MULTI-OBJECTIVE HYBRID CONTROLLER SYNTHESIS

by

John Lygeros, Claire Tomlin, and Shankar Sastry

Memorandum No. UCB/ERL M97/59

15 August 1997

MULTI-OBJECTIVE HYBRID CONTROLLER SYNTHESIS

by

John Lygeros, Claire Tomlin, and Shankar Sastry

Memorandum No. UCB/ERL M97/59

15 August 1997

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Multi-objective Hybrid Controller Synthesis*

John Lygeros, Claire Tomlin and Shankar Sastry

Laboratory for Computer Science
Massachusetts Institute of Technology, NE43-374
545 Technology Square, Cambridge, MA 02139
lygeros@lcs.mit.edu

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

Abstract

The problem of systematically synthesizing hybrid controllers which satisfy multiple control objectives is considered. We present a technique, based on the principles of optimal control, for determining the class of least restrictive controllers that satisfies the most important objective (which we refer to as safety). The system performance with respect to the lower priority objective (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 benchmark problem), and one primarily continuous (a flight vehicle management system).

1 Introduction

Hybrid systems, that is systems that involve the interaction of discrete and continuous dynamics, have attracted the attention of researchers from a number of traditionally distinct fields. Computer scientists have approached the problem by extending techniques that have proved fruitful for discrete systems. The main problem that has been addressed in this setting has been verification, that is formally proving that a given system satisfies certain specifications. One approach to this problem comes from the area of *model checking* [1, 2, 3, 4], where the emphasis is on systems and properties that can be algorithmically verified. In problems where the model checking approach is applicable the verification process can be completely automated; a number of computational tools have been developed to take advantage of this property [5, 6, 7]. A different approach in the computer science literature has been to extend *deductive techniques* [8, 9, 10]. Here the emphasis has been on developing models [11, 12] that provide formal semantics for composition, abstraction, etc. and support proof techniques such as induction on the length of the system executions, invariant assertions and simulation relations. Even though automatic theorem provers can facilitate the process, most of the responsibility of the proof with this approach falls on the designer.

*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.

Researchers in the areas of dynamical systems and control have approached hybrid systems from a "continuous state space and continuous/discrete time" point of view. One effort has been in extending the standard modeling [13, 14, 15] and simulation techniques [16, 17, 18] to capture the interaction between the continuous and discrete components. Another has been in developing new analysis and controller design methodologies by extending existing methodologies such as Lyapunov's theorems [15, 19], discrete event control [20, 21, 22] and optimal control [23, 15, 24].

Our work falls under the last category. In recent years we have developed a methodology for designing controllers for large scale systems, making use of techniques from game theory and optimal control [24, 25]. We have successfully applied these techniques to a number of problems including automated highway systems [26], air traffic management [27] and benchmark examples such as the train gate controller [25]. The focus of our work so far has been on the hybrid phenomena that arise due to the interaction between the multiple agents (e.g. vehicles, aircraft, etc.) in a large scale system. In this paper we focus on the hybrid issues that arise because of the hybrid nature of the dynamics themselves (for example a continuous system being controlled by switches). We are primarily interested in control problems where multiple requirements are imposed on the design. This is usually the case for most realistic systems. For example, when dealing with completely discrete systems the requirements usually considered are those of safety (typically encoded by requirements over the finite runs of the system) and liveness or fairness (typically encoded by requirements over the infinite runs). For conventional control problems, on the other hand, 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).

In such a multi-objective setting some of the requirements are usually assumed to be more important than others, either explicitly or implicitly. The ranking of the requirements can be ignored if the goal is to verify the performance of a given hybrid system, as the objective in this case is to ensure that *all* the requirements are met. The priority is important from the point of view of controller synthesis however, as one would like to ensure that the higher priority specifications are not violated in favor of the low priority ones. This observation implicitly restricts the possible choices of the controllers that can be used to satisfy the lower priority specifications. Ideally one would like to be able to classify the controllers that guarantee the high priority specifications and attempt to optimize the system performance with respect to the lower priority ones within this class.

Here 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. We 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 determine the *largest controlled invariant safe set*, i.e. the largest set of states for which there exists a control such that the safety requirement can be satisfied. In the process we also determine the *class of least restrictive safe controls*, i.e. all the controls that can be used to satisfy the safety requirement for the safe states. The efficiency requirement can then be optimized within this class. 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 [28], which is outlined in Section 2. The design algorithm (presented in Section 3) is motivated by three examples. The first is purely discrete and involves the control of finite automata. Here the safety requirement is assumed to be equivalent to a question of reachability of a region of the state space. Efficiency on the other hand can be encoded by fairness constraints. We show how to determine the least restrictive class of safe controllers, within which one should look for the controllers that satisfy the fairness requirements.

The second example is the steam boiler benchmark problem [29]. Here the plant itself is hybrid, a continuous process (the level of water in the boiler) is to be controlled using discrete controls (pumps being switched on and off). The safety specification is again a question of reachability in the (continuous) state space; we would like the water level to stay within certain bounds. The efficiency requirement on the other hand could be to minimize the number of times the pumps are switched on and off or equalize the "on" time among pumps. Here we only address the question of safety.

Finally, the third example is primarily continuous and is motivated by the design of a flight vehicle management system. We consider the speed and flight path angle dynamics of a passenger aircraft. The plant is two dimensional, highly nonlinear and is influenced by two continuous inputs, the thrust (controlled by the aircraft engine) and the pitch angle (controlled through the elevators). Switching arises from the saturation of the thrust input, which imposes three modes of operation for the aircraft: one where both its velocity and flight path angle are controlled, one where only the velocity is controlled and one where only the flight path angle is controlled. Safety is again encoded by a reachability requirement: the velocity and flight path angle should stay within specified limits (imposed by the engine limitations, wing stall conditions, etc.). We classify the controllers that guarantee safety and establish the mode switching required to implement them. Within this class an efficiency requirement (the magnitude of the linear and angular accelerations) is then optimized.

2 Hybrid System Modeling

2.1 Hybrid Dynamical Systems

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 in various forms ([2, 3, 12]). The model we consider will be similar to models used primarily in model checking (in particular the ones in [30] and [31]). We will take a more input/output approach, along the lines of the reactive module paradigm [32]. For an overview of hybrid models from a dynamical systems perspective see [15].

A hybrid automaton is a dynamical system which determines the evolution and interaction of a finite collection of variables. We consider two distinct kinds of variables:

Definition 1 *A variable is called* **discrete** *if it takes values in a countable set and it is called* **continuous** *if it takes values on a smooth manifold.*

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. For simplicity we will assume here 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** (external), **outputs** (interface) and **state** (private)[1]. 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

[1]The terms in bold and in brackets can be used interchangeably, though we stick to the terms in bold most of the time. The former are more common in control theory while the latter are more common in computer science.

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 that set of times of interest is 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 $[\tau_0', \tau_1][\tau_1', \tau_2] \ldots [\tau_{n-1}', \tau_n]$ 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 where discrete jumps of the state or input occur. We will use $\mathcal{T}$ to denote the set of all such "super-dense" time trajectories and τ to denote an element of $\mathcal{T}$.

Definition 2 *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* $h : X \times U \to Y$, *where* 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 X_C. We assume that f is time invariant[2] and satisfies the standard assumptions for existence and uniqueness of solutions to ordinary differential equation.

Definition 3 *A* **run** *of the hybrid dynamical system H over an interval $T = [t_i, t_f]$ is 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 $u : \tau \to U$ satisfying:*

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

$$\dot{x}(t) = f(q(t), x(t), u(t)) \tag{1}$$
$$q(t) = q(\tau_i') \tag{2}$$
$$(q(t), x(t), u(t), q(t), x(t)) \in E \tag{3}$$

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

It can be shown [28] that the definitions introduced here are rich enough to allow us to model regular dynamical systems, discrete events, autonomous jumps, controlled jumps, etc.

2.2 Graphical Representation

If X_D is a finite set it is very convenient to represent the hybrid automaton by a directed graph. We can associate a graph to a given hybrid automaton H using the following construction:

Nodes: the number of nodes in the graph is equal $|X_D|$. The nodes are indexed by the corresponding discrete state value, $q \in X_D$.

[2]With some additional notation the definitions can be given in terms of the flow of the vector field. The advantage of this is that they would directly extend to other cases, such as time varying vector fields and discrete time systems.

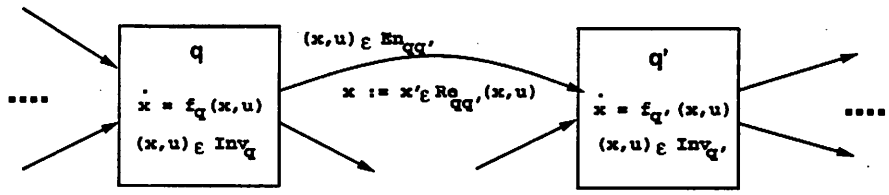

Figure 1: Hybrid automaton graph nodes

Continuous Evolution: to each node, q, we associate a vector field, f_q, defined in terms of f by:

$$\begin{aligned} f_q : X_C \times U &\longrightarrow TX_C \\ (x,u) &\longmapsto f(q,x,u) \end{aligned} \tag{4}$$

The implication is that while in the node q the continuous state evolves according to f_q.

Node Invariants: To each node, q, we associate an invariant:

$$\mathrm{Inv}_q = \bigcup\{(x,u) | x \in X_C, u \in U, (q,x,u,q,x) \in E\} \subset X_C \times U \tag{5}$$

The interpretation is that the system can remain in node q if and only if $(x,u) \in \mathrm{Inv}_q$.

Transition Guards: To the transition from node q to node q' we associate a guard:

$$\mathrm{En}_{qq'} = \bigcup\{(x,u) | x, x' \in X_C, u \in U, (q,x,u,q',x') \in E\} \subset X_C \times U \tag{6}$$

The interpretation is that the transition can take place if and only if $(x,u) \in \mathrm{En}_{qq'}$.

Transition Reset: To the transition from node q to node q' we associate a set valued map:

$$\mathrm{Res}_{qq'}(x,u) = \bigcup\{x' | x' \in X_C, (q,x,u,q',x') \in E\} \subset X_C \tag{7}$$

The interpretation is that if the transition takes place from (x,u) then after the transition the state can find itself in any (q',x') with $x' \in \mathrm{Res}_{qq'}(x,u)$.

The above construction allows us to represent a hybrid automaton graphically as shown in Figure 1. Note that all the information about the discrete transitions (the node invariants, the enabling conditions and the reset relations) is encoded by the set E of the hybrid automaton. Note also that there is no requirement that $q \neq q'$, i.e. loops to the same node are allowed.

2.3 Operations on Hybrid Dynamical Systems

A number of operations can be defined on hybrid dynamical systems [28]. Here we restrict our attention to just one, called interconnection. It 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:

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

Definition 4 *An* **interconnection**, $\mathcal{I}$, *of a collection of hybrid automata,* $\{H_i\}$, *is a partial map* $\mathcal{I} : \hat{U} \to \hat{Y}$.

An interconnection of hybrid automata can be thought of as a pairing $(u_{i,j}, y_{k,l})$ of inputs and outputs. It is only a partial map (i.e. some inputs may be left free), need not be surjective (i.e. some outputs may be left free) and need not be injective (i.e. 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. Also let Π_α denote the projection of a vector valued quantity to the element with index α.

