A GAME THEORETIC APPROACH TO HYBRID SYSTEM DESIGN

by

John Lygeros, Datta N. Godbole, and Shankar Sastry

Memorandum No. UCB/ERL M95/77

12 October 1995

A GAME THEORETIC APPROACH TO HYBRID SYSTEM DESIGN

by

John Lygeros, Datta N. Godbole, and Shankar Sastry

Memorandum No. UCB/ERL M95/77

12 October 1995

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

A Game Theoretic Approach to Hybrid System Design *

John Lygeros, Datta N. Godbole and Shankar Sastry

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

Abstract

We present a design and verification methodology for hybrid dynamical systems. Our approach is based on optimal control and game theory. The hybrid design is seen as a game between two players. One is the disturbances that enter the dynamics. The disturbances can encode the actions of other agents (in a multi-agent setting), the actions of high level controllers or the usual unmodeled environmental disturbances. The second player is the control, which is to be chosen by the designer. The two players compete over a cost function that encodes the properties that the closed loop hybrid system needs to satisfy (e.g. safety). The control "wins" the game if it can keep the system "safe" for any allowable disturbance.

The solution to the game theory problem provides the designer with continuous controllers as well as sets of safe states where the control "wins" the game. The sets of safe sets can be used to construct an interface that guarantees the safe operation of the combined hybrid system. In addition to design, this optimal control methodology can also be used for the verification of hybrid systems as well as the generation of abstractions of the lower layer behavior in terms of the higher layer language (e.g. timed abstractions).

The motivating example for our work is Automated Highway Systems. We show how to cast the lower level, multi-agent control problem in the game theoretic setting and give an algorithm that can produce a safe design.

1 Introduction

The need for Hybrid Control

The demand for increased levels of automation and system integration have forced control engineers to deal with increasingly large and complicated systems. Recent technological ad-

*Research supported by the Army Research Office under grant DAAH 04-95-1-0588 and the PATH program, Institute of Transportation Studies, University of California, Berkeley, under MOU-135.

vances, such as faster computers, cheaper and more reliable sensors and the integration of control considerations in the product design and manufacturing process, have made it possible to extend the practical applications of control to systems that were impossible to deal with in the past. One of the main incentives to move into the area of large scale systems is financial: preliminary studies indicate that increased automation in air traffic management systems, highway systems, chemical process control, power generation and distribution etc. can lead to performance improvement in terms of fuel consumption, efficiency and environmental impact. The size of these businesses is such that minor changes in performance translate to large amounts of money gained or lost.

To deal with large, complex systems engineers are usually inclined to use a combination of continuous and discrete controllers. The reasons why continuous controllers are used are many:

- Interaction with the physical plant, through sensors and actuators, is essentially analog, i.e. continuous, from the engineering point of view.
- Continuous models have been developed, used and validated extensively in the past in most areas that interest control engineers (e.g. electromechanical systems, electromagnetic systems, etc.)
- Powerful control techniques have already been developed for many classes of continuous systems. Moreover, in conjunction with the reliable continuous models proofs of guaranteed performance can be obtained for these techniques.

An equally compelling case can be made in favor of discrete controllers, however:

- Discrete abstractions make it easier to manage the complexity of the system. It is not an accident that most of the work on discrete controllers started in the area of manufacturing, where complex systems were first encountered and modeled.
- Discrete models are easier to compute with, as all computers and algorithms are essentially discrete.
- Discrete abstractions make it easy to introduce linguistic and qualitative information in the controller design.

We will use the term "hybrid systems" to describe systems that incorporate both continuous and discrete dynamics. Engineers and computer scientists have been interested in the control of hybrid systems for some years now, but there is still a lot to be learned about them. This paper will attempt to present a rather general technique for designing hybrid controllers for complex systems.

Multi-Agent Scarce Resource Systems

A very interesting class of systems that are naturally suited for hybrid control are multiagent, scarce resource systems. Their common characteristic is that many agents are trying to make optimum use of a congested, common resource. For example, in highway systems, the vehicles

can be viewed as agents competing for the highway (which plays the role of the resource) while in air traffic management systems the aircraft compete for air space and runway space.

Typically in systems like these the optimum policy for each agent does not coincide with the "common good". Therefore, compromises need to be made. To achieve the common optimum we should ideally have a centralized control scheme that computes the global optimum and commands the agents accordingly. A solution like this may be undesirable, however, for a number of reasons:

- It is likely to be very computationally intensive, as a large centralized computer is needed to make all the decisions.
- It may be less reliable, as the consequences are likely to be catastrophic if the centralized controller is disabled.
- The information that needs to be exchanged may be too large for the available communication capabilities.
- The number of agents may be large and/or dynamically changing. This implies that the solution for the optimal control strategy may be hard to obtain and will have to be constantly updated.

If a completely decentralized solution is unacceptable and a completely centralized solution is prohibitingly complex or expensive, an in-between compromise will have to be obtained. Such a compromise will feature semiautonomous agent operation. In this case each agent is trying to optimize its own usage of the resource and coordinates with neighboring agents in case there is a conflict of objectives. Clearly such a solution is likely to be less efficient than centralized scheme and harder to implement, in a reliable fashion, than a decentralized scheme; it may, however, be the only feasible choice. It should be noted that semiautonomous agent control is naturally suited for hybrid designs. *At the continuous level each agent chooses it's own optimal strategy, while discrete coordination is used to resolve conflicts. This is the class of hybrid systems that we will be most interested in.*

Hybrid Control Methodologies

A common approach to the design of hybrid controllers involves independently coming up with a reasonable design for both the discrete and continuous parts. The combined hybrid controller is then put together by means of interfaces, and verification is carried out to ensure that it satisfies certain properties. Because of the complexity of the system, verification is usually done automatically, using some specialized computer program.

This approach has been motivated by the success of verification techniques for finite state machines and the fact that many hybrid designs already operating in practice need to be verified, without having to redesign them from scratch. Verification algorithms for finite state machines have been in use for years and efficient programs exist to implement them (COSPAN [1], HSIS [2], STATEMATE [3], etc.). They have proved very successful in discrete problems such as communication protocols [4] and software algorithms. The push towards stronger verification techniques has been in the direction of extending the standard finite state machine results to incorporate progressively more complicated continuous dynamics. The first

extension has been for systems with clocks [5] and multi-rate clocks [6]. Theoretical results have established conditions under which problems like these can be solved computationally and algorithms have been developed to implement the verification process (for example timed COSPAN [7] and KRONOS [8]). Verification of timed systems has proved useful in applications such as digital circuit verification [9] and real-time software [10]. Recently the theory has been extended to systems where the dynamics can be modeled by rectangular differential inclusions. The results indicate that, under certain conditions, automatic verification should also be possible for such systems [11], but most applications have been to academic examples rather than actual systems. To our knowledge the only computer package capable of dealing with differential inclusions, HyTech, is still under development [6].

Progress in the direction of automatic verification has been impeded by for two fundamental reasons. The first is **undecidability**. To guarantee that automatic verification algorithm will terminate in finite number of steps with an answer, the system needs to satisfy very stringent technical requirements. It can be shown [12] that relaxing any of these requirements makes the problem undecidable. The second problem is **computational complexity**. Even relatively simple hybrid systems lead to very large numbers of discrete states when looked at from the point of view of automatic verification. Even though efficient algorithms (that make use of heuristics and user input to facilitate the search) exist, the the problem may still be prohibitively large for current computers [10].

A different approach has been to design the hybrid controller so that performance is a-priori guaranteed [13, 14, 15]. This eases the requirements on verification somewhat as a large part of the complexity can be absorbed by careful design. The techniques presented in this paper fit in with this way of thinking. The plan is to start by modeling the systems dynamics at the continuous level. Two factors affect the system evolution at this level. The first is the **control**, that the designer has to determine. The second is the **disturbances** that enter the system, over which we assume no control. We will distinguish three classes of disturbances:

- **Class 1:** Exogenous signals, such as unmodeled forces and torques in mechanical systems, sensor noise, etc.
- **Class 2:** Unmodeled dynamics
- **Class 3:** The actions of other agents, in a multiagent setting.

Disturbances of Class 1 and 2 are standard in classical control theory. Class 3 will be the most interesting one from the point of view of hybrid control. Recall that at this stage we are merely modeling the plant, therefore we assume no cooperation between the agents. As a result, each agent views the actions of its neighbors as uncontrollable disturbances.

In the continuous domain specifications about the closed loop system can be encoded in terms of cost functions. Acceptable performance can be encoded by means of thresholds on the final cost. Our objective is to derive a continuous design for the control inputs that guarantees performance despite the disturbances. If it turns out that the disturbance is such that the specifications can not be met for any controller the design fails. The only way to salvage the situation is to somehow limit the disturbance. For disturbances of Class 3 this may be possible by means of communication and coordination between the agents. The objective

then is to come up with a discrete design that limits the disturbance so that a continuous design is feasible.

Summarizing, our approach to hybrid controller design consists of determining continuous control laws and conditions under which they satisfy the closed loop requirements. Then a discrete design is constructed to ensure that these conditions are satisfied. This process eliminates the need for automatic verification as the hybrid closed loop system is by design guaranteed to satisfy the specifications.

Game Theory

In this paper we will limit our attention to the design of continuous control laws and interfaces between these laws and the discrete world. The design of the continuous laws will be optimal with respect to the closed loop system requirements. An ideal tool for this kind of set up is game theory. In the game theoretic framework the control and the disturbances are viewed as adversaries in a game. The control seeks to improve system performance while the disturbance seeks to make it worse. Games like these do not necessarily have winners and loosers. If we set a threshold on the cost function to distinguish acceptable from unacceptable performance we can say that the control wins the game if the requirement is satisfied for any allowable disturbance, while the disturbance wins otherwise.

The principles involved in game theoretic design are very similar to the ones for optimal control. Very roughly, the designer has to find the best possible control and the worst possible disturbance. If the requirements are met for this pair, it is possible to obtain a satisfactory design (one such design is the "best possible" control). If the requirements are not satisfied the problem can not be solved as is, since there exists a choice of disturbance for which, no matter what the controller does, the closed loop system will fail to satisfy the requirements. In case of disturbances of Class 3 coordination will have to be used to limit the disturbance.

