal{I}(i,j))} \rightarrow Y_{\Pi_1(\mathcal{I}(i,j))} & \text{if } (i,j) \in Pre(\mathcal{I}) \end{cases}$$

If for all $(i,j) \in Pre(\mathcal{I})$, $Y_{\mathcal{I}(i,j)} \subset U_{i,j}$*, operation* $\rightsquigarrow$ *can be repeatedly applied to the right hand side by appropriate map compositions. The construction terminates for each* $u_{i,j}$ *if the right hand side either contains* $u_{i,j}$ *itself or contains only* $u_{k,l} \notin Pre(\mathcal{I})$*. The resulting map will be denoted by* $(u_{i,j} \rightsquigarrow^*)$*.*

Because there are a finite number of inputs, the construction of $(u_{i,j} \rightsquigarrow^*)$ terminates in a finite number of steps. To ensure that an interconnection is well defined as an operation between hybrid automata we impose the following technical conditions:

Definition 5 *An interconnection,* $\mathcal{I}$*, of a collection of hybrid dynamical systems,* $\{H_i\}_{i=1}^N$*, is* **well posed** *if for all* $(i,j) \in Pre(\mathcal{I})$, $Y_{\mathcal{I}(i,j)} \subset U_{i,j}$ *and there are no algebraic loops, i.e. for all* $(i,j) \in Pre(\mathcal{I})$ *the map* $(u_{i,j} \rightsquigarrow^*)$ *does not involve* $u_{i,j}$*.*

Fact 1 *Every well posed interconnection,* $\mathcal{I}$*, of a collection of hybrid dynamical systems,* $\{H_i\}_{i=1}^N$*, defines a new hybrid dynamical system.*

3 Multiobjective Controller Design

We assume that the plant is modeled by a hybrid automaton of the form described in Section 2. We further divide the inputs into two classes, control inputs denoted by u and disturbances denoted by d. The input space is accordingly split into two subspaces, $(u,d) \in U \times D$. The interpretation is that the designer can exercise control over the inputs u but not over the disturbances. Let PC denote the space of piecewise continuous and PC^1 the space of piecewise differentiable functions of the reals and define the set of acceptable inputs by $\mathcal{U} = \{u \in PC | u(t) \in U \quad \forall t\}$ and the set of acceptable disturbances by $\mathcal{D} = \{d \in PC | d(t) \in D \ \forall t\}$.

The controller design should be such that the desired performance is achieved despite the actions of the disturbances. We are interested in a situation where more than one requirement is imposed on the system performance. Here for simplicity, we restrict our attention to the case of two requirements, which we refer to as *safety* and *efficiency*. We assume that these requirements can be encoded by a pair of cost functions, J_1 and J_2 respectively, on the runs of the hybrid automaton with $J_i : PC \times PC^1 \times \mathcal{U} \times \mathcal{D} \rightarrow \mathbb{R}$. The cost functions map a run of the automaton $(q(), x(), u(), d())$ to a real number. Here we restrict our attention to the case where each pair of inputs (u,d) generates a unique state trajectory for a given initial condition (q^0, x^0). We informally refer to hybrid

automata that possess this property as *deterministic hybrid automata.* In this case the cost function can be thought of as a map:

$$J_i : I \times \mathcal{U} \times \mathcal{D} \longrightarrow \mathbb{R} \tag{2}$$

To distinguish acceptable from unacceptable runs we can impose thresholds, C_1 and C_2, on the final costs. A run is acceptable if $J_i(q(), x(), u(), d()) \leq C_i$ for $i = 1, 2$. We also assume that the performance criteria come with an implicit ranking, safety being more important than efficiency.

In order to guarantee that the performance specifications are met despite the action of the disturbances we cast the design problem as a zero sum dynamic game. The two players in the game are the control u and the disturbance d and they compete over the cost functions J_1 and J_2. We seek to determine the best possible control action and the worst possible disturbance. If the performance specifications are met for this pair, then they can also be met for any other choice of the disturbance.

As higher priority is given to safety, the game for J_1 is solved first. If we assume that the game accepts a saddle solution, i.e. there exist input and disturbance trajectories, u_1^* and d_1^* such that:

$$J_1^*(q^0, x^0) = \max_{d \in \mathcal{D}} \min_{u \in \mathcal{U}} J_1(q^0, x^0, u, d) = \min_{u \in \mathcal{U}} \max_{d \in \mathcal{D}} J_1(q^0, x^0, u, d) = J_1(q^0, x^0, u_1^*, d_1^*)$$

then the set $V_1 = \{(q, x) \in X | J_1^*(q, x) \leq C_1\}$ contains all states for which there exists a control such that the objective on J_1 is satisfied for any allowable disturbance. If u_1^* is used as a control law it will guarantee that J_1 is minimized for the worst possible disturbance and, moreover, if the initial state is in V_1 it will also guarantee that the safety requirement is satisfied.

u_1^* however does not take into account the requirements on J_2. To include efficiency in the design let $\mathcal{U}_1(q^0, x^0) = \{u \in \mathcal{U} | J_1(q^0, x^0, u, d_1^*) \leq C_1\}$. Clearly:

$$\mathcal{U}_1(q^0, x^0) \begin{cases} = \emptyset & \text{for } (q^0, x^0) \notin V_1 \\ \neq \emptyset & \text{for } (q^0, x^0) \in V_1, \text{ as } u_1^* \in \mathcal{U}_1(q^0, x^0) \end{cases}$$