Definition 5 *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:*

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

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 6 *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 the map $(u_{i,j} \rightsquigarrow^*)$ does not involve $u_{i,j}$.*

These requirements imply that $(u_{i,j} \rightsquigarrow^*)$ is well defined as a map between the following spaces:

$$(u_{i,j} \rightsquigarrow^*) : X_1 \times \ldots \times X_N \times \Pi_{\hat{U} \setminus Pre(\mathcal{I})}(U_1 \times \ldots \times U_N) \longrightarrow \Pi_{i,j}(U_1 \times \ldots \times U_N)$$

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.*

Proof: Let $H = \{X, U, Y, I, f, E, h\}$ denote the interconnection automaton, defined by $X = X_1 \times \ldots \times X_N$, $U = \Pi_{\hat{U} \setminus Pre(\mathcal{I})}(U_1 \times \ldots \times U_N)$, $Y = Y_1 \times \ldots \times Y_N$, $I = I_1 \times \ldots \times I_N$, $f = [f_i \circ (u_i \rightsquigarrow^*)]_{i=1}^N$, $E \subset X \times U \times X$ with $e = (q, x, u, q', x') \in E$ if and only if $(q_i, x_i, (u_i \rightsquigarrow^*)(q, x, u), q_i', x_i') \in E_i$ for all $i = 1, \ldots, N$ and $h = [h_i \circ (u_i \rightsquigarrow^*)]_{i=1}^N$. ∎

The expression $(u_i \rightsquigarrow^*)$ denotes the map generated by applying $\rightsquigarrow^*$ to the elements $u_{i,1}, \ldots, u_{i,m_i}$. The terms in square brackets have the obvious interpretation as vectors. The symbol $\circ$ denotes composition of maps.

3 Multi-objective Controller Design

3.1 Design Framework

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 but not over the disturbances. This implies that the controller design should be such that the desired performance is achieved despite the actions of 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 $\mathcal{U} = \{u \in PC | u(t) \in U \ \forall t\}$ and the set of acceptable disturbances by $\mathcal{D} = \{d \in PC | d(t) \in D \ \forall t\}$.

For simplicity, we restrict our attention to the case where two requirements are imposed on the system performance; we refer to them 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:

$$J_i : PC \times PC^1 \times \mathcal{U} \times \mathcal{D} \longrightarrow \mathbb{R} \tag{8}$$

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

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*[3]. In this case the cost function can be thought of as a map:

$$J_i : I \times \mathcal{U} \times \mathcal{D} \longrightarrow \mathbb{R} \tag{9}$$

3.2 Controller Synthesis

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 can be met for this pair then they can also be met for any disturbance.

As higher priority is given to safety, the game for J_1 is solved first. Assume that the game admits 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:

$$V_1 = \{(q, x) \in X | J_1^*(q, x) \leq C_1\} \tag{10}$$

contains all states for which there exists a control such that the objective on J_1 is satisfied for the worst possible allowable disturbance (and hence for any allowable disturbance). If u_1^* is used as the control it will guarantee that J_1 is minimized for the worst possible disturbance; moreover, if the initial state is in V_1 it will also guarantee that the performance requirement on J_1 is satisfied.

However, u_1^* does not take into account the requirements on J_2. To introduce efficiency let:

$$\mathcal{U}_1(q^0, x^0) = \{u \in \mathcal{U} | \max_{d \in \mathcal{D}} J_1(q^0, x^0, u, d) \leq C_1\} \tag{11}$$

[3]Nondeterministic hybrid automata, a generalization of nondeterministic finite automata, are not covered here.

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 $\mathcal{U}_1 : X \to 2^{\mathcal{U}}$, that maps to each state the subset of admissible controls which guarantee that the requirement on J_1 is satisfied; in other words, the least restrictive class of safe controls. Within this class we would like to select the control that minimizes the cost function J_2. We again pose the problem as a two person zero sum game. Assume that a saddle solution exists, i.e. there exist u_2^* and d_2^* such that:

$$J_2^*(q^0,x^0)=\max_{d\in\mathcal{D}}\min_{u\in\mathcal{U}_1(q^0,x^0)}J_2(q^0,x^0,u,d)=\min_{u\in\mathcal{U}_1(q^0,x^0)}\max_{d\in\mathcal{D}}J_2(q^0,x^0,u,d)=J_2(q^0,x^0,u_2^*,d_2^*)$$

Then the set:

$$V_2=\{(q,x)\in X|J_2^*(q,x)\leq C_2\} \tag{12}$$

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 min-max problem can only be posed when $\mathcal{U}_1(q^0,x^0)\neq\emptyset$ we have that $V_2\subset V_1$. The control law u_2^* and the set V_2 are such that for all $(q^0,x^0)\in I\cap V_2$ and for all $d\in\mathcal{D}$, $J_i(q^0,x^0,u_2^*,d)\leq C_i$ for $i=1,2$.

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. If the saddle solutions are in feedback form, the controller can be extended to these states using the simple switching scheme:

$$u^*(q,x)=\begin{cases} u_2^*(q,x) & (q,x)\in V_2 \\ u_1^*(q,x) & (q,x)\in X\setminus V_2\end{cases} \tag{13}$$

This will make the operation of the controller hybrid, even when the plant is purely continuous. Such an extension may be particularly useful when trying to design fault tolerant controllers. The occurrence of a fault significantly alters the system dynamics and may lead to severe shrinking of the set V_2. One would like to be able to resort to a controller that guarantees safety, even if the requirements for efficiency have to be violated.

3.3 Technical Issues & Special Cases

The above algorithm may run into technical difficulties, as there is no guarantee that the dynamic games will have a saddle solution, there is no straightforward way of computing $\mathcal{U}_1(q^0,x^0)$ and there is no guarantee that the sets V_1 (and consequently $\mathcal{U}_1(q^0,x^0)$) and V_2 will be non-empty. Fortunately, in the examples considered here (as well as the ones [27, 26]) a solution can be obtained analytically, or using simple numerical calculations. In general, sophisticated optimal control tools [33] can make the solution of more general problems feasible, at least numerically.

Two special cases of the above algorithm deserve explicit mention. The first is the case where 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 in which there is no control. This is for example the case in which 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* reduces to a pair of optimal control problems. For further discussion of this special case the reader is referred to [34].

4 Reachability in Finite Automata

4.1 Problem Description

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 [35] we assume that the set of events is partitioned into two subsets, $\Sigma = \Sigma_u \cup \Sigma_c$, where the events in Σ_c are controllable (in the sense that they can be disabled at will) while the events in Σ_u are uncontrollable[4].

In this setting problems of safety are usually cast as questions of reachability: can the designer ensure that the automaton state will stay in a "good" subset $Q_G \subset Q$ of the state space (or equivalently that it will not enter a "bad" subset $Q_B = Q \setminus Q_G$). Efficiency typically corresponds to questions of fairness or liveness. The distinction is that safety questions can be answered by reasoning over strings of finite length in $L(G)$ while questions of fairness require reasoning over infinite strings. Here we will show how reachability questions can be addressed using the techniques of Section 3.

4.2 System Model

We first cast the finite automaton G into the modeling formalism of Section 2. As there are no continuous variables, X_C, U_C, Y_C and f will be omitted. In the set up of [35], 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 [31]) we assume that the evolution of the system takes place in rounds where a controllable event is followed by an uncontrollable one. More specifically, we capture the evolution of G by a hybrid automaton $H = \{X, D \times U, Y, I, E, h\}$ with $X = Q$, $U = \Sigma_c \cup \{\epsilon\}$, $D = \Sigma_u \cup \{\epsilon\}$, $Y = Q$, $I = Q_0$, $h(q, (d, u)) = q$,

$$E = \left\{ (q_1, (d, u), q_2) \in X \times (D \times U) \times X \;\middle|\; \begin{array}{ll} q_2 = \delta(q_1, d) & \text{if } u = \epsilon \\ q_2 = \delta(q_1, u) & \text{if } d = \epsilon \end{array} \right\}$$

where we assume that $\delta(q, (\epsilon, \epsilon)) = q$. Note that H can "block" some inputs, i.e. there exist states q that for some inputs (d, u), the next state is not defined[5]. This can occur if $d = \epsilon$ and G blocks u, if $u = \epsilon$ and G blocks d or if both $d \neq \epsilon$ and $u \neq \epsilon$. Let $L(H)$ denote the strings of inputs accepted by H.

Fact 2 *The non-blocking traces in $L(G)$ and $L(H)$ are in one to one correspondence, modulo (ϵ, ϵ) transitions.*

[4] Controllable events represent actions that the designer can choose (start processing, send message, etc.) whereas the uncontrollable events are spontaneous actions of the plant (machine breakdown, message lost, etc.).

[5] This phenomenon is common in finite automata whenever the transition relation is a partial map.

Proof: Consider the map T_{GH} that maps a run of G to a run of H by replacing every $d \in \Sigma_u$ by (d, ϵ) and every $u \in \Sigma_c$ by (ϵ, u) and the map T_{HG} that maps a run of H to a run of G by dropping all the ϵ. By definition the maps are injective and surjective over the set of nonblocking traces and $T_{GH} \circ T_{HG}$ and $T_{HG} \circ T_{GH}$ are both the identity. ∎

Fact 2 indicates that the proposed construction does not affect the language accepted by the automaton. Therefore, controlling the automaton H, (where the system evolution is assumed to take place in rounds of uncontrollable and controllable events) is effectively the same as controlling the original automaton G (where this restriction is not imposed). Clearly the dynamics of G can be captured by a much simpler H, one with no ϵ. We introduce the extra notation to help us preserve the "priority" of d over u. We can interpret the transition structure E as saying that $u \neq \epsilon$ is allowed only if $d = \epsilon$.

4.3 Reachability

Assume that a set $Q_B \subset Q$ is given and that the controller is asked to render the states in Q_B unreachable, despite the action of events in Σ_u. To ensure that the automaton H will not block for the saddle solutions (soon to be calculated) and that priority is given to d over u we add two new states, q_G and q_B, and redefine $X = Q \cup \{q_G, q_B\}$, $Q_B = Q_B \cup \{q_B\}$ and $I = I \cup \{q_G\}$. We then complete the transition relation by redefining:

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

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, $m : Q \times Q \to \mathbb{R}$, defined by $m(q_1, q_2) = 0$ if $q_1 = q_2$ and 1 if $q_1 \neq q_2$. It is easy to check that m satisfies the axioms of a metric. The metric induces a map on pairs of subsets of Q by:

$$
\begin{aligned}
m : 2^Q \times 2^Q &\longrightarrow R \\
(Q_1, Q_2) &\longmapsto \min_{(q_1,q_2) \in Q_1 \times Q_2} m(q_1, q_2)
\end{aligned}
$$

In other words, $m(Q_1, Q_2) = 0$ if $Q_1 \cap Q_2 \neq \emptyset$ and $m(Q_1, Q_2) = 1$ if $Q_1 \cap Q_2 = \emptyset$. By abuse of notation we use m to denote both the metric and the map and $m(q, Q_1)$ to denote $m(\{q\}, Q_1)$.