Game theoretic ideas have already been applied in this context to problems with disturbances of Class 1 and 2 and quadratic cost functions. The resulting controllers are the so called H_∞ or L_2 optimal controllers (see for example [16, 17]). We will try to extend these ideas to the multiagent, hybrid setting. In this paper we apply this idea to two examples, the train-gate controller and the problem of vehicle following on a highway.

Extensions: Verification & Abstraction

Optimal control and gaming ideas may prove useful in other hybrid settings. Here we touch upon two of them, verification and abstraction. Standard automatic verification techniques involve some form of exhaustive search, to verify that all possible runs of the systems satisfy a certain property. As discussed above this leads to undecidability and complexity problems. An optimal control approach to verification could be to obtain the worst possible run by solving an optimal control problem, and verifying that the property holds for this run; then it will also hold for all other runs.

Verification of closed loop hybrid systems is better suited for optimal control, rather than game theory, as one of the two players (the controller) has his strategy fixed a-priori. Therefore only the disturbances, trying to do their worst to upset the design, enter the picture.

[18] discusses the application of these ideas to the automated highway example. An interesting class of disturbances that need to be considered in this context is:

- **Class 4:** Commands from the discrete controller

From the point of view of the continuous system (where the optimal control problem is to be solved) these commands can be viewed as signals that make the continuous system switch between control laws, fixed points etc. Optimal control can be used to determine the discrete command sequences that force the continuous system to violate the performance specifications. If the discrete design is such that these command sequences are excluded then the hybrid design is verified. In this paper we use this technique to address a very simple verification example, the leaking gas burner.

The main advantage of the optimal control point of view to verification is that, by removing the requirement for an exhaustive search, the limitations of complexity and undecidability disappear. It is quite likely that many hybrid systems that are undecidable from the conventional verification point of view will be amenable to optimal control verification. On the down side, verification using optimal control requires a lot of input from the user. Moreover, optimal control problems will, in general, be very hard to solve analytically. It is therefore unlikely that this approach can be applied, at least by hand, to systems with more than a few discrete states. For the time being verification using optimal control is no match for conventional verification techniques for those systems to which the latter can be applied.

Optimal control ideas can also be used to generate abstractions of continuous system behavior in terms of discrete languages. For example optimal control can be used to obtain the minimum and maximum times that a hybrid system spends in each discrete state. These bounds can then be used as a rudimentary timed abstraction of the hybrid system. Again the requirement for designer input limits the complexity of the problems that can be handled analytically.

Limitations

The main limitation of the application of gaming and optimal control ideas to hybrid design is that the resulting problems are usually very hard to solve analytically. As a result efficient numerical algorithms are needed to apply these techniques to more complicated problems. In addition extensive designer input is needed during the controller design, verification or abstraction process. Hopefully it will be possible to automate the process, at least partially. However it is still likely that extensive engineering insight will be needed to solve any realistic problems.

This approach is best suited to address questions of "reachability". Problems like these can usually be cast as pursuit evasion problems in the gaming framework. Fortunately pursuit-evasion games have been extensively studied in the literature. Extensions to other important discrete questions such as fairness, liveness and cyclicity requirements should also be possible, however more work is needed in this direction.

2 Game Theoretic Framework

The preceding discussion illustrates that the design of hybrid controllers can be very subtle. The difficulties arize at the interaction between the discrete and continuous components. They are mainly due to inadequate abstraction of the continuous layer performance at the discrete level. This may result in the discrete controller issuing commands that are incompatible with the state of the continuous system. We will try to deal with this difficulty by using game theory to generate continuous controllers and consistent discrete abstractions for the resulting closed loop system. In the case where the continuous design has been carried out separately optimal control will be used for generating abstractions.

2.1 Plant Model

As our starting point we will use a rather general state space model for the continuous plant. Let $x(t) \in \mathbb{R}^n$ represent the state, $u(t) \in \mathbb{R}^m$ represent the input and $d(t) \in \mathbb{R}^p$ represent any disturbance that affects the dynamics, at time t.[1] The plant dynamics will be described by a differential equation and the value of the state at a given time, say t_0:

$$\dot{x}(t) = f(x(t), u(t), d(t), t) \tag{1}$$
$$x(t_0) = x^0 \tag{2}$$

The behavior of the system at time t is assumed to be monitored through a set of outputs $y(t) \in \mathbb{R}^q$. Their value depends on a map:

$$y(t) = h(x(t), u(t), d(t), t) \tag{3}$$

Physical considerations (such as actuator saturation, etc.) impose certain restrictions on the system evolution. We assume that these restrictions are encoded in terms of constraints on the state, inputs and disturbances.[2]

$$x() \in \mathcal{X} \subset PC^1(\mathbb{R}, \mathbb{R}^n) \tag{4}$$
$$u() \in \mathcal{U} \subset PC(\mathbb{R}, \mathbb{R}^m) \tag{5}$$
$$d() \in \mathcal{D} \subset PC(\mathbb{R}, \mathbb{R}^p) \tag{6}$$

Of particular interest is the case where the constraints can be encoded by requiring that the state, input and disturbance lie in a certain set for all times[3], that is for all $t \in \mathbb{R}$:

$$x(t) \in \mathbf{X} \subset \mathbb{R}^n \tag{7}$$
$$u(t) \in \mathbf{U} \subset \mathbb{R}^m \tag{8}$$
$$d(t) \in \mathbf{D} \subset \mathbb{R}^p \tag{9}$$

[1] The design methodology can be modified to apply to systems where the state, input and disturbance evolve in appropriate manifolds and/or the continuous dynamics are described by difference equations. It may also be possible to modify our approach so that it applies to systems for which the plant dynamics themselves are hybrid (e.g. hopping robots).

[2] $PC(\cdot, \cdot)$ denotes the set of piecewise continuous functions whereas $PC^1(\cdot, \cdot)$ represents the set of piecewise differentiable functions.

[3] Even though this class of constraints is easier to work with it excludes certain important cases such as non-holonomic and "isoperimetric" constraints. In its most general form a hybrid design formulation will have to account for constraints like these.

We assume that the differential equation 1 has unique solutions.

Both inputs and disturbances are exogenous functions of time. The difference is that the disturbances are assumed to be beyond the control of the designer and can take arbitrary values within the constraint set $\mathcal{D}$. As discussed in the introduction, the disturbances can represent a number of factors: unmodeled dynamics or environmental inputs, the actions of other agents or the actions of higher layers (in a multi-layered, hierarchical design). The designer's objective is to use the inputs to regulate the outputs, despite the actions of the disturbances. We will assume that the whole state is available for feedback. Extensions to the case of output feedback should also be possible.

In our framework for hierarchical control the desired properties, that the system trajectory needs to satisfy, will be specified by the discrete layers. The continuous layer will be responsible for the regulation, i.e. the selection of inputs u to try to achieve these properties. The interface provides communication between the two layers, providing the lower layer with the desired properties (in continuous terms) and the higher layer with feedback on whether the requirements can be met. Here we will primarily be concerned with the design of the continuous layer and the interface from continuous to discrete.

2.2 Discrete Layer

In this piece of work we will not deal at all with the design of the discrete layer. In terms of the continuous layer the outcome of the discrete design will be represented by:

- A sequence of desired way points x_i^d that should be tracked (i is an integer).
- A set of cost functions:
$$J_i : \mathbb{R}^n \times PC \times PC \longrightarrow \mathbb{R} \tag{10}$$
$i = 1, \ldots, N$ that encode desired properties
- A set of thresholds $C_i, i = 1, \ldots, N$ that specify acceptable limits on the cost functions. An acceptable trajectory must be such that $J_i(x^0, u, d) \leq C_i$ for all $i = 1, \ldots, N$

We assume that the cost functions are ordered in the order of decreasing importance. Qualitatively, the most important cost functions encode things such as safety, while the least important ones encode performance aspects such as resource utilization. The design should be such that the most important constraints are not violated in favor of the less important ones, in other words the design should lead to $J_i \leq C_i$ whenever possible, even if this means that $J_j > C_j$ for some $j > i$. It should be noted that at this stage we make no assumption about the high level design, or even the language used at the higher level. We are merely looking at the higher level from the point of view of the continuous plant, i.e. after the interface. Questions about how the high level objectives, which are usually given linguistically, get parsed to way points, cost functions and thresholds are not addressed.

2.3 Continuous Layer

We now present a technique for systematically constructing controllers which carry out the objectives set by the discrete layer and are optimal with respect to the given cost functions.

2.3.1 Design Principle

At the first stage we treat the design process as a two player, zero sum dynamic game with cost J_1. One player, the control u, is trying to minimize the cost, while the other, the disturbance d, is trying to maximize it. Assume that the game has a saddle point solution, i.e. there exist input and disturbance trajectories, u_1^* and d_1^* such that:

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

Consider the set:

$$V_1 = \{x \in \mathbb{R}^n | J_1^*(x) \leq C_1\}$$

This is the set of all initial conditions 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).

u_1^* can now be used as a control law. 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. u_1^* however does not take into account the requirements on the remaining J_i's. To include them in the design let:

$$\mathcal{U}_1(x^0) = \{u \in \mathcal{U} | J_1(x^0,u,d_1^*) \leq C_1\} \tag{11}$$

Clearly:

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

The set $\mathcal{U}_1(x^0)$ is the subset of admissible controls which guarantee that the requirements on J_1 are satisfied, whenever possible. Within this class of controls we would like to select the one that minimizes the cost function J_2. Again we pose the problem as a zero sum dynamic game between control and disturbance. Assume that a saddle solution exists, i.e. there exist u_2^* and d_2^* such that:

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

Consider the set:

$$V_2 = \{x \in \mathbb{R}^n | J_2^*(x) \leq C_2\}$$

As the minimax problem only makes sense when $\mathcal{U}_1(x^0) \neq \emptyset$ we assume that $V_2 \subset V_1$. V_2 represents the set of 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. To introduce the remaining cost functions to the design we again define:

$$\mathcal{U}_2(x^0) = \{u \in \mathcal{U}_1(x^0) | J_2(x^0,u,d_2^*) \leq C_2\} \tag{12}$$

i.e. the subset of admissible controls that satisfy the requirements on both J_1 and J_2 for any disturbance.