$\mathcal{U}_1$ can be thought of as a feedback map $\mathcal{U}_1 : X \to 2^U$, that maps to each state the subset of admissible controls which guarantees that the requirement on J_1 is satisfied; in other words, the least restrictive class of safe controls. Within this class we would now like to select the control that minimizes the cost function J_2. We again pose the problem as a zero sum dynamic game. Assume that a saddle solution (u_2^*, d_2^*) exists and let $J_2^*(q^0, x^0)$ be the corresponding cost. Then the set $V_2 = \{(q, x) \in X | J_2^*(q, x) \leq C_2\}$ contains the initial conditions for which there exists a control such that for any allowable disturbance the requirements on both J_1 and J_2 are satisfied. As the minimax problem can only be posed when $\mathcal{U}_1(q^0, x^0) \neq \emptyset$ we assume that $V_2 \subset V_1$. The control law u_2^* and the set V_2 are such that for all $(q^0, x^0) \in I \cap V_2$, for all $d \in \mathcal{D}$ and for $i = 1, 2$, $J_i(x^0, u_2^*, d) \leq C_i$.

Note that as $V_2 \subset V_1$ there may still be states for which the requirement for safety can be satisfied whereas that for efficiency can not. The controller can be

extended to these states using the simple switching scheme:

$$u^*(q,x) = \begin{cases} u_2^*(q,x) & \text{if } (q,x) \in V_2 \\ u_1^*(q,x) & \text{if } (q,x) \in X \setminus V_2 \end{cases} \tag{3}$$

This makes the operation of the controller hybrid, even in the case where the plant is purely continuous. Such an extension may be particularly useful when one is trying to design a fault tolerant controller. The occurrence of a fault significantly alters the dynamics and may lead to severe shrinking of the set V_2. In cases like these one would like to resort to a controller that guarantees safety, even if the requirements for efficiency are violated.

Two special cases of the above algorithm deserve explicit mention. The first is the case in which there is no disturbance. The algorithm then calls for the solution to a pair of optimal control problems (rather than games). The optimal solution for J_1 will produce a set of states and classify the least restrictive set of controllers for which the safety requirement can be satisfied. The optimal control problem for J_2 will then attempt to determine the best possible control in terms of efficiency within this class. Application of this special case will be demonstrated in Section 6 on the flight vehicle management system example. The second special case is one where there is no control. This is for example the case when a controller has already been designed and we are asked to verify its operation or determine the sets of initial conditions for which the specifications are satisfied. The verification problem also reduces to a pair of optimal control problems. For further discussion of this special case the reader is referred to [9].

4 Reachability in Finite Automata

Consider a standard, deterministic finite automaton $G = (Q, \Sigma, \delta, Q_0)$ where Q is a finite set of states, Σ a finite set of events, $\delta : Q \times \Sigma \to Q$ a transition relation and $Q_0 \subset Q$ a set of initial states. Let $L(G)$ denote the string of events (language) generated/accepted by G. Following [10] we assume that the set of events is partitioned into two disjoint subsets, $\Sigma = \Sigma_u \cup \Sigma_c$, where the events in Σ_c are controllable (in the sense that they can be disabled) while the events in Σ_u are uncontrollable. In this setting problems of safety are usually cast as questions of reachability: can the designer ensure that the automaton state will stay away from a "bad" set of states $Q_B \subset Q$. Efficiency typically corresponds to questions of fairness or liveness. The distinction made is that safety questions can be decided by reasoning over strings of finite length in $L(G)$ while questions of fairness require reasoning over infinite strings. For our example we will only consider how reachability questions can be addressed using the techniques of Section 3.

We first cast the finite automaton G into the modeling formalism of Section 2. In the set up [10], uncontrollable events are given "priority" over controllable ones, in the sense that they can always take place independent of the action of the controller. To capture this effect (and motivated by a discussion in [5]) we assume that the evolution of the system takes place in rounds where a controllable event

is followed by an uncontrollable one. To ensure that the resulting automaton will not block and that the priority of d over u is preserved we add two new states, q_G and q_B, and a new event, ϵ, and (re)define $X = Q \cup \{q_G, q_B\}$, $Q_B = Q_B \cup \{q_B\}$, $I = I \cup \{q_G\}$, $U = \Sigma_c \cup \{\epsilon\}$, $D = \Sigma_u \cup \{\epsilon\}$ and $Y = Q$. We then form a complete transition relation by defining:

$$
\begin{aligned}
E = \{(q_1,(d,u),q_2) \in X \times (D \times U) \times X \mid \\
q_1 \in Q \Rightarrow \begin{cases} q_2 = \delta(q_1,d) & \text{if } d \neq \epsilon, u = \epsilon, \delta(q_1,d)! \\ q_2 = q_G & \text{if } d \neq \epsilon, u = \epsilon, \delta(q_1,d) \not! \\ q_2 = \delta(q_1,u) & \text{if } d = \epsilon, u \neq \epsilon, \delta(q_1,u)! \\ q_2 = q_B & \text{if } d = \epsilon, u \neq \epsilon, \delta(q_1,u) \not! \\ q_2 = q_G & \text{if } d \neq \epsilon, u \neq \epsilon, \delta(q_1,d) \not! \\ q_2 = q_B & \text{if } d \neq \epsilon, u \neq \epsilon, \delta(q_1,d)! \\ q_2 = q_1 & \text{if } d = \epsilon, u = \epsilon \end{cases} \\
q_1 = q_B \Rightarrow q_2 = q_B \\
q_1 = q_G \Rightarrow q_2 = q_G\}
\end{aligned}
$$

Here $\delta(q,e)!$ is used to denote that the map δ is defined for the pair $(q,e) \in Q \times (\Sigma_c \cup \Sigma_u)$ and $\delta(q,e) \not!$ that it is not.