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 above 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 relationship 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:

$$\begin{aligned} J_1 : I \times (D \times U)^* &\longrightarrow R \\ (q_0, (d, u)) &\longmapsto -\min_{q \in x} m(q, Q_B) \end{aligned}$$

In other words, $J_1(q_0, (d, u)) = 0$ if the run x enters the set Q_B and -1 otherwise. The reachability problem can now be thought of as a game between u and d over the cost function J_1. Consider "feedback" maps $\hat{D} : X \to 2^D$ and $\hat{U} : X \to 2^U$. The following algorithm allows us to determine the least restrictive class of safe controls:

Algorithm: Calculation of safe sets and safe controls

Step 0: Set $i = 1$ and define $Q'_B = Q_B$, $\hat{D}(q) = \{\epsilon\}$ and $\hat{U}(q) = U$ for all $q \in Q'_B$.

Step i: Define:

$$\text{NewQ}_B = \{q \in Q \setminus Q'_B | \exists d_i \in D, q' \in Q'_B \text{ with } (q, (d_i, \epsilon), q') \in E\}$$

If $\text{NewQ}_B \neq \emptyset$ increment i and for all $q \in \text{NewQ}_B$ define $\hat{U}(q) = U$ and

$$\hat{D}(q) = \{d_i \in D | \exists q' \in Q'_B \text{ with } (q, (d_i, \epsilon), q') \in E\}$$

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

$$\hat{U}(q) = \{u_i \in U | (q, (\epsilon, u_i), q') \in E \Rightarrow q' \notin Q'_B\}$$

Define the safe set as $V_1 = Q \setminus Q'_B$. ∎

Fact 3 *The algorithm terminates in at most $|Q|$ steps.*

Proof: The set Q'_B constructed by the algorithm is monotone nondecreasing in the number of steps, starts off with $|Q'_B| \geq |\{q_B\}| = 1$ and is upper bounded by $|X| - 1$ (q_G can never be in Q'_B). The claim follows as $|Q| = |X| - 2$ and the algorithm terminates once Q'_B stops increasing. ∎

Lemma 1 *The system is safe if and only if $I \subset V_1$.*

Proof: First assume $I \subset V_1$. For every $q_0 \in I$ and for every $d^* \in D^*$ choose $u^* \in U^*$ such that for the run $x = \{q_0, q_1, \ldots\}$ generated by (d^*, u^*), $u_i^* \in \hat{U}(q_{i-1})$ if $d_i^* = \epsilon$ and $u_i^* = \epsilon$ if $d_i^* \neq \epsilon$ for all i. Then, by construction of $\hat{U}$, $q_i \in V_1$ for all i, $J_1(q_0, (d^*, u^*)) = -1$ and the system is safe. Note that, as $q_i \in V_1$ it trivially follows that $d_i^* \in \hat{D}(q_{i-1})$ for all i by construction of $\hat{D}$.

Now assume $I \not\subset V_1$. Choose $q_0 \in I \cap V_1^c$ and for every $u^* \in U^*$ choose $d^* \in D^*$ such that for the run $x = \{q_0, q_1, \ldots\}$ generated by (d^*, u^*), $d_i^* \in \hat{D}(q_{i-1})$ for all i. Then, by construction of $\hat{D}$, there exists $i \geq 1$ such that $q_i \in Q_B$. In other words, $J_1(q_0, (d^*, u^*)) = 0$ and the system is unsafe. As above note that by construction of $\hat{U}$ and by the choice of d^*, $u_i^* \in \hat{U}(q_{i-1})$ for all i. ∎

The above discussion reveals that $J_1^*(q_0) = -1$ if $q_0 \in V_1$ and $J_1^*(q_0) = 0$ otherwise. Therefore, as J_1 can take on only two values, any pair (u^*, d^*) that satisfies: $d_i^* \in \hat{D}(q_{i-1})$ and $u_i^* \in \hat{U}(q_{i-1})$ for the corresponding run $x = \{q_0, q_1, \ldots\}$, is a min-max solution. Moreover:

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 in feedback form. The above construction can also be used for standard reachability verification in finite automata, by letting $\Sigma_u = \Sigma, \Sigma_c = \emptyset$. If Q_B is reachable, $\hat{D}$ provides an *error trace* starting at any state $q_0 \in I \cap V_1^c$. In this special case the ϵ construction is not necessary. This approach can in principle also be used to address more general language inclusion problems for regular languages. However, it is likely to be prohibitively expensive computationally, as it would require construction of the automata that accept the languages and complementation of one of them.

5 The Steam Boiler

5.1 Problem Description

Our analysis of the steam boiler problem is based on the description given in [36], which is simpler than the original specification of [29] 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, t, 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 $\dot{p}_i$ switches to P_i. There is no delay when the pumps are switched off. The requirement is that the pumps are switched on and off so that the water level remains between two values M_1 and M_2.

Here we will use three hybrid automata to describe the system, one for the boiler and one for each of the pumps. The specification of [36] 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.

5.2 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 (suppressed to simplify the notation) 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, i.e. $Y_B = X_B$ and $y_B = h_B(x_B, u_B) = x_B$. 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 bounds on the states and inputs: $x_B = [w \quad r]^T \in X_B = \mathbb{R} \times [0, W]$ and $u_B = [\dot{p}_1 \quad \dot{p}_2 \quad d] \in U_B = \{0, P_1\} \times \{0, P_2\} \times [-U_2, U_1]$, where W, U_1, U_2, P_1 and P_2 are positive constants. Following [36] the dynamics are given by:

$$f_B(x_B, u_B) = \begin{bmatrix} \dot{p}_1 + \dot{p}_2 - r \\ d \end{bmatrix}$$

$$E_B = \bigcup_{\substack{x_B \in X_B \\ u_B \in U_B}} (x_B, u_B, x_B)$$

Note that the set E does not allow any discrete jumps of the state.

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 commanded to switch on, hence $x_{p_i} = (q_i, T_i) \in X_{p_i} = \{0, P_i\} \times \mathbb{R}_+$. The evolution of the state is affected by a discrete input, $u_{p_i} = u_i \in U_{p_i} = \{0, 1\}$ that takes the value 0 if the pump is commanded to switch off and 1 if the pump is commanded to switch on. We assume that the pump state is available for measurement, i.e. $h_{p_i}(x_{p_i}, u_{p_i}) = x_{p_i}$. For consistency we restrict the pump initial conditions to:

$$x^0_{p_i} \in I_{p_i} = \left(\bigcup_{T_i \le T_{p_i}} (0, T_i) \right) \cup \left(\bigcup_{T_i \ge T_{p_i}} (P_i, T_i) \right)$$

The dynamics are given by $\dot{T}_i = f_{p_i}(x_{p_i}, u_i) = u_i$ and:

$$\begin{aligned} E_{p_i} \;=\; & ((0, T_{p_i}), 1, (P_i, T_{p_i})) \cup \\ & \left(\bigcup_{T_i \le T_{p_i}} ((0, T_i), 0, (0, 0)) \right) \cup \left(\bigcup_{T_i \le T_{p_i}} ((0, T_i), 1, (0, T_i)) \right) \cup \\ & \left(\bigcup_{T_i \ge T_{p_i}} ((P_i, T_i), 1, (P_i, T_i)) \right) \cup \left(\bigcup_{T_i \ge T_{p_i}} ((P_i, T_i), 0, (0, 0)) \right) \end{aligned}$$

The combined system automaton can be obtained as an interconnection of H_B, H_{p_1} and H_{p_2}. The interconnection map is $\mathcal{I}(\dot{p}_i) = q_i$ for $i = 1, 2$. One can easily see that:

Fact 4 *$\mathcal{I}$ is a well posed interconnection.*

The resulting automaton will have two discrete and four continuous states. We will use $x = ((q_1, q_2), [w \;\; r \;\; T_1 \;\; T_2]^T)$ to denote the overall state.

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 $t \ge 0$. This requirement can be encoded by two cost functions:

$$J_1(x^0, u_1, u_2, d) = -\inf_{t \ge 0} w(t) \quad \text{and} \quad J_1'(x^0, u_1, u_2, d) = \sup_{t \ge 0} w(t) \tag{14}$$

For a given run the requirements are satisfied if and only if $J_1 \le -M_1$ and $J_1' \le M_2$.

5.3 Saddle Solutions and Set of Safe States

We will treat the evolution of the system as a game between the inputs u_1, u_2 and the disturbance d. 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 solution for the game with cost J_1:

$$u_i^*(t) = 1 \text{ for all } t, \quad \text{and} \quad d^*(t) = \begin{cases} U_1 & \text{if } t \le \frac{W - r^0}{U_1} \\ 0 & \text{if } t > \frac{W - r^0}{U_1} \end{cases} \tag{15}$$

Lemma 2 *(u_1^*, u_2^*, d^*) is globally a saddle solution for the game between (u_1, u_2) and d over J_1.*

Proof: We check that the following inequalities hold for all x^0, u_1, u_2 and d:

$$J_1(x^0, u_1^*, u_2^*, d) \leq J(x^0, u_1^*, u_2^*, d^*) \leq J_1(x^0, u_1, u_2, d^*)$$

The details are given in Appendix A. ∎

For cost J_1' the saddle solution can be similarly calculated. Consider the candidate:

$$u_i'^*(t) = 0 \text{ for all } t \quad \text{and} \quad d'^*(t) = \begin{cases} -U_2 & \text{if } t \leq \frac{r^0}{U_2} \\ 0 & \text{if } t > \frac{r^0}{U_2} \end{cases} \tag{16}$$

Lemma 3 *$(u_1'^*, u_2'^*, d'^*)$ is a saddle solution for the game between (u_1, u_2) and d over J_1'.*

Proof: Similar to the proof of Lemma 2, refer to Appendix A. ∎

It should be noted that the saddle solution is not unique in the latter case, 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). For example, any choice of u_i that starts with $u_i(0) = 0$ and does not involve switching to 1 for more than T_{p_i} at a time will lead to the same cost as the saddle solution.

The saddle solutions allow us to determine the set of states for which there exists inputs for the pumps such that the water level is guaranteed to remain between the specified limits for any steam rate. To accomplish this we first need to determine the costs under the saddle solutions. Let $J_1^*(x^0) = J_1(x^0, u_1^*, u_2^*, d^*)$, x^* be the state trajectory generated by (u_1^*, u_2^*, d^*), and $D_i = T_{p_i} - T_i^0$ be the time at which pump i starts pumping water under input $u_i^*(t)$.

Lemma 4 *If $W \leq P_1 + P_2$ then $J_1^*(x^0) = \min\{w^*(D_1), w^*(D_2)\}$.*

Proof: By direct calculation, refer to Appendix A for details. ∎

This fact allows us to calculate the boundary between safe and unsafe initial conditions. In particular, we would like:

$$J_1^*(x^0) \geq M_1 \quad \Rightarrow \quad w^0 \geq \max\{M_1 - w^*(D_1) + w^0, M_1 - w^*(D_2) + w^0\}$$

The calculations in the appendix indicate that $w^*(D_i) - w^0$ is independent of w^0 and is uniquely specified by the values of r^0, T_1^0 and T_2^0. Therefore, the boundary between safe and unsafe states can be thought of as a function $\hat{w} : [0, W] \times \mathbb{R}_+^2 \to \mathbb{R}$, 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 $T_2^0 = 0$ (i.e. pump 2 initially off) and for $T_2^0 \geq T_{p_2}$ (i.e. pump 2 initially fully on) are shown in Figure 2. 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 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 $T_{p_1} = T_{p_2} = 5$.