The process can be repeated for the remaining cost functions. At the $i+1^{\text{st}}$ step we are given a set of admissible controls $\mathcal{U}_i(x^0)$ and a set of initial conditions V_i such that for all $x^0 \in V_i$ there exists $u_i^* \in \mathcal{U}_i(x^0)$ such that for all $d \in \mathcal{D}$, $J_j(x^0, u_i^*, d) \leq C_j$, where $j = 1, \ldots, i$. Assume the two player, zero sum dynamic game for J_{i+1} has a saddle solution, u_{i+1}^*, d_{i+1}^*:

$$\begin{aligned} J_{i+1}^*(x^0) &= \max_{d \in \mathcal{D}} \min_{u \in \mathcal{U}_i(x^0)} J_{i+1}(x^0, u, d) \\ &= \min_{u \in \mathcal{U}_i(x^0)} \max_{d \in \mathcal{D}} J_{i+1}(x^0, u, d) \\ &= J_{i+1}(x^0, u_{i+1}^*, d_{i+1}^*) \end{aligned}$$

Define:

$$V_{i+1} = \{x \in \mathbb{R}^n | J_{i+1}^*(x) \leq C_{i+1}\}$$

and:

$$\mathcal{U}_{i+1}(x^0) = \{u \in \mathcal{U}_i(x^0) | J_{i+1}(x^0, u, d_{i+1}^*) \leq C_{i+1}\} \tag{13}$$

The process can be repeated until the last cost function. The result is a control law u_N^* and a set of initial conditions $V_N = V$ such that for all $x^0 \in V_N$ and for all $d \in \mathcal{D}$ $J_j(x^0, u_N^*, d) \leq C_j$ where $j = 1, \ldots, N$. The controller can be extended to values of the state in the complement of V using the following switching scheme:

$$u^*(x) = \begin{cases} u_N^*(x) & x \in V \\ u_{N-1}^*(x) & x \in V_{N-1} \setminus V \\ \cdots & \cdots \\ u_1^*(x) & x \in \mathbb{R}^n \setminus V_2 \end{cases} \tag{14}$$

The algorithm presented above is sound in theory but can run into technical difficulties when applied in practice:

1. There is no guarantee that the dynamic games will have a saddle solution.
2. There is no straight-forward way of computing $\mathcal{U}_i(x^0)$
3. There is no guarantee that the sets V_i (and consequently $\mathcal{U}_i(x^0)$) will be non-empty.

2.4 Interface

The sets V are such that for all initial conditions in them all requirements on system performance are guaranteed. These sets impose conditions that the discrete switching scheme needs to satisfy. The discrete layer should not issue a new command (way point) if the current state does not lie in the set V for the associated controller. Essentially these sets offer a way of consistently abstracting performance properties of the continuous layer.

It should be noted that, by construction, the sets V_i are nested. Therefore there is a possibility that an initial condition lies in some V_i but not in V. This implies that certain requirements on the system performance (e.g. safety) are satisfied, while others (e.g. efficient resource utilization) are not. This allows the discrete design some more freedom. It may, for

example, choose to issue a new command if it is dictated by safety, even though it violates the requirements of efficiency. This construction provides a convenient way of modeling gradual performance degradation, where lower priority performance requirements are abandoned in favor of higher priority ones.

The design of continuous controllers using game theory as well as the extraction of interfaces in terms of sets of guaranteed performance will be illustrated in the next section by means of a series of examples.

3 Examples

In this section we investigate how the approach developed in Section 2 can be useful in applications. We first consider a classic example from the timed automata verification literature, the "train gate controller". This example is ideal for our purposes because it can be easily cast into the game theoretic framework and the equations are simple enough for complete analysis to be carried out by hand. The second example is vehicle following on an automated highway. Here we proceed analytically as much as possible and use computational tools to complete the design. To simplify the notation, time dependency of the states inputs and outputs will be suppressed unless explicitly stated.

3.1 Train Gate Controller

3.1.1 Problem Statement

The train gate problem set up is shown in Figure 1. We will work on the problem formulation of [19]. For simplicity we will assume that the train is going around on a circular track of length L, where L is large enough to ensure adequate separation between consecutive train appearances. This assumption will be discussed further in Section 3.1.3. We will also assume that the train can be approximated by a point. It is easy to extend the analysis to trains of finite length.

The Train: The train moves clockwise around the track. Let:

$$x_2 \in \left[-\frac{L}{2}, \frac{L}{2}\right)$$

denote the position of the train, with the implicit assumption that x_2 wraps around at $L/2$. The details of the train dynamics are abstracted away by assuming that the train velocity is bounded, i.e.:

$$\dot{x_2} \in [v_1, v_2]$$

From the analysis it will become apparent that in order to guarantee that the problem is well defined we need to assume that:

$$0 < v_1 \leq v_2 < \infty$$

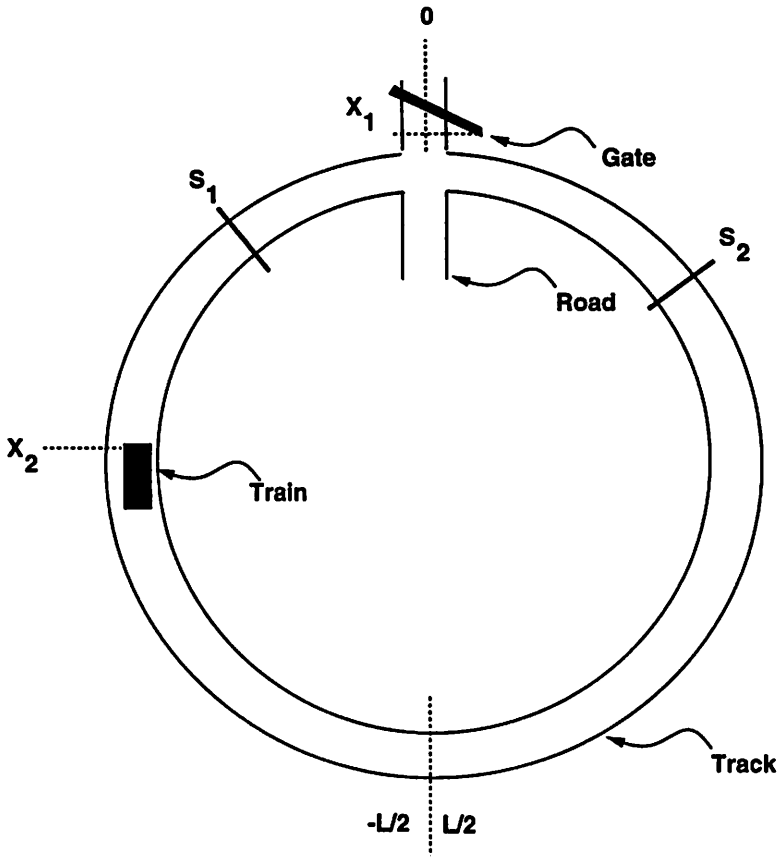

Figure 1: The train-gate set up

The Gate: The crossing is located at position $x_2 = 0$ on the train track. It is guarded by a gate that, when lowered, prevents the cars from crossing the tracks. Let:

$$x_1 \in [0°, 90°]$$

denote the angle of the gate in degrees. Assume that the gate dynamics are described by a first order ODE:

$$\dot{x}_1 = -\frac{1}{2}x_1 + u$$

where u is the input to be chosen by the designer of the gate controller.

The Sensor: The design of [19] is based on discrete sensor measurements. We will assume that there are two sensors located at distances S_1 and S_2 respectively on the track. The sensor at S_1 detects when the train is approaching the crossing, while the one at S_2 detects when it has moved away. From the analysis it will become apparent that in order for the control problem to have a solution we will need to assume that:

$$-\frac{L}{2} < S_1 < 0 < S_2 < \frac{L}{2}$$

Specifications: Two requirements are imposed on the design: safety and throughput. For safety it is required that the gate must be lowered (below a certain threshold) whenever the train reaches the crossing. This can be encoded as:

$$x_2(t) = 0 \Rightarrow x_1 < C_1$$

The value $C_1 = 10°$ will be used in subsequent analysis. For throughput it is required that the gate should be opened whenever it is safe to do so. This is done to maximize the number of cars that can cross the tracks.

3.1.2 Game Theoretic Formulation

The problem can immediately be cast in the game theoretic framework. The two players (agents) are the gate controller, u and the train speed (disturbance), d. The dynamics are linear in the state and affine in the two inputs. Let $x = [x_1\ x_2]^T \in \mathbb{R}^2$ denote the state:

$$\dot{x} = \begin{pmatrix} -1/2 & 0 \\ 0 & 0 \end{pmatrix} x + \begin{pmatrix} 1 \\ 0 \end{pmatrix} u + \begin{pmatrix} 0 \\ 1 \end{pmatrix} d \tag{15}$$

The state is constrained to lie in the set X:

$$x \in X = \{(x_1, x_2) \in \mathbb{R}^2 | x_1 \in [0°, 90°]; x_2 \in \left[-\frac{L}{2}, \frac{L}{2}\right)\} \subset \mathbb{R}^2$$

with the understanding that x_2 wraps around at $L/2$. The input is constrained to lie in the set U:

$$u \in U = [0, 45] \subset \mathbb{R} \tag{16}$$

while the disturbance lies in the set D:

$$d \in D = [v_1, v_2] \subset \mathbb{R}$$

The analysis will reveal that the dynamics and input constraints automatically guarantee the state constraints.

The two players compete over two cost functions J_1 and J_2. J_1 encodes the requirement for safety. Given initial conditions $x^0 \in X$ let:

$$T(x^0) = \min\{t \geq 0 | x_2(t) = 0\} \tag{17}$$

be the first time that the train reaches the crossing. Then the requirement for safety can be encoded by the cost function:

$$J_1(x^0, u, d) = x(T(x^0)) \leq C_1 \tag{18}$$

The requirement for throughput can be encoded by a number of cost functions. A simple one is:

$$J_2(x^0, u, d) = \int_0^\infty (90° - x_1(t))^2 dt$$

Minimizing J_2 implies that the gate is open for as long as possible.