To cast the problem in the setting of Section 3 consider a discrete metric, m, on Q, defined by $m(q_1,q_2) = 0$ if $q_1 = q_2$ and $m(q_1,q_2) = 1$ if $q_1 \neq q_2$. It is easy to check that m satisfies the axioms of a metric. The metric can be extended to subsets of Q in the usual way (i.e. $m(Q_1,Q_2) = 1$ if $Q_1 \cap Q_2 \neq \emptyset$ and $m(Q_1,Q_2) = 0$ otherwise). Let $d = \{d_1, d_2, \ldots\} \in D^*$ denote a sequence in D and $u = \{u_1, u_2, \ldots\} \in U^*$ denote a sequence in U and define their interleaving as $(d,u) = \{(d_1,u_1), (d_2,u_2), \ldots\} \in (D \times U)^*$. As G is assumed to be deterministic, the transition structure defines a unique state trajectory $x = \{q_0, q_1, \ldots\} \in X^*$ for every $q_0 \in I$ and every $(d,u) \in (D \times U)^*$. The defining relation is $(q_i,(d_{i+1},u_{i+1}),q_{i+1}) \in E$. The metric can be used to assign a cost to this run by:

$$J(q_0,(d,u)) = -\min_{q \in x} m(q,Q_B)$$

The reachability problem can now be thought of as a game between u and d over the cost function J. Consider "feedback" maps $\hat{D} : X \to 2^D$ and $\hat{U} : X \to 2^U$. The following algorithm produces the least restrictive class of safe controls:
Step 0: Set $i = 1$. Define $Q'_B = Q_B$, $\hat{D}(q) = \{\epsilon\}$, $\hat{U}(q) = U$ for all $q \in Q'_B$.
Step i: Define $\text{NewQ}_B = \{q \in Q \setminus Q'_B \mid \exists d_i \in D, q' \in Q'_B \ni (q,(d_i,\epsilon),q') \in E\}$. If $\text{NewQ}_B \neq \emptyset$ increment i and define for all $q \in \text{NewQ}_B$ $\hat{D}(q) = \{d_i \in D \mid \exists q' \in Q'_B \ni (q,(d_i,\epsilon),q') \in E\}$ and $\hat{U}(q) = U$. Redefine $Q'_B = Q'_B \cup \text{NewQ}_B$ and return to step i. If $\text{NewQ}_B = \emptyset$, then for all $q \in X \setminus Q'_B$ define $\hat{D}(q) = D$ and $\hat{U}(q) = \{u_i \in U \mid (q,(\epsilon,u_i),q') \in E \Rightarrow q' \notin Q'_B\}$.

Lemma 1 *The algorithm terminates in at most $|X|$ steps. The system is safe if and only if $I \subset V_1 = Q \setminus Q'_B$.*

Corollary 1 *$\hat{U}$ defines the least restrictive class of controls that can guarantee that the system stays safe whenever it starts safe.*

Clearly, the least restrictive class of safe controls is already in feedback form. The above construction can also be used for reachability verification in finite automata, by letting $\Sigma_u = \Sigma, \Sigma_c = \emptyset$. $\hat{D}$ provides an *error trace* starting at any state $q_0 \in I \cap V_1$. In this special case the ϵ construction is not necessary.

5 The Steam Boiler

Our analysis of the steam boiler problem is based on the description of [2], which is simpler than the original specification of [11] in that the effect of faults on the system is not considered. The steam boiler consists of a tank containing water and a heating element that causes the water to boil and escape as steam. The water is replenished by two pumps which at time t pump water into the boiler at rates $\dot{p}_1(t)$ and $\dot{p}_2(t)$ respectively. At every time pump i can either be on ($\dot{p}_i(t) = P_i$) or off ($\dot{p}_i(t) = 0$). There is a delay T_{p_i} between the time pump i is ordered to switch on and the time $\dot{p}_i$ becomes P_i. There is no delay when the pumps are switched off. The requirement is that the water level remains between two values M_1 and M_2. We will use three hybrid automata to describe the system, one for the boiler and one for each of the pumps. The specification of [2] also includes a valve that, together with the pumps, can be used to bring the water level to a desirable initial condition before the heating element is turned on and the boiling starts. As the valve is only used to set the initial condition, its operation will be ignored in our safety calculations.

