ELSEVIER

Theoretical Computer Science 138 (1995) 141–168

Theoretical Computer Science

Viability in hybrid systems

Wolf Kohn[a], Anil Nerode[b,*,1], Jeffrey B. Remmel[c,2], Alexander Yakhnis[d,3]

[a]*Intermetrics Corporation, Bellevue, Washington, USA*

[b]*Mathematical Sciences Institute, Cornell University, Ithaca, NY 14853, USA*

[c]*Department of Mathematics and Department of Computer Science, University of California at San Diego, San Diego, CA92093, USA*

[d]*Sciences Institute, Cornell University, Ithaca, NY, 14853, USA*

Abstract

Hybrid systems are interacting systems of digital automata and continuous plants subject to disturbances. The digital automata are used to force the state trajectory of the continuous plant to obey a performance specification. For the basic concepts and notation for hybrid systems, see Kohn and Nerode (1993), and other papers in the same volume. Here we introduce tools for analyzing enforcing viability of all possible plant state trajectories of a hybrid system by suitable choices of finite state control automata. Thus, the performance specification considered here is that the state of the plant remain in a prescribed viability set of states at all times (Aubin, 1991). The tools introduced are local viability graphs and viability graphs for hybrid systems. We construct control automata which guarantee viability as the fixpoints of certain operators on graphs. When control and state spaces are compact, the viability set is closed, and a non-empty closed subset of a viability graph is given with a sturdiness property, one can extract finite state automata guaranteeing viable trajectories. This paper is a sequel to Kohn and Nerode (1993), especially Appendix II.

1. Introduction

When we try to control the behavior of a dynamical system evolving in time, the fundamental problem is to extract control functions $c(t)$ which force the state $x(t)$ of the dynamical system to remain at all times in a prescribed set, the viability set (**VS**). In his book *Viability Theory* [5], Aubin gathers together the literature on viability of state trajectories of continuous and discrete dynamical systems and puts it in coherent

*Corresponding author: E-mail: anil@math.cornell.edu.

[1]Supported in part by Army Research Office contract DAAL03-91-C-0027 and by DARPA-US ARMY AMCCOM (Picatinny Arsenal, NJ) contract DAAA21-92-C-0013 to ORA Corp.

[2]Supported in part by Army Research Office contract DAAL03-91-C-0027 and NSF grant DMS-9306427.

[3]Supported by DARPA-US ARMY AMCCOM (Picatinny Arsenal, NJ) contract DAAA21-92-C-0013 to ORA Corp.

SSDI 304-3975(94)00150-2

form. Aubin points out that many problems of evolutionary theory, economics, and the sciences are problems of viability, or of enforcing viability by choice of appropriate control functions. We are interested in the somewhat different case when control can be forced by a finite state control automaton rather than a traditional feedback control law. Aubin develops viability theory for continuous time systems. He also spends a few pages on discrete time systems. Now hybrid systems are interacting systems of digital automata and continuous plants subject to disturbances. The digital automata are usually intended to be control automata used to force the state trajectory of the continuous plant to obey a performance specification. The fundamental problem of hybrid systems is to extract, given a continuous plant simultion model and the performance specification on plant state trajectories, a finite control automaton which will force the hybrid system to meet the performance specification. Hybrid systems have mixed continuous time, discrete time states. These states are the simultaneous states of the continuous plant and of the finite state control automaton. We wish to capture the conditions under which *finite state* control automata exist which guarantee viability.

Aubin generally emphasizes non-deterministic systems, with many evolutions possible for given initial conditions, for which set-valued analysis is the appropriate tool. Here we limit definitions to the deterministic case because generalization of definitions to the non-deterministic case is both straightforward and notationally opaque. Aubin points out that there is a dearth of general theorems on the existence of feedback control functions producing viable solutions to general problems, but that when a specific feedback control function, plant, and viability set are given in advance, one can fruitfully investigate whether the feedback control can enforce viability. The situation is similar for viability in hybrid systems. For hybrid systems we start one step later. Suppose that we are already given a continuous feedback control function which enforces viable trajectories for a plant. Then we can fruitfully investigate how to extract a finite automaton which exhibits controllable and observable behavior and enforces the same viability.

Here is one view of the stages in the extraction of a control policy for ensuring that a continuous plant obeys its performance specification independent of the particular brand of control theory used (see Fig. 1).

1. In the first state, identify the space of control policies which meet the viability constraint.
2. In the second stage, extract from that space a mathematical control policy leading to an evolution of plant state behavior which has a prescribed optimality property. This is a mathematically optimal policy in some prescribed sense.
3. In the third stage, since mathematically optimal control policies are not usually directly implementable in hardware and software, extract and implement an approximation to that optimal policy which still meets the viability constraint and yields evolutionary behavior sufficiently close to optimal to be satisfactory to the user.

In this paper we investigate for simple hybrid systems only the first stage. In Section 8 we sketch why this development is part of the foundations of Kohn–Nerode distributed hybrid control [26]. See also [28, 29, 32, 33].

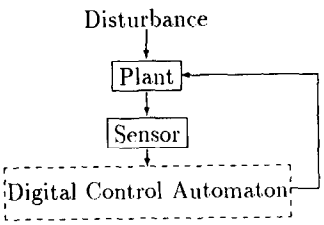

Fig. 1.

2. Simple hybrid systems

All hybrid systems will be simple hybrid systems as defined in [25] with fixed control intervals $[0,\Delta]$, $[\Delta,2\Delta]$,... Here is a brief description of a simple hybrid system consisting of a continuous plant interacting with a control automaton at times $n\Delta$, $n=0,1,\ldots$, see [25] for more details.

The plant runs open loop inside an interval of time $[n\Delta,(n+1)\Delta]$ based on the control function c_n and the disturbance d_n supplied at time $n\Delta$ for use in $[n\Delta,(n+1)\Delta]$. The control automaton receives as input at time $n\Delta$ the current state x of the plant, then runs open loop with no further inputs till time $(n+1)\Delta$. Based on its state at time $(n+1)\Delta$, the control automaton transmits a new control function c_{n+1} to the plant to be used in the interval $[(n+1)\Delta,(n+2)\Delta t]$, and so on. We shall assume that our plant is described by a vector first order ordinary differential equation

$$\dot{x}=f(x,c,d)$$

with parameters, c,d. We assume that

$$\dot{x}(t)=f(x(t),\tilde{c}(t),\tilde{d}(t))$$

is such that for any time t_0, for any initial state $s(t_0)$, for any admissible control and disturbance functions $\tilde{c}(t),\tilde{d}(t)$ defined on $[0,\infty]$, there is a unique solution $x(t)$ defined on $[0,\infty]$ satisfying the differential equation. That is, we assume that there is a field of solutions over the whole space, each uniquely determined by any one point. This is mostly for convenience. Generlizations of the methods of this paper to the case when there are no, many, or limited trajectories for given initial conditions are notationally cumbersome, but nevertheless can be developed in a relatively straightforward manner. We call $x(t)$ the trajectory of the plant state.

To repeat, the control automaton plant receives as input at time $n\Delta$ the current state $x(n\Delta)$ of the plant based on its operation over the previous interval $[(n-1)\Delta,n\Delta]$. At time $n\Delta$, the plant is in a certain state, the initial plant state for the next interval of time. The current control function is shifted to the new one imposed by the control automaton output at that time. So the initial value of $\dot{x}$ for the interval $[n\Delta,(n+1)\Delta]$ is not inherited from the previous interval, but is computed from the

differential equation based on current plant state and the initial value of the new control function and the new disturbance at $n\Delta$. At times $n\Delta$, the vector field generally changes direction abruptly. This is characteristic of hybrid control.

Abuse of notation. We shall assume that the differential equation describing our plant is autonomous so that the behavior in any interval of length Δ is the translate of behavior in any other interval of length Δ. Thus we may as well assume that all admissible control functions $c(t)$ and disturbance functions $d(t)$ are defined on $[0,\Delta]$ and by abuse of notation, translate them by $n\Delta$ for use on the interval $[n\Delta,(n+1)\Delta]$. Hence the differential equation on interval $[n\Delta,(n+1)\Delta]$ now reads

$$\dot{x}(t)=f(x(t-n\Delta),c(t-n\Delta),d(t-n\Delta)),$$

where, from now on, $c(t),d(t)$ both have domain $[0,\Delta]$. In summary, we have a state space S for the plant, a set C of admissible control functions on $[0,\Delta]$, a set D of admissible disturbance functions on $[0,\Delta]$, such that when an admissible control $c=c(t)$ and disturbance $d=d(t)$ are given, the state trajectory $x(t)$ on the interval $[0,\Delta]$ is uniquely determined.

Δ-Plant automata. The continuous plant induces an automaton, which we call the Δ-plant automaton associated with the simple hybrid system. It has two input alphabets, the set D of admissible disturbance functions and the set C of admissible control functions. Its set of internal states is the set of plant states. Its state transition function assigns to input letters control $c(t)$ and disturbance $d(t)$ and automaton current state s_0, the new automaton state $x(\Delta)$ where $x(t)$ is a plant state trajectory such that $x(0)=s_0$ and $x(t)$ is the solution of the differential equation

$$\dot{x}=f(x(t),c(t),d(t)).$$

Proposition 2.1. *Suppose we are given disturbance function $\tilde{d}(t)$ and control function $\tilde{c}(t)$ on $[0,\infty]$. Suppose that for all integers $n\geqslant 0$,*

1. *$d_n\in D$, where $d_n(t)=\tilde{d}(t-n\Delta)$, and*
2. *$c_n\in C$, where $c_n(t)=\tilde{c}(t-n\Delta)$.*

Suppose that $x(t)$ is the solution to

$$\dot{x}=f(x(t),\tilde{c}(t),\tilde{d}(t)),$$

Suppose that the associated Δ-automaton is started at time 0 in state $x(0)$ with inputs c_0,d_0, and receives inputs c_n,d_n at time $n\Delta$, and instantaneously changes state at these times based on its transition function. Then $x(n\Delta)$ is the state of the Δ-automaton at time $n\Delta$.