3.1.3 Controller Design

The system dynamics given in equation (15) are simple enough to allow us to write an analytic expression for the value of the state at time t:

$$x_1(t) = e^{-t/2}x_1^0 + \int_0^t e^{-(t-\tau)/2}u(\tau)d\tau \quad (19)$$

$$x_2(t) = x_2^0 + \int_0^t d(\tau)d\tau \quad (20)$$

Lemma 1 *If $x^0 \in X$ and the input constraints of 16 are satisfied, $x(t) \in X$ for all $t \geq 0$.*

Proof: Under the wrap around assumption on x_2 the only thing we need to show is that $x_1(t) \in [0, 90]$ for all $t \geq 0$. Indeed:

$$\begin{aligned} x_1(t) &= e^{-t/2}x_1^0 + \int_0^t e^{-(t-\tau)/2}u(\tau)d\tau \\ &\leq e^{-t/2}x_1^0 + 45\int_0^t e^{-(t-\tau)/2}d\tau \\ &= e^{-t/2}x_1^0 + 90(1 - e^{-t/2}) \\ &\leq 90e^{-t/2} + 90(1 - e^{-t/2}) \\ &= 90 \\ x_1(t) &\geq e^{-t/2}x_1^0 \\ &\geq 0 \end{aligned}$$

□

Design for Safety

Our first goal is to find the safe set of initial conditions and the control that makes them safe. In other words we are looking for $x^0 \in X$ and u^* with $u^*(t) \in U$ such that for all d satisfying $d(t) \in D$, $J(x^0, u^*, d) \leq C_1$.

Because of the nature of the cost it is rather difficult to formulate this problem in the standard optimal control framework and tackle it with the usual tools (Dynamic Programming, Maximum Principle, etc.). Fortunately the dynamics are simple enough to allow us to guess a candidate saddle solution:

$$u^*(t) \equiv 0 \quad (21)$$

$$d^*(t) \equiv v_2 \quad (22)$$

The state trajectories for the candidate saddle strategy are:

$$\begin{aligned} x_1(t) &= e^{-t/2}x_1^0 \\ x_2(t) &= x_2^0 + v_2 t \end{aligned}$$

From (17) $x_2(T(x^0)) = 0$, hence $T(x^0) = -x_2^0/v_2$. Therefore, from (18):

$$J_1^*(x^0) = J_1(x^0, u^*, d^*) = e^{x_2^0/(2v_2)}x_1^0$$

Lemma 2 *(u^*, d^*) is globally a saddle solution.*

Proof: First fix $d = d^*$ and vary u. Then:

$$\begin{aligned} x_1(t) &= e^{-t/2}x_1^0 + \int_0^t e^{-(t-\tau)/2}u(\tau)d\tau \\ x_2(t) &= x_2^0 + v_2 t \end{aligned}$$

Hence $T(x^0) = -x_2^0/v_2$ again. Therefore:

$$\begin{aligned} J_1(x^0, u, d^*) &= e^{x_2^0/(2v_2)}x_1^0 + \int_0^{-x_2^0/v_2} e^{(x_2^0/v_2+\tau)/2}u(\tau)d\tau \\ &= J_1^*(x^0) + \int_0^{-x_2^0/v_2} e^{(x_2^0/v_2+\tau)/2}u(\tau)d\tau \\ &\geq J_1^*(x^0) \end{aligned}$$

since:

$$u(t) \in [0, 45] \Rightarrow \int_0^{-x_2^0/v_2} e^{(x_2^0/v_2+\tau)/2}u(\tau)d\tau \geq 0$$

Now fix $u = u^*$ and vary d. Then:

$$\begin{aligned} x_1(t) &= e^{-t/2}x_1^0 \\ x_2(t) &= x_2^0 + \int_0^t d(\tau)d\tau \end{aligned}$$

Let $T'(x^0)$ be the time that the train reaches the crossing.

$$x_2(T'(x^0)) = 0 \Rightarrow \int_0^{T'(x^0)} d(\tau)d\tau = -x_2^0$$

But $d(t) \in [v_1, v_2]$ therefore:

$$\begin{aligned} & \int_0^t d(\tau)d\tau \leq \int_0^t v_2 d\tau = v_2 t \\ \Rightarrow\ & T'(x^0) \geq -x_2^0/v_2 = T \\ \Rightarrow\ & J_1(x^0, u^*, d) = e^{-T'(x^0)/2}x_1^0 \leq e^{x_2^0/(2v_2)}x_1^0 \\ \Rightarrow\ & J_1(x^0, u^*, d) \leq J_1^*(x^0) \end{aligned}$$

Summarizing, for any admissible (u, d):

$$J_1(x^0, u^*, d) \leq J_1^*(x^0) \leq J_1(x^0, u, d^*) \tag{23}$$

Therefore (u^*, d^*) is a global saddle solution to the game. □

The saddle solution can now be used to calculate the safe set of initial conditions:

Lemma 3 *The set of safe initial conditions is:*

$$V = \left\{ x^0 \in X | x_2^0 > 0 \text{ or } x_2^0 \leq 2v_2 \ln(\frac{C_1}{x_1^0}) \right\} \tag{24}$$

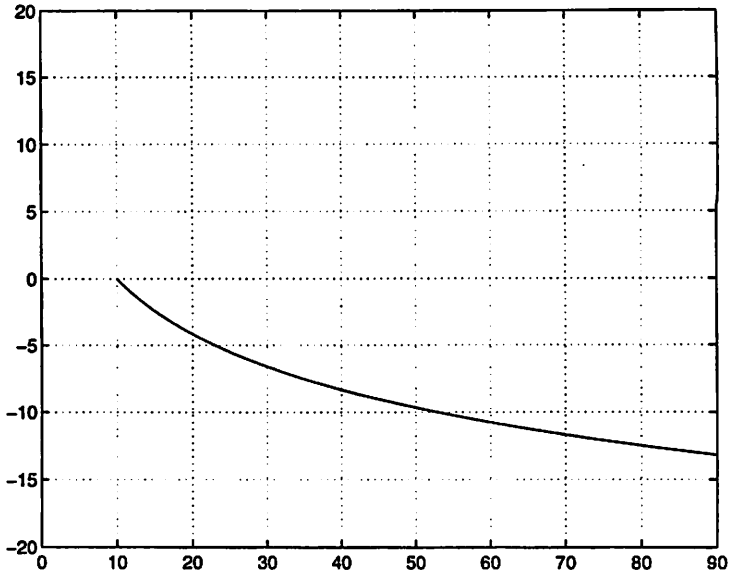

Figure 2: Safe Set of Initial Conditions

Proof: For sufficiently large values of L, all initial conditions with $x_2^0 > 0$ are safe (the train has already passed the crossing). For $x_2^0 \leq 0$ safety is equivalent to:

$$J_1^*(x^0) = e^{x_2^0/(2v_2)} x_1^0 \leq C_1$$

Since v_2 and x_1^0 are positive and the exponential is monotone this is equivalent to:

$$x_2^0 \leq 2v_2 \ln(\frac{C_1}{x_1^0})$$

□

The safe set for $C_1 = 10°$ and $v_2 = 3m/s$ is shown in Figure 2.

Design for Throughput

As:

$$J_2(x^0, u, d) = \int_0^\infty (90° - x_1(t))^2 dt$$

and, by Lemma 1, $x_1(t) \leq 90°$, maximizing throughput (minimizing J_2) is equivalent to maximizing $x_1(t)$. From the proof of Lemma 1 this is equivalent to setting:

$$u(t) \equiv 45 \tag{25}$$

"Optimal" Controller

The optimal controller can be obtained by combining the designs for safe and efficient operation. As safety takes precedence over efficiency the resulting controller will be:

$$u = \begin{cases} 0 & x \in S^c \\ 45 & x \in S \end{cases} \tag{26}$$

where $S = \text{interior}(V)$.

It should be noted that, by design, the controller of 26 will be safe. Moreover any controller which uses $\hat{S} \subset S$ in the place of S will also be safe, but not as efficient in terms of throughput. These observations are summarized below:

Theorem 1 *A switching controller of the form of (26) with switching taking place at $\hat{S}$ will be safe if $\hat{S} \subset S$ and $x^0 \in V$.*

Proof: The theorem follows directly from Lemmas 1, 2, 3 and the fact that $\hat{S} \subset S \subset V$. □

If discontinuous controls are undesirable, the controller can be made smooth by applying:

$$u = (1 - f(x))45$$

where f is any smooth function satisfying:

$$\begin{aligned} f : \mathbb{R}^2 &\longrightarrow \mathbb{R} \\ x &\longmapsto \begin{cases} 0 & x \in \hat{S} \\ 1 & x \in \hat{S}^c \end{cases} \end{aligned}$$

As above, the resulting controller will be safe if $\hat{S} \subset S$.

Discrete Controller

Due to its bang-bang nature, the optimal controller can easily be implemented by a discrete scheme, using the discrete sensor and an appropriate actuator as an interface.

Theorem 2 *The discrete control scheme will be safe if:*

$$\begin{aligned} S_1 &\leq 2v_2 \ln(\frac{C_1}{90^\circ}) \\ S_2 &> 0 \end{aligned}$$

Proof: When viewed as an input-output system in the continuous domain, the combination of the discrete controller and the interface looks like:

$$u = \begin{cases} 0 & S_1 \leq x_2 < S_2 \\ 45 & \text{otherwise} \end{cases} \tag{27}$$

Here we assume that all discrete transitions take place instantaneously. Let $\hat{S} = \{x \in X | S_1 \leq x_2 < S_2\}$. By Theorem 1 the controller will be safe if $\hat{S} \subset S$, i.e. $S_2 > 0$ and $S_1 \leq 2v_2 \ln(\frac{C_1}{x_1^0})$ for all $x_1^0 \in [0^\circ, 90^\circ]$. As the logarithm function is monotone, the above conditions are the same as the ones in the theorem statement. □

The Possible Role of Coordination

What if the sensors were placed improperly, for example $S_1 > 2v_2 \ln(\frac{C_1}{90^\circ})$. The above analysis can not guarantee a safe controller. One solution may be to modify the hardware and provide additional sensors. An alternative would be to obtain a promise from the train that it will slow down once it enters the sensor range, i.e. promise that $\dot{y} \in [v_1, v_2']$ with $v_2' < v_2$. For appropriate choices of v_2' (in particular $v_2' \leq \frac{S_1}{\ln(C_1/90^\circ)}$) safe operation with a discrete controller will still be possible. This example indicates how inter-agent communication (which can be used to provide such promises) can bias the game in the controllers' favor and help the designer produce an acceptable design.