5.1 System Model

The boiler is modeled by a hybrid automaton, $H_B = \{X_B, U_B, Y_B, I_B, f_B, E_B, h_B\}$, with a single discrete state and two continuous states, the water level, w, and the rate at which steam escapes, r. We assume that both states are available for measurement. The system evolution is influenced by two discrete inputs, $\dot{p}_1$ and $\dot{p}_2$, and one continuous input, the derivative of the steam rate, d. The physical properties of the boiler impose the bounds $r(t) \in [0, W]$ and $d(t) \in [-U_2, U_1]$ for all t. Following [2] the dynamics are given by:

$$f_B(x_B, u_B) = \begin{bmatrix} \dot{p}_1 + \dot{p}_2 - r \\ d \end{bmatrix}; \quad E_B = \bigcup_{\substack{x_B \in X_B \\ u_B \in U_B}} (x_B, u_B, x_B); \quad h_B(x_B, u_B) = x_B$$

Note that the set E does not allow any discrete jumps of the state. It is assumed that W, U_1, U_2, P_1 and P_2 are positive constants.

Each pump can also be modeled by a hybrid automaton, $H_{p_i} = \{X_{p_i}, U_{p_i}, Y_{p_i}, I_{p_i}, f_{p_i}, E_{p_i}, h_{p_i}\}$, with two discrete states $q_i = 0$ and $q_i = P_i$ that reflect if the pump is on or off and one continuous state, T_i, that reflects the time that

has elapsed since the pump was ordered to switch on. The evolution of the state is affected by a discrete input that takes the value 0 if the pump is ordered to switch off and 1 if the pump is ordered to switch on. The dynamics are given by $f_{p_i}(x_{p_i}, u_i) = u_i$, $h_{p_i}(x_{p_i}, u_{p_i}) = x_{p_i}$ and

$$E_{p_i} = \left(\bigcup_{T_i \leq T_{p_i}} ((0,T_i),0,(0,0)) \right) \cup \left(\bigcup_{T_i \leq T_{p_i}} ((0,T_i),1,(0,T_i)) \right) \cup ((0,T_{p_i}),1,(P_i,T_{p_i})) \cup \left(\bigcup_{T_i \geq T_{p_i}} ((P_i,T_i),1,(P_i,T_i)) \right) \cup \left(\bigcup_{T_i \geq T_{p_i}} ((P_i,T_i),0,(0,0)) \right)$$

The combined system automaton can be obtained as the interconnection $\mathcal{I}(\dot{p}_i) = q_i$ for $i = 1, 2$ of H_B, H_{p_1} and H_{p_2}. The resulting automaton will have four discrete and four continuous states. If the outputs q_1 and q_2 of the pumps are suppressed, the definition of the interconnection automaton gives:

$$\begin{aligned} x &= ((q_1,q_2),[w \;\; r \;\; T_1 \;\; T_2]^T) \in X = \{0,P_1\} \times \{0,P_2\} \times \mathbb{R} \times [0,W] \times \mathbb{R}_+^2 \\ u &= ((u_1,u_2),d) \in U = \{0,1\}^2 \times [-U_2,U_1] \\ y &\in Y = \mathbb{R} \times [0,W] \times \mathbb{R}_+^2 \\ x^0 &\in I = I_B \times I_{p_1} \times I_{p_2} \end{aligned}$$

with dynamics:

$$f(x,u) = \begin{bmatrix} q_1 + q_2 - r \\ d \\ u_1 \\ u_2 \end{bmatrix}; \quad E = E_B \times E_{p_1} \times E_{p_2}; \quad h(x,u) = (w, r, T_1, T_2)$$

By a slight abuse of notation the value of the discrete states and inputs is used in the vector field definition. Without loss of generality assume that all runs of the automaton begin at $t = 0$. Our goal is to design a feedback controller for u_1 and u_2 that keeps the water level in the interval $w(t) \in [M_1, M_2]$ for all $t \geq 0$. This requirement can be encoded by two cost functions $J_1(x^0, u_1, u_2, d) = -\inf_{t \geq 0} w(t)$ and $J_1'(x^0, u_1, u_2, d) = \sup_{t \geq 0} w(t)$. For a given run (x^0, u) the requirements are satisfied if and only if $J_1(x^0, u) \leq -M_1$ and $J_1'(x^0, u) \leq M_2$.

5.2 Saddle Solutions, Safe States and Safe Controls

For any initial condition $x^0 = ((q_1^0, q_2^0), [w^0 \;\; r^0 \;\; T_1^0 \;\; T_2^0]^T)$, consider the following candidate saddle solutions:

$$u_i^*(t) = 1 \text{ for all } t; \quad d^*(t) = \begin{cases} U_1 & \text{if } t \leq \frac{W - r^0}{U_1} \\ 0 & \text{if } t > \frac{W - r^0}{U_1} \end{cases} \tag{4}$$

$$u_i'^*(t) = 0 \text{ for all } t; \quad d'^*(t) = \begin{cases} -U_2 & \text{if } t \leq \frac{r^0}{U_2} \\ 0 & \text{if } t > \frac{r^0}{U_2} \end{cases} \tag{5}$$