Example. Here is a Δ-plant automaton arising from a differential equation

$$\dot{x}=f(x,c)$$

in a different way. Here we allow delays as disturbances, that is, the only disturbances are delays d in changing from one control function c_0 to another c_1. Let C_p be the admissible set of control functions on $[0,\Delta]$, let $C=C_p\times\mathbf{VS}\times C_p$. Let the set of delays $D=[0,b]$ for a $b<\Delta$. We model a delay $d\leqslant b<\Delta$ before the change from the control c_0 used at time 0 to the control c_1 used at and after time d as follows. Define an associated Δ-automaton as follows. The states are the plant states x. One input is a pair $(c_0,c_1)\in C_p\times C_p$ and the other input is a disturbance $d\in D$. Suppose that the state of the Δ-automaton at time 0 is s_0, the inputs are a pair of controls (c_0,c_1) and disturbance d. What is the corresponding state transition from time 0 to time Δ? Define

$$c_{01}(t)=\begin{cases} c_0(t) & \text{for } 0\leqslant t<d,\\ c_1(t-d) & \text{for } d\leqslant t\leqslant\Delta.\end{cases}$$

The new state of the Δ-automaton at time Δ is $x(\Delta)$, where $x(t)$ solves

$$\dot{x}=f(x(t),c_{01}(t))$$

with initial condition $x(0)=s_0$. We will use this model below for a generalization of the leaky water tank example of [1].

3. Local viability

We designate a subset **VS** of plant states as the viability set [5]. We shall usually assume that **VS** is closed and compact. A trajectory $x(t)$ over an interval of time $[0,\Delta]$ is called *viable* if for all t in that interval, $x(t)\in\mathbf{VS}$. Similarly, a trajectory $x(t)$ extending over $[0,\infty]$ is *viable* if for all $n\geqslant 0$, the trajectory $x_n(t)=x(t-n\Delta)$ over $[0,\Delta]$ is viable. Suppose we are given a simple hybrid system with control intervals $[n\Delta,(n+1)\Delta]$, $n=0,1,\ldots$, and viability set **VS**, with associated Δ-plant automaton. We give three definitions of local graphs associated with viability.

1. The abstract local viability graph. The abstract viability graph is an obvious analogue to the viability kernels of continuous systems in [5]. Non-empty closed compact subsets of this graph and closed viability sets lead to finite automata that enforce viability.
2. The sturdy local viability graph. Non-empty closed compact subsets lead to finite automata that force viability and also are "safe" under small errors in state and control measurements.
3. The e-sturdy local viability graph. This represents those hybrid systems with a sensor of plant states with error bounded by a fixed e. This leads to finite state control automata whose analog to digital converter, or sensor of plant state (see [25]) has error bounded by e and also enforces viability.

We shall not attempt to develop the last two extensively here.

3.1. The abstract local viability graph

Nodes. Define the nodes of the local viability graph as those pairs

$$(c_0, s_0) \in C \times \mathbf{VS}$$

such that for any disturbance $d_0 \in D$, the trajectory $x(t)$ determined by d_0, control c_0 and initial state s_0, is viable.

Remark.
1. The c_0 should be interpreted as the control used in the current control interval.
2. The s_0 should be interpreted as the state at the beginning of the current control interval.
3. Disturbances are not directly used in determining controls. Normally, we observe disturbances only through their effect on plant state. We need to choose controls that maintain viability of trajectories no matter what disturbance is encountered.

Edges. There is a directed edge from node (c_0, s_0) to node (c_1, s_1) if and only if
1. (c_0, s_0) and (c_1, s_1) both nodes of the local viability graph and
2. There is a disturbance $d_0 \in D$ such that the trajectory $x(t)$ with disturbance d_0 and control c_0 and initial condition $x(0) = s_0$ has $x(\Delta) = s_1$.

Remark.
1. Condition (2) ensures the Δ-automaton with inputs d_0 and c_0 is a local s_0 has a transition to new state s_1.
2. Note that in (2), $x(t)$ is a viable trajectory because (c_0, s_0) is a local viability node.
3. We call (c_0, s_0) the tail, (c_1, s_1) the head of the directed edge.
4. There may be nodes of the abstract local viability graph that are not tails or heads of any edges. In the definition of the abstract viability graph below, nodes that are not tails of edges are dropped at the beginning of the construction.

3.2. The sturdy local viability graph

Suppose again we are given a simple hybrid system with control intervals $[n\Delta, (n+1)\Delta]$, $n = 0, 1, \ldots$, and viability set **VS**.

Definition 3.1. Call a pair $(c_0, s_0) \in C \times \mathbf{VS}$ sturdy if there exists a neighborhood U of c_0 in C and neighborhood V of s_0 in **VS** such that for every $(\bar{c}, \bar{s}) \in U \times V$, $(\bar{c}, \bar{s})$ is a node of the abstract local viability graph.

This says that for any fixed disturbance, varying the initial state and control preserves the viability of resulting trajectories over $[0, \Delta]$.

Sturdy nodes. The sturdy local viability graph has as nodes those pairs $(c_0, s_0) \in C \times \mathbf{VS}$ such that (c_0, s_0) is a sturdy node.

Sturdy edges. The edges of the sturdy viability graph are those edges of the abstract local viability graph that consists of pairs of sturdy nodes.

Remark.

1. A pair (U, V) for sturdy pair (c_0, s_0) should be interpreted as a "latitude" in knowledge of plant state s_0 and control c_0 permitted that still produces viable trajectories no matter what acceptable disturbance occurs. This latitude is required to extract finite state control automata guaranteeing viable trajectories.
2. Here is an explanation of "sturdy". If arbitrarily small changes in initial control and state change behavior from viable to non-viable trajectories, the system should not be regarded as controllable. In the interpretation of [25, Appendix II], the existence of such a pair (U, V) should be regarded as a physical realizability requirement for the analog to digital converter for any proposed finite state control automaton.

Sturdy local viability graph. This is a variant of the model in [32]. Assume that S is a metric space. Suppose we are given a positive real $e>0$, interpreted as a bound for measurement error in plant state in the following sense. The plant state as measured is always less than e from the true state. This is to represent a fixed bound on observability of plant state by a fixed sensor. Define the local viability graph with sensor error bound e as follows.

Nodes. The e-sensor nodes of the e-sensor local viability graph are those pairs $(c_0, s_0) \in C \times S$ such that for every disturbance $d \in D$, every state $\bar{s}$ with distance $(s, \bar{s}) < e$, the trajectory over $[0, \Delta]$ satisfying

$$\dot{x}(t) = f(x(t), c_0(t), d(t)).$$

with initial condition $x(0) = \bar{s}$ is viable.

Remark. The difference between this and the definition of sturdy is that e diameter neighborhoods V of state s_0 are required, while no latitude is allowed for control c_0 at all.

Edges. The edges of the e-sensor local viability graph are just the edges of the local viability graph with e-sensor nodes.

All of these graphs can be thought of as abstract non-deterministic automata. Here is an example.

The local viability automaton. The input alphabet of this automaton is the set of viable plant states **VS**. The set of local viability automaton states is the set of controls. The non-deterministic transition relation maps a pair $(c_0, s_1) \in C \times \mathbf{VS}$ to a control c_1 iff there exists an edge in the abstract local variability graph with tail (c_0, s_0) and head (c_1, s_1). This is a partially defined transition relation. The interpretation is that c_0 should be thought of as the control used in the previous control interval which has,

due to a disturbance, produced the current plant state s_1. Then, with input letter s_1 when in local viability automaton state c_0, the local viability automaton moves to local viability automaton state c_1 and outputs letter c_1.

We note that there is another natural automaton which can be defined from the local viability graph. For this alternative automaton, the input alphabet is the set of admissible disturbances D. The set of states for the alternative automaton is the set $\mathbf{VS} \times C$ where **VS** is the set of viable plant states and C is the set of controls. The non-deterministic transition relation maps a pair $\langle d_0, (c_0, s_0)\rangle \in D \times (C \times \mathbf{VS})$ to a pair $(c_1, s_1) \in C \times \mathbf{VS}$ iff there exists an edge in the abstract local viability graph with tail (c_0, s_0) and head (c_1, s_1) where $s_1 = x(\Delta)$ for the viable trajectory $x(t)$ determined by disturbance d_0, control c_0, and initial condition $x(0) = s_0$. The interpretation is that with input letter d_0 in state (c_0, s_0), the automaton moves to state (c_1, s_1) with output letter c_1. This is also a partially defined transition relation. However we shall not pursue this alternative automaton because in applications, one does not observe the disturbance directly but only observes the effect of the disturbance by its effect on the plant state. Thus the alternative automaton is not implementable in practice.

4. Graph operators and graph kernels

In this section we introduce the tools needed to discuss viability over $[0, \infty]$. Assume we are given a directed graph T which consists of a non-empty set T of nodes together with a subset E of $T \times T$ of its directed edges such that each node is incident on at least one edge. It is convenient to identify the graph with its set of edges since the nodes can be recovered from the edges. If (a, b) is an edge of T, then a is called its tail, b is called its head. Each subset T' of T defines a subgraph with edges $E' = E \cap (T' \times T')$. A path is a finite or infinite sequence of edges such that the head of each edge is the tail of the next. An end node of a graph is a node which is not the tail of any edge in that graph. Let $P(T)$ denote the power set of T, i.e. the set of all subsets of T.

Definition 4.1. Suppose graph T is given.

1. Define a monotone decreasing operator $F: P(T) \to P(T)$ by letting $F(T')$ be the set of nodes of T' which are not end nodes of T' and which are on at least one edge of T'.
2. For each ordinal α, define an operator $F^\alpha: P(T) \to P(T)$ by transfinite induction as
   (a) $F^0(T') = F(T')$,
   (b) $F^{\alpha+1}(T') = F(F^\alpha(T'))$,
   (c) $F^\lambda(T') = \bigcap_{\alpha<\lambda} F^\alpha(T')$ if λ is a limit ordinal.