Extensions

The discrete design can easily be modified to deal with multiple trains, by introducing a counter between the sensor and the controller. The counter is incremented whenever a train crosses S_1 and is decremented whenever one crosses S_2. The state *Train Far* is interpreted as the counter reading 0. Provided that the conditions of Theorem 1 are met and all initial conditions are chosen appropriately the resulting closed loop design will be safe.

With small changes in the analysis, safe designs can also be obtained for systems with delays (e.g. between the occurrence of an event and sensing it, and between command and execution). With a bit more work designs with the additional constraint $\ddot{y} \in [a_1, a_2]$ can also be obtained. We will forgo these calculations in favor of the more interesting and challenging vehicle following calculation.

3.2 Vehicle Following

3.2.1 Problem Statement

Consider two vehicles (labeled A and B) moving along a single lane highway (Figure 3). Assume that the vehicles have lengths L_A and L_B and let x_A and x_B denote their positions with respect to a fixed reference on the road. Assume that vehicle B is leading, i.e. $x_B > x_A > 0$. The problem we are interested in is the vehicle following problem: we assume no control over vehicle B and try to control vehicle A.

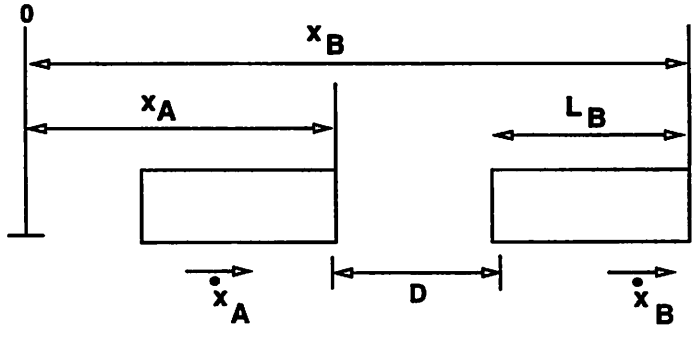

Figure 3: Vehicle Following

Leading Vehicle: The dynamics of the leading vehicle will be abstracted by a second order ordinary differential equation:

$$\ddot{x}_B = d$$

with $d(t) \in [d_{min}, d_{max}] \subset \mathbb{R}$ and $-\infty < d_{min} \leq 0 \leq d_{max} < \infty$. The values of d_{min} and d_{max} are dictated by the road and vehicle conditions. We will use $d_{min} = -5ms^{-2}$ and $d_{max} = 3ms^{-2}$. This abstraction is justified by the laws of motion and the assumed sensor arrangement to be discussed shortly.

To make the problem more realistic, a restriction on the speed of vehicle B is imposed:

$$\dot{x}_B \in [v_{min}, v_{max}]$$

For highway operation it is assumed that vehicles will not be allowed to go backwards, therefore $v_{min} = 0$ will be used. v_{max} is imposed by engine limitations. One objective of the controllers we design will be fuel efficiency. As a consequence the engine will not have to be pushed to its limits for maximum speed and therefore v_{max} will not feature in the calculations. Therefore, we will assume $v_{max} = \infty$ to simplify the analysis.

Trailing Vehicle: The dynamics of the trailing vehicle will be approximated by a third order ordinary differential equation:

$$\dddot{x}_A = b_A(\dot{x}_A, \ddot{x}_A) + a_A(\dot{x}_A)v_A$$

a_A and b_A are complicated nonlinear functions of the state with $a_A(\dot{x}_A) \neq 0$. The first two derivatives on x_A arise from the laws of motion. The third has to do with the operation of the engine, which can be modeled by the throttle angle acting through an integrator on some nonlinear dynamics (involving engine time constants, aerodynamic drag, etc.). For our purposes the details of the nonlinear functions b_A and a_A are not important. Following the designs of [20], we will assume that feedback linearization has already been carried out, i.e.:

$$\begin{aligned} v_A(t) &= \frac{-b_A(\dot{x}_A, \ddot{x}_A) + u}{a_A(\dot{x}_A)} \\ \dddot{x}_A &= u \end{aligned}$$

We will design controllers for the resulting linear dynamics.

As for the leading vehicle we assume that the dynamics are constrained by the engine, tire and road conditions. More specifically it is required that:

$$\begin{aligned} \dot{x}_A &\in [v_{min}, v_{max}] = [0, \infty)ms^{-1} \\ \ddot{x}_A &\in [a_{min}, a_{max}] \approx [d_{min}, d_{max}] \\ u(t) &\in [j_{min}, j_{max}] \end{aligned}$$

The design will use $a_{min} = -5ms^{-2}, a_{max} = 3ms^{-2}, j_{min} = -50ms^{-3}$ and $j_{max} = 50ms^{-3}$.

Sensors: Based on current technology, it is assumed that vehicle A is equipped with sensors that can measure its own velocity and acceleration, as well as relative position and velocity with respect to vehicle B. We will assume that the acceleration of vehicle B can not be measured and is not communicated to vehicle A (as it is, for example, for the platoon following scenario of [21]).

The vehicles are also equipped with communication devices. It is assumed that they will only be used to exchange discrete messages for coordinating the agent operation. In case the vehicle is equipped with multiple, redundant sensors or communication capabilities it will be assumed that the necessary sensor fusion has been carried out before hand and the designer has access to a unique and accurate measurement.

Specifications: The objective is to design a controller for vehicle A. The requirements we impose on the design are safety, passenger comfort and efficiency. It is assumed that safety takes precedence over the other two requirements. Comfort and efficiency will be treated as equally important and a compromise between them will be sought. Safety will be assumed to mean no collision between vehicles A and B, at any time[4]. For passenger comfort it is required that the input u is kept "as small as possible". In the transportation literature, $\dddot{x}_A \leq 2.5ms^{-3}$ is often quoted as the limit for a comfortable ride. Finally for efficiency it is required that all maneuvers (for example convergence of the vehicle spacing to a desired value) take place "as quickly as possible". All the above statements are quantified in the next section.

3.2.2 Game Theoretic Formulation

Both the design specifications and the system dynamics are independent of the absolute vehicle position. To completely remove the absolute position from the problem we introduce a new variable to measure the spacing between vehicles A and B:

$$D = x_B - x_A - L_B$$

All pertinent information can now be encoded by the state vector:

$$x = \begin{bmatrix} \dot{x}_A \\ \ddot{x}_A \\ D \\ \dot{D} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \in \mathbb{R}^4$$

From Section 3.2.1 the dynamics are:

$$\begin{aligned} \dot{x} &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} d \\ &= Ax + Bu + Dd \\ x(0) &= x^0 \end{aligned}$$

[4]Different definitions of safety, for example no high relative velocity collisions can also be accommodated in this framework

For the vehicle following problem we are interested in regulating the spacing and relative velocity to a desired fixed point. This requirement can be encoded in terms of two outputs:

$$\begin{aligned} y &= \begin{bmatrix} D \\ \dot{D} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x \\ &= Cx \end{aligned}$$

Note that the assumed sensor arrangement can provide full state measurements.

To complete the picture we also need to encode all the constraints in the new coordinates. From the discussion in Section 3.2.1 it is required that, for all t:

$$\begin{aligned} x(t) \in X &= \left\{ x \in \mathbb{R}^4 | x_1 \in [v_{min}, v_{max}], x_2 \in [a_{min}, a_{max}], x_4 + x_1 \in [v_{min}, v_{max}] \right\} \\ u(t) \in U &= [j_{min}, j_{max}] \\ d(t) \in D &= [d_{min}, d_{max}] \end{aligned}$$

The analysis can be greatly simplified if the input constraints are modified somewhat to guarantee that the state constraints are satisfied. In particular we will assume that:

$$u(t) \in \begin{cases} [j_{min}, j_{max}] & \text{if } x_2 \in (a_{min}, a_{max}) \\ [0, j_{max}] & \text{if } x_2 = a_{min} \\ [j_{min}, 0] & \text{if } x_2 = a_{max} \end{cases} \tag{28}$$

$$d(t) \in \begin{cases} [d_{min}, d_{max}] & \text{if } x_4 + x_1 \in (v_{min}, v_{max}) \\ [0, d_{max}] & \text{if } x_4 + x_1 = v_{min} \\ [d_{min}, 0] & \text{if } x_4 + x_1 = v_{max} \end{cases} \tag{29}$$

This will ensure that the constraints on x_2 and $x_1 + x_4$ will never be violated if $x^0 \in X$.

The only state constraint that we have to worry about is $x_1 \in [v_{min}, v_{max}]$. For the reasons discussed above we will not be concerned with the upper bound too much. We can work around the lower bound by assuming that $\ddot{x}_A = x_2$ becomes zero when $x_1 = v_{min}$ is reached (recall that when x_1 reaches v_{min}, $x_2 \leq 0$). This modification is not as far fetched as it may seem at first, for two reasons:

1. The brake input does not have to go through the extra integration of the engine throttle input. It effectively acts directly as the acceleration input. Even though the engine can be used for deceleration, physical considerations imply that the brakes have to be used if the vehicle is to stop completely in which case, x_2 can be considered as playing the role of the input (instead of $\dot{x}_2$).

2. What happens after x_1 reaches x_{min} is not very important, at least from the safety point of view. Recall that the constraint (29) guarantees that $x_B \geq v_{min} = 0$. Therefore, if no collision occurs until $x_1 = 0$ none is going to occur from then on, unless vehicle A starts accelerating again.

The two players are the unmeasured acceleration, d, of vehicle B and the controller, u, of vehicle A. They compete in three fronts over the cost functions:

1. **Safety:**

$$J_1(x^0, u, d) = -\inf_{t \geq 0} x_3(t) \tag{30}$$

A safe maneuver is one where:

$$J_1(x^0, u, d) \leq C_1 = 0\ m$$

Allowing $J_1 = 0$ meters makes the limiting case (where the vehicles just touch with zero relative velocity) acceptable.