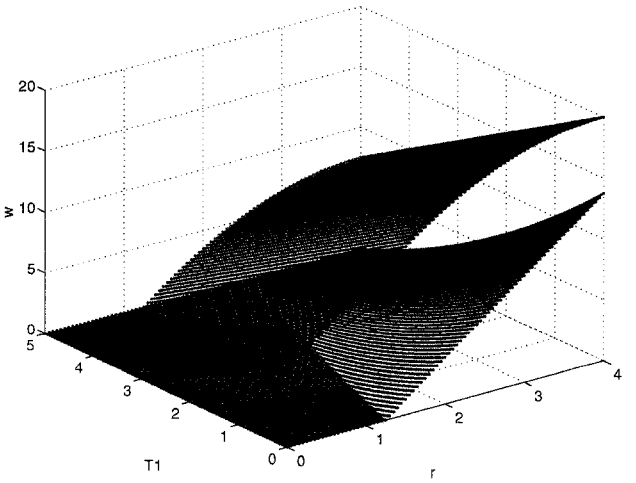

Fig. 1. Lower limit on w to avoid draining

Lemma 2 *(u_1^*, u_2^*, d^*) and $(u_1'^*, u_2'^*, d'^*)$ are saddle solutions for the game between (u_1, u_2) and d over cost functions J_1 and J_1'.*

It should be noted that the saddle solution is not unique in the case of J_1', as far as the u_i are concerned. In particular, any u_i such that $\dot{w}(t) \leq 0$ for all t will produce the same maximum water level (equal to the initial water level).

The saddle solution calculation for J_1 allows us to determine the water levels, w^0, that are safe with respect to M_1 as a function of r^0, T_1^0 and T_2^0. In particular, the boundary between safe and unsafe states can be thought of as a function $\hat{w} : [0, W] \times \mathbb{R}_+^2 \rightarrow \mathbb{R}$, which maps (r^0, T_1^0, T_2^0) to the minimum water level required for safety. The level sets of $\hat{w}$ for $T_2^0 = 0$ (pump 2 is initially off and for $T_2^0 \geq T_{p_2}$ (pump 2 is initially fully on) are shown in Figure 1. The safety boundary for any other value of T_2 will be a similar surface lying between the two surfaces of the figure. Safety ($w(t) \geq M_1$) can be maintained as long as the current value of the water level is on or above the corresponding surface. As expected the higher the value of T_2 the more states are safe (the surface moves down). The parameters used in the figure were $M_1 = 0$, $U_1 = 0.5$, $W = 4$, $P_1 = P_2 = 2.5$ and $T_{p_1} = T_{p_2} = 5$.

The states that are safe with respect to M_2 can be similarly determined. On the boundary between safe and unsafe states, $w^0 = M_2$ (the only situation where J' becomes safety critical) and:

$$r^0 = \hat{r}(T_1^0, T_2^0) = \begin{cases} 0 & \text{if } T_1^0 < T_{p_1} \text{ and } T_2^0 < T_{p_2} \\ P_1 & \text{if } T_1^0 \geq T_{p_1} \text{ and } T_2^0 < T_{p_2} \\ P_2 & \text{if } T_1^0 < T_{p_1} \text{ and } T_2^0 \geq T_{p_2} \\ P_1 + P_2 & \text{if } T_1^0 \geq T_{p_1} \text{ and } T_2^0 \geq T_{p_2} \end{cases}$$

The interpretation is that any initial condition such that either $w^0 < M_2$ or $w^0 = M_2$ and $r^0 \geq \hat{r}(T_1^0, T_2^0)$ is safe with respect to J'.

The calculation of the safe set also allows us to classify the controls that can keep the system safe provided it starts safe (w^0 and r^0 in the ranges discussed above). The class of safe controls is given in a state feedback form.

Lemma 3 *A control law for (u_1, u_2) is safe with respect to M_1 if and only if:*

$$u_1 \in \{0,1\} \;\; u_2 \in \{0,1\} \;\; \text{if } w > \hat{w}(r,0,0)$$
$$u_1 = 1 \;\; u_2 \in \{0,1\} \;\; \text{if } \hat{w}(r,0,0) \geq w > \hat{w}(r,T_1,0)$$
$$u_1 \in \{0,1\} \;\; u_2 = 1 \;\; \text{if } \hat{w}(r,0,0) \geq w > \hat{w}(r,0,T_2)$$
$$u_1 = 1 \;\; u_2 = 1 \;\; \text{if } w \leq \hat{w}(r,T_1,T_2)$$

Note that, as $\hat{w}$ is monotone in T_1 and T_2, the last condition is satisfied if and only if all other conditions fail. The two middle conditions may in fact overlap, therefore there is some nondeterminism in the choice of safe controls: some states may be safe with either one or the other pump on, but not neither. Similarly:

Lemma 4 *A control law for (u_1, u_2) is safe with respect to M_2 if and only if:*

$$u_1 \in \{0,1\} \;\; u_2 \in \{0,1\} \;\; \text{if } w < M_2 \text{ or } r > \hat{r}(T_1,T_2)$$
$$u_1 = 0 \;\; u_2 \in \{0,1\} \;\; \text{if } w \geq M_2 \text{ and } \hat{r}(T_1,T_2) \geq r > \hat{r}(0,T_2)$$
$$u_1 \in \{0,1\} \;\; u_2 = 0 \;\; \text{if } w \geq M_2 \text{ and } \hat{r}(T_1,T_2) \geq r > \hat{r}(T_1,0)$$
$$u_1 = 0 \;\; u_2 = 0 \;\; \text{if } w \geq M_2 \text{ and } r \leq \min\{\hat{r}(T_1,0), \hat{r}(0,T_2)\}$$