Proposition 4.2. *Suppose that $T' \subseteq T$.*

1. *Then T' is a fixed point of F if and only if every node of T' is the initial node of some infinite path in T'.*

2. *There is a least ordinal* α *such that*

$$F^{\alpha+1}(T')=F^{\alpha}(T').$$

3. *If* α *is the least ordinal such that* $F^{\alpha+1}(T')=F^{\alpha}(T')$, *then* $F^{\alpha}(T')$ *is the largest fixed point of* F *contained in* T'.

Proof.

1. $F(T')$ is T' less the end nodes of T' and the nodes on no edge. Thus if T' is a fixed point of F, it has no end nodes, and no nodes which are not part of any edge. Suppose that $F(T')=T'$. Then we can start with any node p_0 of T' and define by induction an infinite path $p_0, p_1, p_2, \ldots$ of T' by choosing p_{n+1} to be a node of T' such that p_n is the tail and p_{n+1} is the head of an edge of T'. Thus every node of a fixed point T' of F begins an infinite path of T'. Conversely, suppose $T' \subseteq T$ and every node of T' starts an infinite path of T'. The lack of end nodes means T' is fixed under F.
2. By transfinite induction, for any $\alpha \leqslant \beta$ and any $T' \subseteq T$,

$$F^{\alpha}(T') \supseteq F^{\beta}(T').$$

Thus there is a β smaller than the next infinite cardinal after the cardinal of T, $card(T)^+$, such that

$$F^{\beta+1}(T')=F^{\beta}(T').$$

Otherwise, points in T from the differences $\{F^{\alpha+1}(T')-F^{\alpha}(T'): \alpha < card(T)^+\}$ would be a subset of T of cardinality larger than the cardinal of T, a contradiction.
3. If α is the least such β, then $F^{\alpha}(T')$ is a fixed point under F. Suppose that $T'' \subseteq T'$ is a fixed point. Then every node of T'' begins an infinite path of T''. By transfinite induction, all nodes on this path will in all $F^{\beta}(T')$, therefore in $F^{\alpha}(T')$. So the latter is the largest fixed point within T'. □

Remark. Proposition 4.2 can also be derived from a strengthening of the Tarski fixed point theorem for monotone increasing maps on complete partial orders. Simply use the monotone increasing operator

$$G(T')=T-F(T') \text{ on } P(T).$$

Proposition 4.3. *Suppose that*:

(i) *The nodes of* T *are elements of a separable metric space and*

(ii) T' *is a subgraph of* T *such that for every ordinal* α *and for every end node of* $F^{\alpha}(T')$ *or node on no edge of* $F^{\alpha}(T')$, *there is a neighborhood containing that node and no other node of* $F^{\alpha}(T')$. (*Here we interprete* F^0 *to be the identity map on* $P(T)$.)

Then:

1. *The least ordinal* α *such that* $F^{\alpha}(T')=F^{\alpha+1}(T')$ *is a countable ordinal.*
2. *If* T' *is closed, then* $F(T')$ *is closed.*

3. *If T' is closed, then so are all $F^{\alpha}(T')$*
4. *If T' is closed, then so is the maximal fixed point of F under T'.*

Proof. By separability, there is a countable dense subset of the metric space. Due to (ii), at least one new element of that countable dense subset is eliminated whenever a node is eliminated, that is, at any α for which $F^{\alpha}(T) \neq F^{\alpha+1}(T)$. So the process terminates at a countable ordinal. Note that (ii) implies that in removing all end nodes from a closed T', we are intersecting T' with a closed set, and hence get a closed set, which is (2). Then (3) follows by transfinite induction, since the intersection of closed sets is closed, and (4) follows from (3). □

Remark. One can also prove Proposition 4.3 by an application of Cantor's argument for the perfect kernel of a closed set in a separable space. The separability condition is satisfied in our intended applications.

Dual Scott Topology. The dual Scott topology on the power set $P(T)$ is generated by the following open sets. For each finite subset T_1 of T, declare that

$$[T_2 \in P(T) \mid T_1 \cap T_2 = \emptyset]$$

is open. Open sets in the dual Scott topology are precisely arbitrary unions of these open sets arising from finite subsets of the nodes of T in this way.

Proposition 4.4. *Let $G: P(T) \to P(T)$ be defined by $G(T') = T - F(T')$. Suppose that G is continuous in the dual Scott topology. Then $F^{\omega}(T')$ is the maximal fixed point of F within T'.*

Proof. By Proposition 4.2, it suffices to show that $F^{\omega}(T')$ *is a fixed point of F. Since F maps sets into smaller sets,*

$$F(F^{\omega}(T')) \subseteq F^{\omega}(T').$$

So it suffices to show that

$$T' - F(F^{\omega}(T')) \subseteq T' - F^{\omega}(T').$$

Suppose, then, that $p \in T' - F(F^{\omega}(T'))$. Note that $P(T - \{p\})$ is an open set in the dual Scott topology and $F(F^{\omega}(T')) \in P(T - \{p\})$. According to dual Scott continuity, there is a finite $T_1 \subseteq T$ such that whenever T'' is disjoint from T_1, then p is not in $F(T'')$. But, applied to $T'' = F^{\omega}(T') = \bigcap_{n \in \omega} F^{\omega}(T')$ this says that if $T_1 = \{p_1, \ldots, p_k\}$, then there exist $n_1, \ldots, n_k \in \omega$ such that $p_i \notin F^{n_i}(T')$. Letting m be the maximum of $\{n_1, \ldots, n_k\}$, and using the fact that F maps sets into smaller sets, we see that T_1 is disjoint from $F^{m}(T')$, so that p is in $T' - F^{m+1}(T')$, and hence by the monotonicity of F, is in $T' - F^{\omega}(T')$, as was to be proved. □

Remark. We can also prove Proposition 4.4 by applying the fixed point theorem for continuous functions on complete partial orderings directly to $G(T') = T - F(T')$. The theorem needed was already in [27], the earliest topological form of Kleene's second recursion theorem.

Remark. The fixed point may be an $F^n(T)$ for a finite n. This happens for a natural operator associated with the leaking water automaton $A(g,h)$ defined in Section 6.

5. Viability graphs

We now want to discuss not only the local viability of the trajectory over the current and next interval of time, but viability of trajectories on $[0,\infty]$. With the apparatus of Section 4, we can define the abstract, sturdy or e-sensor viability graphs of a simple hybrid system with control intervals $[n\Delta,(n+1)\Delta]$, $n=0,1,\ldots$ with viability set $\mathbf{VS}\subseteq S$. The intention is that this graph captures all control polices (see below) which enforce that the simple hybrid system produces only viable trajectories over $[0,\infty]$, even when we allow small changes in state s_0 and control c_0.

Definition 5.1. The *abstract* (respectively, *sturdy*, *e-sensor*) viability graph is the kernel of the abstract (respectively, sturdy, e-sensor) local viability graph.

Viability automaton. The abstract viability automaton has a transition corresponding to each edge of the abstract viability graph with tail (c_0,s_0) and head (c_1,s_1) as described above. Any admissible disturbance $\bar{d}$ over $[0,\infty]$ determines by restriction to $[n\Delta,(n+1)\Delta]$ and translation back to $[0,\Delta]$ a sequence $d_n\in D$ of disturbances on $[0,\Delta]$. If we start at a node (c_0,s_0) of the abstract viability graph with disturbance d, this process produces an "execution sequence" of the abstract viability automaton and a viable trajectory on $[0,\infty]$ as long as there is indeed a node in the abstract viability graph. However, the abstract viability graph may be empty. Moreover, it can be quite difficult to prove that the abstract viability graph is non-empty. The same remarks apply to the sturdy and e-sensor viability graphs.

Control policies. In interpreting an edge with tail (c_0,s_0) and head (c_1,s_1), what we envisage is that at the beginning of any control interval, c_0 represents the control used in the previous interval for the plant physical controller, s_0 represents the plant state at the start of the previous control interval, c_1 is a possible choice of control for the current interval given that the trajectory $x(t)$ determined by c_0,s_0, and the admissible disturbance determined by $\bar{d}$ has $x(\Delta)=s_1$. With this in mind, a control policy can be defined as a map on $C\times S$ to $P(C)$ which assigns to a pair (c_0,s_1) the set of choices of c_1, any of which is permitted under the control policy. Alternately, a control policy is simply a subset CP of $C\times S\times C$ consisting of triples (c_0,s_1,c_1). The largest control policy is the universal policy $C\times S\times C$, which permits any choice of c_1, the smallest is

the null policy, which is devoid of choice. Because edges with tail (c_0, s_0) and head (c_1, s_1)) that make up the edges of the abstract, sturdy, and e-sensor local viability graphs and viability graphs all lead naturally to triples of the form $(c_0, s_1, c_1) \in C \times \mathbf{VS} \times C$, a policy can be interpreted on any of these graphs.

Definition 5.2. An edge of the abstract local viability graph with tail (c_0, s_0) and head (c_1, s_1) is an abstract policy edge for policy CP if (c_0, s_1, c_1) is in CP. The policy graph for a policy consists of its policy edges. A path consisting of abstract policy edges is an abstract policy path for CP. (An abstract policy path is just a path in the abstract local viability graph that can arise by following the policy.)