2. **Comfort:**

$$J_2(x^0, u, d) = \sup_{t \geq 0} |u(t)| \tag{31}$$

A comfortable maneuver is one where:

$$J_2(x^0, u, d) \leq C_2 = 2.5 m s^{-3}$$

3. **Efficiency:**

$$J_3(x^0, u, d) = \int_0^\infty (y(\tau) - y_d)^T P (y(\tau) - y_d) d\tau \tag{32}$$

where y_d is the desired fixed point for a given maneuver and P is positive definite.

The solution to the system equations can be obtained using the variation of constants formula [22]:

$$x(t) = e^{At}x^0 + \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau + \int_0^t e^{A(t-\tau)}Dd(\tau)d\tau$$

Using the formula:

$$e^{At} = I + At + \frac{(At)^2}{2!} + \ldots$$

and the fact that A is nilpotent ($A^3 = 0$) we obtain:

$$e^{At} = \begin{bmatrix} 1 & t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -t^2/2 & 1 & t \\ 0 & -t & 0 & 1 \end{bmatrix}$$

This leads to:

$$x(t) = \begin{bmatrix} x_1^0 + tx_2^0 \\ x_2^0 \\ -t^2x_2^0/2 + x_3^0 + tx_4^0 \\ -tx_2^0 + x_4^0 \end{bmatrix} + \int_0^t \begin{bmatrix} t - \tau \\ 1 \\ -(t-\tau)^2/2 \\ -t + \tau \end{bmatrix} u(\tau)d\tau + \int_0^t \begin{bmatrix} 0 \\ 0 \\ t - \tau \\ 1 \end{bmatrix} d(\tau)d\tau \tag{33}$$

This equation is valid, under the assumed constraints (28, 29), as long as $x_1(t) \in [v_{min}, v_{max}]$.

3.2.3 Design for Safety

As for the train gate problem, we will start by guessing a saddle solution. Let:

$$T_1 = \frac{a_{min} - x_2^0}{j_{min}} \qquad T_2 = -\frac{x_1^0 + x_4^0}{d_{min}}$$

Consider the candidate saddle strategy:

$$u^*(t) = \begin{cases} j_{min} & \text{if } t \leq T_1 \\ 0 & \text{if } t > T_1 \end{cases} \tag{34}$$

$$d^*(t) = \begin{cases} d_{min} & \text{if } t \leq T_2 \\ 0 & \text{if } t > T_2 \end{cases} \tag{35}$$

In other words, both A and B try to come to a stop as quickly as possible, under the given constraints.

For the candidate saddle solution $x(t)$ can be written explicitly:

- If $0 \leq t \leq \min\{T_1, T_2\}$:

$$x(t) = e^{At}x^0 + \begin{bmatrix} t^2/2 \\ t \\ -t^3/6 \\ -t^2/2 \end{bmatrix} j_{min} + \begin{bmatrix} 0 \\ 0 \\ t^2/2 \\ t \end{bmatrix} d_{min}$$

- If $T_1 \leq t \leq T_2$:

$$x(t) = e^{At}x^0 + \begin{bmatrix} tT_1 - T_1^2/2 \\ T_1 \\ -T_1^3/6 + tT_1(T_1 - t)/2 \\ -tT_1 + T_1^2/2 \end{bmatrix} j_{min} + \begin{bmatrix} 0 \\ 0 \\ t^2/2 \\ t \end{bmatrix} d_{min}$$

- If $T_2 \leq t \leq T_1$:

$$x(t) = e^{At}x^0 + \begin{bmatrix} t^2/2 \\ t \\ -t^3/6 \\ -t^2/2 \end{bmatrix} j_{min} + \begin{bmatrix} 0 \\ 0 \\ tT_2 - T_2^2/2 \\ T_2 \end{bmatrix} d_{min}$$

- If $t \geq \max\{T_1, T_2\}$:

$$x(t) = e^{At}x^0 + \begin{bmatrix} tT_1 - T_1^2/2 \\ T_1 \\ -T_1^3/6 + tT_1(T_1 - t)/2 \\ -tT_1 + T_1^2/2 \end{bmatrix} j_{min} + \begin{bmatrix} 0 \\ 0 \\ tT_2 - T_2^2/2 \\ T_2 \end{bmatrix} d_{min}$$

Let T_3 be the stopping time for vehicle A. Then:

- If $0 \leq T_3 \leq T_1$:

$$T_3^2 j_{min}/2 + T_3 x_2^0 + x_1^0 = 0$$
$$\Rightarrow \quad T_3 = \frac{-x_2^0 - \sqrt{(x_2^0)^2 - 2j_{min}x_1^0}}{j_{min}}$$

Note that the second solution would lead to $T_3 < 0$ (assuming that $j_{min} < 0$ and $x_1^0 \geq 0$).

- If $T_1 \leq T_3$:

$$(T_1 T_3 - T_1^2) j_{min} + T_3 x_2^0 + x_1^0 = 0$$
$$\Rightarrow \quad T_3 = \frac{(a_{min} - x_2^0)^2 - 2j_{min}x_1^0}{2j_{min}a_{min}}$$

From the discussion at the end of Section 3.2.2 the times of interest are now restricted to the interval $t \in [0, T_3]$. A simple calculation shows that:

Lemma 4 *If $x^0 \in X$ then $x(t) \in X$ for all $t \in [0, T_3]$.*

To calculate $J_1(x^0, u^*, d^*)$ recall that $x_3(t)$ is a differentiable function of time, with derivative $x_4(t)$, defined on a compact interval $[0, T_3]$. Therefore:

Lemma 5 *There exist $\hat{T} \in [0, T_3]$ such that:*

$$J_1(x^0, u^*, d^*) = -x_3(\hat{T})$$

Moreover:

$$\hat{T} \in \{0, T_3, \{T \in (0, T_3) | x_4(T) = 0\}\}$$

The calculations for analytically determining $J_1^*(x^0) = J_1(x^0, u^*, d^*)$ from this lemma are rather messy. However if the set of times T where $x_4(T) = 0$ is finite we can easily carry out the calculation numerically. After a few steps of algebra we can determine that:

- For $0 \leq T \leq \min\{T_1, T_2\}$:

$$T = \frac{d_{min} - x_2^0 \pm \sqrt{(d_{min} - x_2^0)^2 + 2j_{min}x_4^0}}{j_{min}}$$

- If $T_1 \leq T \leq T_2$:

$$T = \frac{x_4^0 + T_1^2 j_{min}/2}{a_{min} - d_{min}}$$

- If $T_2 \leq T \leq T_1$:

$$T = \frac{-x_2^0 \pm \sqrt{(x_2^0)^2 + 2j_{min}(T_2 d_{min} + x_4^0)}}{j_{min}}$$

- If $T \geq \max\{T_1, T_2\}$:

$$T = \frac{x_4^0 + T_1^2 j_{min}/2 + T_2 d_{min}}{a_{min}}$$

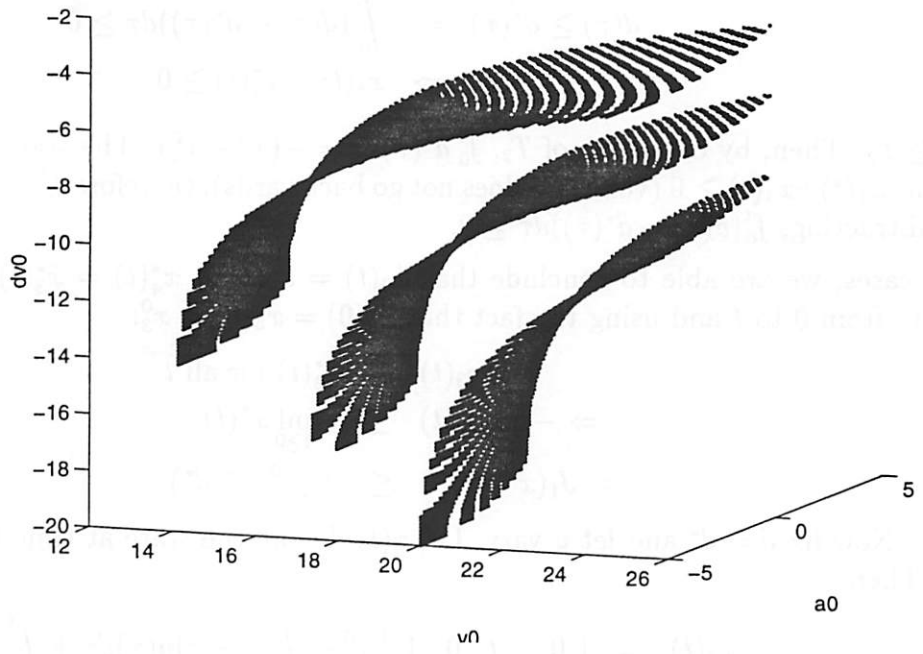

Figure 4: Safe set of initial conditions for $x_3^0 = 15, 20$ and 25 meters

A computer program was written to calculate $J_1^*(x^0)$ for various values of x^0 and the constants given in Section 3.2.1. It should be noted that x_3^0 only enters the calculations as a constant offset on $x_3(t)$. Figure 4 shows the surface where $J_0^*(x^0) = 0$ for some values of x_3^0. Any initial condition on or above these surfaces will not lead to a collision under the strategy (u^*, d^*).