6 Flight Vehicle Management System

The flight vehicle management system (FVMS) example is based on the dynamic aircraft equations and the design specification of [12]. The equations model the speed and the flight path angle dynamics of a commercial aircraft in still air. The inputs to the equations are the thrust T, accessed through the engine throttle, and the pitch angle θ, accessed through the elevators, and the outputs are the speed V and the flight path angle γ. There are three primary modes of operation. In **Mode 1**, T is between its specified operating limits ($T_{min} < T < T_{max}$), both T and θ can be used as inputs and both V and γ can be controlled as outputs. In **Mode 2**, T saturates ($T = T_{min} \vee T_{max}$) and is no longer available as an input. The only input is θ, and the only controlled output is V. Finally, in **Mode 3**, T also saturates, the input is again θ but the controlled output is γ. Within Modes 2 and 3 there are two submodes depending on whether $T = T_{min}$ (idle thrust) or $T = T_{max}$ (maximum thrust).

Safety dictates that V and γ must remain within specified limits. For ease of presentation we simplify this *safety envelope*, S, of [12] to $S = \{(V,\gamma) | (V_{min} \leq V \leq V_{max}) \wedge (\gamma_{min} \leq \gamma \leq \gamma_{max})\}$, where $V_{min}, V_{max}, \gamma_{min}, \gamma_{max}$ are constants. We would like to design a control scheme which will cause the aircraft to reach

a target operating point $(V,\gamma)_{target}$ in S from any initial operating point in S. The resulting trajectory $(V(t),\gamma(t))$ must not exit the envelope at any time and must satisfy acceleration constraints imposed for passenger comfort.

6.1 System Model

The flight path angle dynamics of the aircraft can be summarized using two state variables, $x = [V\ \ \gamma]^T \in \mathbb{R}\times S^1$, where V (m/s) is the airspeed and γ (rad) is the flight path angle. The dynamics of the system are given by:

$$\dot{V} = -\frac{a_D V^2}{m} - g\sin\gamma + (\frac{1}{m})T \tag{6}$$

$$\dot{\gamma} = \frac{a_L V(1-c\gamma)}{m} - \frac{g\cos\gamma}{V} + (\frac{a_L V c}{m})\theta \tag{7}$$

where T (N) is the thrust, m (kg) is the mass of the aircraft, g (m/s^2) is gravitational acceleration, a_L and a_D are the lift and drag coefficients, c is a small positive constant and θ is the aircraft pitch angle. For these equations to be meaningful we need to assume that $X \subset (0,\infty)\times[-\pi/2,\pi/2]$. Clearly this will be the case for realistic aircraft. Physical considerations also impose constraints on the inputs: $U = [T_{min},T_{max}]\times[\theta_{min},\theta_{max}]$.

To guarantee safety we need to ensure that $x(t)\in S$ for all t. Let ∂S denote the boundary of S. The requirement that the state stays within S can be encoded by a cost function:

$$J_1(x^0,u) = -\min_{t\geq 0}(x(t)-\partial S) \tag{8}$$

by defining:

$$x(t)-\partial S = \begin{cases} \min_{y\in\partial S}\|x(t)-y\| & \text{if } x\in S \\ -\min_{y\in\partial S}\|x(t)-y\| & \text{if } x\notin S \end{cases}$$

Here $\|\cdot\|$ denotes the Euclidean metric on $\mathbb{R}^2$. To ensure that the state stays within S we impose the threshold $J_1(x^0,u)\leq 0$.

Cost functions involving the linear and angular accelerations can be used to encode the requirement for passenger comfort:

$$J_2(x^0,u) = \max_{t\geq 0}(\dot{V}(t)) \ \text{ and } \ J_2'(x^0,u) = \max_{t\geq 0}(V(t)\dot{\gamma}(t)) \tag{9}$$

The requirement that the linear and angular acceleration remain within the limits determined for comfortable travel are encoded by thresholds $J_2(x^0,u)\leq 0.1g$ and $J_2'(x^0,u)\leq 0.1g$.

6.2 Safe States and Least Restrictive Safe Controls

To find the controls that keep the state within the safety envelope we solve the optimal control problem $J_1^*(x^0) = \min_{u\in\mathcal{U}} J_1(x^0,u)$, $u^*(x^0) = \arg\min_{u\in\mathcal{U}} J_1(x^0,u)$.

Proposition 1 (Optimally Safe Controls) *The optimally safe input is:*

$$u^*(x^0) = \begin{cases} (T_{max}, \theta_{min}) \; \forall x^0 = (V,\gamma) \in S \cap \{(V,\gamma) : \frac{\gamma - \gamma_{min}}{\gamma_{max} - \gamma_{min}} > \frac{V - V_{min}}{V_{max} - V_{min}}\} \\ (T_{min}, \theta_{max}) \; \forall x^0 = (V,\gamma) \in S \cap \{(V,\gamma) : \frac{\gamma - \gamma_{min}}{\gamma_{max} - \gamma_{min}} < \frac{V - V_{min}}{V_{max} - V_{min}}\} \end{cases}$$

The optimal control calculation allows us to determine the set of safe states and the class of controls that renders this set safe. If $J_1^*(x^0) > 0$ there is no control that will keep the trajectory starting at $x^0 \in S$ within S. If, however, $J_1^*(x^0) \leq 0$ there exists at least one (and maybe multiple) such safe controls. Our goal is to determine $V_1 = \{x^0 \in S | J_1^*(x^0) \leq 0\}$ and $\mathcal{U}_1(x^0) = \{u \in \mathcal{U} | J_1(x^0, u) \leq 0\}$.