To determine J'^*, note that $w'^*(t) \leq w^0$ for any t, therefore, $J'^*(x^0) = w^0$ and any state with $w^0 \leq M_2$ is safe with respect to J'. However, as noted earlier, the u_i^* are not the unique minimizers of J', as

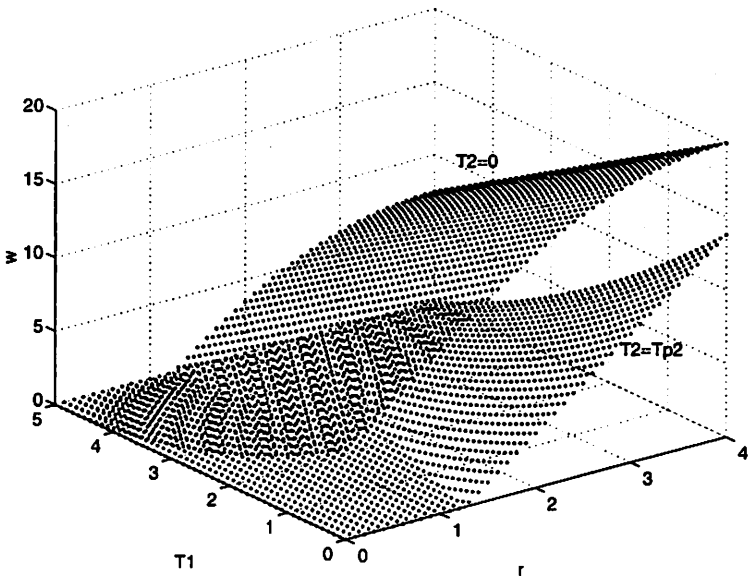

Figure 2: Lower limit on w to avoid draining

any controls such that $\dot{w} \leq 0$ whenever $w = w^0$ achieve the same value of J'. As $\dot{w} = q_1 + q_2 - r$, this observation leads to a boundary between safe and unsafe states. On the boundary, $w^0 = M_2$ (the only situation where J' becomes safety critical) and $r^0 = \hat{r}(T_1^0, T_2^0)$ where:

$$\begin{array}{rcl} \hat{r} : \mathbb{R}^2 & \longrightarrow & \mathbb{R} \\ (T_1^0, T_2^0) & \longmapsto & \left\{ \begin{array}{cl} 0 & \text{if } T_1^0 < T_{p_1} \text{ and } T_2^0 < T_{p_2} \\ P_1 & \text{if } T_1^0 \geq T_{p_1} \text{ and } T_2^0 < T_{p_2} \\ P_2 & \text{if } T_1^0 < T_{p_1} \text{ and } T_2^0 \geq T_{p_2} \\ P_1 + P_2 & \text{if } T_1^0 \geq T_{p_1} \text{ and } T_2^0 \geq T_{p_2} \end{array} \right. \end{array}$$

Pictorially, this boundary is shown in Figure 3. 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'.

5.4 The Class of Least Restrictive, Safe Controls

The calculation of the safe set also allows us to classify the controls that can keep the system safe (water level between M_1 and M_2) 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 5 *A control law for (u_1, u_2) is safe with respect to M_1 if and only if:*

$$\begin{array}{l} u_1 \in \{0,1\} \text{ and } u_2 \in \{0,1\} \text{ if } w > \hat{w}(r,0,0) \\ u_1 = 1 \text{ and } u_2 \in \{0,1\} \text{ if } \hat{w}(r,0,0) \geq w > \hat{w}(r,T_1,0) \\ u_1 \in \{0,1\} \text{ and } u_2 = 1 \text{ if } \hat{w}(r,0,0) \geq w > \hat{w}(r,0,T_2) \\ u_1 = 1 \text{ and } u_2 = 1 \text{ if } w \leq \hat{w}(r,T_1,T_2) \end{array}$$

Proof: The proof is a corollary of Lemma 2 and the properties of the saddle solution. For the "if" part (proposed scheme is *safe*) it suffices to show that any u satisfying the above conditions leads to

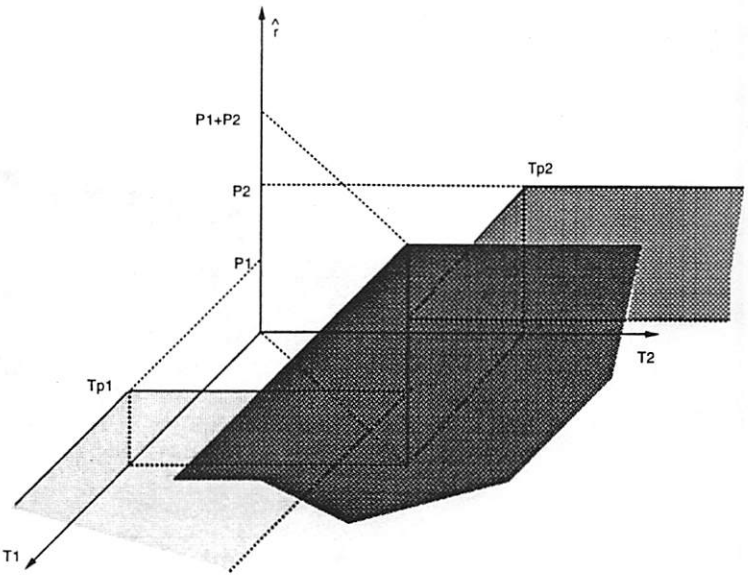

Figure 3: Lower limit on r to avoid overflow

a state trajectory with $w(t) \geq \hat{w}(r, T_{p_1}, T_{p_2}) = M_1$ for all t (under the underlying assumption that $W \leq P_1 + P_2$). The first three cases are relevant only if $w(t) \geq \hat{w}(r, 0, 0)$, $w(t) \geq \hat{w}(r, T_1, 0)$ and $w(t) \geq \hat{w}(r, 0, T_2)$ respectively. As $\hat{w}$ is monotone in T_1 and T_2, the lower bounds are greater than or equal to $\hat{w}(r, T_{p_1}, T_{p_2})$ in all three cases. Therefore, the last case ($w \leq \hat{w}(r, T_1, T_2)$) is the only one we have to worry about. Safety in this case is guaranteed by Lemma 2.

For the "only if" part (proposed scheme is *least restrictive*) we only need to worry about the last three cases (the first case is trivially least restrictive). In the last three cases at least one of the u_i is restricted to be $u_i = 1$. If $u_i = 0$ is used instead, the conditions on the three cases and the monotonicity of $\hat{w}$ with respect to T_i imply that the resulting jump to $T_i = 0$ will result in $w < \hat{w}(r, T_1, T_2)$. In this situation, Lemma 2 guarantees that there exists a d (for example $d = d^*$) and a $t \geq 0$ such that $w(t) < M_1$. Therefore, any control scheme violating the proposed restrictions is potentially unsafe. ■

Note that, as $\hat{w}$ is monotone in T_1 and T_2, the condition on the last case is enabled if and only if all other conditions fail. The two middle conditions may overlap however. 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).

Lemma 6 *A control law for (u_1, u_2) is safe with respect to M_2 if and only if:*

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

Proof: The proof follows as a corollary of Lemma 3. For the "if" part, if $w = M_2$ (the only safety critical situation), the restrictions imposed on u guarantee that $\dot{w} \leq 0$ for d'^* (and hence any d). For

the "only if" part, any control violating the conditions of the lemma will potentially result in $\dot{w} > 0$ when $w = M_2$. ∎

Note again the nondeterminism in the choice of control in the middle two cases (the system may be safe with either one or the other pump on but not both).

The class of controls specified by the lemmas are least restrictive in the sense that *any* control will have to satisfy the lemma conditions to guarantee safety. If the control needs to satisfy other, secondary, objectives on top of safety, Lemmas 5 and 6 give the class of controls in which the optimum for the secondary objectives should be sought. The above calculations also lead to conditions for the existence of safe controls.

Corollary 2 *Safe control laws exist if* $W \leq P_1 + P_2$ *and* $\hat{w}(W, 0, 0) \leq M_2$. *Safe control laws do not exist if* $W > P_1 + P_2$.

6 Flight Vehicle Management Systems

The flight vehicle management system (FVMS) example is based on the dynamic aircraft equations and the design specification of [37]. The equations model the speed and the flight path angle dynamics of a commercial aircraft in still air. The control inputs to the equations are the thrust T, accessed through the engine throttle, and the pitch angle θ, accessed through the elevators. The outputs we wish to control are the speed V and the flight path angle γ. There are three primary modes of operation:

1. **Mode 1:** The thrust T is between its specified operating limits ($T_{min} < T < T_{max}$), the control inputs are T and θ, and both V and γ are controlled outputs.
2. **Mode 2:** The thrust saturates ($T = T_{min} \vee T = T_{max}$) and thus it is no longer available as a control input; the only input is θ, and the only controlled output is V.
3. **Mode 3:** The thrust saturates ($T = T_{min} \vee T = T_{max}$); the input is again θ, and 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 regulations for the aircraft dictate that V and γ must remain within specified limits: for ease of presentation we simplify this *safety envelope*, S, of [37] to

$$S = \{(V, \gamma) | (V_{min} \leq V \leq V_{max}) \cap (\gamma_{min} \leq \gamma \leq \gamma_{max})\} \tag{17}$$

where $V_{min}, V_{max}, \gamma_{min}, \gamma_{max}$ are constant values.

We would like to design a control scheme, an FVMS, to drive the aircraft between operating points in S. The resulting trajectory $(V(t), \gamma(t))$ must satisfy acceleration constraints imposed for passenger comfort, and must not exit the envelope at any time. Here we describe the minimally restrictive set of controllers which guarantees safe operation of the aircraft, by classifying all of the control inputs that keep the $(V(t), \gamma(t))$ trajectory within the safety envelope and establishing the mode switching logic required for safety. An "efficiency" requirement for passenger comfort is then optimized within the class of safe controls.