To complete the safety calculation we need to show:

Lemma 6 *(u^*, d^*) is globally a saddle solution for cost $J_1(x^0, u, d)$.*

Proof: For (u^*, d^*) to be a saddle point we need to show that a unilateral change in strategy leaves the player who decided to change worse off. Let $x^*(t)$ denote the state at time t under the inputs (u^*, d^*).

$$x_3^*(t) = \begin{bmatrix} 0 & -t^2/2 & 1 & t \end{bmatrix} x^0 - \int_0^t \frac{(t-\tau)^2}{2} u^*(\tau) d\tau + \int_0^t \frac{t-\tau}{2} d^*(\tau) d\tau$$

Fist fix u^* and allow d to vary. Let $x(t)$ denote the state at time t under the inputs (u^*, d). Then:

$$\begin{aligned} x_4^*(t) &= \begin{bmatrix} 0 & -t & 0 & 1 \end{bmatrix} x^0 - \int_0^t (t-\tau) u^*(\tau) d\tau + \int_0^t d^*(\tau) d\tau \\ x_4(t) &= \begin{bmatrix} 0 & -t & 0 & 1 \end{bmatrix} x^0 - \int_0^t (t-\tau) u^*(\tau) d\tau + \int_0^t d(\tau) d\tau \\ \Rightarrow x_4(t) - x_4^*(t) &= \int_0^t (d(\tau) - d^*(\tau)) d\tau \end{aligned}$$

We need to distinguish two cases:

1. $t \leq T_2$. The bounds on d imply that:

$$\begin{aligned} d(\tau) \geq d^*(\tau) &\Rightarrow \int_0^t (d(\tau) - d^*(\tau))d\tau \geq 0 \\ &\Rightarrow x_4(t) - x_4^*(t) \geq 0 \end{aligned}$$

2. $t \geq T_2$. Then, by definition of T_2, $\int_0^t d^*(\tau)d\tau = -(x_1^0 + x_4^0)$. The state constraints imply that $x_1(t)+x_4(t) \geq 0$ (vehicle B does not go backwards), therefore $\int_0^t d(\tau)d\tau \geq -(x_1^0+x_4^0)$. Subtracting, $\int_0^t (d(\tau) - d^*(\tau))d\tau \geq 0$.

In both cases, we are able to conclude that $\dot{x}_3(t) = x_4(t) \geq x_4^*(t) = \dot{x}_3^*(t)$. Integrating this inequality from 0 to t and using the fact that $x_3^*(0) = x_3(0) = x_3^0$:

$$\begin{aligned} x_3(t) &\geq x_3^*(t) \text{ for all } t \\ \Rightarrow -\inf_{t\geq 0} x_3(t) &\leq -\inf_{t\geq 0} x_3^*(t) \\ \Rightarrow J_1(x^0, u^*, d) &\leq J_1(x^0, u^*, d^*) \end{aligned} \tag{36}$$

Now fix $d = d^*$ and let u vary. Let $x(t)$ denote the state at time t under the inputs (u, d^*). Then:

$$\begin{aligned} x_4(t) &= \begin{bmatrix} 0 & -t & 0 & 1 \end{bmatrix} x^0 - \int_0^t (t-\tau)u(\tau)d\tau + \int_0^t d^*(\tau)d\tau \\ \Rightarrow x_4(t) - x_4^*(t) &= \int_0^t (t-\tau)(u^*(\tau) - u(\tau))d\tau \end{aligned}$$

As above we need to distinguish two cases:

1. $t \leq T_1$. The bounds on u imply that:

$$\begin{aligned} u(\tau) \geq u^*(\tau) &\Rightarrow \int_0^t (t-\tau)(u^*(\tau) - u(\tau))d\tau \leq 0 \\ &\Rightarrow x_4(t) - x_4^*(t) \leq 0 \end{aligned}$$

2. $t \geq T_1$. Recall that $d^*(t)$ is piecewise constant, with a discontinuity at T_2 (which may be either greater or less than T_1). Therefore $x_4(t)$ and $x_4^*(t)$ are piecewise differentiable, with derivatives $-x_2(t)$ and $-x_2^*(t)$ respectively. By definition of T_1, $x_2^*(t) = a_{min} \leq x_2(t)$ in the interval of interest. Therefore, as $x_4(0) = x_4^*(0) = x_4^0$, $x_4^*(t) \geq x_4(t)$.

In both cases $\dot{x}_3(t) = x_4(t) \leq x_4^*(t) = \dot{x}_3^*(t)$. Using the fact that $x_3^*(0) = x_3(0) = x_3^0$ and integrating:

$$\begin{aligned} x_3(t) &\leq x_3^*(t) \text{ for all } t \\ \Rightarrow -\inf_{t\geq 0} x_3(t) &\geq -\inf_{t\geq 0} x_3^*(t) \\ \Rightarrow J_1(x^0, u, d^*) &\geq J_1(x^0, u^*, d^*) \end{aligned} \tag{37}$$

Combining inequalities 36 and 37:

$$J_1(x^0, u^*, d) \leq J_1^*(x^0) \leq J_1(x^0, u, d^*) \tag{38}$$

for all d and u. By definition, (u^*, d^*) is a saddle solution. □

Lemma 6 implies that the surfaces of Figure 4 are 2D slices of the 3D boundary of the safe set $V \subset X$ for various values of x_3^0.

3.2.4 Completing the Design

The analysis of the previous section provides a design that will satisfy the requirements for safety. By construction any initial condition in the safe set (on or above the corresponding surface of Figure 4) will not lead to a collision, if the control of equation 34 is applied. In the interior of the safe set, where safety is guaranteed, the other performance aspects (passenger comfort and efficiency) also become important. In order to complete the continuous design we now need to come up with a controller that optimizes with respect to these performance aspects in the interior of the safe set.

The safe set also provides guidelines for the discrete design. The discrete level should be such that initial conditions outside the safe set are never encountered. Such initial conditions may be caused for example in a multilane highway, when vehicle B changes lane in front of vehicle A. The safe set provides requirements on the vehicle states for a safe lane change to take place. It is up to the discrete layer to guarantee that these requirements are satisfied.

3.3 The Leaking Gas Burner

To illustrate how optimal control ideas can be used to carry out verification consider the "leaking gas burner" example of [6]. Even though this problem is rather simple and amenable to other techniques for verification it highlights some of the advantages of verifying using optimal control.

3.3.1 Problem Statement

The gas burner has two states, "normal" and "leaking". There are two rules governing the leaking process:

1. Leaks are detected and repaired within D_1 seconds
2. No leak occurs within D_2 seconds of the last leak being fixed.

The gas burner can be modeled by the hybrid automaton of Figure 5.

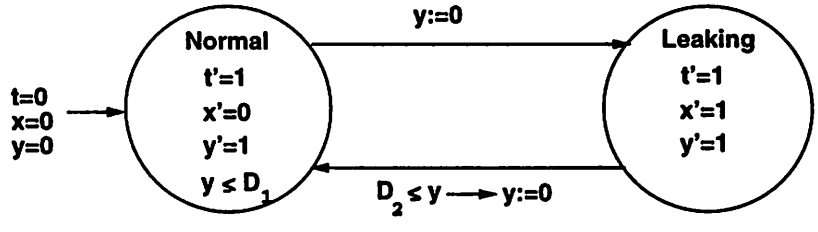

Figure 5: The leaking gas burner hybrid automaton

The requirement is that the accumulated time of leakage does not exceed $\alpha\%$ in any interval larger that $T > 0$ seconds, i.e.

$$t \geq T \Rightarrow x < \alpha t$$

3.3.2 Optimal Control Formulation

To cast the verification as an optimal control define:

$$x([a,b]) \in [0,1]$$

to be the percentage of leaking time in the interval $[a,b] \subset \mathbb{R}$ with $0 \leq a < b$. Let:

$$d : [a,b] \longrightarrow \{0,1\}$$

be the leaking times, with $d(t) = 0$ being "normal" and $d(t) = 1$ "leaking". A typical d is shown in Figure 6.

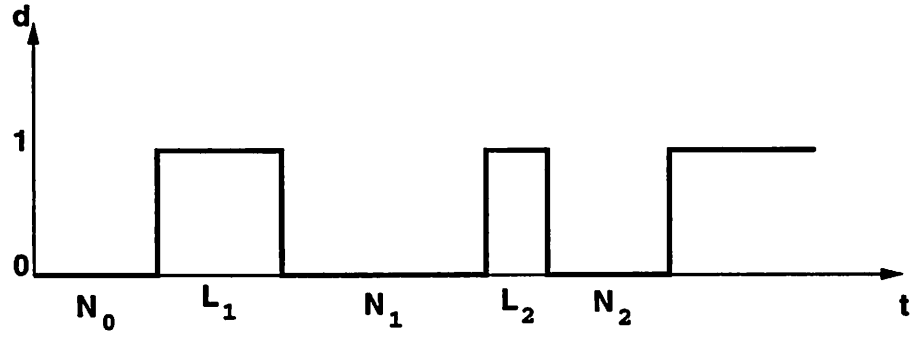

Figure 6: Typical leaking pattern

We will use $d = \{N_0, L_1, N_1, L_2, \ldots\}$ to denote this sequence. The leaking rules limit the set of acceptable u's to:

$$\mathcal{D} = \{d : [a,b] \to \{0,1\} | L_i \leq D_1, N_i \geq D_2\} \tag{39}$$

We will treat the leaking process as an optimization problem. In particular we will try to solve:

$$\max_{d \in \mathcal{D}} x([a,b]) = \max_{d \in \mathcal{D}} \frac{L_1 + L_2 + \ldots + L_k}{b-a}$$

The requirement on the accumulated leaking time can be posed as:

$$x([a,b]) \leq \alpha \text{ for all } 0 \leq a < b \text{ with } b - a \geq T \tag{40}$$

3.3.3 Optimal Solution

We will seek the worst possible disturbance d. First consider the case where $a = 0, b > 0$, which trivially generalizes to general a, b. The problem is simple enough to allow us to guess a maximizing d:

Lemma 7 *A global maximizer of* $x([a,b])$ *is*

$$d = \{0, D_1, D_2, D_1, \ldots\} \in \mathcal{D}$$

Proof: Consider an arbitrary $d = \{N_0, L_1, N_1, L_2, \ldots\} \in \mathcal{D}$. Apply the following algorithm:

- **Step 1:** If $N_0 > 0$ "slide" L_1 left by N_0, filling up with 0 on the right. The resulting d will be:
$$d_1 = \{0, L_1, N_1', L_2, \ldots\}$$
Note that $d_1 \in \mathcal{D}$ as $N_1' = N_0 + N_1 \geq D_2$.
- **Step 2:** If $L_1 < D_1$ "slide" part of L_2 left to "fill up" L_1.
$$d_1' = \{0, D_1, N_1', L_2', \ldots\}$$
If L_2 is not enough use some of L_3 and so on. Note that $d_1' \in \mathcal{D}$ as $L_2' = L_2 - (D_1 - L_1) \leq D_1$.
- **Step 3:** If $N_1' > D_2$ "slide" L_2' left, until:
$$d_2 = \{0, D_1, D_2, L_2', N_2', \ldots\}$$
Note that $d_2 \in \mathcal{D}$ as $N_2' = N_2 + (N_1 - D_2) \geq D_2$.