We start by analyzing the system equations (6, 7) along ∂S. Consider an arbitrary point $x^0 \in \partial S$. We can distinguish three cases. If $f(x^0, u)$ points "inside" S for all $u \in U$ then all controls are safe for the given point x^0, i.e. $\mathcal{U}_1(x^0) = U$. If $f(x^0, u)$ points "outside" S for some u, let $\hat{U} \subset U$ be the controls for which this happens. These inputs are unsafe for the point x^0, i.e. $\mathcal{U}_1(x^0) = U \setminus \hat{U}$. Finally, if $f(x^0, u)$ points outside S for all $u \in U$ then all controls are unsafe for the given point x^0, i.e. $\mathcal{U}_1(x^0) = \emptyset$. A special case of the second situation is one where $f(x^0, u)$ is tangent to ∂S for some u and points outside for all others. In this case, the set of controls that make $f(x^0, u)$ tangent to ∂S will be exactly u^*. This allows us to extend the safe set construction to the interior of S. The system equations are integrated backwards using u^* from that point to determine the boundary of the safe set in the interior of the envelope.

Consider, for example, the left hand edge of ∂S (Figure 2). The complete set of controls moves from being safe to unsafe as γ varies from γ_{min} to γ_{max}. We can determine which values of (T, θ) in U are unsafe along ∂S by determining where along this boundary the vector field is tangent to ∂S. We calculate this by setting $\dot{V} = 0$, $T = \hat{T}$ in equation (6). Solving for $\hat{T}$ leads to $\hat{T}(\gamma) = a_D V_{min}^2 + mg \sin\gamma$. $\hat{T}(\gamma)$ does not depend on θ, so the safe set of inputs are all (T, θ) for which $T(\gamma) \geq \hat{T}(\gamma)$. When γ is such that $\hat{T}(\gamma) = T_{min}$, the cone of vector fields points completely "inside" S; when γ is such that $\hat{T}(\gamma) = T_{max}$, the cone of vector fields points completely "outside" S, and T_{max} is the unique thrust input which keeps the system trajectory inside S. We define γ_1 and γ_2 to be such that $\hat{T}(\gamma_1) = T_{max}$ and $\hat{T}(\gamma_2) = T_{min}$ and calculate the boundary of the safe set of states on the interior of the envelope by integrating the system equations backward in time from (V_{min}, γ_1) using the constant control (T_{max}, θ_{min}). We denote this part of the safe set boundary in the interior of S as ∂V_1^1, and the point of intersection of ∂V_1^1 with the upper edge of ∂S as (V_1, γ_{max}).

A similar calculation along the upper edge of ∂S using equation (7) yields the values of θ for which the vector field becomes tangent to ∂S: $\hat{\theta}(V) = \frac{m}{a_L V c}\left(\frac{g \cos\gamma_{max}}{V} - \frac{a_L V(1 - c\gamma_{max})}{m}\right)$. The set of safe inputs in this case is all (T, θ) for which $\theta(V) \leq \hat{\theta}(V)$. When V is such that $\hat{\theta}(V) = \theta_{min}$, θ_{min} is the unique pitch angle input which keeps the system trajectory inside S. The calculations may be repeated for the right hand side and lower boundaries of S.

We are now in a position to describe explicitly the safe set of states V_1 and the safe controls $\mathcal{U}_1(x^0)$. Define the boundary of V_1 as

$$\begin{aligned}\partial V_1 = \{(V,\gamma)|\ &(V = V_{min}) \wedge (\gamma_{min} \leq \gamma \leq \gamma_1) \vee (V,\gamma) \in \partial V_1^1 \vee \\ &(\gamma = \gamma_{max}) \wedge (V_1 \leq V \leq V_{max}) \vee (V = V_{max}) \wedge (\gamma_4 \leq \gamma \leq \gamma_{max}) \vee \\ &(V,\gamma) \in \partial V_1^2 \vee (\gamma = \gamma_{min}) \wedge (V_{min} \leq V \leq V_2)\}\end{aligned}$$

V_1 is defined as the set enclosed by ∂V_1. This is depicted in Figure 2. $\mathcal{U}_1(x^0)$ is