Proposition 5.3. *The trajectories on* $[0, \infty]$ *produced by a control policy and admissible disturbances on* $[0, \infty]$ *are all viable iff all infinite policy paths are abstract policy paths.*

Proof. If all infinite policy paths of a control policy are abstract policy paths, then no matter what the admissible disturbance on $[0, \infty]$, if a trajectory is produced on $[0, \infty]$ by making choices of control solely in accordance with the policy and starting at a node of the abstract viability graph, that trajectory is always viable. Conversely, suppose that a control policy allows the choice of an infinite policy path with an edge not in the abstract viability graph. Then that edge allows us to exhibit an admissible disturbance for which the trajectory on $[0, \infty]$ corresponding to that policy path is not viable. □

Corollary 5.4. *Suppose we are given a simple hybrid system, a* Δ, *and a closed viability set* **VS**. *Suppose that* S, C *are separable metric spaces, and that the set* T *of nodes of the abstract (respectively, sturdy, e-sensor) local variability graph is closed. Then the set of nodes of the abstract (respectively sturdy, e-sensor error) viability graph is also closed.*

Proposition 5.5. *Suppose that* **VS** *is closed. Suppose that for any fixed* t_0 *with* $0 \leqslant t_0 \leqslant \Delta$ *and any disturbance* d *in* D, *the map* $(s_0, c_0) \to x(t_0)$ *with domain* $S \times C$ *is continuous. Then the set of nodes of the abstract local viability graph is closed.*

Proof. We prove that the complement of the set of nodes of the local viability graph is open. Suppose that (c_0, s_0) is not a node of the local abstract viability graph. We need an open set containing that point and disjoint from that set of nodes.

Suppose that (c_0, s_0) is not in the abstract local viability graph. Thus there must exist a disturbance d_0 such that the trajectory $x(t)$, obtained from s_0, c_0, and d_0, is not viable. Then for some $t_0 \in [0, \Delta]$, the value $x(t_0)$ is in the open set $S - \mathbf{VS}$. Continuity and the fact that $S - \mathbf{VS}$ is open imply that there exist neighborhoods U_d of c_0, V_d of s_0 such that the image of $U_d \times V_d \times \{d\}$ under $(c, s) \to x(t_0)$ is a subset of $S - \mathbf{VS}$. In this case, $U_d \times V_d$ is an open set containing (c_0, s_0) and every $(\bar{c}, \bar{s}) \in U_d \times V_d$ has the property that the trajectory $x(t)$ determined by disturbance d, control $\bar{c}$, and initial condition $\bar{s}$ has $x(t_0)$ outside **VS**, so that x is not viable. It then follows that (c_0, s_0, c_1)

is not in the local viability graph for any c_1. Hence, $U_d \times V_d \times C$ is a neighborhood of (c_0, s_0, c_1) disjoint from the nodes of the local viability graph. This completes the proof. □

Remark. It is a mild assumption on the underlying differential equation for the plant that for any fixed t_0 and d_0, the state $x(t_0)$ is jointly continuous in initial condition $s_0 \in S$ and parameter $c_0 \in C$. This is the source of Proposition 5.5.

Proposition 5.6. *Suppose we are given a simple hybrid system, a Δ, and a closed viability set* **VS**. *Suppose that S, C are separable metric spaces, and that the set T of nodes of the abstract (respectively, sturdy, e-sensor) local viability graph is closed. Moreover, suppose that every closed subgraph of T' of T has the property that for every end node of T' or node on no edge of T', there is a neighborhood containing that node and no other node of T'. Then the set of nodes of the abstract (respectively, sturdy, e-sensor error) viability graph is also closed.*

Proof. This proposition follows immediately from Proposition 4.3 where $T = T'$. □

We end this section with a simple example of where the hypothesis of Proposition 5.6 hold.

Proposition 5.7. *Suppose that*

1. **VS** *is closed and for any fixed t_0 with $0 \leqslant t_0 \leqslant \Delta$ and any disturbance d in D, the map $(s_0, c_0) \to x(t_0)$ with domain $S \times C$ continuous.*
2. *C and D are compact and the map $(s_0, c_0, d_0) \to x(\Delta)$ with domains $S \times C \times D$ is continuous.*

Then the set of nodes of the abstract local viability graph T is closed and every closed subgraph of T' of T has the property that for every end node of T' or node on no edge of T', there is a neighborhood containing that node and no other node of T'.

Proof. The proof of Proposition 5.5 shows that T is closed. Now suppose that T' is a closed subgraph of T. We must show that for any node (c_0, s_0) in T' which has the property that for any disturbance $d_0 \in D$, if $s_1 = x(\Delta)$ where $x(t)$ is the trajectory determined by (c_0, s_0, d_0), there is no $c_1 \in C$ such that $(c_1, s_1) \in T'$, then there is neighborhood U containing (c_0, s_0) such that for every $(\bar{c}_0, \bar{s}_0)$ in U and any disturbance $d_0 \in D$, if $\bar{s}_1 = x(\Delta)$ where $x(t)$ is the trajectory determined by $(\bar{c}_0, \bar{s}_0, d_0)$, then there is no $c_1 \in C$ such that $(c_1, \bar{s}_1) \in T'$. Let (c_0, s_0) be such a node and fix a disturbance d and let $s_1 = x(\Delta)$ where $x(t)$ is the trajectory determined by (c_0, s_0, d). Since T' is closed, for each $c \in C$, there is a neighborhood $V_c \times U_c \subseteq C \times S$ of (c, s_1) such that $T' \cap (V_c \times U_c) = \emptyset$. Thus the set of V_c's for $c \in C$ cover C and since C is compact, we can find $V_{c_1}, \ldots, V_{c_n}$ which cover C. It follows, that if $U_d = \bigcap_{i=1}^{n} U_{c_i}$, then $s_1 \in U_d$ and $C \times U_d$ is disjoint from T'. By the continuity of the map $(c_0, s_0, d) \to x(\Delta)$, there is a neighborhood $J_d \times K_d \times H_d \subseteq C \times S \times D$ of (c_0, s_0, d) such that for every

$(\bar{c}_0, \bar{s}_0, \bar{d}) \in J_d \times K_d \times H_d$, $\bar{x}(\Delta) \in U_d$ where $\bar{x}(t)$ is the trajectory determined by $(\bar{c}_0, \bar{s}_0, \bar{d})$. Now the set of H_d's for $d \in D$ cover D and since D is compact, we can find $H_{c_1}, \ldots, H_{c_m}$ which cover D. But then if $U = \bigcap_{i=1}^{m} J_{d_i} \times K_{d_i}$, it follows that $(c_0, s_0) \in U$ and for every $(c, s) \in U$ and every disturbance $d \in D$, if $s = x(\Delta)$ where $x(t)$ is the trajectory determined by (c, s, d), then $x(\Delta) \in \bigcup_{i=1}^{m} U_{d_i}$ and hence $C \times \{x(\Delta)\}$ is disjoint for T' as desired. This completes the proof and shows that under the hypothesis of the proposition, the abstract viability graph will be closed. □

6. Example, the leaky water tank

We use as our example a generalization of the water pump example of [1]. We generalize it in three ways. First, we allow bounded measurement error e in water level. Second, we permit a more elaborate dynamics for both refilling the tank and for the tank leak. Third, we monitor water level only every Δ units of time. That is, the system runs open loop inside the control intervals. The plant consists of a water pump and a leaking water tank. The set of plant states S is the set of pairs $s = (y, pmp)$, $y \geqslant 0$, $pmp \in \{pon, poff\}$. Here y is the water level. In pump state *pon* the pump is on, in pump state *poff* the pump is off. The dynamics of water level and leakage are supplied by

$$\dot{y} = \begin{cases} f_1(y) & \text{if the pump is on,} \\ f_2(y) & \text{if the pump is off,} \end{cases} \tag{1}$$

where f_1 and f_2 are continuous functions such that $0 < b_0 < f_1(y) \leqslant a_0$ for all y and $0 > -b_1 > f_2(y) \geqslant -a_1$ for all y, where $a_0 > b_0 > 0$ and $a_1 > b_1 > 0$.

Two numbers u, v are given, with $0 < u < v$. The water level $y(t)$ is required to be in the interval $[u, v]$ at all times. It is assumed that the error in measurement m of plant state y is at most e.

Thus the set of states S of the water level y is

$$[(y, pmp) \mid u - e \leqslant y \leqslant v + e,\ pmp \in \{pon, poff\}]$$

and the set of viable states **VS** of the water level y is

$$[(y, pmp) \mid u \leqslant y \leqslant v,\ pmp \in \{pon, poff\}].$$

There are only four control orders allowed, namely,

$$C = \{(pon, pon), (poff, poff), (pon, poff), (poff, pon)\}.$$

Here is what they do.

1. The control order (pon, pon) means that the pump will simply be instructed to stay on during the whole interval $[0, \Delta]$. Thus the control order (pon, pon) on $[0, \Delta]$, with initial state s_0 at the beginning of the interval and delay d, forces the water level trajectory $y(t)$ to satisfy

$$\dot{y} = f_1(y)$$

with $y(0)=s_0$ no matter what the delay. It is easy to check that our inequalities on $f_1(y)$ will ensure that $y(t)$ is a viable trajectory no matter what the delay $d \leqslant b < \Delta$ if s_0 is initially in the interval $[u, v-a_0 \cdot \Delta]$. Moreover, if s_0 is initially in the interval $S^i(pon, pon, s_0)=[u+e, v-a_0 \cdot \Delta - e]$, the water level trajectory $y(t)$ which satisfies

$$\dot{y}=f_1(y)$$

with $y(0)=\bar{s}$ where $|\bar{s}-s_0|<e$ will also be a viable trajectory on $[0, \Delta]$. Finally, it again easily follows from our inequalities on f_1 that $y(\Delta)$ must lie in the interval

$$S^f(pon, pon, s_0)=[s_0+b_0 \cdot \Delta, s_0+a_0 \cdot \Delta].$$

2. The control order (*poff*, *poff*) means that the pump will simply be instructed to stay off during the whole interval $[0, \Delta]$. Thus the control order (*poff*, *poff*) on $[0, \Delta]$ with initial state s_0 at the beginning of the interval and delay d forces the water level trajectory $y(t)$ to satisfy

$$\dot{y}=f_2(y)$$

with $y(0)=s_0$ no matter what the delay. It is easy to check that our inequalities on $f_2(y)$ will ensure that $y(t)$ is a viable trajectory, no matter what the disturbance $d \leqslant b < \Delta$ if s_0 is initially in the interval $[u+a_1 \cdot \Delta, v]$. Moreover, if s_0 is initially in the interval $S^i(poff, poff, s_0)=[u+a_1 \cdot \Delta + e, v-e]$, the water level trajectory y(t) which satisfies