Steps 2 and 3 are repeated for the remaining L and N. Let $\hat{d}$ denote the resulting d:

$$\hat{d} = \{0, D_1, D_2, D_1, D_2, \ldots, L, N\}$$

where $0 \leq L \leq D_1$ and $N \geq 0$. Note that all the steps of the algorithm preserve the leaking times therefore both d and $\hat{d}$ lead to the same $x([a,b])$. If $L < D_1$ and $N > 0$, "fill up" L until $L = D_1$ or $N = 0$. The resulting d will be:

$$\begin{aligned} \hat{d}' &= \{0, D_1, D_2, D_1, D_2, \ldots, D_1, N'\} \\ &\text{or} \\ \hat{d}' &= \{0, D_1, D_2, D_1, D_2, \ldots, L', 0\} \end{aligned}$$

In either case it will lead to $x'([a,b]) \geq x([a,b])$. If $N' > D_2$ make $N' = D_2$ and add a new interval of leaking time. Again this will lead to a higher value of $x([a,b])$. The process can be repeated until the maximizer:

$$\hat{d} = \{0, D_1, D_2, D_1, D_2, \ldots, L, N\}$$

with $0 \leq L \leq D_1$ and $0 \leq N \leq D_2$ with the restriction that $L = D_1$ if $N > 0$. It should be noted that this maximizer is not unique. □

The above calculation trivially generalized to any a and b. Let k be the integer part of $\frac{b-a}{D_1+D_2}$. Then the worst case accumulated leaking time is given by:

$$\hat{x}([a,b]) = \max_{d \in \mathcal{D}} x([a,b]) = \begin{cases} \frac{(k+1)D_1}{b-a} & \text{if } (b-a) - k(D_1+D_2) \geq D_1 \\ \frac{(b-a)-kD_2}{b-a} & \text{if } (b-a) - k(D_1+D_2) \leq D_1 \end{cases} \tag{41}$$

Theorem 3 *The accumulated leaking time specification is satisfied if and only if D_1, D_2, T are such that $\hat{x}([a,b]) \leq \alpha$.*

The proof is a direct corollary of Lemma 7. Figure 7 shows a graph of $\hat{x}([a,b])$ for various values of (D_1, D_2) and $T = 60$ seconds (as in [6]). The dividing line between acceptable and unacceptable designs is a horizontal plane at height α.

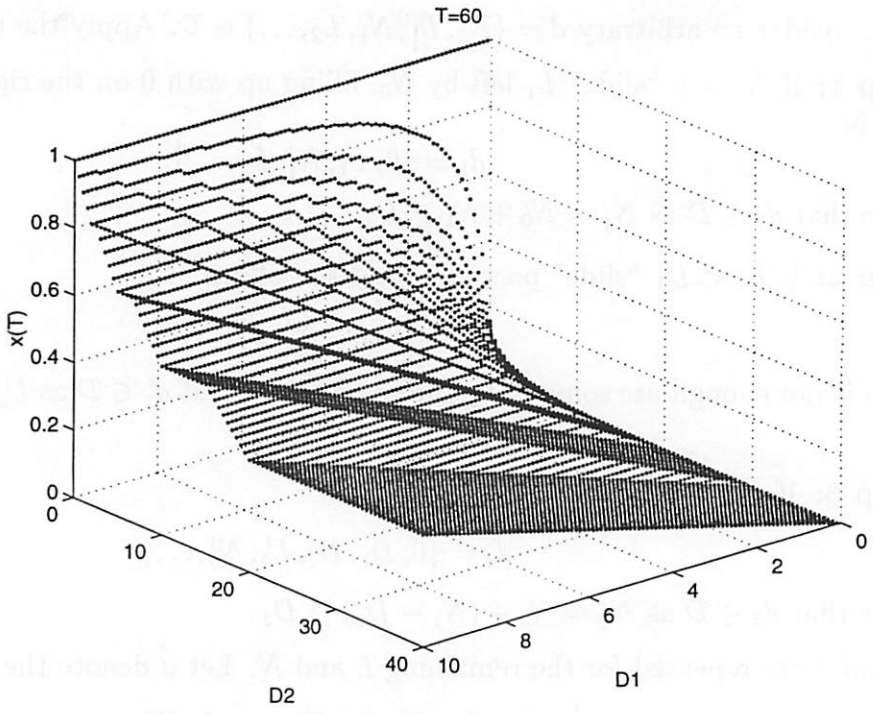

Figure 7: Worst case leaking percentage for D_1, D_2

4 Concluding Remarks

Hierarchical, hybrid system design and verification has attracted significant attention over the last few years. In this paper we presented an approach to the design of hybrid controllers for complex systems. Our approach fits nicely with a multiagent scenario. The starting point was game theory in the continuous domain. The continuous design was treated as a game between the controller and the disturbances, that can be used to model the behavior of other agents, among other things. In addition to optimal continuous controllers the solution to the game also provides requirements on the discrete switching policy. A hybrid controller is guaranteed to meet certain performance specifications if the discrete part obeys these requirements.

Our approach was illustrated by means of two examples: the train-gate controller and the vehicle following problem. We also discussed how similar ideas can be used to solve problems of verification and discrete abstraction generation. We illustrated the application to verification by a simple example, the leaking gas burner, a timed system with an integrator variable.

To facilitate the application of our methodology we need to derive conditions under which the problems are solvable (saddle and optimal solutions exist, etc.). In this respect we hope to be able to extend older results in game theory [23] and optimal control [24]. We also need to develop algorithms for determining the sets of guaranteed performance, V_i, or at least conservative approximations to them. Extensions of previous results may also suffice for this problem.

The most important missing part is a technique for designing the higher (discrete) levels. The approach presented here only gives continuous designs and switching guidelines.

The designer still has to come up with a discrete design that follows the guidelines. The problem is more difficult because of the fact that even the descriptive language of the discrete layer is not specified a-priori. An interesting issue that needs to be addressed in the discrete setting is what happens if the abstractions indicate that conflicting objectives (e.g. safety and capacity) can not be met. As discussed in the introduction, the problem may still be solvable in this case by using inter-agent coordination. The role of coordination is to cut down on the set of allowable disturbances generated by other agent actions and hence bias the game in the controllers favor. The questions of how are the higher levels to be designed to achieve sufficient coordination and where is the line between feasible and infeasible performance requirements are very interesting and still require a lot of work to be answered.

References

[1] Z. Har'El and R. Kurshan, *Cospan User's Guide.* AT&T Bell Laboratories, 1987.

[2] Adnan Aziz, et al., "HSIS: a BDD-based environment for formal verification," in *ACM/IEEE International Conference on CAD*, 1994.

[3] M. Heymann, "Hierarchical decomposition of hybrid systems." (preprint), 1994.

[4] A. Hsu, F. Eskafi, S. Sachs, and P. Varaiya, "Protocol design for an automated highway system," *Discrete Event Dynamic Systems*, vol. 2, no. 1, pp. 183–206, 1994.

[5] R. Alur, C. Courcoubetis, and D. Dill, "Model checking for real-time systems," *Logic in Computer Science*, pp. 414–425, 1990.

[6] 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.), pp. 209–229, New York: Springer Verlag, 1993.

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

[8] C. Daws and S. Yovine, "Two examples of verification of multirate timed automata with KRONOS," in *Proc. 1995 IEEE Real-Time Systems Symposium, RTSS'95*, (Pisa, Italy), IEEE Computer Society Press, Dec. 1995.

[9] F. Balarin, *Iterative Methods for Formal Verification of Digital Systems.* PhD thesis, University of California, Berkeley, 1994.

[10] F. Balarin, K. Petty, and A. L. Sangiovanni-Vincentelli, "Formal verification of the patho real-time operating system," in *IEEE Control and Decision Conference*, pp. 2459–2465, 1994.

[11] A. Puri and P. Varaiya, "Decidebility of hybrid systems with rectangular differential inclusions," in *Computer Aided Verification*, pp. 95–104, 1994.

[12] T. Henzinger, P. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata," in *STOCS*, 1995.

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

[14] M. S. Branicky, V. S. Borkar, and S. K. Mitter, "A unified framework for hybrid control: Background, model and theory," Tech. Rep. LIDS-P-2239, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, 1994.

[15] 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.), pp. 297–316, New York: Springer Verlag, 1993.

[16] T. Basar and P. Bernhard, *H^∞-Optimal Control and Related Minimax Design Problems.* Birkhauser, 1991.

[17] J. C. Doyle, K. Glover, P. P. Khargonekar, and B. A. Francis, "State-space solutions to standard H_2 and H_∞ control problems," *IEEE Transactions on Automatic Control*, vol. 34, no. 8, pp. 831–847, 1989.

[18] A. Puri and P. Varaiya, "Driving safely in smart cars," in *American Control Conference*, pp. 3597–3599, 1995.

[19] A. Puri and P. Varaiya, "Verification of hybrid systems using abstractions," in *Hybrid Systems II, LNCS 999*, Springer Verlag, 1995.

[20] D. Godbole and J. Lygeros, "Longitudinal control of the lead car of a platoon," *IEEE Transactions on Vehicular Technology*, vol. 43, no. 4, pp. 1125–1135, 1994.

[21] J. K. Hedrick, D.McMahon, V. Narendran, and D. Swaroop, "Longitudinal vehicle controller design for IVHS system," in *American Control Conference*, pp. 3107–3112, 1991.

[22] F. M. Callier and C. A. Desoer, *Linear System Theory.* Springer-Verlag, 1991.

[23] T. Basar and G. J. Olsder, *Dynamic Non-cooperative Game Theory.* Academic Press, 2nd ed., 1994.

[24] L. Berkovitz, *Optimal Control Theory.* Springer-Verlag, 1974.