6.1 System Model and Problem Specification

The flight path angle dynamics of the aircraft can be summarized using two continuous 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:

$$\dot{V} = \frac{T - D}{m} - g \sin \gamma, \quad \dot{\gamma} = \frac{L}{mV} - \frac{g \cos \gamma}{V} \tag{18}$$

where T (N) is the thrust, m (kg) is the mass of the aircraft, g (m/s^2) is gravitational acceleration and L and D are the aerodynamic lift and drag forces. The aerodynamic forces can be modeled by:

$$L = a_L V^2 (1 + c(\theta - \gamma)), \quad D = a_D V^2 (1 + b(1 + c(\theta - \gamma))^2) \tag{19}$$

where a_L and a_D are the lift and drag coefficients, b and c are small positive constants, and θ is the aircraft pitch angle. We assume that the pilot has direct control over the thrust T and the pitch angle θ, thus $u = [T, \theta]$[6]. Substituting (19) into (18) and assuming that b is small enough to neglect the quadratic term in $(\theta - \gamma)$ in the drag, leads to:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = f(x, u) = \begin{bmatrix} -\frac{a_D x_1^2}{m} - g \sin x_2 + \frac{1}{m} u_1 \\ \frac{a_L x_1 (1 - c x_2)}{m} - \frac{g \cos x_2}{x_1} + \frac{a_L c x_1}{m} u_2 \end{bmatrix} \tag{20}$$

For these equations to be meaningful we need to assume that $x_1 > 0$ and that x_2 is bounded above and below by a realistic angle limit. Physical considerations also impose constraints on the inputs,

$$U = [T_{min}, T_{max}] \times [\theta_{min}, \theta_{max}] \tag{21}$$

In our calculations we use the following aircraft parameters and state and input limits, which correspond to a $DC-8$ at cruising speed, at an altitude of $35000ft$: $m = 85000$kg, $c = 6$, $a_L = 30$, $a_D = 2$, $T_{min} = 40000$ N, $T_{max} = 80000$ N, $\theta_{min} = -22.5°$, $\theta_{max} = 22.5°$, $V_{min} = 180$ m/s, $V_{max} = 240$ m/s, $\gamma_{min} = -22.5°$ and $\gamma_{max} = 22.5°$. The bounds on the pitch angle θ and the flight path angle γ are chosen to be symmetric about zero for ease of computation. In actual flight systems, the positive bound on these angles is greater than the negative bound. Also, the angles chosen for this example are much higher than what are considered acceptable for passenger flight ($\pm 10°$).

To guarantee safety we need to ensure that the aircraft trajectory $x(t) \in S$ for all t. The additional constraint of passenger comfort is satisfied if the aircraft linear and angular acceleration remain bounded by $0.1g$:

$$\begin{aligned} |\dot{x}_1(t)| &\leq 0.1g \\ |x_1(t)\dot{x}_2(t)| &\leq 0.1g \end{aligned} \tag{22}$$

6.2 Optimal Control Inputs and Safe Set of States

Safety is maintained by operating within the largest subset V_1 of S which can be rendered invariant by using a control input $u \in \mathcal{U}$. Let ∂S denote the boundary of S, ∂V_1 denote the boundary of V_1. We calculate the set V_1 by solving an optimal control problem over a time interval $[t, t_f]$. As we are interested only in whether or not the state leaves S, we define t_f to be the first time at which the state leaves S:

$$t_f = \inf\{\tau \in \mathbb{R} | x(\tau) \notin S, x(t) \in S\} \tag{23}$$

[6]The bounds on the dynamics which arise from the relationship between the engine throttle and the forward thrust, and the elevators and the aircraft pitch are introduced in the calculation through the constraints on the inputs and the state variables.

and we let t be free. If t_f exists, then for ease of notation we set $t_f = 0$ and consider negative initial times t (without loss of generality, as the dynamics are time invariant). The cost function $J_1(x, t, u(\cdot))$ depends only on the state at the terminal time:

$$J_1(x, t, u(\cdot)) = l(x(0)) \tag{24}$$

where $l(x)$ is such that:

$$\begin{array}{lll} l(x) > 0 & x \in S \backslash \partial S & \text{[Safe]} \\ l(x) = 0 & x \in \partial S & \text{[Boundary]} \\ l(x) < 0 & x \in \mathbb{R}^n \backslash S & \text{[Unsafe]} \end{array} \tag{25}$$

The optimally safe control input $u^*(\cdot) \in \mathcal{U}$ is therefore the one which maximizes $J_1(x, t, u(\cdot))$:

$$u^*(\cdot) = \arg \max_{u(\cdot) \in \mathcal{U}} J_1(x, t, u(\cdot)), \quad J_1^*(x, t) = \max_{u(\cdot) \in \mathcal{U}} J_1(x, t, u(\cdot)) \tag{26}$$

The Hamiltonian is given by

$$H_1(x, p, u) = pf(x, u) \tag{27}$$

where $p \in T^*\mathbb{R}^2$ is the costate. The optimal Hamiltonian is thus

$$\begin{aligned} H_1^*(x, p) &= \max_{u \in U} H_1(x, p, u) = H_1(x, p, u^*) \\ &= \max_{u \in U} [p_1(-\frac{a_D x_1^2}{m} - g \sin x_2 + \frac{1}{m} u_1) + p_2(\frac{a_L x_1 (1 - c x_2)}{m} - \frac{g \cos x_2}{x_1} + \frac{a_L c x_1}{m} u_2)] \end{aligned}$$

For a given initial time t, the safe set of states $V_1(t)$ is, from equations (24) and (25),

$$V_1(t) = \{x \in S | \exists u(\cdot) \in \mathcal{U}, J_1(x, t, u(\cdot)) \geq 0\} \tag{28}$$

$$= \{x \in S | J_1^*(x, t) \geq 0\} \tag{29}$$

If we let $t \to -\infty$, the set $V_1(t)$ becomes the "steady state" safe set:

$$V_1 \equiv V_1(-\infty) = \{x \in S | J_1^*(x, -\infty) \geq 0\} \tag{30}$$

with boundary $\partial V_1 = \{x \in S | J_1^*(x, -\infty) = 0\}$. If $J_1^*(x, t)$ is a smooth function of x and t, meaning that there are no "shocks", or discontinuities of $J_1^*(x, t)$ as a function of x as t evolves, then $J_1^*(x, t)$ satisfies the Hamilton-Jacobi equation:

$$\frac{\partial J_1^*(x, t)}{\partial t} = -H_1^*(x, \frac{\partial J_1^*(x, t)}{\partial x}) \tag{31}$$

with boundary condition $J_1^*(x, 0) = l(x)$. In order to compute the steady state solution $J_1^*(x, -\infty)$ of (31), we assume that no shocks exist, and set the left hand side of the Hamilton-Jacobi equation to zero. Thus, $\frac{\partial J_1^*(x, -\infty)}{\partial x}$ is normal to the vector field $f(x, u^*)$.

Consider the following construction. Define each edge of ∂S separately, as

$$l_1^1(x) = x_1 - V_{min}, \; l_1^2(x) = -x_2 + \gamma_{max}, \; l_1^3(x) = -x_1 + V_{max}, \; l_1^4(x) = x_2 - \gamma_{min} \tag{32}$$

with $x \in S$. We introduce the following notation: $J_1^i(x, t, u(\cdot)) = l_1^i(x(0))$ is the cost function for edge i, $H_1^i(x, p, u)$ is the corresponding Hamiltonian, and $p_i = \partial l_1^i(x)/\partial x$ is the inward pointing normal to $l_1^i(x) = 0$. Starting with $l_1^1(x)$, $p_1 = [1, 0]^T$, so that along this boundary, $u_1^* = T_{max}$ but u_2 is indeterminate. Because of the loss of dependency of the optimal Hamiltonian on u_2, the points in

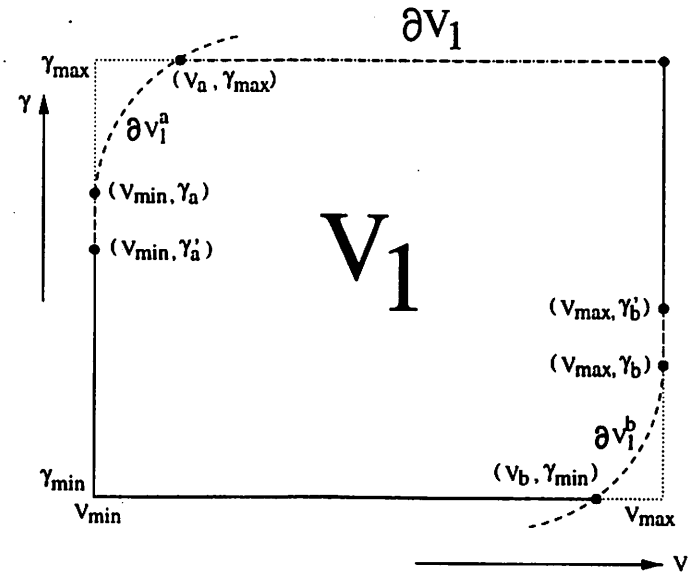

Figure 4: The safe set of states, V_1, and its boundary ∂V_1

$\{x \in S | l_1^1(x) = 0\}$ are called *abnormal extremals.* Define $(V_{min}, \gamma_a) = \{x \in S | l_1^1(x) = 0 \cap H_1^*(x) = 0\}$ and calculate γ_a by:

$$\gamma_a = \sin^{-1}(\frac{T_{max}}{mg} - \frac{a_D V_{min}^2}{mg}) \tag{33}$$

Integrate the system dynamics

$$\dot{x} = f(x, u^*), \quad x(0) = (V_{min}, \gamma_a) \tag{34}$$

backwards from $t = 0$ to $t = -T$, where T is chosen to be large enough so that the solution to (34) intersects $\{x \in S | l_1^2(x) = 0\}$. The optimal control u_2^* is required for this calculation. At the abnormal extremal (V_{min}, γ_a), any $u_2 \in [\theta_{min}, \theta_{max}]$ may be used. However, as we integrate the system, we leave the abnormal extremal regardless of the choice of u_2 instantaneously, and u_2^* is uniquely determined. For all $u_2 \in [\theta_{min}, \theta_{max}]$, for all $\delta \in \mathbb{R}^+$, the inward pointing normal to $f(x(-\delta), [u_1^* \; u_2]^T)$ is such that p_2 is negative, thus, $u_2^* = \theta_{min}$. Denote the point of intersection of the solution of (34) with $\{x \in S | l_1^2(x) = 0\}$ as (V_a, γ_{max}), and the solution to (34) between (V_{min}, γ_a) and (V_a, γ_{max}) as ∂V_1^a, as shown in Figure 4. In this example, the abnormal extremal was not complicated enough to cause difficulties in the construction; the general situation is considered in [38]. Repeat this calculation for the remaining three boundaries. Only $\{x \in S | l_1^3(x) = 0\}$ contains a point at which $H_1^*(x)$ vanishes. We denote this point as (V_{max}, γ_b) where:

$$\gamma_b = \sin^{-1}(\frac{T_{min}}{mg} - \frac{a_D V_{max}^2}{mg}) \tag{35}$$

and similarly calculate ∂V_1^b and V_b, as shown in Figure 4.

Lemma 7 *For the aircraft dynamics (20) with flight envelope S given by (17) and input constraints*

(21), the safe set of states V_1 is the set enclosed by ∂V_1, given by

$$\begin{aligned}\partial V_1 = \{(V,\gamma)|\ & (V = V_{min}) \wedge (\gamma_{min} \le \gamma \le \gamma_a) & \vee \\ & \partial V_1^a & \vee \\ & (\gamma = \gamma_{max}) \wedge (V_a \le V \le V_{max}) & \vee \\ & (V = V_{max}) \wedge (\gamma_b \le \gamma \le \gamma_{max}) & \vee \\ & \partial V_1^b & \vee \\ & (\gamma = \gamma_{min}) \wedge (V_{min} \le V \le V_b)\}\end{aligned} \tag{36}$$

Proof: The safe set is $\{x \in S | J_1^*(x, -\infty) \ge 0\}$, where $J_1^*(x, -\infty)$ is the steady state solution to the Hamilton-Jacobi equation. Starting with $\{x \in S | l_1^1(x) = 0\}$, the optimal Hamiltonian $H_1^{1*}(x,p)$ satisfies:

$$H_1^{1*}(x,p) \begin{cases} < 0 & x \in S \cap l_1^1(x) = 0 \cap \gamma > \gamma_a \\ = 0 & x \in S \cap l_1^1(x) = 0 \cap \gamma = \gamma_a \\ > 0 & x \in S \cap l_1^1(x) = 0 \cap \gamma < \gamma_a \end{cases}$$

so that $\{x \in S | (V = V_{min} \cap (\gamma_{min} \le \gamma \le \gamma_a)\}$ is safe with respect to $l_1^1(x)$.