$$\dot{y}=f_1(y)$$

with $y(0)=\bar{s}$ where $|\bar{s}-s_0|<e$ will also be a viable trajectory on $[0, \Delta]$. Finally, it again easily follows from our inequalities on $f_2(y)$ that $y(\Delta)$ must lie in the interval

$$S^f(poff, poff, s_0)=[s_0-a_1 \cdot \Delta, s_0-b_1 \cdot \Delta]$$

3. The control order (*pon*, *poff*) means that if at the start of the interval the pump is on, then the pump will be turned off after some delay $d \leqslant b$. Thus the control order (*pon*, *poff*) on $[0, \Delta t]$ with initial state s_0 at the beginning of the interval and delay d forces the water level trajectory $y(t)$ to satisfy

$$\dot{y}=\begin{cases} f_1(y) & \text{for } 0 \leqslant t < d, \\ f_2(y) & \text{for } d \leqslant t \leqslant \Delta, \end{cases}$$

where $y(0)=s_0$. It is easy to check that our inequalities on $f_1(y)$ and $f_2(y)$ will ensure that $y(t)$ is a viable trajectory, no matter what the delay $d \leqslant b$ if s_0 is initially in the interval $[u+a_1 \cdot \Delta, v-a_0 \cdot b]$ Moreover, if s_0 is initially in the interval $S^i(pon, poff, s_0)=[u+a_1 \cdot \Delta t + e, v-a_0 \cdot b - e]$, the water level trajectory $y(t)$ which satisfies

$$\dot{y}=\begin{cases} f_1(y) & \text{for } 0 \leqslant t < d, \\ f_2(y) & \text{for } d \leqslant t \leqslant \Delta \end{cases}$$

with $y(0)=\bar{s}$ where $|\bar{s}-s_0|<e$ and $0\leqslant d\leqslant b$ will also be a viable trajectory on $[0,\Delta]$. Finally, it again easily follows from our inequalities on $f_1(y)$ and $f_2(y)$ that the $y(\Delta)$ must lie in the interval

$$S^f(pon, poff, s_0)=[s_0-a_1\cdot\Delta, s_0+a_0\cdot b].$$

4. The control order (*poff*, *pon*) means that if at the start of the interval the pump is off, then the pump will be turned on after some delay $d\leqslant b$. Thus the control order (*poff*, *pon*) on $[0,\Delta t]$ with initial state s_0 at the beginning of the interval and delay d forces the water level trajectory $y(t)$ to satisfy

$$\dot{y}=\begin{cases} f_2(y) & \text{for } 0\leqslant t<d,\\ f_1(y) & \text{for } d\leqslant t\leqslant\Delta,\end{cases}$$

where $y(0)=s_0$. It is easy to check that our inequalities on $f_1(y)$ and $f_2(y)$ will ensure that $y(t)$ is a viable trajectory, no matter what the disturbance $d\leqslant b$ if s_0 is initially in the interval $[u+a_1\cdot b, v-a_0\cdot\Delta]$. Moreover, if s_0 is initially in the interval $S^i(pon, poff, s_0)=[u+a_1\cdot b+e, v-a_0\cdot\Delta-e]$, the water level trajectory $y(t)$ which satisfies

$$\dot{y}=\begin{cases} f_2(y) & \text{for } 0\leqslant t<b,\\ f_1(y) & \text{for } d\leqslant t\leqslant\Delta\end{cases}$$

with $y(0)=\bar{s}$ where $|\bar{s}-s_0|<e$ and $0\leqslant d\leqslant b$ will also be a viable trajectory on $[0,\Delta]$. Finally, it again easily follows from our inequalities on $f_1(y)$ and $f_2(y)$ that the $y(\Delta)$ must lie in the interval

$$S^f(pon, poff, s_0)=[s_0-a_1\cdot b, s_0+a_0\cdot\Delta].$$

If we write $c_0=(poff, pon)$, then we will let $S^f(c_0, s_0)$ denote $S^f(poff, pon, s_0)$, etc. With this notation, we see that $((c_0, s_0),(c_1, s_1))$ is an abstract e-sturdy local viability node if

1. s_0 is in $S^i(c_0, s_0)$;
2. $(c_1, s_1)\in S^i(c_1, s_1)$;
3. $s_1\in S_f(c_0, s_0)$ and there is a delay $d\leqslant b<\Delta$ such that the trajectory $y(t)$ determined by c_0, s_0, and d has $y(\Delta)=s_1$.

Remark

1. If we endow our finite set of controls C with the discrete topology, then the sturdy local viability graph is the same as the e-sensor local viability graph and hence the sturdy viability graph is the same as the e-sensor viability graph.
2. Consider now the special case considered in [1]. This is the case when f_1, f_2 are constant functions. In this special case, translating the characterization into inequalities and manipulating, one can tediously write out the inequalities characterizing the nodes and branches of the e-sensor local viability graph. Alternatively, due to the fact that only simplifying linear inequalities are involved, we can avoid writing this out and derive existence and algorithms from either the real linear programming algorithm or, equivalently, from the Tarski decision method for

dense linearly ordered abelian groups, to see that in principle these inequalities can be computed. The situation is the same for the condition that an edge be in the e-sensor viability graph.

Of course if Δ is too large, the e-sensor viability graph may be empty. We need to determine those Δ for which the e-sensor viability graph using Δ is non-empty.

Instead of a direct attack on this problem, we shall instead define a class of finite state control automata $A(g,h), 0<g<h$. We show there are choices of g, h, Δ such that all runs of the automaton $A(g,h)$ yield paths in the e-sensor viability graph, thus showing the e-sensor viability graph is non-empty and that in this case, viability can be enforced by a finite state control automaton. The automaton changes state only at times $n\Delta$, when a certain test is satisfied by the measurement of plant state m at $n\Delta$. It instantaneously changes state and issues to the pump one of the four control orders c above, executed as described above. The intention is that if c is of the form $(pon, ?)$, then the pump was on at time $n\Delta$, that is, the state of the pump at that time was *son*. If c is of the form $(poff, ?)$, then the pump was off at time $n\Delta$, that is, the state of the pump at time $n\Delta$ was *soff*.

We proceed to define an automaton $A(g,h)$ for each pair of real parameters g, h such that $0<g<h$. This is the sequential automaton below.

1. The input alphabet is the non-negative reals, regarded as measurements m of plant state y.
2. The internal states are the two element set $\{son, soff\}$
3. The output alphabet is the set of controls

$$\{(pon, pon), (poff, poff), (pon, poff)), (poff, pon)\}.$$

4. In real time the automaton does not receive input or change state except at times $n\Delta$.

A. Suppose the automaton is in state *son* at time $n\Delta$ and the measurement of plant state is m. Then instantaneously,

1. if $m \geqslant h$, the automaton outputs $(pon, poff)$ and shifts its state to *soff*,
2. if $m<h$, then the automaton remains in state *son* and outputs (pon, pon).

B. Suppose that the automaton is in state *soff* and receives input measurement m at time $n\Delta$. Then, instantaneously,

1. if $m \leqslant g$, then the automaton outputs $(poff, pon)$ and shifts to state *son*, and
2. if $m>g$, then the automaton remains in state *soff* and outputs $(poff, poff)$.

We seek values for Δ and the parameters g, h which ensure that when controls are chosen by the automaton, the water level $y(t)$ is viable no matter what the disturbances $0 \leqslant d_0, d_1, \ldots \leqslant b$ encountered in the successive intervals $[n\Delta, (n+1)\Delta]$. This will prove the e-sensor viability graph is non empty.

Necessary conditions. We can derive necessary conditions on the parameters g and h to guarantee that the control automaton $A(g,h)$ produces only viable trajectories by

analyzing the plant trajectories for given input measurements and states of $A(g,h)$. Consider the following two cases.

Case 1: Suppose that the plant state is *son* at time $t_k = k\Delta$ and the control automaton at that time receives measurement $m_k < h$. By the assumption on e, if the actual water level at time t_k is $y(t_k)$, then

$$y(t_k) - e \leqslant m_k \leqslant y(t_k) + e.$$

Thus

$$m_k - e \leqslant y(t_k) \leqslant m_k + e.$$

Also suppose that the pump is on at time t_k. In this case the automaton remains in state *son* and the pump remains on for the next Δ seconds. Then, since the plant trajectory $y(.)$ between t_k and $t_{k+1} = t_k + \Delta$ must satisfy

$$0 < b_0 \leqslant \dot{y}(t) \leqslant a_0,$$

it is easy to see that $y(t)$ is a strictly increasing function in this interval and that

$$y(t_{k+1}) \leqslant y(t_k) + a_0\Delta \leqslant m_k + a_0\Delta + e \leqslant h + a_0\Delta + e.$$

Now if we find that the measurement received at time t_{k+1}, m_{k+1}, is still less than h, then of course the automaton will continue to be in state *son*, so that the pump will remain on, the plant trajectory $y(.)$ between t_{k+1} and t_{k+2} will be strictly increasing, and $y(t_{k+2}) \leqslant h + a_0\Delta + e$. We continue on this way until we find the least $l > k$ such that the measurement received at time t_l is greater than or equal to h. By our analysis, the actual plant state $y(t_l)$ will be bounded by $h + a_0\Delta + e$. At that time the automaton will output (*pon*, *poff*) which in effect orders that the pump be turned off and the state be switched to state *soff*. What happens to the trajectory $y(t)$ between times t_l and $t_{l+1} = t_l + \Delta$? There exists a $\tau_l \leqslant b < \Delta$ such that the trajectory satisfies

$$0 < b_0 \leqslant \dot{y}(t) \leqslant a_0 \qquad \text{if } t_l \leqslant t \leqslant t_l + \tau_l,$$

$$0 > -b_1 \geqslant \dot{y}(t) \geqslant -a_1 \quad \text{if } t_l + \tau_l < t \leqslant t_{l+1}.$$