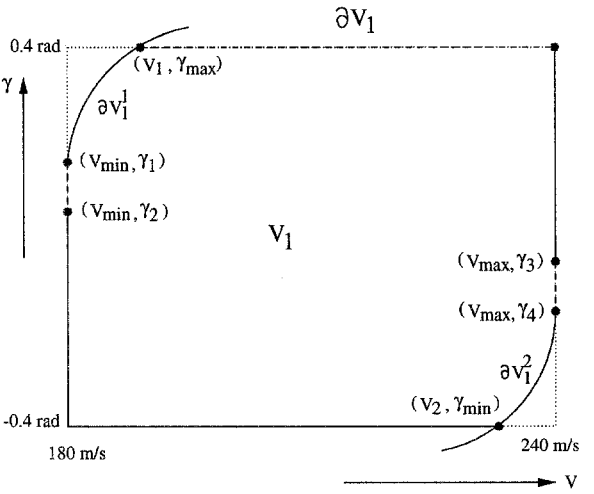

Fig. 2. The safe set of states, V_1, and its boundary ∂V_1

defined by the feedback map $G : S \to 2^U$:

$$\begin{aligned}G(V,\gamma) = \{\ &\emptyset, (V,\gamma) \in S \backslash V_1 \\ &\theta \leq \hat{\theta}(V) \wedge T_{min} \leq T \leq T_{max}, (V,\gamma) \in (\gamma = \gamma_{max}) \wedge (V_1 \leq V \leq V_{max}) \\ &\theta_{min} \leq \theta \leq \theta_{max} \wedge T \geq \hat{T}(\gamma), (V,\gamma) \in (V = V_{min}) \wedge (\gamma_2 \leq \gamma \leq \gamma_1) \\ &\theta_{min} \leq \theta \leq \theta_{max} \wedge T \leq \hat{T}'(\gamma), (V,\gamma) \in (V = V_{max}) \wedge (\gamma_4 \leq \gamma \leq \gamma_3) \\ &\theta = \theta_{min} \wedge T = T_{max}, (V,\gamma) \in \partial V_1^1 \\ &\theta = \theta_{max} \wedge T = T_{min}, (V,\gamma) \in \partial V_1^2 \\ &\theta_{min} \leq \theta \leq \theta_{max} \wedge T_{min} \leq T \leq T_{max}, \text{else}\}\end{aligned}$$

This map defines the *least restrictive* control scheme which satisfies the safety requirement and it determines the mode switching logic. On ∂V_1^1 and ∂V_1^2, the system must be in **Mode 2** or **Mode 3**. Anywhere else in V_1, any of the three modes is valid as long as the input constraints of G are satisfied. In the region $S \backslash V_1$ no control inputs are safe.

6.3 Additional Constraints for Passenger Comfort

Within the class of safe controls, a control scheme which addresses the passenger comfort (efficiency) requirement can be constructed. To do this, we solve the optimal control problem for J_2 and J_2', for $x^0 \in V_1$. From this calculation, we determine the set of "comfortable" states and controls:

$$V_2 = \{x^0 \in V_1 : J_2^*(x^0) \leq 0.1g \wedge J_2'^*(x^0) \leq 0.1g\} \tag{10}$$

$$\mathcal{U}_2(x^0) = \{u \in \mathcal{U}_1 : J_2(x^0, u) \leq 0.1g \wedge J_2'(x^0, u) \leq 0.1g\} \tag{11}$$

These sets may be easily calculated by substituting the bounds on the accelerations into equations (6, 7):

$$T \leq 0.1mg + a_D V^2 + mg \sin\gamma \text{ and } \theta \leq \frac{0.1mg}{a_L V^2 c} - \frac{1 - c\gamma}{c} + \frac{mg\cos\gamma}{a_L V^2 c} \tag{12}$$

These constraints provide upper bounds on the thrust and the pitch angle which may be applied at any point (V, γ) in V_2.

References

1. J. Lygeros, *Hierarchical Hybrid Control of Large Scale Systems.* PhD thesis, Department of Electrical Engineering, University of California, Berkeley, 1996.
2. T. A. Henzinger and H. Wong-Toi, "Using HyTech to synthesize control parameters for a steam boiler," in *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, LNCS, Springer Verlag, 1996.
3. R. Alur, C. Courcoubetis, T. A. Henzinger, and P. H. Ho, "Hybrid automaton: An algorithmic approach to the specification and verification of hybrid systems," in *Hybrid System*, no. 736 in LNCS, pp. 209–229, Springer Verlag, 1993.
4. A. Deshpande, *Control of Hybrid Systems.* PhD thesis, Department of Electrical Engineering, University of California, Berkeley, 1994.
5. A. Puri, *Theory of Hybrid Systems and Discrete Event Systems.* PhD thesis, Department of Electrical Engineering, University of California, Berkeley, 1995.
6. N. Lynch, R. Segala, F. Vaandrager, and H. Weinberg, "Hybrid I/O automata," in *Hybrid Systems III*, no. 1066 in LNCS, pp. 496–510, Springer Verlag, 1996.
7. R. Alur and T. A. Henzinger, *Computer-Aided Verification.* 1996. to appear.
8. M. S. Branicky, *Control of Hybrid Systems.* PhD thesis, Massacussets Institute of Technology, 1994.
9. J. Lygeros, D. N. Godbole, and S. Sastry, "Optimal control approach to multiagent, hierarchical system verification," in *IFAC World Congress*, 1996.
10. P. J. G. Ramadge and W. M. Wonham, "The control of discrete event dynamical systems," *Proceedings of the IEEE*, vol. Vol.77, no. 1, pp. 81–98, 1989.
11. J.-R. Abrial, "The steam-boiler control specification problem," in *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, LNCS, Springer Verlag, 1996.
12. C. S. Hynes and L. Sherry, "Synthesis from design requirements of a hybrid system for transport aircraft longitudinal control." (preprint), NASA Ames Research Center, 1996.