We now prove that for $x \in \partial V_1^a$, $J_1^{1*}(x, -\infty) = 0$. $J_1^{1*}(x, -\infty)$ satisfies:

$$(\frac{\partial J_1^{1*}(x, -\infty)}{\partial x}) f(x, u^*) = 0$$

where $\partial J_1^{1*}(x, -\infty)/\partial x$ is the inward pointing normal to $\{x | J_1^{1*}(x, -\infty) = 0\}$. At each point x in $\{x | J_1^{1*}(x, -\infty) = 0\}$, $f(x, u^*)$ is tangent to $\{x | J_1^{1*}(x, -\infty) = 0\}$. Thus the solution $x(t)$ to $\dot{x} = f(x, u^*)$ evolves along $J_1^{1*}(x, -\infty) = 0$. By construction, $x \in \partial V_1^a$ satisfies $J_1^{1*}(x, -\infty) = 0$.

Repeating this analysis for $\{x \in S | l_1^3(x) = 0\}$, we can prove that $(V = V_{max}) \wedge (\gamma_b \le \gamma \le \gamma_{max})$ is safe with respect to $l_3(x)$, and $x \in \partial V_1^b$ satisfies $J_1^{3*}(x, -\infty) = 0$. On the remaining boundaries, $H_1^{2*}(x,p)$ and $H_1^{4*}(x,p)$ respectively are greater than zero, so $(\gamma = \gamma_{max}) \wedge (V_a \le V \le V_{max})$ and $(\gamma = \gamma_{min}) \wedge (V_{min} \le V \le V_b)$ are safe with respect to $l_1^2(x)$ and $l_1^4(x)$. ∎

6.3 The Least Restrictive Safe Control Scheme

The safe set of control inputs U_1 can be characterized as a set-valued feedback map $U_1 : S \to 2^U$:

Lemma 8 *The safe set of control inputs is $U_1(x) = U \cap \hat{U}_1(x)$, where:*

$$\begin{aligned}\hat{U}_1(V,\gamma) = \{\ & \emptyset & & \text{if } (V,\gamma) \in S \backslash V_1 \\ & T \ge T_a(\gamma) & & \text{if } (V = V_{min}) \wedge (\gamma_{min} \le \gamma \le \gamma_a) \\ & \theta = \theta_{min} \wedge T = T_{max} & & \text{if } (V,\gamma) \in \partial V_1^a \\ & \theta \le \theta_c(V) & & \text{if } (\gamma = \gamma_{max}) \wedge (V_a \le V \le V_{max}) \\ & T \le T_b(\gamma) & & \text{if } (V = V_{max}) \wedge (\gamma_b \le \gamma \le \gamma_{max}) \\ & \theta = \theta_{max} \wedge T = T_{min} & & \text{if } (V,\gamma) \in \partial V_1^b \\ & \theta \ge \theta_d(V) & & \text{if } (\gamma = \gamma_{min}) \wedge (V_{min} \le V \le V_b) \\ & \theta_{min} \le \theta \le \theta_{max} \wedge T_{min} \le T \le T_{max} & & \text{else}\}\end{aligned} \tag{37}$$

where

$$T_a(\gamma) \ = \ a_D V_{min}^2 + mg \sin\gamma \tag{38}$$

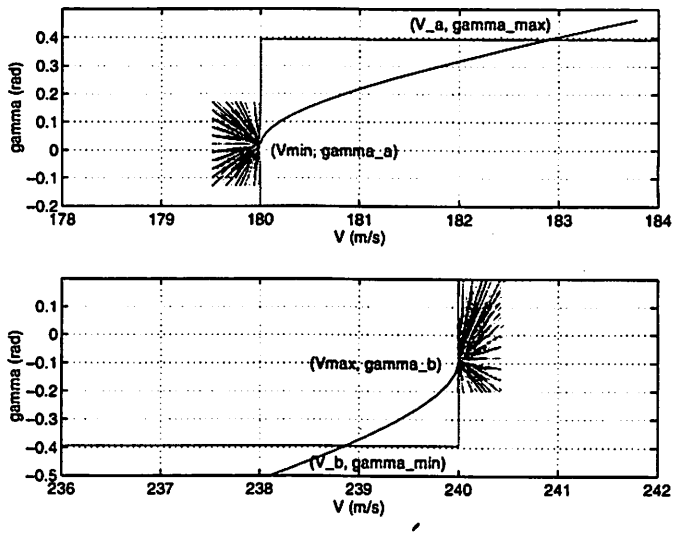

Figure 5: Upper left boundary and lower right boundary of the safe set

$$T_b(\gamma) = a_D V_{max}^2 + mg\sin\gamma \tag{39}$$

$$\theta_c(V) = \frac{m}{a_L V c}\left(\frac{g\cos\gamma_{max}}{V} - \frac{a_L V(1-c\gamma_{max})}{m}\right) \tag{40}$$

$$\theta_d(V) = \frac{m}{a_L V c}\left(\frac{g\cos\gamma_{min}}{V} - \frac{a_L V(1-c\gamma_{min})}{m}\right) \tag{41}$$

Proof: Consider the left side of ∂S. For each x in $\{x \in S | l_1^1(x) = 0\}$, denote by $(T_a(\gamma), \theta_a(\gamma))$ the values of (T, θ) for which the vector field $f(x, [T, \theta]^T)$ becomes tangent to $l_1^1(x) = 0$ (i.e. $\dot{V} = 0$). Setting $\dot{V} = 0$ leads to equation (38), for all $\theta_a(\gamma) \in [\theta_{min}, \theta_{max}]$. Therefore, the safe set of inputs along $\{x \in S | l_1^1(x) = 0\}$ are all $T \in [T_{min}, T_{max}]$ with $T \geq T_a(\gamma)$ and all $\theta \in [\theta_{min}, \theta_{max}]$. At the point (V_{min}, γ_a'), where $\gamma_a' = \{\gamma | T_a(\gamma) = T_{min}\}$ the cone of vector fields $f([V_{min}, \gamma_a'], U)$ points completely inside S. At $\gamma_a = \{\gamma | T_a(\gamma) = T_{max}\}$ the cone of vector fields points completely outside S, and T_{max} is the unique thrust which keeps the system trajectory tangent to S. This is illustrated in Figure 5, which shows the upper left boundary of the safe set, and the cone of controls at the point (V_{min}, γ_a).

The calculation may be repeated for the right side of ∂S: $\{x \in S | l_1^3(x) = 0\}$. Here, let $T_b(\gamma)$ be the value of the input thrust for which $f(x, [T_b(\gamma), \theta]^T)$ is tangent to $l_1^3(x) = 0$, thus $T_b(\gamma)$ is given by equation (39). The safe set of inputs along $\{x \in S | l_1^3(x) = 0\}$ are all $T \in [T_{min}, T_{max}]$ with $T \leq T_b(\gamma)$ and all $\theta \in [\theta_{min}, \theta_{max}]$. At the point (V_{max}, γ_b), where $\gamma_b = \{\gamma | T_b(\gamma) = T_{min}\}$, T_{min} is the unique thrust which keeps the system trajectory tangent to S (lower right boundary of the safe set, in Figure 5).

Similar calculations along the upper and lower sides of ∂S yield that the values of θ for which the vector field becomes tangent to ∂S are $\theta_c(V)$ and $\theta_d(V)$ of equations (40) and (41). ∎

In Figure 4, the portions of ∂V_1 for which all control inputs are safe $(U_1(x) = U(x))$ are indicated with solid lines; those for which only a subset are safe $(U_1(x) \subset U(x))$ are indicated with dashed lines. The map defines the *least restrictive safe control scheme* and determines the mode switching logic. On ∂V_1^a and ∂V_1^b, 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 equation (37) are satisfied. In the regions

$S \backslash V_1$ (the upper left and lower right corners of S), no control inputs are safe.

6.4 Additional Constraints for Passenger Comfort

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

$$J_2(x, u(\cdot)) = \max_{t \geq 0} |\dot{x}_1(t)|, \quad J_2'(x, u(\cdot)) = \max_{t \geq 0} |x_1(t)\dot{x}_2(t)| \tag{42}$$

The requirement that the linear and angular acceleration remain within the limits determined for comfortable travel are encoded by thresholds:

$$J_2(x, u(\cdot)) \leq 0.1g, \quad J_2'(x, u(\cdot)) \leq 0.1g \tag{43}$$

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:

$$\begin{array}{rclcrcl} J_2^*(x) & = & \min_{u \in \mathcal{U}_1} J_2(x, u(\cdot)), & & u^*(x) & = & \arg\min_{u \in \mathcal{U}_1} J_2(x, u(\cdot)) \\ J_2'^*(x) & = & \min_{u \in \mathcal{U}_1} J_2'(x, u(\cdot)), & & u'^*(x) & = & \arg\min_{u \in \mathcal{U}_1} J_2'(x, u(\cdot)) \end{array} \tag{44}$$

From this calculation, we determine the set of "comfortable" states and controls:

$$V_2 = \{x \in V_1 | J_2^*(x) \leq 0.1g \wedge J_2'^*(x) \leq 0.1g\} \tag{45}$$

$$\mathcal{U}_2(x) = \{u \in \mathcal{U}_1 | J_2(x, u(\cdot)) \leq 0.1g \wedge J_2'(x, u(\cdot)) \leq 0.1g\} \tag{46}$$

These sets may be easily calculated by substituting the bounds on the accelerations into equation (20) to get

$$\begin{array}{ccccc} -0.1mg + a_D V^2 + mg\sin\gamma & \leq & T & \leq & 0.1mg + a_D V^2 + mg\sin\gamma \\ -\frac{0.1mg}{a_L V^2 c} - \frac{1-c\gamma}{c} + \frac{mg\cos\gamma}{a_L V^2 c} & \leq & \theta & \leq & \frac{0.1mg}{a_L V^2 c} - \frac{1-c\gamma}{c} + \frac{mg\cos\gamma}{a_L V^2 c} \end{array} \tag{47}$$

These constraints provide lower and upper bounds on the thrust and the pitch angle which may be applied at any point (V, γ) in V_2. Figure 6 illustrates the set V_2, within the safe set V_1.

7 Conclusions

We have presented a methodology for synthesizing controllers to satisfy multiple performance requirements for hybrid systems. In this paper we have restricted our attention to two requirements, safety and efficiency; the methodology easily extends to an arbitrary number. We have illustrated the key features of our approach using three examples, a purely discrete system, a continuous system controlled by discrete inputs, and a continuous system with discrete modes of operation induced by input saturation.