Then trajectory $y(t)$ over interval $[t_l, t_{l+1}]$ reaches its maximum at time $t = t_l + \tau_l$. This maximum value is bounded by $y(t_l) + a_0\tau_1 \leqslant y(t_l) + a_0 b \leqslant h + a_0\Delta + e + a_0 b$. After time $t_l + \tau_l$, $y(t)$ is strictly decreasing for the rest of the interval. Pick h so that

$$h + a_0 b + a_0\Delta + e \leqslant v.$$

This ensures that if we use the control automaton $A(g,h)$, the water level never becomes greater than v. There is also a lower bound imposed on h derived from the fact that the minimum value of $y(t)$ in the interval $[t_l, t_{l+1}]$ must be greater than or equal to u. Since we assume that $m_l \geqslant h$, we know that $y(t_l) \geqslant h - e$. If we assume that there is no delay in turning the pump off, then $y(t)$ could be strictly decreasing in the interval. It is easy to see in that situation that $y(t_{l+1})$ could be as small as $h - e - a_1\Delta$. Moreover, it could be that $h - e - a_1\Delta - e \leqslant g$ so that $m_{l+1} \leqslant g$. In that situation the pump will be off and our control automaton will tell the pump to turn on. However,

there could be a maximum delay of time b before the pump turns on and the water level once again starts to increase. Thus, there could be a further drop of $a_1 b$ in the water level during this delay so that the water level could become as small as $h-e-a_1\Delta-a_1 b$. Thus, we must also assume that $h-a_1 b-a_1\Delta-e\geqslant u$ or equivalently that $u+a_1 b+a_1\Delta+e\leqslant h$. In Case 2, we will deal with the case when $m_{l+1}>g$.

Case 2: Suppose that at time t_k the plant state is *soff* and the control automaton receives a measurement $m_k>g$. Again, the actual water level $y(t_k)$ satisfies

$$m_k-e\leqslant y(t_k)\leqslant m_k+e.$$

Assume also that the pump is off at time t_k. Then the automaton remains in state *soff* and the pump remains off for the next Δ time period. Then, since the plant trajectory $y(.)$ between t_k and $t_{k+1}=t_k+\Delta$ must satisfy

$$0>-b_1\geqslant\dot{y}(t)\geqslant -a_1,$$

$y(t)$ is a strictly decreasing function in this interval and

$$y(t_{k+1})\geqslant y(t_k)-a_1\Delta\geqslant m_k-a_1\Delta-e\geqslant g-a_1\Delta-e.$$

If we find that the measurement received at time t_{k+1}, m_{k+1}, is still greater than g, then the control automaton will continue to be in state *soff* and the pump will remain off. The plant trajectory $y(.)$ between t_{k+1} and t_{k+2} will be strictly decreasing, and $y(t_{k+2})\geqslant g-a_1\Delta-e$. We continue on this way until we find the least $l>k$ such that the measurement received at time t_l is less than or equal to g. By our analysis, the actual plant state $y(t_l)$ is bounded below by $g-a_1\Delta-e$. At that point, the automaton will output (*poff*, *pon*) which in effect issues the order for the pump to be turned on and the state to be switched to state *son*. Again we can analyze what happens to the trajectory $y(t)$ between times t_l and $t_{l+1}+\Delta$. There exists a $\tau_l\leqslant b<\Delta$ such that the trajectory satisfies

$$0>-b_1\geqslant\dot{y}(t)\geqslant -a_1 \quad \text{if } t_l\leqslant t\leqslant t_l+\tau_l,$$

$$0<b_0\leqslant\dot{y}(t)\leqslant a_0 \quad \text{if } t_l+\tau_l<t\leqslant t_{l+1}.$$

Then the trajectory $y(t)$ over the interval $[t_l, t_{l+1}]$ reaches its minimum at time $t=t_l+\tau_l$. This minimum value is bounded below by $y(t_l)-a_1\tau_1\geqslant y(t_l)-a_1 b\geqslant g-a_1\Delta-e-a_1 b$. Then, after time $t_l+\tau_l$, $y(t)$ is strictly increasing. If we pick g so that

$$g-a_1 b-a_1\Delta-e\geqslant u,$$

then by using the control automaton, the water level never becomes less than u. There is also upper bound on g which comes from the fact that the maximum value of $y(t)$ in the interval $[t_l, t_{l+1}]$ must be less than or equal to v. Since we are assuming that $m_l\leqslant g$, we know that $y(t_l)\leqslant g+e$. If we assume that there is no delay in turning the pump on, then $y(t)$ could be strictly increasing in the interval. In this situation, $y(t_{l+1})$ could be as large as $g+e+a_0\Delta$. Note that the case when $m_{l+1}<h$ was handled in Case 1. However, it could be that $g+e+a_0\Delta+e\geqslant h$ so that $m_{l+1}\geqslant h$. In that situation, the

pump will be on and our controller will tell the pump to turn off. However, there could be a maximum delay of time b before the pump turns off and the water level once again starts to decrease. Thus, there could be a further rise of $a_0 b$ in the water level during this delay. The water level could become as large as $g+e+a_0\Delta+a_0 b$. Thus, we must also assume that $g+a_0 b+a_0\Delta+e\leqslant v$ or equivalently that $g\leqslant v-a_0 b-a_0\Delta-e$.

This ends the discussion of necessary conditions. When turned around as sufficient conditions we get the following proposition.

Proposition 6.1. *Suppose we are given a maximum delay of b, $a\Delta>b>0$ and a measurement error bound $e\geqslant 0$. Choose the numbers (g,h) so that*

$$u+a_1\cdot b+a_1\cdot\Delta+e\leqslant g<h\leqslant v-a_0\cdot b-a_0\cdot\Delta-e.$$

Suppose that either the initial water level is between $u+e$ and $v-a_0 b$ and the pump is on, or alternately that the initial water level is between $u+a_1\cdot b$ and $v-e$ and the pump is off. Suppose that initially the pump and the control automaton are both in the "on" state or both in the "off" state. If the automaton $A(g,h)$ makes a run with the initial conditions, the water level $y(t)$ is viable no matter what the disturbances.

Proof. Using the facts established in Cases 1 and 2 we can prove Proposition 6.1 in a straightforward manner. One simply proceeds by induction on k to prove that if we follow the policy associated with $A(g,h)$, then in each successive interval $[t_k,t_{k+1}]$, the trajectory of the plant $y(t)$ will always satisfy $u\leqslant y\leqslant v$. See also [29]. □

Remark

1. By picking $\Delta>h$, we guarantee that if initially the plant state and the initial state of $A(g,h)$ are such that:
   (a) If the initial state of $A(g,h)$ is *soff*, then the pump is off, and
   (b) if the initial state of $A(g,h)$ is *son*, then the pump is on,
   then at some time before the end of each $(n\Delta,(n+1)\Delta)$ interval, the automaton state and the state of the pump will necessarily correspond.
2. The inequalities on g and h in this proposition automatically impose the following upper bound on the size of the control interval Δ:

$$\Delta\leqslant\frac{v-u-b(a_0+a_1)-2e}{a_0+a_1}.$$

3. To avoid having the pump continually alternating between the states *son* and *soff* in each pair of successive intervals one should also ensure that there is a sufficient distance between g and h. However, we will not deal with this issue here.
4. If we strengthen the hypothesis of Proposition 6.1 to assume that g and h satisfy

$$u+a_1\cdot b+a_1\cdot\Delta+e<g<h<v-a_0\cdot b-a_0\Delta-e,$$

then we can modify the inequalities in the definition of the $A(g,h)$-automaton and Proposition 6.1 will continue to hold. For example, we could change conditions **A**–**B** to read

A*. Suppose the automaton is in state *son* at time $n\Delta$ and the measurement of plant state is m. Then instantaneously,
(a) if $m>h$, the automaton outputs (*pon*, *poff*) and shifts its state to *soff*, and
(b) if $m\leqslant h$, then the automaton remains in state *son* and outputs (*pon*, *pon*).

B*. Suppose that the automaton is in state *soff* and receives input measurement m at time $n\Delta$. Then, instantaneously,
(a) if $m<g$, then the automaton outputs (*poff*, *pon*) and shifts to state *son*, and
(b) if $m\geqslant g$, then the automaton remains in state *soff* and outputs (*poff*, *poff*).

The non-deterministic automaton $NDA(g,h)$. The argument above succeeds even though water level y and the measurement m input to automaton $A(g,h)$ in state *son* with water level y can differ by up to e. This shows that in the definition of the output value of $A(g,h)$ when it is in state *son* and the water level is y where y in the interval $h-e\leqslant y\leqslant h+e$, whether we define that output value as (*pon*, *pon*) or (*pon*, *poff*) does not affect viability. Similarly, in the definition of the output value of $A(g,h)$ when it is in state *soff* and the water level is y, if y is in the interval $g-e\leqslant y\leqslant g+e$, whether we define that output value as (*poff*, *pon*) or (*poff*, *poff*) does not affect viability. This allows us to define a non-deterministic automaton $NDA(g,h)$, guaranteeing viability, which combines the non-deterministic sensor which maps y to m with the sequential automaton $A(g,h)$, and which has additional allowed transitions for the two cases alluded to in the previous paragraph. The input alphabet of $NDA(g,h)$ is the set of water level values. $NDA(g,h)$ is defined as follows.

A. Suppose the automaton $NDA(g,h)$ is in state *son* at time $n\Delta$ and the water level is y. Then instantaneously,
1. If $y>h+e$, the automaton shifts its state to *soff* with output value $c_1=(pon, poff)$.
2. If $y<h-e$, the automaton remains in state *son* with output value $c_1=(pon, pon)$.
3. If $h-e\leqslant y\leqslant h+e$, the automaton may remain in state *son* with output value $c_1=(pon, pon)$ or shift into state *soff* with output value $c_1=(pon, poff)$.