The notions of "maximal safe set" and "least restrictive safe controller" are central to our formulation. They allow us to deal with the multi-objective nature of the problem by solving a sequence of nested two player, zero sum games. These notions are also important in the hierarchical control context. Assume that a number of controllers are synthesized (using the methodology introduced here for example), each designed to deal with a particular situation, and we are asked to develop a discrete supervisor to switch between them. The maximal safe sets for each controller provide necessary

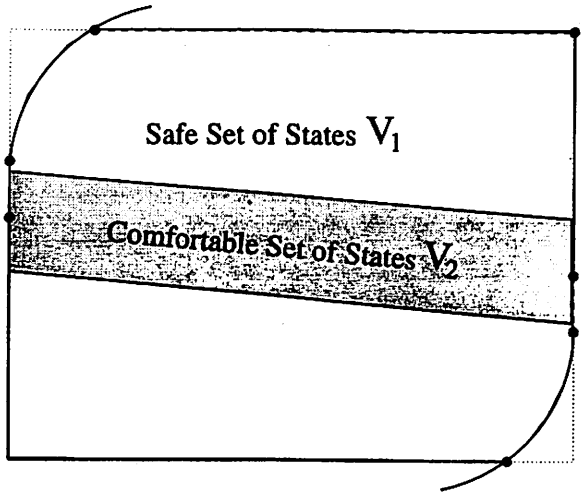

Figure 6: Comfort constraint intersected with the safe set (V_1) resulting in the "comfortable" set of states (V_2)

enabling conditions for the transitions of the supervisor; a particular controller should be invoked only if the current value of the state lies in the corresponding safe set.

In the examples considered here the maximal safe sets and least restrictive safe controllers naturally emerged from the calculations. We would like to develop a formal methodology to capture this procedure. The techniques used in the last example (FVMS) seem to be the most promising in this respect. We are currently working on formalizing these techniques in the context of semi-permeable surface calculation in pursuit evasion games. Semi-permeable surfaces form the boundary of the maximal safe set and define regions where there are limitations on the allowable controls.

The methods presented in this paper can also have important implications for the introduction of new controllers into so-called *legacy systems* for real time control. Legacy systems come equipped with a controller with a guaranteed domain of validity, say V_1[7]. Assume one would like to retro-fit the system with a new experimental controller with unknown domain of validity, presumably in an attempt to improve performance. This addition should be done in a way that does not compromise the safety of the system. One way of accomplishing this is to utilize the experimental controller only in the interior of the validity set V_1 and resort to the legacy controller as soon as the state approaches the boundary of V_1. Our methods are useful for systematically computing the switching logic among the controllers and determining the switching boundaries.

References

[1] R. Alur and D. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, pp. 183–235, 1994.

[2] X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "An approach to the description and analysis of hybrid systems," in *Hybrid System* (R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, eds.), no. 736 in LNCS, pp. 149–178, New York: Springer Verlag, 1993.

[7] Actually, the legacy controllers frequently involve state based switching between controllers designed for "safety" and "performance", not unlike the systems discussed here.

[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* (R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, eds.), no. 736 in LNCS, pp. 209–229, New York: Springer Verlag, 1993.

[4] T. Henzinger, P. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata," in 27^{th} *Annual Symposium on the Theory of Computing, STOC'95*, pp. 373–382, ACM Press, 1995.

[5] R. P. Kurshan, *Computer-aided verification of coordinating processes; the automata-theoretic approach.* Princeton University Press, 1994.

[6] C. Daws, A. Olivero, and S. Yovine, "Verifying ET-LOTOS programs with KRONOS," in *Proc. 7th. IFIP WG G.1 International Conference of Formal Description Techniques, FORTE'94* (D. Hogrefe and S. Leue, eds.), (Bern, Switzerland), pp. 227–242, Formal Description Techniques VII, Chapman & Hall, Oct. 1994.

[7] T. A. Henzinger, P. H. Ho, and H. W. Toi, "A user guide to HYTECH," in *TACAS 95: Tools and Algorithms for the Construction and Analysis of Systems* (E. Brinksma, W. Cleaveland, K. Larsen, T. Margaria, and B. Steffen, eds.), no. 1019 in LNCS, pp. 41–71, Springer Verlag, 1995.

[8] C. Heitmayer and N. Lynch, "The generalized railroad crossing: A case study in formal verification of real-time systems," in *Proc. ICCC Real-Time Systems Symposium*, (San Juan, Puerto Rico), 1994.

[9] Z. Manna and A. Pnueli, *Temporal Verification of Reactive Systems: Safety.* New York: Springer-Verlag, 1995.

[10] M. Branicky, E. Dolginova, and N. Lynch, "A toolbox for proving and maintaining hybrid specifications." preprint, 1996. presented in *Workshop on Hybrid Systems*, Cornell University, October 12-16.

[11] Z. Manna and A. Pnueli, *The Temporal Logic of Reactive and Concurrent Systems: specification.* Berlin: Springer-Verlag, 1992.

[12] 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.

[13] R. W. Brockett, "Hybrid models for motion control systems," in *Perspectives in Control* (H. Trentelman and J. Willems, eds.), Birkhäuser, 1993.

[14] A. Nerode and W. Kohn, "Models for hybrid systems: Automata, topologies, controllability, observability," in *Hybrid System* (R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, eds.), no. 736 in LNCS, pp. 317–356, New York: Springer Verlag, 1993.

[15] M. S. Branicky, *Control of Hybrid Systems.* PhD thesis, Massachussets Institute of Technology, 1994.

[16] L. Tavernini, "Differential automata and their simulators," *Nonlinear Analysis, Theory, Methods and Applications*, vol. 11(6), pp. 665–683, 1987.

[17] A. Deshpande, A. Gollu, and L. Semenzato, "The SHIFT programming language and run-time system for dynamic networks of hybrid automata," Tech. Rep. UCB-ITS-PRR-97-7, Institute of Transportation Studies, University of California, Berkeley, 1997.

[18] M. Anderson, D. Bruck, S. E. Mattsson, and T. Schonthal, "Omsim- an integrated interactive environment for object-oriented modeling and simulation," in *IEEE/IFAC joint symposium on computer aided control system design*, pp. 285–90, 1994.

[19] L. Hou, A. Michel, and H. Ye, "Stability analysis of switched systems," in *IEEE Conference on Decision and Control*, pp. 1208–1214, 1996.

[20] O. Maler, A. Pnueli, and J. Sifakis, "On the synthesis of discrete controllers for timed systems," in *Theoretical Aspects of Computer Science*, no. 900 in LNCS, pp. 229–242, Springer Verlag, 1995.

[21] M. Heymann, F. Lin, and G. Meyer, "Control synthesis for a class of hybrid systems subject to configuration-based safety constraints," in *Hybrid and Real Time Systems*, no. 1201 in LNCS, pp. 376–391, Springer Verlag, 1997.

[22] M. Lemmon, J. A. Stiver, and P. J. Antsaklis, "Event identification and intelligent hybrid control," in *Hybrid System* (R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, eds.), no. 736 in LNCS, pp. 268–296, New York: Springer Verlag, 1993.

[23] A. Nerode and W. Kohn, "Multiple agent hybrid control architecture," in *Hybrid System* (R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, eds.), no. 736 in LNCS, pp. 297–316, New York: Springer Verlag, 1993.

[24] J. Lygeros, D. N. Godbole, and S. Sastry, "Multiagent hybrid system design using game theory and optimal control," in *IEEE Conference on Decision and Control*, pp. 1190–1195, 1996.

[25] J. Lygeros, D. N. Godbole, and S. Sastry, "A game theoretic approach to hybrid system design," in *Hybrid Systems III*, no. 1066 in LNCS, pp. 1–12, Springer Verlag, 1996.

[26] J. Lygeros, D. N. Godbole, and S. Sastry, "A verified hybrid controller for automated vehicles," Tech. Rep. UCB-ITS-PRR-97-9, Institute of Transportation Studies, University of California, Berkeley, 1997. (to appear in the Special Issue on Hybrid Systems of the IEEE Transactions on Automatic Control).

[27] C. Tomlin, G. Pappas, and S. Sastry, "Conflict resolution for air traffic management: a case study in multi-agent hybrid systems," Tech. Rep. UCB/ERL M96/38, Electronic Research Laboratory, University of California Berkeley, 1996.

[28] J. Lygeros, *Hierarchical Hybrid Control of Large Scale Systems.* PhD thesis, Department of Electrical Engineering, University of California, Berkeley, 1996.

[29] J.-R. Abrial, "The steam-boiler control specification problem," in *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control* (J.-R. Abrial, E. Börger, and H. Langmaack, eds.), no. 1165 in LNCS, Springer Verlag, 1996.

[30] A. Deshpande, *Control of Hybrid Systems.* PhD thesis, Department of Electrical Engineering, University of California, Berkeley, 1994.

[31] A. Puri, *Theory of Hybrid Systems and Discrete Event Systems.* PhD thesis, Department of Electrical Engineering, University of California, Berkeley, 1995.

[32] R. Alur and T. Henzinger, "Reactive modules," in *Proceedings of the 11th Annual Symposium on Logic in Computer Science*, pp. 207–218, IEEE Computer Society Press, 1996.

[33] A. L. Schwartz, *Theory and Implementation of Numerical Methods Based on Runge-Kutta Integration for Solving Optimal Control Problems.* PhD thesis, Department of Electrical Engineering, University of California, Berkeley, 1996.

[34] J. Lygeros, D. N. Godbole, and S. Sastry, "Optimal control approach to multiagent, hierarchical system verification," in *IFAC World Congress*, 1996.

[35] 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.

[36] 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* (J.-R. Abrial, E. Börger, and H. Langmaack, eds.), no. 1165 in LNCS, pp. 265–282, Springer Verlag, 1996.

[37] 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.

[38] C. Tomlin, S. Sastry, and R. Montgomery, "Computing safe sets using the Hamilton-Jacobi equation." (to be published), 1997.