B. Suppose that the automaton $NDA(g,h)$ is in state *soff* at time $n\Delta$ and the water level is y. Then, instantaneously,
1. If $y>g+e$, then the automaton remains in state *soff* with output value $c_1=(poff, poff)$.
2. If $y<g-e$, then the automaton shifts to state *son* with output value $c_1=(poff, pon)$.
3. If $g-e\leqslant y\leqslant g+e$, then the automaton may equally well remain in state *soff* with output value $c_1=(poff, poff)$, or shift to state *son* with output value $c_1=(poff, pon)$.

Here is the control policy $P(NDA(g,h))$ corresponding to the automaton $NDA(g,h)$. It consists of those triples (c_0, y, c_1) described in the six clauses below, one for each of the six clauses in the definition of $P(NDA(g,h))$. The question mark in the clause is a variable ranging over the two element set $\{pon, poff\}$.

1. If $y > h+e$, then $(?, pon), y, (pon, poff)) \in P(NDA(g,h))$.
2. If $y < h-e$, then $((?, pon), y, (pon, pon)) \in P(NDA(g,h))$.
3. If $h-e \leqslant y \leqslant h-e$, then $((?, pon), y, (pon, pon)) \in P(NDA(g,h))$ and $((?, pon), y, (pon, poff) \in P(NDA(g,h))$.
4. If $y > g+e$, then $((?, poff), y, (poff, poff)) \in P(NDA(g,h))$.
5. If $y < g+e$, then $((?, poff), y, (poff, pon)) \in P(NDA(g,h))$.
6. If $g-e \leqslant y \leqslant g+e$, then $((?, poff), y, (poff, poff)) \in P(NDA(g,h))$ and $((?, poff), y, (poff, pon)) \in P(NDA(g,h))$.

By the same analysis as above we get the following proposition.

Proposition 6.2. *Suppose we are given a maximum delay of $b, a\Delta > b > 0$ and a measurement error bound $e \geqslant 0$. Suppose that the numbers (g,h) satisfy*

$$u + a_1 \cdot b + a_1 \cdot \Delta + e < g < h < v - a_0 \cdot b - a_0 \cdot \Delta - e.$$

Let

$$A = \{(pon, pon)\} \times [u, h+e] \cup \{(pon, poff)\} \times [g-e, v-a_0 b] \cup \{(poff, poff)\}$$
$$\times [g-e, v] \cup \{(pon, pon)\} \times [u + a_1 b, h+e].$$

Suppose that (c_0, s_0) is in A and (c_0, s_1, c_1) is in $P(NDA(g,h))$. Then for any admissible disturbance over interval Δ, if $x(t)$ is the trajectory over Δ for that disturbance starting from state s_0 using control c_0 and $s_1 = x(\Delta)$, then $x(t)$ is viable and (c_1, s_1) is also in A.

7. Finite coverings and finite automata

Suppose we are given a simple hybrid system with control intervals

$$[n\Delta, (n+1)\Delta],$$

$n = 0, 1, \ldots$. As we have said, the abstract viability graph, if non-empty is a non-deterministic automaton which enforces viability if started on a node of the abstract viability graph. If we knew how to implement the abstract viability automaton, we could enforce that all trajectories produced on $[0, \infty]$ are viable no matter what the disturbance as long as the automaton is started in a state s_0 with a control c_0 such that there is a node (c_0, s_0) of the abstract viability graph. But this automaton has been obtained by a pure mathematical fixed point argument with little constructive content. This automaton generally has a highly non-constructive transition relation.

We want to investigate cases when there is a finite state control automaton guaranteeing viable trajectories over $[0, \infty]$. In principle, these finite automata are

implementable. Here we prove a simple theorem ensuring finite automata. It is suited to applications where the set of controls is finite. Theorems with weaker hypotheses for the case of compact controls are deferred to a sequel.

Proposition 7.1. *Suppose that S is the set of plant states, C is the set of controls, and* **VS** *is the set of viable states. Suppose also that*

1. *R is a non-empty closed subset of the abstract viability graph.*
2. *For any* $(\bar{c}_0, \bar{s}_0) \in R$ *and any disturbance* $d \in D$, *if* $x(t)$ *is the resulting trajectory, there exists a* $\bar{c}_1 \in C$ *such that* $(\bar{c}_1, x(\Delta)) \in R$. *(Note in the language of Section 4, this say that R is a fixed point of the operator F.)*
3. *The spaces, S, C are compact metric spaces.*
4. *The viability set* **VS** *is a closed subset of S.*
5. *Let* $R_0 R_1$ *be, respectively, the projections of R on its first coordinate* c_0 *and on its second coordinate* s_0. *Assume that R has the following "sturdiness property".*
   For any $r = (c_0, s_0)$ *in R, there exists a pair of open sets* $U_r \subseteq C$, $V_r \subseteq$ **VS**, *such that* $(c_0, s_0) \in U_r \times V_r$ *and* $(U_r \cap R_0) \times (V_r \cap R_1) \subseteq R$.

Then there exist finite state control automata which, regarded as control policies, have infinite policy paths which are policy paths of R. That is, these are finite state automata which can produce viable trajectories from certain initial conditions no matter what the disturbance. (We do not assert that every policy path of R is a policy path for the automaton.)

Proof. Property (2) ensures that we can construct an R-automaton from R exactly as we constructed the local viability automaton from the local viability graph. The R-automaton is an automaton with the desired property, but is usually is not a finite state automaton. So the object is to replace it by a finite state automaton which approximates it. We base our construction of a finite automaton which ensures viable plant trajectories on a choice of bases for the open sets of the topological spaces involved, see [25, Appendix II]. Suppose that we are given a basis B_S for the open sets for S and a basis B_C for the open sets of C. Suppose that we use the product basis $B_C \times B_S$ as a basis of open sets for the product space $C \times S$. That is, the basis of open sets for $C \times S$ consists of products $U \times V$, where $U \in B_C$ and $V \in B_S$. We use as a base for **VS** the intersections of the base sets for S with **VS** and similarly for R_0 and R_1. Note that $R \subseteq R_0 \times R_1 \subseteq C \times$ **VS**.

Note that since the projection of a closed compact set is closed and compact, both R_0 and R_1 are closed and compact. Clearly, $R \subseteq R_0 \times R_1$. As r ranges over, the open sets $(U_r \cap R_0) \times (V_r \cap R_1)$ of $C \times$ **VS** cover the compact set $R_0 \times R_1$. Therefore, there is a finite sequene $r_1, \ldots, r_n \in R$ such that the $(U_{r_1} \cap R_0) \times (V_{r_1} \cap R_1), \ldots, (U_{r_n} \cap R_0) \times (V_{r_n} \cap R_1)$ cover $R_0 \times R_1$.

The $(U_{r_i} \cap R_0)$ generate a finite subtopology of the topology on R_0, which we call the small topology on R_0. Let $U_1, \ldots, U_k$ be a list of all distinct non-empty join irreducibles for the small topology on R_0. We choose the set $\{U_1, \ldots, U_k\}$ as the set of states for our desired finite state control automaton. Similarly, the $V_{r_i} \cap R_1$ generate

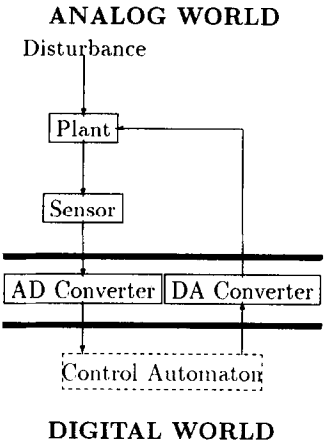

Fig. 2.

a finite topology R_1, which we call the small topology on R_1. Let $V_1, \ldots, V_p$ be a list of all distinct non-empty join irreducibles of the small topology on R_1. We choose the set $\{V_1, \ldots, V_p\}$ as the set of input symbols of the finite state control automaton. The set $\{c_1, \ldots, c_p\}$ will be our output alphabet for the finite control automaton where $U_{r_{k_1}} \times V_{r_{k_1}}, \ldots, U_{r_{k_p}} \times V_{r_{k_p}}$ are such that $V_{r_{k_i}} \subseteq V_i$ and $r_{k_i} = (c_i, s_i)$ for $i = 1, \ldots, p$.

For each pair (U_i, V_j) consisting of an automaton state and an automaton input symbol, we define the new state and the output as follows. Let $U_{r_{k_i}} \times V_{r_{k_i}}$ and $U_{r_{k_j}} \times V_{r_{k_j}}$ be such that $U_{r_{k_i}} \subseteq U_i$ and $V_{r_{k_j}} \subseteq V_j$. Then the new state of the automaton is the join irreducible that contains c_j and the output symbol is c_j (see Fig. 2).

It is easy to see that this automaton assures viability of the trajectory for any admissible disturbance when started in state s_0 using control c_0 for which there is a (c_0, s_0) in R, This was our objective, and ends the proof. □

Remarks

1. This presentation makes the analog to digital converter of [25] defined on R_0 rather than S, since the intention is to have all execution sequences policy edges of R. With this in mind, we think of plant state s in R_0 as being converted by the analog to digital converter into the join irreducible V_i containing s. This is then an input symbol to the finite automaton. If one prefers to have the analog to digital converter defined on all of S, then adds a letter $\perp$ to the input alphabet for the automaton and regard any s in the open set $S - R_0$ as converted to $\perp$. In this case, we should also add $\perp$ to the automaton output alphabet to be used when the input symbol is $\perp$.
2. Note that the finite automaton constructed in Proposition 7.1 may be non-deterministic since for a given V_j there may be several $U_{r_{k_i}} \times V_{r_{k_i}}$'s such that $V_{r_{k_i}} \subseteq V_j$.
3. We can easily generalize to cover e-sensor and sturdy viability graphs.