A Additional Proofs

Lemma 2 *(u_1^*, u_2^*, d^*) is globally a saddle solution for the game between (u_1, u_2) and d over J_1.*

Proof: We check that the following inequalities hold for all x^0, u_1, u_2 and d:

$$J_1(x^0, u_1^*, u_2^*, d) \leq J(x^0, u_1^*, u_2^*, d^*) \leq J_1(x^0, u_1, u_2, d^*)$$

The state evolution, $x^*(t)$, under the saddle input is:

$$q_i^*(t) = \begin{cases} 0 & \text{if } t \leq T_{p_i} - T_i^0 \\ P_i & \text{if } t \geq T_{p_i} - T_i^0 \end{cases}$$

$$w^*(t) = w^0 + \begin{cases} -\frac{U_1 t^2}{2} - r^0 t & \text{if } t \leq T_r \wedge t \leq D_1 \wedge t \leq D_2 \\ -W(t - T_r) - \frac{U_1 T_r^2}{2} - r^0 T_r & \text{if } t \geq T_r \wedge t \leq D_1 \wedge t \leq D_2 \\ P_1(t - D_1) - \frac{U_1 t^2}{2} - r^0 t & \text{if } t \leq T_r \wedge t \geq D_1 \wedge t \leq D_2 \\ P_1(t - D_1) - W(t - T_r) - \frac{U_1 T_r^2}{2} - r^0 T_r & \text{if } t \geq T_r \wedge t \geq D_1 \wedge t \leq D_2 \\ P_2(t - D_2) - \frac{U_1 t^2}{2} - r^0 t & \text{if } t \leq T_r \wedge t \leq D_1 \wedge t \geq D_2 \\ P_2(t - D_2) - W(t - T_r) - \frac{U_1 T_r^2}{2} - r^0 T_r & \text{if } t \geq T_r \wedge t \leq D_1 \wedge t \geq D_2 \\ P_1(t - D_1) + P_2(t - D_2) - \frac{U_1 t^2}{2} - r^0 t & \text{if } t \leq T_r \wedge t \geq D_1 \wedge t \geq D_2 \\ P_1(t - D_1) + P_2(t - D_2) - W(t - T_r) - \frac{U_1 T_r^2}{2} - r^0 T_r & \text{if } t \geq T_r \wedge t \geq D_1 \wedge t \geq D_2 \end{cases}$$

$$r^*(t) = \begin{cases} r^0 + U_1 t & \text{if } t \le T_r \\ W & \text{if } t \ge T_r \end{cases}$$
$$T_i^* = T_i^0 + t$$

where $T_r = \frac{W - r^0}{U_1}$ is the time at which the steam rate reaches its maximum value under disturbance $d(t) = U_1$ and $D_i = T_{p_i} - T_i^0$ is the time at which pump i starts pumping water under input $u_i^*(t)$. Even though there are eight possible expressions for $w(t)$, once the initial condition (and hence D_1, D_2 and T_r) is fixed only four of them need concern us. For example, if $D_1 \le T_r \le D_2$ the only possibilities are:

$$w^*(t) = w^0 + \begin{cases} -\frac{U_1 t^2}{2} - r^0 t & \text{if } t \le D_1 \le T_r \le D_2 \\ P_1(t - D_1) - \frac{U_1 t^2}{2} - r^0 t & \text{if } D_1 \le t \le T_r \le D_2 \\ P_1(t - D_1) - W(t - T_r) - \frac{U_1 T_r^2}{2} - r^0 T_r & \text{if } D_1 \le T_r \le t \le D_2 \\ P_1(t - D_1) + P_2(t - D_2) - W(t - T_r) - \frac{U_1 T_r^2}{2} - r^0 T_r & \text{if } D_1 \le T_r \le D_2 \le t \end{cases}$$

First fix $u_i = u_i^*$ and allow d to vary. Let $x(t)$ denote the state evolution starting at x^0 under (u_1^*, u_2^*, d). Then:

$$w(t) = w^*(t) - \int_0^t (r(\tau) - r^*(\tau)) d\tau$$

But, by the assumed constraints on r and d, $r(t) \le r^*(\tau)$ for all t. Therefore $w(t) \ge w^*(t)$ for all t and $J_1(x^0, u_1^*, u_2^*, d) = -\inf_{t \ge 0} w(t) \le J_1(x^0, u_1^*, u_2^*, d^*)$.

Now fix $d = d^*$ and allow u_1 and u_2 to vary. Again let $x(t)$ denote the state evolution starting at x^0 under (u_1, u_2, d^*). The constraints on the switching imply that $q_i^*(t) \ge q_i(t)$ for all $t \ge 0$ (with the obvious notational interpretation). But:

$$w(t) = w^*(t) - \int_0^t (q_1^*(\tau) + q_2^*(\tau) - q_1(\tau) - q_2(\tau)) d\tau$$

Therefore, $w(t) \le w^*(t)$ and $J_1(x^0, u_1, u_2, d^*) = -\inf_{t \ge 0} w(t) \ge J_1(x^0, u_1^*, u_2^*, d^*)$. ∎

Lemma 3 *$(u_1'^*, u_2'^*, d'^*)$ is a saddle solution for the game between (u_1, u_2) and d over J_1'.*

Proof: As there is no delay in switching the pumps off, the state evolution, $x'^*(t)$, under the saddle input is:

$$q_i'^*(t) = 0 \text{ for all } t$$
$$w'^*(t) = w^0 + \begin{cases} \frac{U_2 t^2}{2} - r^0 t & \text{if } t \le \frac{r^0}{U_2} \\ -\frac{(r^0)^2}{2U_2} & \text{if } t \ge \frac{r^0}{U_2} \end{cases}$$
$$r'^*(t) = \begin{cases} r^0 - U_2 t & \text{if } t \le \frac{r^0}{U_2} \\ 0 & \text{if } t \ge \frac{r^0}{U_2} \end{cases}$$
$$T_i'^* = 0 \text{ for all } t$$

As above, first fix u and allow d to vary. The resulting state trajectory will satisfy $r(t) \ge r'^*(t)$ and therefore $w(t) \le w'^*(t)$ for all t. Hence, $J_1'(x^0, u_1'^*, u_2'^*, d) \le J_1'(x^0, u_1'^*, u_2'^*, d'^*)$. Likewise, if we fix $d = d'^*$ and allow u to vary, the resulting trajectory will satisfy $q_i(t) \ge q_i'^*(t)$ and therefore $w(t) \ge w'^*(t)$ for all t. Hence, $J_1'(x^0, u_1, u_2, d'^*) \ge J_1'(x^0, u_1'^*, u_2'^*, d'^*)$. ∎

Lemma 4 *If* $W \leq P_1 + P_2$ *then* $J_1^*(x^0) = \min\{w^*(D_1), w^*(D_2)\}$.

Proof: To facilitate the algebra consider the derivative of the water level $\dot{w}^*$ which is given by:

$$\dot{w}^*(t) = \begin{cases} -U_1 t - r^0 & \text{if } t \leq T_r \wedge t \leq D_1 \wedge t \leq D_2 \\ -W & \text{if } t \geq T_r \wedge t \leq D_1 \wedge t \leq D_2 \\ P_1 - U_1 t - r^0 & \text{if } t \leq T_r \wedge t \geq D_1 \wedge t \leq D_2 \\ P_1 - W & \text{if } t \geq T_r \wedge t \geq D_1 \wedge t \leq D_2 \\ P_2 - U_1 t - r^0 & \text{if } t \leq T_r \wedge t \leq D_1 \wedge t \geq D_2 \\ P_2 - W & \text{if } t \geq T_r \wedge t \leq D_1 \wedge t \geq D_2 \\ P_1 + P_2 - U_1 t - r^0 & \text{if } t \leq T_r \wedge t \geq D_1 \wedge t \geq D_2 \\ P_1 + P_2 - W & \text{if } t \geq T_r \wedge t \geq D_1 \wedge t \geq D_2 \end{cases}$$

This expression leads us to distinguish the following cases:

Case 1: If $W > P_1 + P_2$, then, for t large enough (in particular $t \geq \max\{T_r, D_1, D_2\}$), $\dot{w}^*(t) = P_1 + P_2 - W < 0$ therefore $w^* \to -\infty$. Clearly in this case a game winning strategy does not exist for u_i for any initial condition, as d can always force the water to drop below any level.

Case 2: If $W \leq P_1 + P_2$ we can distinguish three further cases:
Case 2.1*:* If $W \leq \min\{P_1, P_2\}$, then:

$$\dot{w}^*(t) \begin{cases} < 0 & \text{if } t \leq \min\{D_1, D_2\} \\ \geq 0 & \text{if } t \geq \min\{D_1, D_2\} \end{cases}$$

Therefore, $J_1^*(x^0) = w^*(\min\{D_1, D_2\})$.
Case 2.2*:* If $P_1 \leq W \leq P_2$, then:

$$\dot{w}^*(t) \begin{cases} < 0 & \text{if } t \leq D_1 \\ \geq 0 & \text{if } D_1 \leq t \leq D_2 \wedge t \leq \frac{P_1 - r^0}{U_1} \\ < 0 & \text{if } D_1 \leq t \leq D_2 \wedge t > \frac{P_1 - r^0}{U_1} \\ \geq 0 & \text{if } t \geq D_2 \end{cases}$$

Therefore, $J_1^*(x^0) = \min\{w^*(D_1), w^*(D_2)\}$. By symmetry, the same will be true if $P_2 \leq W \leq P_1$.
Case 2.3*:* If $\max\{P_1, P_2\} \leq W$, then:

$$\dot{w}^*(t) \begin{cases} < 0 & \text{if } t \leq \min\{D_1, D_2\} \\ \geq 0 & \text{if } D_1 \leq t \leq D_2 \wedge t \leq \frac{P_1 - r^0}{U_1} \\ < 0 & \text{if } D_1 \leq t \leq D_2 \wedge t > \frac{P_1 - r^0}{U_1} \\ \geq 0 & \text{if } D_2 \leq t \leq D_1 \wedge t \leq \frac{P_2 - r^0}{U_1} \\ < 0 & \text{if } D_2 \leq t \leq D_1 \wedge t > \frac{P_2 - r^0}{U_1} \\ \geq 0 & \text{if } t \geq \max\{D_1, D_2\} \end{cases}$$

Again, $J_1^*(x^0) = \min\{w^*(D_1), w^*(D_2)\}$.

Overall, if we restrict our attention to Case 2 (where there is some hope that the system will be safe), the above relations indicate that:

$$J_1^*(x^0) = \min\{w^*(D_1), w^*(D_2)\}$$

In fact, some algebra reveals that for safety one needs:

$$w^0 \geq \max\left\{ \begin{array}{c} M_1 + W(D_1 - T_r) + \frac{U_1 T_r^2}{2} + r^0 T_r, \\ M_1 - P_1(D_2 - D_1) + W(D_2 - T_r) + \frac{U_1 T_r^2}{2} + r^0 T_r \end{array} \right\} \text{ if } T_r \leq D_1 \leq D_2$$

$$w^0 \geq \max\left\{ \begin{array}{c} M_1 + W(D_2 - T_r) + \frac{U_1 T_r^2}{2} + r^0 T_r, \\ M_1 - P_2(D_1 - D_2) + W(D_1 - T_r) + \frac{U_1 T_r^2}{2} + r^0 T_r \end{array} \right\} \text{ if } T_r \leq D_2 \leq D_1$$

$$w^0 \geq \max\left\{ \begin{array}{c} M_1 + \frac{U_1 D_1^2}{2} + r^0 D_1, \\ M_1 - P_1(D_2 - D_1) + W(D_2 - T_r) + \frac{U_1 T_r^2}{2} + r^0 T_r \end{array} \right\} \text{ if } D_1 \leq T_r \leq D_2$$

$$w^0 \geq \max\left\{ \begin{array}{c} M_1 + \frac{U_1 D_2^2}{2} + r^0 D_2, \\ M_1 - P_2(D_1 - D_2) + W(D_1 - T_r) + \frac{U_1 T_r^2}{2} + r^0 T_r \end{array} \right\} \text{ if } D_2 \leq T_r \leq D_1$$

$$w^0 \geq \max\left\{ \begin{array}{c} M_1 + \frac{U_1 D_1^2}{2} + r^0 D_1, \\ M_1 - P_1(D_2 - D_1) + \frac{U_1 D_2^2}{2} + r^0 D_2 \end{array} \right\} \text{ if } D_1 \leq D_2 \leq T_r$$

$$w^0 \geq \max\left\{ \begin{array}{c} M_1 + \frac{U_1 D_2^2}{2} + r^0 D_2, \\ M_1 - P_2(D_1 - D_2) + \frac{U_1 D_1^2}{2} + r^0 D_1 \end{array} \right\} \text{ if } D_2 \leq D_1 \leq T_r$$

■