Example. The leaky water tank revisited. We briefly indicate how the theorem applies to the leaky water tank example. Our set R is the set A as defined in Proposition 6.2. $S = \mathbf{VS} = [u, v]$ with the standard topology and $C = \{(pon, pon), (pon, poff), (poff, pon), (poff, poff)\}$ with the discrete topology. Clearly, A is a closed subset of $C \times S$. The non-deterministic automaton $NDA(g, h)$ and corresponding $P(NDA(g, h))$ shows that A is a subset of the abstract viability graph which satisfies conditions (2) and (5) of Proposition 7.1, see [29] for a detailed proof.

8. Viability in autonomous hybrid control

We discuss the relation of viability to Kohn–Nerode autonomous hybrid control [26] based on relaxed calculus of variations. Disturbances are hereby omitted from the model. The informal justification for this omission is that small disturbances are reflected in small deviations in plant state, and that the analog to digital converters of [25] work correctly in spite of small deviations in plant state. So as long as the plant state fluctuations resulting from the disturbances are within the error tolerated by the analog to digital plant sensor, the system still operates as specified. We do not give detailed conditions for the validity of the control automaton extraction process outlined below. We do remark that we assume that the Lagrangian $L(x, u, c)$ is lower semicontinuous and that the control and state spaces are compact [10]. The purpose of the outline below is to indicate how to show a viability graph is non-empty in applications of autonomous hybrid control. We are using this extraction process for a number of systems, examples to be published in later papers.

1. Corresponding to the autonomous problem, we formulate a non-negative Lagrangian $L(x, u, c)$ on plant state trajectories which results from applying possible control functions c of time to produce the trajectory, in such a way that the smaller the value of its integral along a trajectory, the better the performance. (Here u plays the role of $\dot{x}$.) The original Lagrangian is usually non-convex in u. The performance specification for the plant is reformulated as the requirement that the plant state trajectory is viable with respect to a viability set and has an integral within a prescribed ε of its minimum. A control function of time that does this from a given initial state is called an ε-optimal control function. That is all that is required in actual applications.
2. Second, $L(x, u, c)$ is convexified to get an $L^{**}(x, u, c)$ which is convex in u. The main existence theorem of the relaxed calculus of variations is applied to such convexified problems to find a measure-valued control function of time $c(t)$ resulting in a state trajectory $x(t)$ minimizing the integral of $L^{**}(x, u)$ on the trajectory. This is a so-called relaxed control function as introduced by Young [36] and Warga [35]. The measures are on the space of control values. Because our control problem is autonomous, this control function of time for each state can be converted by Bellman's dynamic programming method to a control policy, a function of state that tells what control to use in each state. (Most optimal measure-valued control

policies are not directly implementable, a fact that led many experts to neglect relaxed control for twenty years till Kohn developed his declarative control [13–17, 19–21].)

3. The method of covers used in [25, Appendix II] and in this paper can then be used to extract a finite cover for this optimal control policy. When this finite cover is implemented as a finite state control automaton with analog to digital and digital to analog converters according to [25, Appendix II], the resulting control automaton enforces a control policy which assures ε-optimal control. A length Δ for control intervals has to be extracted at the same time. The control policy is, in the language of this paper, a function of state s_0 and previous control c_0.
4. The control values issued by the finite automaton, finite in number, can be taken as measures on the space of controls with finite support. Each of these measures represents a finite chattering control built from a finite set of controls altogether. In [10], Caratheodory's theory on convex sets is used to see that the chattering control which approximates to the optimal control minimizing the convexification $L^{**}(x,u)$ corresponds to expressing the absolute minimum for the convexified problem as a convex combination of some local minima for the original non-convex problem. See also [7, 8].
5. If the optimal policy is jointly continuous in state and control and the viability set has a non-empty interior, then the viability kernel can often be proved to be non-empty.
6. If a control policy making the system meet performance requirements is obtained from any other control theory, the same outline can be used to prove viability kernels non-empty and to prove the existence of finite control automata which enforce viable trajectories.
7. This is in accord with the heuristic principle that to prove viability kernels non-empty, whether for conventional [5] or hybrid systems, it is at present usually necessary to have a construction of feedback control laws to which we can approximate. But, given this, one still needs compactness and uniformity in parameters to show that a viability graph is non-empty.

Remark

1. The question as to how, given an ε, to choose a Δ so as to be sure of being able to compute a finite automaton with Δ length control intervals which achieves ε-optimality is a major one, see [10] for a general algorithm with roughly the hypotheses of [7].
2. In the outline above disturbances are not modelled. When disturbances are included and the controls are required to succeed no matter what the disturbances, relaxed control is less well developed subject; see [35], last chapter. But what is clear is that if one insists that the controls chosen by the control policy not be very sensitive to small changes in previous state and control, then one is faced with choosing control automata and a control interval length Δ which execute a policy so that the policy edges are in the viability graph. So it seems that it is important to understand these graphs.

Acknowledgment

We are much indebted to A. Benveniste for several helpful suggestions.

References

[1] R. Alur, C. Courcoubetis, T. Henzinger and Pei-Hsin Ho, Hybrid Automata: an algorithmic approach to the specification and verification of hybrid systems, in [11].
[2] J.P. Aubin, *Convex Analysis and Optimization* (Pitman, London, 1982).
[3] J.P. Aubin, *Differential Inclusions, Set Values Maps and Viability* (Springer, Berlin, 1984).
[4] J.P. Aubin, *Set Valued Analysis* (Birkhauser, Basel, 1990).
[5] J.P. Aubin, *Viability Theory* (Birkhauser, Basel, 1991).
[6] J.P. Aubin and I. Ekeland, *Applied Non-Linear Analysis* (Wiley, New York, 1984).
[7] I. Ekeland, *Infinite Dimensional Optimization and Convexity*, University of Chicago Lecture Notes in Mathematics (University of Chicago Press, Chicago, 1983.)
[8] I. Ekeland and R. Teman, *Convex Analysis and Variational Problems* (Elsevier, New York, 1976).
[9] A.F. Filippov, *Differential Equations with Discontinuous Right Hand Part* (Kluwer Academic Publishers, Dordecht, 1988).
[10] X. Ge, W. Kohn and A. Nerode, Algorithms for chattering approximations to relaxed optimal controls, MSI Tech. Report 94-23, 1994.
[11] R. Grossman, A. Nerode, H. Rischel and A. Ravn (eds.), *Hybrid Systems*, Springer Lecture Notes in Computer Science (Springer, Berlin, 1993).
[12] J. Guckenheimer and A. Nerode, Simulation for hybrid and nonlinear control, in: *Proc. IEEE 31st CDC*, Vol. 3 (1992) 2981–2983.
[13] W. Kohn, A Declarative theory for rational controllers, in: *Proc. 27th CDC* (1988) 130–136.
[14] W. Kohn, Hierarchical control systems for autonomous space robots, in: *Proc. AIAA* (1988).
[15] W. Kohn, Application of declarative hierarchical methodology for the flight telerobotic services, Boeing Document G-6630-061, Final Report of NASA-Ames Research Service request 2072, Job Order T1988 (1988).
[16] W. Kohn, The rational tree machine: technical description and mathematical foundations, IR and DBE-499, Tech. Document 905-10107-1, Boeing Computer Services (1989).
[17] W. Kohn, Rational algebras: a constructive approach, IR and DBE-499, Tech. Document D-905-10107-2 (1989).
[18] W. Kohn, Cruise missile mission planning: a declarative control approach, Boeing Computer Services Tech. Report (1989).
[19] W. Kohn, Declarative multiplexed rational controllers, in: *Proc. 5th IEEE Internat. Symp. Intelligent Cont.* (1990) 794–803.
[20] W. Kohn, Advanced architecture and methods for knowledge-based planning and declarative control, Boeing Computer Services Tech. Document IRD BCS-021 in ISMIS 91 (1990).
[21] W. Kohn and K. Carlsen, Symbolic design and analysis in control, in: *Proc. 1988 Grainger Lecture Series*, U. of Illinois (1989) 40–52.
[22] W. Kohn and A. Murphy, Multiple agent reactive shop control, *ISMIS 91.*
[23] W. Kohn and A. Nerode, An autonomous control theory: an overview, in: *Proc. IEEE CACSD92, Napa Valley* (1992).
[24] W. Kohn and A. Nerode, Multiple agent autonomous control systems, in: *Proc. 31st IEEE CDC Tucson*, AZ (1993) 2956–2966.
[25] W. Kohn and A. Nerode, Models for hybrid systems: automata, topologies, controllability, observability, in [11].
[26] W. Kohn and A. Nerode, Multiple agent autonomous control, a hybrid systems architecture, *Logical Methods* (Birkhauser, Basel, 1993).
[27] A. Nerode, General topology and partial recursive functionals, in: *Summaries of Talks at the AMS Summer Institute in Mathematical Logic*, Cornell (1957).

[28] A. Nerode, J.B. Remmel and A. Yakhnis, Hybrid systems and continuous sensing games in: *Proc. 9th IEEE Conf. on Intelligent Control* (1993).
[29] A. Nerode, J.B. Remmel and A. Yakhnis, Hybrid system games: extraction of control automata with small topologies, MSI Tech. Report 93–102 (1993).
[30] A. Nerode and A. Yakhnis, Modelling hybrid systems as games, CDC92 (1992) 2947–2952.
[31] A. Nerode and A. Yakhnis, Hybrid games and hybrid systems, MSI Tech. Report 93–77 (1993).
[32] A. Nerode and A. Yakhnis, Control automata and fixed points of set-valued operators for discrete sensing hybrid systems, MSI Tech. Report 93–105 (1993).
[33] A. Nerode and A. Yakhnis, An example of extraction of a finite control automaton and A. Nerode's AD-converter for a discrete sensing hybrid system, MSI Tech. Report 93–104 (1993).
[34] L.E. Neustadt, *Optimization* (Princeton University Press, Princeton, NJ, 1976).
[35] J. Warga, *Optimal Control of Differential and Functional Equations* (Academic Press, New York, 1972).
[36] L.C. Young, *Optimal Control Theory* (Chelsea, New York, 1980).